

REAL-TIME IMPULSE-BASED SIMULATION OF RIGID BODY SYSTEMS FOR HAPTIC DISPLAY

**Beeling Chang
J. Edward Colgate**

Department of Mechanical Engineering
Northwestern University
Evanston, IL 60208, USA

bchang@nwu.edu

colgate@nwu.edu

ABSTRACT

To date, haptic interfaces have rarely been used to display complex virtual environments such as those involving 3D mechanical assembly operations, or the use of hand tools. Of the limitations which have been encountered, one is the absence of suitable simulation techniques which provide a

approach. In section 2, the idea of a virtual coupling and the impulse-based method are reviewed. In section 3, we distinguish between haptic simulation and computer graphic simulation, noting restrictions which apply specifically to haptic simulation. Then, we extend the techniques used in impulse-based simulation for use in haptic simulation in section 4. The new approach based on this work has been implemented for a class of simple planar virtual environments and interfaced to both one and two degree-of- freedom haptic displays. Section 5 shows the experimental results and discusses some unexpected problems.

2. HAPTIC SIMULATION OVERVIEW

Haptic display of dynamical objects requires a virtual environment simulator, a controller, and a haptic device. Figure 1 outlines these basic elements and the overall structure of the system. The simulation algorithm loops through three steps: integration of states of dynamic bodies, collision detection/contact determination, and collision response. The controller, then, displays the reaction force to the user. This section briefly reviews the virtual coupler and impulse-based simulation.

2.1 Virtual coupling: bridge between the haptic interface and the virtual environment

Colgate (1995) has recently proposed a novel approach to the haptic display of complex systems which should allow stability to be guaranteed without any simulation-specific parameter tuning. In practice, a small number of parameters (as few as two) would be tuned during device calibration; thereafter, a user could design a new environment with an assurance of stability. The basic idea is illustrated in Figure 1. There are two key elements: one, the environment is simulated by some method that is guaranteed to be *discrete time passive*, or nearly so; two, the handle of the virtual tool is connected to the handle of the haptic display via a multi-dimensional coupling consisting of stiffness and damping. More details can be found in (Brown and Colgate, 1997).

2.2 Impulse-based simulation

Mirtich and Canny (Mirtich and Canny, 1994) have proposed a new approach to three-dimensional rigid body simulation. In their approach, the Lin-Canny closest features (Lin, 1993) algorithm and scheduling scheme are used as a collision detection system. Based on separation distance and body velocities/accelerations, a heap of time-to-collide estimates is created and prioritized. Integration is performed up to the expected time of collision of the top pair in the prioritized heap. After the integration, collision detection is executed. If no collision occurs, the time of impact for this pair of objects is recomputed and the heap is prioritized again. If a collision does occur, the collision response module must find the correct impulses to prevent interpenetration. Also, the time of

collision for all pairs of objects involving either of these two objects needs to be recomputed and updated in the heap.

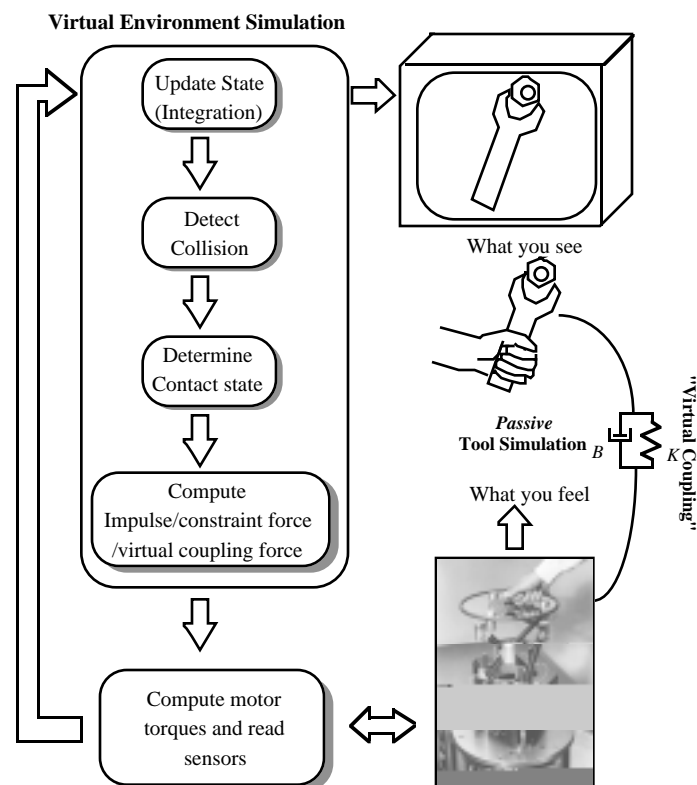


Figure 1. The computational steps in the dynamic multibody simulation, and overall structure of the system.

