A Generalized Framework for Interactive Dynamic Simulation for MultiRigid Bodies

Wookho Son, Kyunghwan Kim, Nancy M. Amato, *Member, IEEE*, and Jeffrey C. Trinkle, *Member, IEEE*

***Abstract*—This paper presents a generalized framework for dynamic simulation realized in a prototype simulator called the Interactive Generalized Motion Simulator (I-GMS), which can simulate motions of multirigid-body systems with contact interaction in virtual environments. I-GMS is designed to meet two important goals: generality and interactivity. By generality, we mean a dynamic simulator which can easily support various systems of rigid bodies, ranging from a single free-flying rigid object to complex linkages such as those needed for robotic systems or human body simulation. To provide this generality, we have developed I-GMS in an object-oriented framework. The user interactivity is supported through a haptic interface for articulated bodies, introducing interactive dynamic simulation schemes. This user-interaction is achieved by performing push and pull operations via the PHANToM haptic device, which runs as an integrated part of I-GMS. Also, a hybrid scheme was used for simulating internal contacts (between bodies in the multirigid-body system) in the presence of friction, which could avoid the nonexistent solution problem often faced when solving contact problems with Coulomb friction. In our hybrid scheme, two impulse-based methods are exploited so that different methods are applied adaptively, depending on whether the current contact situation is characterized as "bouncing" or "steady." We demonstrate the user-interaction capability of I-GMS through on-line editing of trajectories of a 6-degree of freedom (dof) articulated structure.**

***Index Terms*—Articulated dynamics, contact dynamics, dynamic simulation, haptic interaction, interactive simulation, object-oriented design, rigid-body contact.**

I. Introduction

DYNAMIC simulation (possibly with contact interactions) arises in many engineering application domains such as virtual reality (VR), graphics, robot motion simulators, and computer games. The VR community is motivated by the desire to enhance the realism of the virtual environment in which the VR user is immersed. Robot engineers and motion animators want to describe at a very high level a task or action for a complex mechanical system and then use dynamic simulations to determine input trajectories to accomplish those tasks.

A simulation environment can be thought of containing multirigid-body systems, each of which consists of a number of passive bodies, called *freebodies*, 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 multirigid-body systems in the environment.

Most current prototype dynamic simulators have been developed for specific types of environments, such as simulating motions of robot manipulators or interactions between the robot manipulator/free body and the environment. Generally, they have not been widely applied to more complex structures such as multibranch linkages with movable bases or free bodies with frequent contact interaction. This is due to limitations in their design, which makes it extremely difficult to accommodate more general environments. Although many proprietary dynamic simulation packages are general and powerful enough to handle most application purposes, it is quite difficult or impossible for robotics researchers to extend or customize them to meet some experimental or research needs. Also, their user-interaction capabilities are very limited with most providing no interactive capabilities at all. Those that do support user interaction generally use keyboard input or graphical interfaces to enable parameter tuning.

Our general goal is to design and develop a general-purpose dynamic simulator which supports user interaction through haptics and is easily extensible to accommodate various kinematics and dynamics. A dynamic simulator with these qualities could be used as a backbone system by many researchers so that they could build their applications on top of it by providing the necessary customization.

Dealing with contacts, possibly with frictional effects among rigid bodies, is crucial to realize physically-plausible simulation in virtual environments. This is because true virtual environments would enable multiple types of interaction between bodies in contact or collision. During simulation of a system of rigid bodies in contact, there are various contact modes that need to be considered such as rolling, sliding, and breaking of contact. These should be dealt with cautiously as microscopic behaviors by incorporating appropriate contact dynamics whenever contact occurs. On the other hand, "bouncing" and "steady" contacts are two frequently occurring contact situations which should be viewed as macroscopic motions during simulation of a system of rigid bodies in contact. So far, no single existing method has been found to work well for both these contact situations.

Manuscript received July 2, 2002; revised December 27, 2002. This work was supported in part by the National Science Foundation under CAREER Award CCR-9624315 and Grants IIS-9619850, EIA-9805823, and EIA-9810937 and by the Texas Higher Education Coordinating Board under Grant ARP-036327-017. This paper was recommended by Associate Editor C. T. Lin.

W. Son is with the Virtual Reality Department, Electronics and Telecommunications Research Institute (ETRI), Taejon 305-350, Korea (e-mail: whson@etri.re.kr).

K. Kim is with the Wooshin Mechatronics Co. Ltd., Seoul 150-816, Korea (e-mail: kimk@wooshin-m.com).

N. M. Amato is with the Department of Computer Science, Texas A&M University, College Station, TX 77843-3112 USA (e-mail: amato@cs.tamu.edu).

J. C. Trinkle is with the Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590 USA (e-mail: trink@cs.rpi.edu).

Digital Object Identifier 10.1109/TSMCB.2003.818434

1083-4419/04$20.00 © 2004 IEEE

TABLE I
COMPARISON OF PUBLICLY AVAILABLE DYNAMIC SIMULATORS

	TOOLBOX [1]	*RCCL* [2]	*Multibody* [3]	*Isaac* [4]	*Impulse* [5]	*Berkelman* [6]	*Stanford* [7]	*I − GMS* [8]
Multi-body support			√					√
Extensibility			√					√
Haptic interactiion							√	√
Contact simulation				√	√	√	√	√
Rapid prototyping	√	√	√					√

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 updating the dynamics due to outside disturbances such as contact. This has many applications such as teleoperation of robots for remote inspection, virtual training, etc. Haptic interaction enables people to 'perform' work which requires touching or sophisticated grasping of any part without the physical presence of the operator in the scene. To our knowledge, no known simulator supports interactive dynamic simulation via haptic feedback while simulating dynamic motions of multirigid-bodies in the general sense. In particular, the use of user interaction for on-line creation and modification of trajectories of articulated robots is unknown so far.

Although much work has been done, research has not yet provided a dynamic simulator which is extensible to multiple various rigid-body systems or one which supports haptic user interaction. This paper describes the design and development of a multirigid-body dynamic simulator called the Interactive Generalized Motion Simulator (I-GMS), which addresses four different issues mentioned above; *object-oriented framework*, *extensibility*, *efficient contact simulation*, and *interactive simulation*.

The paper is organized as follows: Section II describes the related work in three different, related categories of topic. In Section III, we give the system overview in terms of the design principle and the dynamic models used for both simulation and haptic interaction. Section IV gives the experimental results for the interactive dynamic simulation for a linkage structure, which is applied to on-line trajectory modification for a robot manipulator. Finally, Section V concludes our work.

II. RELATED WORKS

Some simulators currently available in the public domain are compared according to some features of interest to us:

1) articulated-body dynamics;
2) methods for handling rigid-body contact with friction;
3) interactive simulation via haptic interaction (see Table I).

A. Existing Dynamic Simulators

Some packages are specifically for simulating robot manipulators (e.g., Robotics TOOLBOX [1], SPACELIB [9], and ROBOOP [10], the former being written in Matlab, while the latter two are developed in C++). RCCL [2] is a special language in C for robot control which provides Unix-based integration of external software modules. A graphic simulator is available for robot control in the simulation environment using RCCL [11]. ARCL [12] is a more modular and portable version of RCCL. Lee *et al.* [13] provide an interactive package that can be run in tandem with a physical robotics environment. Mirtich [5] has implemented an impulse-based approach to deal with contact interaction. Multibody [3] has been developed for simulating motions of articulated bodies. This package is similar to I-GMS in some features in that it is designed in an object-oriented manner. However, while it is intended as a basic system upon which more sophisticated dynamic simulators could be built, it does not handle user interaction and requires some effort to tailor it to a specific application. Symbolic Dynamic/Fortran Simulation Toolkit (SD/FAST) [14] generates codes for a particular physical model (in this case, rigid-body dynamics) of mechanical systems by taking a short description of an articulated system of rigid bodies. Isaac [4] is an ongoing project aimed at extending VR into the realm of multibody dynamics, focusing on unifying existing major technologies in geometric modeling, model-driven dynamic simulation, collision handling, motion control, sensory feedback, knowledge representation of the environement, planning, and computational modeling. There are also a number of efforts which specifically focus on human body simulation by developing dynamic control algorithms that deal with realistic human motions [15], [16]. Also, [17] uses various control methods to implement a physics-based torso simulation for humanoid robot, and applied to the task of performing a continous sequence of smooth movements by articulated agents.