Obviously, the impulse-based method is suited for non-linked (unilaterally constrained) rigid body environments. For linked (bilaterally constrained) rigid body systems, a constraint-based method is probably preferred. In impulse-based simulation, multiple contacts at one time step are not allowed. Instead of dealing with multiple contacts directly, a variable time step integration method is used and impulses are treated as a sequence of collisions over time. Therefore, impulses can be handled individually. This avoids the need to solve the Linear Complementarity Problem (Baraff, 1994) for a multiple contact situation. From the complexity of programming point of view, impulse-based method is superior to constraint-based methods.

Further, the impulse-based method, which does not rely on constraints at all, simply calculates sufficient impulses to prevent penetration. It is, however, relatively easy to ensure that impulses do not add energy to a rigid body system. With this approach, the state of each rigid body can be integrated individually, making implicit integration methods a reasonable choice. Thus, the impulse-based method should be compatible

with discrete time passive integration techniques which will preserve energy and momentum (Simo and Wong, 1991) of the rigid body systems as well.

What is less clear is how well an impulse-based method might work in a true real-time setting in which the computational time required to advance the simulation is consistently below the elapsed real time. This topic will be discussed the following section.

3. SIMULATION REQUIREMENTS FOR HAPTIC DISPLAY

The term "real-time" as used by various groups carries various meanings. The computer graphics community treats "real-time" simulation as one in which the average speed of the simulation is as great as that at which the physical system evolves. Significant variations in the speed of the simulation are tolerated. We call this *pseudo* real-time simulation. The definition of *true* real-time is that the computational time is always smaller than the integration timestep at any point in the integration. This means that the simulation time and real time stay essentially in lockstep. Pseudo real-time simulation allows the use of variable stepsize integration techniques to slow down the speed (i.e. computational time is larger than integration stepsize) when the simulation encounters some complicated situation such as intensive collisions, and to speed up again when that situation is resolved. True real-time simulation is necessary for haptic display because the interface must interact energetically with the real world. A simulation which slows down periodically is liable to result in instability.

In order to achieve true real-time simulation, it is important to minimize the computational effort per time step that is required. Consequently, the integration method, collision detection algorithm, and collision response need to be carefully chosen to accomplish these requirements. Following the definition of true real-time, the highest achievable update rate is determined by the worst case scenario (the longest time to update the states of all the dynamic bodies at some time step) of the whole simulation process. For a variable order and variable stepsize integration method, as used in (Mirtich and Canny, 1994), it is impossible to predict the computational time necessary to integrate the equations forward for a determined period of time. Another difficulty is that with adaptive stepsize the information output to the haptic interface will not occur at regular intervals in time. Therefore, in order to achieve true real time simulation, methods that have a well defined minimum speed should be used. This then allows the real-time simulation to be synchronized with real-time control of the haptic interface.

Our experiments show that the haptic perceptual system needs not only a real-time simulation, but also an extremely fast update rate (500 Hz - 1kHz). Lower rates in a haptic display are not only perceptually undesirable but are unacceptable because they promote instability. Based on these considerations, some requirements of the true real-time simulation for haptic display include:

- Fixed time step simulation. It is preferred to use a fixed integration formula that incurs the same computational cost in each integration step.
- A guaranteed upper bound on the computational time required to advance the simulation time an amount corresponding to the update rate of the haptic display. Thus, if we wish to update outputs to the haptic display at 2 msec intervals, no more than 2 msec (or, more reasonably, 1.5 msec) should be required to complete the computations necessary to advance the simulation time by 2 msec.
- A guarantee that advances in simulation time correspond to advances in simulation state. Some approaches to simulating systems with unilateral constraints "get stuck", which means that the simulation time advances while the simulation state does not. The consequence is obviously an unrealistic behavior.
- Realistic models of unilateral constraint. Coulomb's model of friction and Poisson's model of restitution, for instance, are well-accepted empirical relations. Ideally, these models would be built into the simulation method. These models can, however, be very difficult to implement, especially in real-time and when multiple points of contact exist.

4. MODIFICATIONS OF THE IMPULSE-BASED SIMULATION FOR HAPTIC DISPLAY

As discussed in the previous section, some aspects of the impulse-based method must be adjusted in order to meet the requirements for haptic display. Considerations include:

- No bodies should overlap
- No energy growth
- Real-time and high update rate performance
- Constant time step integration
- Computational accuracy
- Realistic frictional behavior
- Realistic restitution behavior

The first four guidelines are essential for an application of haptic display and the last three are desirable to emulate the physical world. These guidelines will be further discussed to consider which method is suitable for implementation of haptic display.

4.1 Integration method

The numerical solution of ODEs is necessarily a part of impulse-based simulation. In order to provide a discrete-time passive simulation, an integration method which does not add energy to the system and which preserves accuracy and stability is required. (Brown and Colgate, 1997) gives this issue a detailed discussion.

4.2 Collision detection

As we discussed in the previous section, haptic display requires that the algorithms be optimized for each time step (the maximum achievable update rate is based on the worst case). For the collision detection algorithm, therefore, we wish to optimize the worst-case time of collision. Fortunately, based on the algorithm in (Lin, et al., 1994), constant collision detection time could be obtained by tracking the closest points at each state without using bounding boxes or similar techniques. The closest points at each state could then be used as the initial conditions for the next step. We can expect an almost constant time to complete the collision detection algorithm even if there is no collision.

4.3 Impact determination

Once a collision occurs, the impact between the bodies needs to be resolved. It is important to note when and where the impulses should be applied to the system for this will affect the simulation results. Also, which "impact state"¹ is used to compute impulses will lead to different results. To introduce this problem, assume that external forces are constant during one time step, and the coefficient of restitution for all contacts is one (no energy loss). We will consider several ways of setting up the impact state. Figure 2 shows the simulation of impact between two bodies.

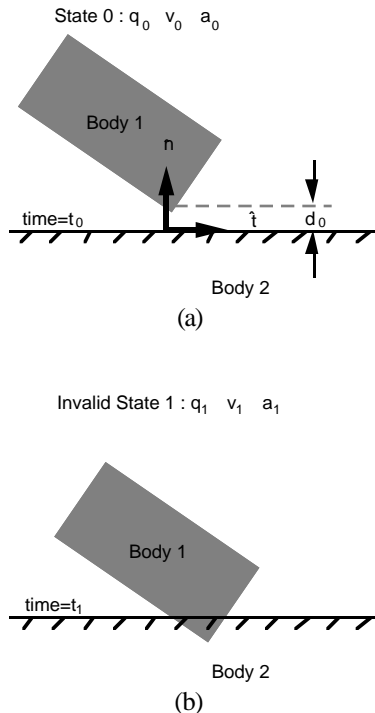


Figure 2. (a) State 0 represents the previous valid state. (b) Invalid state 1 shows a collision after one time step integration (c) Simulator adds proper impulse/force and chooses a reasonable position (this could yield a possible set of position, velocity from (b)) to correct the invalid state 1 to valid state 1. ($0 \leq d_1 \leq d_0$)

Choice 1: Exact collision state

Intuitively, the exact collision state (i.e. true impact state) is the best choice for reasons of accuracy and energy conservation. This is what the physical world does. Yet, the exact collision state is usually found by forward evolution and backtracking, using a binary search or other iterative methods. This is time-consuming. The algorithm could spend a large amount of time searching for the exact answer, which would achieve accuracy, but limit real-time performance. In a true real-time control system, this method is unacceptable.

Choice 2: Previous valid state ($q_{\text{impact}}, v_{\text{impact}}$)

$$a_{\text{impact}} = (q_0, v_0, a_0)^2$$

Now, we may think of retrieving the previous state as the impact state. The advantage of using the previous state (state 0) as the impact state is that adding impulses will not change the energy of the system (position and velocity of the body are always consistent). However, suppose that v_0 is non-negative in state 0; i.e., the bodies are receding from one another. A collision may still occur if a large enough acceleration exists. Yet, because v_0 is non-negative, no impulses will be added to the "impact state", and the new state 1 remains the same as state 0. Next, the system integrates another fixed time step, and penetration occurs again because of acceleration. The impact state will now be chosen as state 1 (i.e. state 0), and contact velocity still remains non-negative. No impulses are applied. The new state 2 will remain the same as state 1 which

¹ The "impact state" is a computational state and does not exist in any time instance. Hence, it does not need to obey any physical meaning.