More general and powerful simulation packages that can be found in the proprietary systems lack user-interaction capability when it is considered in the context of simulation [14], [18]–[21].

B. Articulated Dynamics

Walker *et al.* [22] have performed pioneering work in dynamics of articulated rigid bodies or constrained bodies. They have developed an $O(n^3)$ algorithm for computing forward dynamics of an n link manipulator, which explicitly builds the mass matrix for the system and inverts it to solve for joint accelerations. Featherstone's algorithm [23] is an $O(n)$ time method for computing forward dynamics, which exploits structural recursiveness for a loop-free n-link manipulator. Extensions of this basic serial chain algorithm to handle tree-like linkages and floating linkages are provided in [5]. In particular, recursive algorithms with $O(\log n)$ time complexity on $O(n)$ processors were presented in [24] and [25], which is applicable to articulated-bodies with kinematic trees and close loops. These are generalized coordinate approaches, meaning that there are as many state variables as degrees of freedom in the system. In these approaches, the constraints are automatically enforced because there are no invalid state configurations.

A maximal coordinate method is given in Baraff [26], which runs in $O(n)$ time using Lagrange multipliers. This provides the algorithmic framework for computing linkage dynamics with general constraint settings. In the multiplier method, more state variables are employed than there are degrees of freedom, which means that constraints must be continuously enforced unlike the generalized coordinate methods. This is because of drift problems due to coordinate redundancy.

C. Contact Simulation

Forces or impulses must be applied at contact points to prevent two contacting (or colliding) bodies from interpenetrating. Various methods have been developed for computing contact reaction forces, each with their own strengths and weaknesses.

Penalty methods are the simplest approach to computing contact forces [27], [28]. These methods introduce restoring forces when objects interpenetrate each other and do not strictly enforce nonpenetration. Instead, they keep penetrations negligible relative to the scale of the system. Constraint-based methods cast the contact-force computation problem as a nonlinear complementarity problem (NCP) [29]–[32].

Friction can be incorporated into the framework of NCP by modifying or adding constraints to it. To accomplish this, the Coulomb friction cone is usually approximated by a pyramid. Frictional constraints nullify the convexity of the LCP formulation and this in turn can cause solution nonexistence. This is because the use of Coulomb's friction law with the principles of classical rigid body dynamics introduces mathematical inconsistencies. As such, most existing constraint-based formulations do not guarantee a solution [29], [32]–[34]. They occasionally fail to give solutions depending on the frictional assumptions and contact status. Thus, these NCP-based contact dynamic formulations with Coulomb friction and acceleration variables have difficulties when applied to simulation, since they may yield inconsistencies (due to no solution cases) or indeterminate results (due to multiple solution cases).

To overcome these limitations, Song *et al.* [35] used a compliant contact model to derive stability criteria that yield a unique solution in terms of accelerations and forces for planar systems.

Impulse-based method model all contacts between bodies by collisions at contact points [5], [27], [36]. If friction and restitution are incorporated into the collision model, then sequences of collision impulses can approximate persistent contact modes such as rolling, sliding, and resting. The major advantage of an impulse-based method is that only a single contact point is handled each time; the key is that many of the computational problems associated with simultaneous contacts as in LCP methods are avoided. In this method, there is always a solution when computing the collision impulses between two colliding bodies. Impulse-based methods show promise for real-time simulation because they offer computational efficiency and solution existence coupled with reasonable physical accuracy [5].

Unlike other numerical methods for rigid-body formulations with friction, time-stepping methods easily allows for the incorporation of impulses. These relatively new methods do not explicitly identify impulsive forces. Instead, they use the integrals of the forces over each time step, which are finite even if there are impulsive forces [37]–[41]. These methods avoid the nonexistent solution problem that plagues acceleration-based methods. An advantage of this method is that it handles "steady contact" well because it implicitly assumes a zero coefficient of restitution.

D. Interactive Simulation via Haptics

Early work on haptic interaction has focused on haptic rendering of graphical environments, which focused on simulating the forces generated by contact with a virtual model so that a person's sense of touch can be used to interrogate the model. This force feedback coupled with the visual display has been used to realize surface shading, friction and texture [42], [43]. Thompson *et al.* introduced a tracing algorithm that supports haptic rendering of non-uniform rational B-splines (*NURBS*) surfaces without the use of any intermediate representation [44].

Colgate *et al.* [45] were the first to propose a virtual coupling between a simulation and a haptic display to simplify design and ensure stability [45]. There are some dynamic simulators with haptic interaction capability. The use of an impulse-based simulation as a general purpose multibody simulator for haptic display is presented in [46]. Haptic interaction for a point contact for rigid body dynamics is studied in [47]. Berkelman *et al.* [6] provide high-fidelity tool-based haptic interaction for a dynamic simulated rigid-body environment. In this system, a user can grasp a tool handle on the haptic device to interact in real time with the simulated environment while feeling the detailed reaction forces of the tool due to solid contacts, friction, and texture. A haptic interaction method for a virtual hand was presented for grasping dynamic objects and physical modeling of plasticity [48]. The use of an interpolation scheme for local update of an intermediate representation for the haptic device with simulation at a slower servo rate has been proposed in [47], [49].

Donald and Henle used haptics to browse and edit abstract representations of animation trajectories [50]. This approach uses a vector field method to allow the user to manipulate high degree-of-freedom motion-captured data with a lower degree-of-freedom control space.

To our knowledge, a framework provided for multirigid-body dynamic simulation in [7] is the most relevant to our work in terms of haptic interaction capability. This framework has been used to develop a simulator that can model interaction between generalized articulated mechanical systems and permit direct "hand-on" haptic interaction with the virtual environment. In fact, it has provided only a push operation using a stiff spring attached between the user's real finger and a "virtual proxy" as a means of haptic interaction, whereas I-GMS is providing both push and pull operations. This has been used to realize an interactive method of commanding a virtual robot to manipulate an object in a virtual world as is discussed in [51].

III. System Overview

In general, a simulation (virtual) environment is comprised of several multirigid-body systems which could differ in nature (i.e., in terms of dynamics and kinematics). Having different simulators for each different multirigid-body system does not lend itself to systematic construction of virtual environments

from the application programmer's view and complicates the development of an efficient real-time dynamic simulator from the viewpoint of developer.

To remedy this, we need a generic framework in which incremental addition of functionalities can be performed with relatively little effort. This will, at the same time, facilitate an easy and systematic composition of a complex environment. We use an object-oriented approach, which is well suited to this purpose. This is due to the reusability of classes through class derivation and the virtual function mechanism for function overloading, which are two powerful features of object-oriented design.

A. Construction of an Environment

In I-GMS, we introduce the term *MultiBody* to refer to a system of bodies which are under the influence of the same dynamic motion equation during simulation. For example, a free-flying robot, a robotic manipulator, and a free-falling cube are all considered as individual *MultiBodies*, each of which may use different motion equations for dynamic update during simulation. Generally, a virtual environment contains multiple *MultiBodies*. We also introduce the terms *FixedBody* and *FreeBody* to distinguish an unmovable body from a movable body. These are the basic building blocks for constructing any kind of *MultiBody* in three-dimensional (3-D) space. In other words, a *MultiBody* is a collection of one or more *FreeBodies* and *FixedBodies* that are connected by joints.

The generality of I-GMS is that various kinds of robots and mechanical systems can be constructed through appropriate composition of both *FreeBodies* and *FixedBodies* in a hierarchical manner, such as human-body models, robot manipulators, etc. Thus, a *MultiBody* is characterized by its component body types. For example, we consider a robot manipulator as a single *MultiBody*, which has a *FixedBody* as a base and multiple *FreeBody* as linkage bodies. On the other hand, a free-flying robot is considered a *MultiBody* with a *FreeBody* as a floating base. Also, a free-falling cube is considered to be a *MultiBody* that has a *FreeBody* as its only component. Many such free bodies could coexist at the same time without connections (e.g., a particle system), in which case, each individual body is considered a separate *MultiBody*. An obstacle in the environment is a *MultiBody* with one or more *FixedBodies* stacked together. This hierarchy in the environment is illustrated in Fig. 1. I-GMS's expressive power is illustrated by the exemplary construction of various mechanical systems shown in Table II.