² ($q_{\text{impact}}, v_{\text{impact}}, a_{\text{impact}}$) are position, velocity and acceleration of the body which are used in the "impact state". Sub-indices of 0 and 1 indicate the previous legitimate state and the current illegitimate state. An upper-index of "+" indicates a legitimate inter-state between 0 and 1.

is also the same as state 0. Consequently, State 0 = State 1 = State 2 = = State n. The system gets stuck.

Choice 3: $(q_{\text{impact}} \ v_{\text{impact}} \ a_{\text{impact}}) = (q_0 \ v_1 \ a_1)$

In order to prevent the previous situation (initial condition of v_0 could be positive), we may consider v_1 instead of v_0 because v_1 is negative in most cases. If we use the previous q_0 and current v_1 as the "impact state", the impact may add energy (if $\text{norm}(v_1) > \text{norm}(v_0)$) into the system because the position is not consistent with its velocity. This choice may relieve the previous "getting stuck" problem as long as the energy gain is not too severe or we have some way to bound the energy gain.

Choice 4: Limited searching for a better contact state

If we try to compute a more precise collision time dt ($0 < dt < h$, h is fixed time step) with a given limited number of iterative steps, we could integrate state 0 by dt , instead of h . To implement the limited searching for better choice, we could use a binary search³ or lower bound on possible collision time (the scheduling scheme). The new position (q_0^+) of the body 1 does not penetrate body 2. We could use q_0^+ and v_0^+ as the impact state. Intuitively, this is a good choice because it is more accurate than the previous two impact states, but it is limited in the amount of additional computation to be performed. Yet, it still cannot avoid the disadvantage of the second choice (using the previous state). When d_0 in state 0 is close to ϵ , dt is very close to zero. Limited searching cannot integrate forward anymore. This will cause the impact state of this choice (q_0^+ and v_0^+) to remain the same as state 0 (q_0 and v_0). Consequently, it becomes the second choice. It gets stuck occasionally (if v_0 is once positive).

Choice 5: Combination of inter-state of position and velocity $(q_{\text{impact}} \ v_{\text{impact}} \ a_{\text{impact}}) = (q_0^+ \ v_1 \ a_1)$ or $(q_0 \ v_0^+ \ a_1)$

Now, one possible way is mixing the choice 3 and choice 4 to define a new impact state. The combination of inter-state position and velocity has better accuracy and avoids getting stuck in most situations. Yet, energy gain is still a concern.

Choice 6: Hybrid of force and impulse

to manage for a constant force over time. A second order trapezoidal integration method can be used to achieve a constant system energy. This property does not exist for a variable external force within a time step. The energy gain adds an unpredictable factor to the stability problem of the simulation if we do not have a way to bound it.

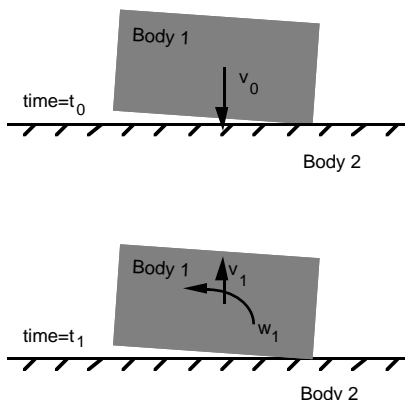


Figure 3. Two bodies first collide at time t_0 ; the impulse is applied at the correct contact point (right vertex of Body 1). After application of the impulse, if Body 1 penetrates Body 2 within the next time step, then a problem could arise in trying to find the correct collision location within one real time step.

display is limited by the size of the heap, given by $N \times M + \frac{M \times (M-1)}{2}$.