In I-GMS, we can use the above-described hierarchical composition method to build multirigid-body systems as shown in the examples in Fig. 2. Fig. 2(a) describes a 6-degree of freedom (dof) robot manipulator, while Fig. 2(b) shows a human model with 34-dof which has all the necessary skeletal structure such as head, neck, arms, elbows, torso, waist, legs, and feet.

B. Object-Oriented Design

There exist several different basic dynamic engines in I-GMS. Through function overloading, these are available to different types of multirigid-body systems which need different

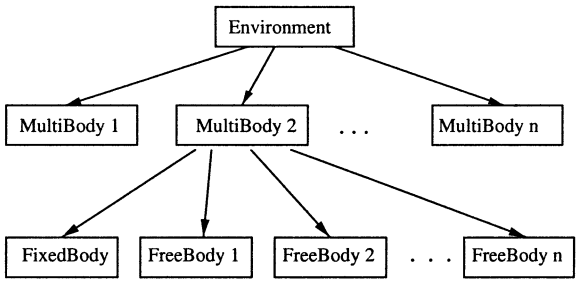

Fig. 1. Environment represented as a hierarchical structure of geometric components.

dynamics. Thus, a simulation driver can simply invoke a function of a standard name for motion simulation, without requiring detailed knowledge of the underlying structure of the system (see Fig. 3). Internally, the dynamic functions (kinematic functions, also) are provided as *virtual functions* through class derivations from the basic *MultiBody* class. This feature of object-oriented design provides the extensibility in I-GMS. In fact, a variety of multirigid-body systems can be supported through class derivation as needed. In particular, I-GMS is developed in the C++ programming language due to its desirable features which support our objectives, such as reusability and efficiency for real-time implementation. I-GMS 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. Users can construct a virtual environment through an appropriate composition of these entities in a hierarchical manner. 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.

I-GMS provides a framework so that applications can be written by following several high-level steps:

- specification of a multirigid-body system (through an input file to the system);
- building an appropriate internal functions;
- specification of a simulation scenario by a high-level driver.

With this philosophy in mind, construction of the virtual environment itself requires careful examination of the hierarchical characteristics inherent in the system so that the virtual function concept can be exploited with the *Body* class, which propagates characteristics of kinematic and dynamic functionalities to its derived classes. This is indicated in Fig. 4. Here, classes *FreeBody* and *FixedBody* are derived from the parent class *Body* for the purpose of differentiating between movable and unmovable. We can also think of deriving another class from each of these, such as *attFixedBody* and *attFreeBody*, respectively, where *att* (which is an acronym for "attribute") could be any relevant name, depending on the application. For our purposes, we have named them *RigidDynFixedBody* and *RigidDynFreeBody* to indicate that all the bodies are used for rigid-body dynamic simulation. On the other hand, the names

TABLE II
CONSTRUCTION OF MULTI-RIGID-BODY SYSTEMS IN I-GMS

System	Components, Connections, and Fixed/Movable Base
Human Body	*FreeBody* as a moving base and multiple serial connections of *Freebodies* from the base
Robot manipulator	*FreeBody* or *FixedBody* as a base and a serial connection of *Freebodies*
Pendulum	*FixedBody* as a base and a serial connection of *FreeBodies*
Mobile robot	Single *FreeBody* as the chassis
Walking robot	*FreeBody* as a torso and multiple branch-outs of serial linkages from the torso
Robot hand	*FreeBody* as a palm and multiple branch-outs of serial linkages from the palm

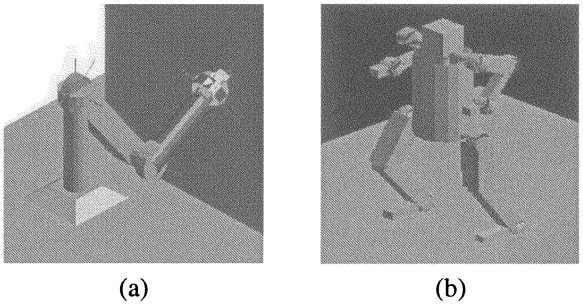

Fig. 2. (a) Six-dof robot manipulator (left). (b) Human model with 34-dof (right).

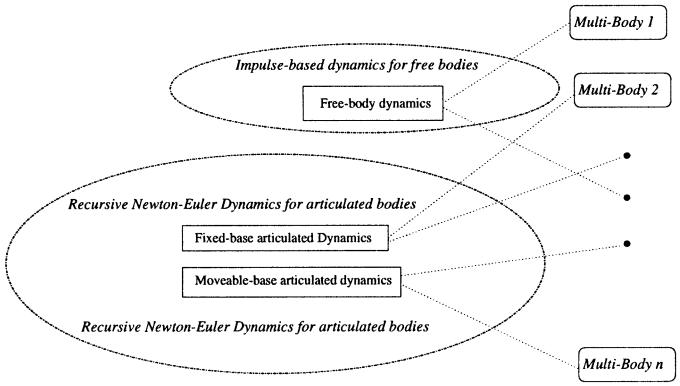

Fig. 3. Realization of dynamics in I-GMS.

obprmFixedBody and *obprmFreeBody* are used for a motion planning application, called the Obstacle-Based Probablistic Roadmap Method (OBPRM) [52]. In fact, I-GMS is used as the geometric and kinematic backbone of the motion planner where the planning environment (robot and obstacles) is constructed as defined above. The major classes available within I-GMS are listed in Table III along with brief descriptions of their functionalities.

The extensibility of I-GMS is mainly achieved through exploiting *class derivation* and *virtual functions* to apply common kinematic and dynamic functionalities to various types of rigid body systems. For example, to simulate a human-body model, we need to derive a new class (say, *HumanBody*) from the base class *MultiBody*. Since *MultiBody* already has all the necessary dynamic functions (and kinematic functions) as members, we need to overload these functions. This can be achieved by using *virtual functions*. In this way, the same names can be used for these functions across different multibody systems in the simulation driver at the high-level. These virtual functions

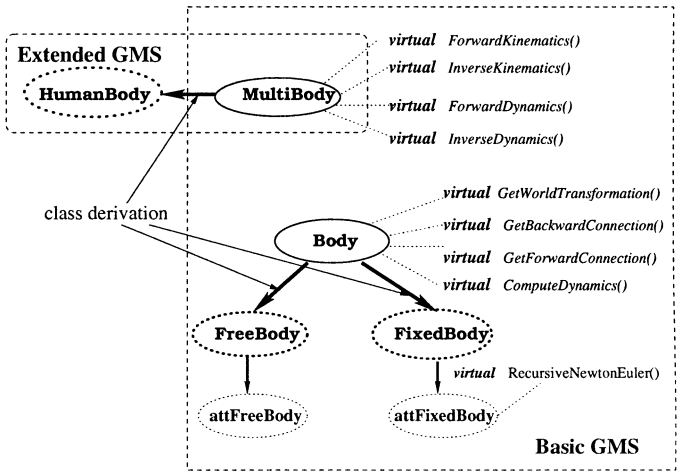

Fig. 4. Class hierarchy within the I-GMS.

are *ForwardKinematics*, *InverseKinematics*, *ForwardDynamics*, and *InverseDynamics*.

Dynamics for other complex mechanisms such as biped, quadped, or hexaped can be supported in the same way. In general, in I-GMS, adding support for a different type of multirigid-body system does not require substantial reimplementation effort. Instead, we just need to overload the virtual functions provided in the class derivation. This gives great generality to I-GMS in terms of extensibility.

C. Dynamic Model

The classical rigid-body dynamics is used for free body, while the recursive Newton–Euler equation is extended to accommodate both the movable base and the multilinkage characteristics of general articulated structures. A detailed description of this extension is shown using the modified DH parameter notation as given in [53]. In particular, Euler parameters and modified Rodrigues parameters (MRP) [54] are used to represent orientation parameters for free rigid bodies, since they provide a singularity-free representation.

D. Dynamics for multiLinkage Structures

The governing equations of motion for the multibody systems are Newton-Euler equations. We have distinguished between linked and nonlinked bodies in deriving the motion equations. The dynamics of linked bodies are further categorized into those having multibranch or single-branch connections and fixed or movable base. As described in Section III-B, these motion equations are used to implement the underlying dynamic engines of I-GMS.

TABLE III
OBJECT CLASSES AVAILABLE IN I-GMS

Basic Object Classes Available in I-GMS.	
Class	**Description**
Environment	The simulation environment on which the dynamic simulation is performed
MultiBody	A multi-rigid-body system. An environment is comprised of a finite set of *MultiBody*
Body	Each component body of the multi-rigid-body system
FixedBody	A body which is unmovable
FreeBody	A body which is movable (floating)
Polyhedron	The entity which deals with the geometry of the *Body*
Connection	Deals with connection between two *Body*'s using DH parameters
DHparameters	Contains the DH parameters and deals with associated transformations.
Transformation	Deals with the position and orientation between two different coordinate frames.
Contact	Deals with contact informations such as contact point, normal, contact frame, etc
Input	Plays as a buffer scheme from which a geometric structure of the simulation environment is built in a hierarchical manner, whether it being linked or non-linked rigid-body systems.
Matrix	Matrix representation and its associated operations.
Vector3D	Vector representation in 3D space and its associated operations.

We have extended the recursive Newton-Euler dynamics algorithm [53] (used for the fixed-base case) to describe the inverse dynamics. Here, the base is considered as a free-falling body in deriving the equations of inverse dynamics. Thus, we have attached a moving frame to the base, which results in an additional 6 dof for representing its position and orientation. This situation is depicted in Fig. 5.

To consider the movable base, 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 [53].)

Outward iteration: As a base case, we have

$$\begin{aligned} {}^{\mathbf{1}}\dot{\mathbf{v}}_{C_1} &= {}^{\mathbf{1}}\mathbf{R}_0(\dot{\mathbf{v}}_B + g) \\ {}^{\mathbf{1}}\omega_1 &= {}^{\mathbf{1}}\mathbf{R}_0\omega_B \\ {}^{\mathbf{1}}\dot{\omega}_1 &= {}^{\mathbf{1}}\mathbf{R}_0\dot{\omega}_B. \end{aligned}$$

Here, $\dot{v}_B$, ω_B, and $\dot{\omega}_B$ are the linear acceleration, angular velocity, and angular acceleration of the center of mass of the base body, respectively. ${}^i\dot{v}_{C_i}$ refers to the linear acceleration of the mass center of the ith body, and g is the gravity vector.

$i : 1 \rightarrow n-1$

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

where ${}^iP_j$ and ${}^iR_j$ refer to the position and orientation of the jth body frame origin in the ith body frame, respectively. (${}^iP_{C_i}$ refers to the position of the mass center of the ith body in the jth body frame.) Also, ${}^iv_i$ and ${}^i\dot{v}_i$ refer to the linear velocity and acceleration of the ith body frame origin, respectively. (Analogously, ${}^i\omega_i$ and ${}^i\dot{\omega}_i$ refer to the angular velocity and acceleration of the ith body frame origin, respectively. $\dot{\theta}_i$ and $\ddot{\theta}_i$ are the velocity and acceleration of the ith joint angle.) ${}^iI_{C_i}$ is the inertia tensor of the ith body in a frame located at the center of mass of the ith body, and m_i is the total mass of the ith body. ${}^iF_i$ and ${}^iN_i$ refer to the force and torque acting at the center of mass of the ith body-fixed origin, respectively. (Note that ${}^1R_0$, in the base case above, is an inverse of the orientation of the transformation from the inertia frame $\{\mathbf{0}\}$ to the base body.)

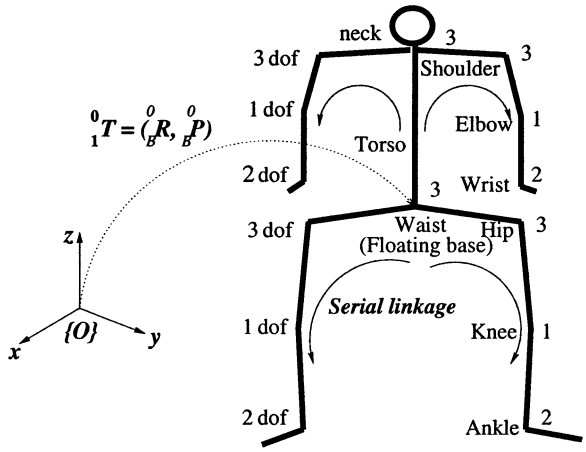

Fig. 5. Human body model as a multibranch linkage structure with a free-falling base.

Multiple-branch linkage connections are taken into consideration during the *inward iteration* in order to accommodate the external force due to contacts at a link in computing appropriate force and torque acting on it.

Inward iterations: $i : n \rightarrow 1$

$$\begin{aligned} {}^if_i &= \sum_{j\in O_i} {}^i\mathbf{R}_j\,{}^jf_j + {}^iR_0 \sum_{j\in O_i} {}^0f_{E_j} + {}^iF_i \\ {}^in_i &= \sum_{j\in O_i} \left[{}^iR_j\,{}^jn_j + {}^iP_{i+1} \times {}^iR_j\,{}^jf_j\right] \\ &\quad - \sum_{j\in O_i} {}^jc_j \times {}^0f_{E_j} + {}^iN_i + {}^iP_{C_i} \times {}^iF_i \\ \tau_i &= {}^in_i^T\,{}^i\hat{Z}_i. \end{aligned}$$

Here, f_{E_j} refers to the jth one among all the external forces acting on the ith link, O_i refering to the set of indices corresponding to all the branching links from the ith body. Also, ${}^{i}f_i$ and ${}^{i}n_i$ refer to the force and torque exerted on the ith body, respectively, while τ_i is the torque required on the ith joint. Note that ${}^{j}c_j$ is the transformation from a body-fixed frame (indexed by j) to the contact on it. The boldfaced terms account for both the effects due to multibranch links on an incident body and external forces acting on it.

Also, the force and moment acting at the floating base of the multilinkage structure is defined as

$$\mathbf{f_B} = {}^{\mathbf{0}}\mathbf{R_B}\,{}^{\mathbf{1}}\mathbf{f_1}$$
$$\mathbf{n_B} = {}^{\mathbf{0}}\mathbf{R_B}\,{}^{\mathbf{1}}\mathbf{n_1}$$

where f_B and n_B are the force and moment acting at the base body, ${}^{0}R_B$ 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 body-fixed local frame, respectively.

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

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

where

- pB (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 to represent the position and orientation of the base. The state-space representation of the inverse dynamics is the following:

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

where

- $H(X)$ $(N+6) \times (N+6)$ inertia matrix;
- $C(X,V)$ $(N+6) \times (N+6)$ matrix specifying centrifugal and Coriolis's effects;
- $G(X)$ $(N+6) \times 1$ vector specifying gravity effect;
- U $(N+6) \times 1$ vector specifying generalized force generated by actuation;
- U_E $(N+6) \times 1$ vector specifying generalized force generated by external forces.

E. Hybrid Simulation for Rigid Body Contact With Coulomb Friction

As a first step toward correctly handling contact situations among free rigid bodies, I-GMS introduces an adaptive scheme for handling two different contact situations ("bouncing contact" and "steady contact"),which commonly occur during simulation of a system of rigid-bodies [55]. In reality, a bouncing contact will be gradually diminished to a "steady contact" after sliding and rolling, where it is considered that no more bouncing occurs. In simulation, we consider bouncing to have ended when the bounces are so small that, numerically, the contact is steady state. Correctly tracking these changes in contact situations and applying appropriate contact dynamics is very important for physically correct motion simulation. Unfortunately, no single existing method handles all the contact situations well.