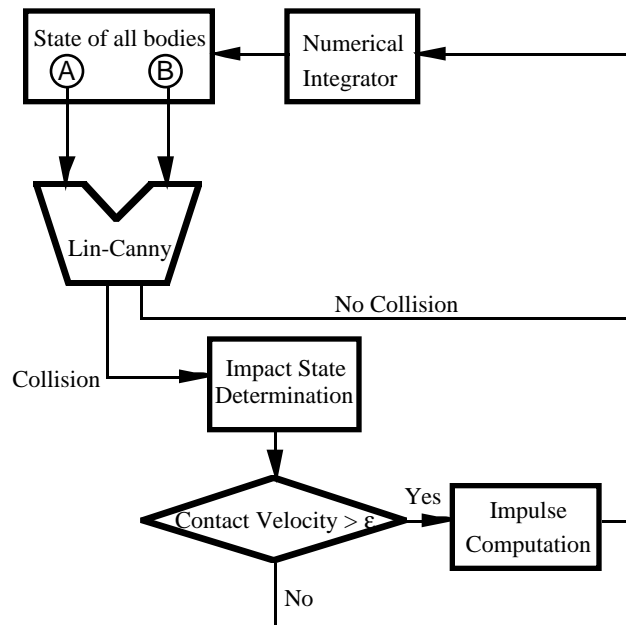


Figure 4. Simulation algorithm for haptic display.

5. EXPERIMENT SETUP AND RESULTS

This implementation uses a fixed-time integration technique, a spatial coherence based collision detection algorithm, and the impulse-based collision response method. Figure 4 shows the overall algorithm for haptic display. Currently, the impact state determination is based on the choice number 5 in section 4.3. Multiple collisions within one time step are processed as serial collisions in one simulation time step. The algorithm solves simultaneous collisions as a series of individual collisions. It updates the state of the body immediately after each collision, thus affecting the impact state of the next collision.

A test environment containing user-defined virtual objects has been built. An environment containing N fixed bodies and M dynamic (free) bodies has been tested. The user can manipulate one of the dynamic bodies through the handle and simultaneously experience the virtual force from the haptic display. Figure 5 shows a virtual environment which contains one dynamic body and six static bodies. The dynamic body is constrained to the handle (a short line segment shows the position of the handle), using a virtual coupling. The program creates a heap of body pairs in which each pair contains at least one dynamic body. Each step of the simulation loops through this heap to track the closest points, detect collisions and resolve impulses for each pair. The update rate of the haptic

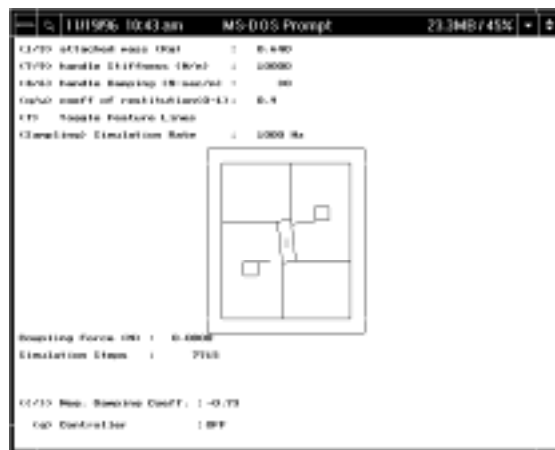


Figure 5. Simulation of six fixed bodies and one dynamic body in the virtual environment.

Two haptic displays, one single axis, and another two-axis, have been used for testing this impulse-based method. The controller (implemented as an interrupt service routine), virtual environment simulator, and graphic display run on a single processor. Virtual environments of heap size 15 (e.g. $N=7, M=2$ or $N=4, M=3$) have been realized at an update rate of 1KHz

on a Pentium 90Mhz personal computer in the DOS environment.

Despite the high update rate which the impulse-based method can achieve, we still encountered two unexpected problems. One is a “sticky” feeling, the other is the absence of frictional force.

We have previously described the phenomenon by which a body “gets stuck”: the algorithm re-assigns the previous position each time a collision is detected to avoid interpenetration. In the cases when a user is pushing to produce repeated collisions, a continual re-assignment of the previous position in fact produces the appropriate physical behavior. That is, so long as the applied force lies inside the friction cone, the dynamic body will hold position, as is appropriate. When the user begins to pull, however, a “sticky force” will arise in some situations. For instance, in the case of two rectangles with edge to edge contact, if the contact rotational velocity between them is high, a sticky force could be produced. Unless the applied pulling force is large enough to avoid another immediate collision in the present time step, the algorithm will retrieve the previous position and assign it to the current state. Given that the position remains unchanged, the virtual coupling will extend to create a tensile force. (See Fig. 6.)