To deal with bouncing contact interactions between bodies, we have used the impulse-based approach introduced by Mirtich [5]. In this scheme, all types of contact (colliding, rolling, sliding, and resting) are modeled as a series of collision impulses between the bodies in contact. This method works well on systems of bodies where the contact modes change rapidly. However, they cannot efficiently handle more than a few simultaneous and persistent contacts. To handle "steady contact," we use an implicit time-stepping method for simulating systems of rigid bodies developed by Stewart and Trinkle [39]. Unlike other methods which take an instantaneous point of view [32], this relatively new method does not explicitly identify impulsive forces. Instead, the method uses the integrals of the forces over each time step, which are finite even if there are impulsive forces.

In particular, the Stewart and Trinkle's method is formulated as a complementarity problem which is distinguished as follows: Given $g: R^n \to R^n$, find z such that

$$z \geq 0, \quad g(z) \geq 0, \quad z^T g(z) = 0.$$

Here, z and $g(z)$ are vectors of equal length (n), roughly described as relative accelerations and contact forces expressed in coordinate frames attached to the contact points (assumed isolated). This complimentarity constraint enforces the idea that a nonzero force can only exist $(f(z) > 0)$ if the contact is being maintained $(z = 0)$. And a contact force may not exist $(f(z) = 0)$ if the contact is breaking $(z > 0)$. Problems with solution existence of the individual complementarity problems have been at least partially solved by casting the dynamics problem in terms of impulse and meomentum variables rather than force and acceleration variables [29]. In fact, a linearized version of this complementarity problem (LCP), which is applied to our hybrid scheme, is proven to have solution(s) which are unique for most problems, although uniqueness is not guaranteed [56]. [Anitescue *et al.* proved the existence of the solution(s) of this LCP formulation [40].] This effectively has allowed the search for solutions to include impulsive forces (as finite impulses).

Our adaptive scheme for handling contact during simulation is represented in the state transition diagram shown in Fig. 6. During simulation of rigid bodies, whenever a bouncing contact is detected, Mirtich's algorithm is applied, while Stewart's method is applied to cases of steady contact. A steady contact is detected when the bounces become so small that it is considered that the bouncing has ended. We use a tolerance to detect this during contact simulation. All these transitions in contact dynamics occur automatically during simulation by our adaptive scheme which keeps track of changes in the contact situation.

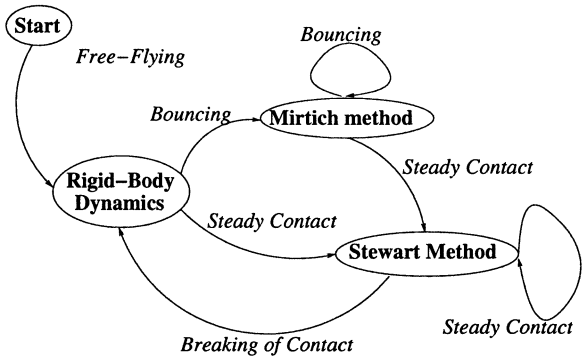

Fig. 6. Control flow of adaptive contact simulation.

In particular, the simulation for contact situations is driven by contacts. This means that a "Contact" class is instantiated at each time step in which a collision is detected and an explicit (via Mirtich's algorithm) or implicit (via Stewart's LCP algorithm) calculation of the impulse is performed using the kinematics and dynamic information available within it. This approach provides scalability when it is applied to the multiple contact case. The formulation for computing contact dynamics is correspondingly scaled in a natural manner in proportion to the complexity of the number of contacts. In this way, we are able to maintain real-time capability, while at the same time achieving physically correct motions.

The LCP formulations were solved using a numerical library developed based on Lemke's algorithm which is a pivoting method similar to the simplex method for linear programming. Specifically, we have used the mathematical library, *Meschach* [57], which is written in C. We call *Meschach*'s functions from I-GMS.

F. Interactive Simulation Through Haptic Interaction

In robotics, handling the interaction between the manipulator and the environment has been a fundamental capability for carrying out successful robot task. Thus, much work has been done toward developing efficient control strategy to deal with this issue. In particular, a few effective force control strategies are presented in [58] for a theoretical and experimental treatment of robot interaction control: stiffness and impedance control, motion control, and some advanced force and position control schemes.

In the context of robot interaction with the environment, the capability of editing the trajectory of a robot is a very powerful tool for environments which change dynamically. For example, obstacles in the environment can move, invalidating a preplanned trajectory. In this situation, replanning is very costly as is widely discussed in [59]. An efficient way of modifying the invalid portion of the planned trajectory would offer a potentially more efficient solution.

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 offline 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. In particular, the user interaction in I-GMS is focused on the online editing and creation of trajectories for articulated robots. (Refer to [8].)

1) On-Line Trajectory Modification: The PHANToM haptic device [60] 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 (1) in Section III-F2]. 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.

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.

Note that a user usually performs haptic interaction in a sporadic manner. Hence 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. 7.

2) Dynamics for Haptic Interaction: **Push operation**: For the push operation, I-GMS considers the haptic interaction on a multirigid-body system as an external force applied to it by the user, acting at a contact point on the body surface. Thus, haptic touch on an articulate structure 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)]:

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

$\mathbf{U_E}$ accounts for the joint torque vector corresponding to the contact force c due to collision as follows:

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

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 $d_{\text{penetration}}$ is the penetration distance between the haptic device and the virtual object.

In our case, $d_{\text{penetration}}$ is determined by the distance between the the actual position of the haptic touch and the projection of the haptic position [we call the projected position of the haptic touch the surface contact point (SCP)].

Pull operation: For the pull operation, I-GMS uses an impedance controller approach introduced by [61]. The impedance controller calculates the force F from the virtual spring, damper, and mass. This method is very similiar to the cartesian impedance approach used by [17], except we used

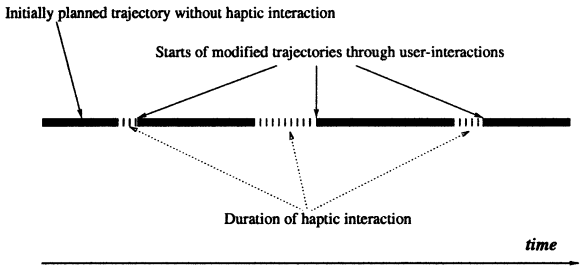

Fig. 7. Interactive simulation as a sequence of interleaved operations with I-GMS.

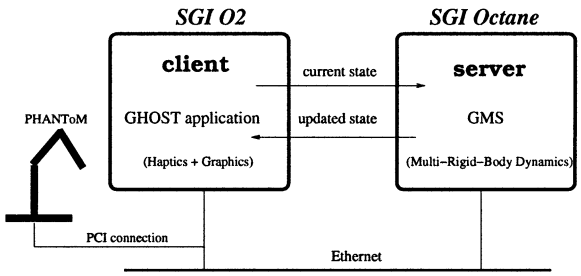

Fig. 8. Integration of I-GMS with PHANToM.

haptics as a means of interaction. 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 (3), which represents a special case of impedance control.

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

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 $\dot{X}_{\text{endeff}}$ and $\dot{X}_{ph}$ 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 U at the joints, which are calculated using the Jacobian $J(\theta_{\text{endeff}})$ as in the following relation:

$$U = J(\theta_{\text{endeff}})^T F. \tag{5}$$

This U 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 consisted of a 3-dof PHANToM haptic device [60], an SGI O2 graphics workstation (graphics display), and an SGI Octane (dynamic computation server). The PHANToM is a force-feedback device in which an operator generates position commands by moving a pencil-like interface.

Our PHANToM cannot exert moments since although the finger tip has 6 dof, it has only 3 dof for exerting forces. It has a finger controller with three active degrees of freedom (dof) and a 3 dof passive gimbal that permits users to feel the force that arises from point interactions with virtual objects. A multilateral connection exists between the PHANToM, the SGI O2 (the graphics display), and the SGI Octane (the 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 applications which were developed using the C++ *General Haptic Open Software Toolkit* (GHOST SDK) [62] in such a way that both haptic and dynamic computations occur at the same servo cycle. This was necessary to reflect I-GMS's state change due to dynamic computation appropriately within the GHOST application. This integration is described in Fig. 8.

Motion simulation of any articulated figure involves trajectory generation, computation of inverse dynamics, and computation of forward dynamics. Interactive simulation needs an additional step for performing haptic interaction whenever there is a contact (touch) by the user during the simulation. The overall computational cost for performing these steps increases in proportion to the dofs of the multirigid-body system. Thus, the dynamic simulation will be slower for complex models, causing some visual lag at each time step which in turn may result in nonreal-time simulation. It poses an even more serious problem when interactive simulation is performed during real-time simulation, since stable haptic interaction requires 1 kHz servo rate, which is very hard to attain for a high dof model. This situation is illustrated by the approximate computational costs for articulated structures of various dof as shown in Table IV.

To achieve realistic feedback, all steps should be executed with 1-kHz frequency. However, in our applications, the servo rate was usually around 250 Hz for a 6-dof robot manipulator, which is four times slower than needed. Thus, to achieve realistic interaction, we use two techniques: one is the distribution of the computations over the network and the other is the use of an interpolation of the system's state between network relays. Having a sound interpolation scheme is very important for the distributed computation. This is because it is quite difficult to achieve a reasonable dynamic feedback upon touch if the delay is too large to be compensated for by just a simple interpolation.

4) Distributed Computation of Haptics: 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 transmission of data somewhat faster than that of TCP, but this is at the cost of reliable transmission of data packets. However, for our own purposes, transmitting data at a faster servo rate is more crucial than the possible minimum loss of data to maintain realistic haptic interaction between the simulator and the user.

To ensure reliable haptic interaction through maintaining a high servo rate, we have used a simple interpolation scheme when implementing the above-mentioned distributed computation. If the remote server does not return the dynamics results after a certain preset time duration, the client (haptic computation) uses the results computed at a previous time cycle. To

TABLE IV
COMPUTATIONAL OVERHEADS MEASURED FOR SYSTEMS OF VARIOUS COMPLEXITIES

	$3-dof\ robot$	$6-dof\ robot$	$34-dof\ human\ model$
Inverse dynamics	0.7 ms	1.5 ms	8.5 ms
Servo rate	750 Hz	250 Hz	40 Hz

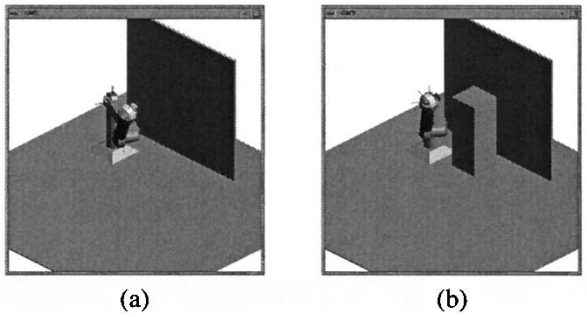

Fig. 9. (a) Original environment. (b) Changed environment.

preset this time duration which in turn determines the servo rate, I-GMS uses an alarm clock available within the Unix system. This mechanism allows I-GMS to poll the socket on the client side at a specified time interval to check if renewed data have been received. If there are renewed data available, I-GMS uses it to update the system's state. Otherwise, it uses the interpolated data. The preset time interval is adjustable within I-GMS, and setting it to approximately 2 ms gives reasonable haptic interaction from our experimentation.

IV. SIMULATION EXAMPLE

We have demonstrated interactive simulations on a 6-dof robot manipulator through online editing of preplanned trajectories. We consider a simple scenario where a 6-dof robot manipulator (see Fig. 9) 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. 9(a), it is not very hard to plan the trajectory and have the robot to follow it. A straight-line trajectory is given in Fig. 10. However, when an obstacle is introduced in the way of preplanned trajectory, as shown in Fig. 9(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.

A. Push for 6-Dof Robot Manipulator

An interactive way of modifying an existing trajectory is an efficient way of avoiding the costly preprocessing. In our scheme of modifying the trajectory, 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.

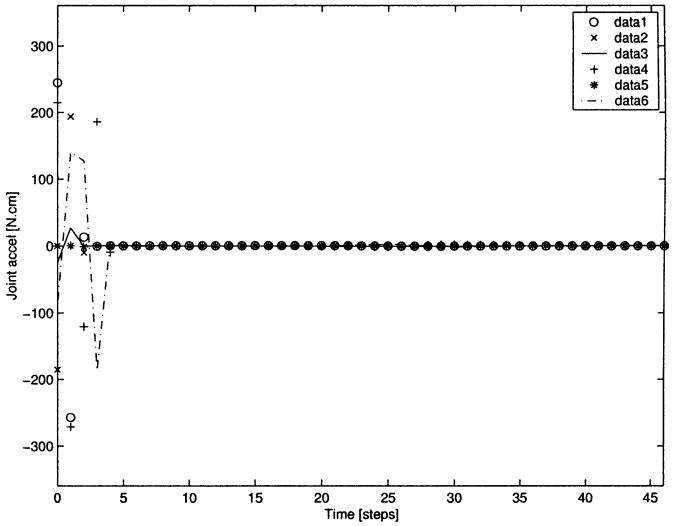

Fig. 10. Nominal trajectory.

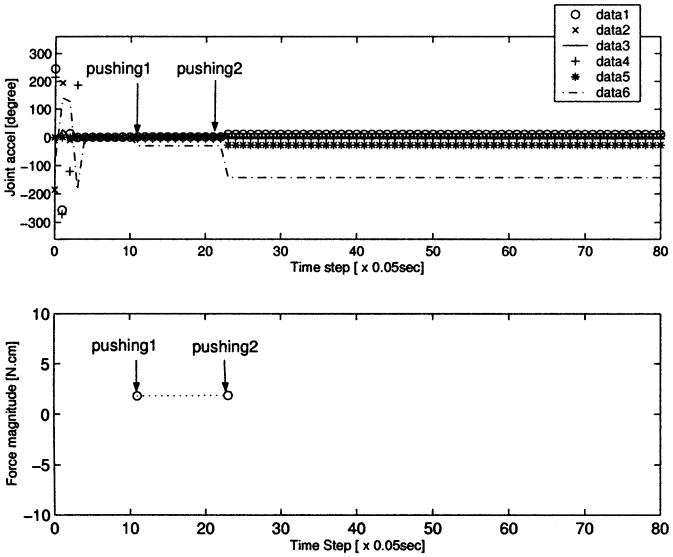

Fig. 11. Trajectory disturbed by push operation.

The initial steps in Fig. 11 (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 Fig. 11 (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 Fig. 11 (bottom). 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. 7) during the interleaved operations. In fact, the

Fig. 12. Simulation snapshots of the 6-dof robot manipulator following the modified trajectory.

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 toward 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 (2) at time steps 11 and 23 (the magnitude unit is newton-centimeters).

The simulation snapshots of the modified trajectory are given in Fig. 12. This shows the 6-dof manipulator taking a detour around the obstacle rather than taking the original straight-line trajectory to avoid the obstacle. This example shows that just a few haptic pushes at appropriate points on the manipulator bodies could change the preplanned trajectory to avoid colliding with an obstacle. (In Fig. 12, all the legend items data1, data2, …, correspond to joint1, joint2, …, respectively.)

B. 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 was not placed in collision.

The plots in Fig. 13 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 toward 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 tradeoffs in terms of advantages and disadvantages in using these two modes of operation for the trajectory modifications. We note that relatively greater forces are required for the dragging operation than the push operation and this is considered 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 relatively much harder for the user to guide the articulated bodies toward a target position, since the pushing operation is an indirect way of achieving the desired motion. Thus, it usually takes more sophisticated effort for the user to be able to manipulate the articulated bodies haptically, but the pushing operation requires less force to move the articulated bodies.

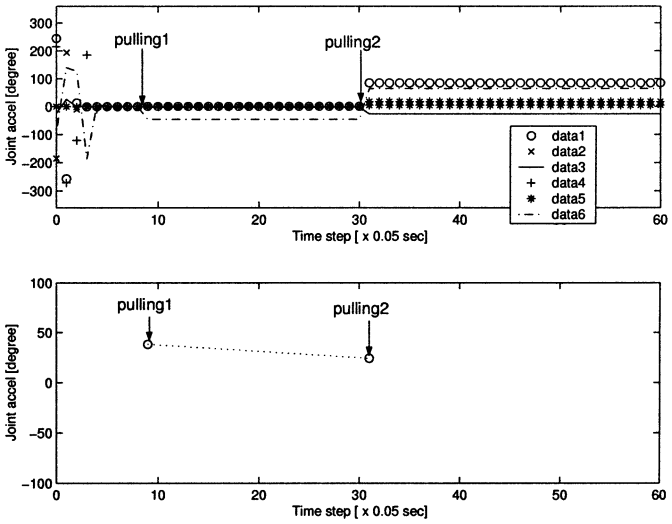

Fig. 13. Trajectory disturbed by pull operation.

V. Conclusion and Future Work

We have proposed a generic framework for a generalized dynamic simulator I-GMS, and have provided a prototype implementation of it which supports haptic user interaction. This framework is general enough to accommodate new multirigid-body systems (mostly complex linkage structures) whenever they are needed. Simulation results of our prototype system showed some promise toward its claim of extensibility and interactivity.

In particular, we have developed a hybrid scheme to simulate contact with Coulomb friction of a system of rigid bodies. This scheme has the capability to adaptively change contact dynamics depending on two contact situations: "bouncing" and "steady." So far, no existing contact dynamic formulation handles all types of the contact situations well, since most instantaneous formulations (such as most LCPs) do not guarantee solution existence. This is an important step toward a more sophisticated dynamic simulator to deal with general situations of contacts.

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 as well, once performance issues can be resolved to ensure stable haptic interaction.

There are still some stability issues remaining to be solved. Especially, dynamic simulation with contact involves fine step size tuning during simulation which requires applying an adaptive step size for correct simulation without numerical failure. In

a hybrid scheme like the one introduced here, using the correct step size for the integration is extremely important since it helps avoid some erratic system states which are unexpected when one type of contact dynamics is applied. The stability of the haptic interaction could be improved further by devising good interpolation schemes for enhancing the dynamic feedback upon haptic touch by the user in the case of highly articulated structures.

We would also like to work on incorporating haptic interaction for free bodies having a secondary 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 secondary contact. This is a complicated problem that requires exact contact mechanics to predict physically correct contact modes 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] *Robotics Toolbox for Use With Matlab (Release 4)*, CSIRO Div. Manufacturing Technol., Canberra, Australia, 1996.

[2] V. Hayward and R. P. Paul, "Robot manipulator control under unix—RCCL: A Robot control C library," *Int. J. Robotics Res.*, vol. 5, no. 4, pp. 94–111, 1986.

[3] B. Mirtich. (1996) Multibody dynamic package. [Online]. Available: http://www.merl.com/projects/rigidBodySim/multibodyDyn

[4] J. Cremer. (1994) Issac. [Online]. Available: http://www.cs.uiowa.edu/

[5] B. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, Univ. California, Berkley, CA, 1996.

[6] P. J. Berkelman, R. L. Hollis, and D. Baraff, "Interaction with a real-time dynamic environment simulation using a magnetic levitation haptic interface device," in *Proc. IEEE Int. Conf. Robot. Automat.*, Detroit, MI, May 1999, pp. 3261–3266.

[7] D. C. Ruspini and O. Khatib, "A framework for multicontact multibody dynamic simulation and haptic display," in *Proc. IEEE Int. Conf. Intel. Robotic Syst.*, Oct. 2000, pp. 1322–1327.

[8] W. Son, K. Kim, and N. M. Amato, "An interactive generalized motion simulator (I-GMS) in an object-oriented framework," in *Proc. Comput. Animation Conf*, Philadelphia, PA, May 2000, pp. 176–181.

[9] *SPACELIB in MATLAB Version 1.0*, Univ. Brescia, Brescia, italy, 1998.

[10] R. Gourdeau, "Object-oriented programming for robotic manipulator simulation," *IEEE Trans. Robot. Automat. Mag.*, vol. 4, pp. 21–29, Sept. 1999.

[11] V. Hayward and R. P. Paul, "Robot manipulator control under unix, RCCL: A robot control c libary," *Int. J. Robotic Res.*, vol. 5, no. 4, pp. 94–111, 1986.

[12] "ARCL Robot Programming System," CSIRO Division Manufacturing Technol., Canberra, Australia, 1991.

[13] P. U. Lee, D. C. Ruspini, and O. Khatib, "Dynamic simulation of interactive robotic environment," in *Proc. IEEE Int. Conf. Robot. Automat.*, San Diego, CA, May 1994, pp. 1147–1152.

[14] *SD/FAST User's Manual*, Symbolic Dyn., , 2001.

[15] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, "Animating human athletics," in *Proc. SIGGRAPH*, Los Angeles, CA, Aug. 1995, pp. 71–78.

[16] E. Kokkevis, D. Metaxas, and N. I. Badler, "User-controlled physics-based animation for articulated figures," in *Proc. Comput. Animation*, Houston, TX, June 1996, pp. 16–25.

[17] M. J. Mataric, "Making complex articulated agents dance: An analysis of control methods drawn from robotics, animation, and biology," *J. Autonomous Agents MultiAgent Syst.*, vol. 2, no. 1, pp. 23–44, 1999.

[18] Delmia Corp.. (2002) Igrip, Auburn Hills, MI. [Online]. Available: http://www.delmia.com

[19] MSC Software Corp., Santa Ana, CA. (2000) ADAMS. ADAMS and M. D. Inc.. [Online]. Available: http://www.adams.com

[20] (2000) MathEngine—The Dynamics Toolkit* 2.0 SDK. [Online]. Available: http://www.mathengine.com

[21] MSC Software, Santa Ana, CA, Visual Nastran 4d, Redwood City, CA, http://www.krev.com, 2002.

[22] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanism," *ASME J. Dyn. Syst., Meas., Contr.*, vol. 104, pp. 205–211, 1982.

[23] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *Int. J. Robotics Res.*, vol. 2, no. 1, pp. 13–30, 1983.

[24] ——, "A divide-and-conquer articulated-body algorithm for parallel $o(\log n)$ calculation of rigid-body dynamics: Part1: Basic algorithm," *Int. J. Robotics Res.*, vol. 18, no. 9, pp. 867–875, 1999.

[25] ——, "A divide-and-conquer articulated-body algorithm for parallel $o(\log n)$ calculation of rigid-body dynamics: Part 2: Trees, loops and accuracy," *Int. J. Robotics Res.*, vol. 18, no. 9, pp. 876–892, 1999.

[26] D. Baraff, "Linear-time dynamics using Lagrange multipliers," in *Proc. SIGGRAPH*, New Orleans, LA, Aug. 1996, pp. 137–146.

[27] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," *Comput. Graphics*, vol. 22, no. 4, pp. 289–298, 1998.

[28] M. H. Railbert and J. K. Hodgins, "Animation of dynamic legged locomotion," *Comput. Graphics*, vol. 25, no. 4, pp. 349–358, 1991.

[29] P. Lötstedt, "Numerical simulation of time-dependent contact friction problems in rigid body mechanics," *Zeitschrift für Angewandte Mathematik and Mechanik*, vol. 5, no. 2, pp. 370–393, 1984.

[30] D. Baraff, "Dynamic simulation of nonpenetrating rigid bodies," Ph.D. dissertation, Cornell Univ., Ithaca, NY, 1992.

[31] J. S. Pang and J. C. Trinkle, "Complementarity formulations and existence of solutions of dynamic multirigid-body contact problems with coulomb friction," *Math. Programming*, vol. 73, no. 4, pp. 199–226, 1996.

[32] J. C. Trinkle, J. S. Pang, S. Sudarsky, and G. Lo, "On dynamic multirigid-body contact problems with coulomb friction," *Zeitschrift für Angewandte Mathematik and Mechanik*, vol. 77, no. 4, pp. 267–279, 1997.

[33] D. Baraff, "Issues in computing contact forces for nonpenetrating rigid bodies," *Algorithmica*, vol. 10, no. 2–4, pp. 292–352, 1993.

[34] ——, "Fast contact force computation for nonpenetrating rigid bodies," in *Proc. SIGGRAPH*, Orlando, FL, Aug 1994, pp. 23–34.

[35] P. Song, P. Kraus, and V. Kumar, "Analysis of rigid body dynamic model for simulation of systems with frictional contacts," *ASME J. Applied Mechanics*, vol. 68, no. 1, pp. 118–128, 2001.

[36] J. K. Hahn, "Realistic animation of rigid bodies," in *Proc. SIGGRAPH*, vol. 22, Atlanta, GA, Aug 1988, pp. 299–308.

[37] J. Moreau, *Nonsmooth Mechanics and Applications: Unilateral Contact and Dry Friction in Finite Freedom Dynamics*. Berlin, Germany: Springer-Verlag, 1988, pp. 1–82.

[38] M. Marques, *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*. Berlin, Germany: Birkhauser-Verlag, 1993.

[39] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and coulomb friction," *Int. J. Numerical Methods Eng.*, vol. 39, no. 4, pp. 2673–2691, 1996.

[40] M. Anitescu, F. A. Potra, and D. Stewart, "Time-Stepping for three-dimensional rigid body dynamics," Dept. Math., Univ. Iowa, Iowa City, IA, Tech. Rep. 98–02, 1998.

[41] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid-body dynamics with coulomb friction," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, May 2000, pp. 162–169.

[42] D. C. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphi," in *Proc. SIGGRAPH*, Los Angeles, CA, Aug. 1997, pp. 345–352.

[43] C. Zilles and K. Salisbury, "A constraint-based god-object method for haptic display," in *Proc. IEEE Int. Conf. Intell. Robotic Syst.*, Pittsburgh, PA, Aug. 1995, pp. 146–151.

[44] T. V. T. Il, D. E. Johnson, and E. Cohen, "Direct haptic rendering of sculptured models," in *Proc. Symp. Interactive 3D Graphics*, Providence, RI, Apr. 1997, pp. 167–176.

[45] J. E. Colgate, M. C. Stanley, and J. M. Brown, "Issues in the haptic display of tool use," in *Proc. IEEE Int. Conf. Intell. Robotic Syst.*, Pittsburgh, PA, Aug. 1999, pp. 140–145.

[46] B. Chang and J. E. Colgate, "Real-time impulse-based simulation of rigid body system for haptic display," in *Proc. ASME Dyn. Syst. Contr. Divisions*, Houston, TX, Apr. 1997, pp. 1–8.

[47] S. Vedula and D. Baraff, "Force feedback in interactive dynamic simulation," in *Proc. First PHANToM User's Group Workshop*, Dedham, MA, Sept 1996, pp. 25–31.
[48] V. Popescu, G. Burdea, and M. Bouzit, *Proc. Comput. Animation Virtual Reality Simulation Modeling Haptic Glove*, Geneva, Switzerland, May 1999, pp. 195–200.
[49] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate representation for stiff virtual objects," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, Research Triangle Park, NC, Mar. 1995, pp. 203–210.
[50] B. R. Donald and F. Henle, "Using haptic vector fields for animation motion control," in *Proc. IEEE Int. Conf. Robotic Automat.*, San Francisco, CA, Apr. 2000, pp. 3435–3442.
[51] O. Khatib, O. Brock, K. S. Chang, F. Conti, D. C. Ruspini, and L. Sentis, "Robotics and interactive simulation," *Commun. ACM*, vol. 45, no. 3, pp. 46–51, 2002.
[52] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Proc. Int. Workshop Algorithmic Foundations Robotics*, Houston, TX, Mar. 1998, pp. 155–168.
[53] J. J. Craig, *Introduction to Robotics, Mechanics, and Control*. Reading, MA: Addison-Wesley, 1986.
[54] P. Tsiotras, J. L. Junkins, and H. Schaub, "Higher-order cayley transforms with applications to attitude representations," *J. Guidance, Contr. Dyn.*, vol. 20, no. 3, pp. 528–535, 1997.
[55] W. Son, J. C. Trinkle, and N. M. Amato, "An interactive generalized motion simulator (I-GMS) in an object-oriented framework," in *Proc. IEEE Int. Conf. Robotics Automat.*, Seoul, Korea, May 2001, pp. 3789–3794.
[56] R. W. Cottle, J. S. Pang, and R. E. Stone, *The Linear Complementarity Problem*, ser. Computer Science and Scientific Computing. New York: Academic, 1992.
[57] *Meschach: Matrix Computations in C*. Canberra, Australia: Center Math. Applicat., School Math. Sci., Australian Nat. Univ., 1994.
[58] B. Siciliano and L. Villani, *Robot Force Control*. Boston, MA: Kluwer, 2000.
[59] J. C. Latombe, *Probot Motion Planning*. Boston, MA: Kluwer, 1990.
[60] T. H. Massie and J. K. Salisbury, "The phantom haptic interface: A device for probing virtual objects," in *ASME Haptic Interfaces for Virtual Environment Teleoperator Systems*. New York: ASME, 1994, pp. 295–302.
[61] N. Hogan, "Impedance control: An approach to manipulation," *J. Dyn. Syst., Meas., Contr.*, vol. 107, no. 4, pp. 1–23, 1985.
[62] *GHOST Software Developer's Toolkit Programmer's Guide Version 1.21*, Sensable Technologies, Woburn, MA, 1999.

Wookho Son was born in Kyoungju, Korea, in 1964. He received the B.S. degree in computer science from Yonsei University, Seoul, Korea, in 1987 and the M.S. and Ph.D. degrees in computer science from the Texas AM University, College Station, in 1996 and 2001, respectively.

He is currently a senior research scientist with the Virtual Reality Research Center, Electronics and Telecommunications Research Institute (ETRI), Taejeon, Korea. His research interests include robotics, virtual reality (especially haptic interaction), and physically based dynamic simulation.

Kyunghwan Kim was born in Milryang, Korea, in 1967. He received the B.E. degree in electrical engineering from Yonsei University, Seoul, Korea in 1992 and the M.E. and the Ph.D. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan in 1994 and 1997, respectively.

From September 1997 to August 1999, he was a postdoctoral research associate with the University of Wisconsin, Madison and Texas A&M University, College Station. From September 1999 to June 2002, he was a senior research scientist with the Korea Institute of Science and Technology (KIST), Seoul. He is currently a director of Wooshin Mechatronics Co., Ltd., Seoul. His research interests include in robotics (especially, human-computer interface), micro and nano robotics, and semiconductor handling.

Nancy M. Amato (S'93–M'95) received the B.S. and A.B. degrees in mathematical sciences and economics, respectively, from Stanford University, Stanford, CA, and the M.S. and Ph.D. degrees in computer science from the University of California, Berkeley, and the University of Illinois at Urbana-Champaign, respectively.

She is currently an associate professor of computer science at Texas A&M University, College Station. Her main areas of research focus are motion planning, high-performance computing, and computational biology and geometry.

She is an Associate Editor of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION and of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. She is a member of the Computing Research Association's Committee on the Status of Women in Computing Research (CRA-W), and she directs the CRA-W's Distributed Mentor Program (http://www.cra.org/Activities/craw/dmp/). She is a recipient of a CAREER Award from the National Science Foundation and was an AT&T Bell Laboratories Ph.D. Scholar.

Jeffrey C. Trinkle (S'84–M'03) received the bachelor's degree in physics in 1979 and engineering science and mechanics in 1979 from Ursinus College, Collegeville, PA, and Georgia Institute of Technology, Atlanta, respectively. He received the Ph.D. degree from the Department of Systems Engineering at the University of Pennsylvania, Philadelphia, in 1987.

He was a member of the Fiber Composites Group at Lawrence Livermore National Laboratory, Livermore, CA, for two and one half years before returing to graduate school in 1982. Since 1987, he has held faculty positions in the Department of Systems and Industrial Engineering at the University of Arizona, Tucson, and the Department of Computer Science, Texas A&M University, College Station. He was also a Principal member of Technical Staff with the Intelligent Systems and Robotics Center, Sandia National Laboratories, Albuquerque, NM, from 1998 to 2003. He is currently a Professor and Chairman of the Department of Computer Science at Rensselaer Polytechnic Institute, Troy, NY. His primary research interests lie in the areas of robotic manipulation planning, mutibody dynamics, and automated fixturing.