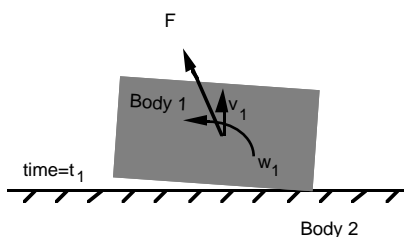


Figure 6. Unless external force can overcome the repeated collisions caused by the rotational velocity in one time step, the position of the body will remain unchanged. A sticky force is produced.

The problem of “getting stuck” results mainly from the incorrect impact state at the “contact” situation (i.e., large rotational velocity causes the difficulty in determining the correct collision point). For the resting contact of a pair of bodies in the physical world, no impulse exists, instead a contact force exists between them. If one of the bodies is static, the velocity of the other should also drop to zero. If both of bodies are dynamic, they should maintain the same velocity at the contact point.

Secondly, although Coulomb friction is built into the impulse-based method, users cannot feel it. One reason is that friction occurs during body contact. The stipulation that impulses be applied only one contact point at a time creates problems. For instance, if a user applies a large external force to the body in the case of a pair of nearly parallel edges, the impulse applied to one end of the edge can produce a rotational velocity and result in an immediate collision of its another end

(in the next time step). The algorithm will retrieve the previous position and assign it to the current state. The position of the body is held.

One way to deal with this problem is to identify the “resting contact” situation and to treat it with special techniques. A counter can be used to detect if there are continual collisions over time. When a “resting contact” situation is detected, the algorithm resets the velocity to zero before it adds the impulses to the body. This helps to avoid repeated collisions and allows the body to slide. However, when a user actually manipulates one body to slide on another, he/she will only feel a “stop and go” sliding behavior instead of frictional sliding. This behavior arises because of the interplay between the externally applied forces and the collision impulses.

6. CONCLUSIONS

In order to feel realistic frictional behavior of dynamic systems using haptic display, various “contact” situations must be identified. In the impulse-based method, all kinds of contact are unified into a single collision model. This does help to simplify the complexity of rigid body dynamics, yet the resting contact phenomenon is only approximated. Using micro-collisions to approximate the resting contact looks acceptable on the screen, but does not feel right through the haptic display.

Another issue that needs to be addressed is that when using a fixed time step method, multiple collisions can occur within one time step. It is unclear in this situation how to incorporate friction into the impact analysis. If we still treat the event as a sequence of collisions, separated by a simulation time step, a poor result can be expected.

However, the appeal of impulse-based simulation is that it yields solutions to many challenges that arise in rigid body simulation for haptic display. In this method, constraints are not implemented at all; impulses are computed sufficient to prevent interpenetration when collisions occur. This makes it easy to ensure that the impulses do not add energy to a rigid body system. Also with this approach, the state of each rigid body can be integrated individually, making implicit integration methods applicable. Best of all, this method is computationally efficient, conceptually simple, and easy to implement from a programming point of view. In addition, the models of Coulomb friction and Poisson's restitution may be built into the impulse-based method.

7. REFERENCES

- Baraff, D., 1994, “Fast Contact Force Computation for Nonpenetrating Rigid Bodies,” *Computer Graphics (Proc. SIGGRAPH)*, Orlando, FL, Vol. 28, pp. 23-34.
- Brown, J. M., 1996, “A Passive Implementation of

Force-Reflecting Interfaces," *IEEE Virtual Reality Annual International Symposium*, Seattle, WA, pp. 202-207.

Colgate, J. E., Stanley, M. C. and Brown, J. M., 1995, "Issues in the Haptic Display of Tool Use," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, Vol. 3, pp. 140-145.

Gillespie, R. B., 1996a, "Haptic Display of Systems with Changing Kinematic Constraints: The Virtual Piano Action," Ph. D., Stanford University.

Gillespie, R. B., 1996b, "Stable user-specific haptic rendering of the virtual wall," *International Mechanical Engineering Congress and Exhibition*, K. Danai, ed., ASME, Atlanta, GA, Vol. DSC-Vol. 58, pp. 397-406.

Keller, J. B., 1986, "Impact with Friction," *ASME Journal of Applied Mechanics*, Vol. 53, pp. 1-4.

Klatzky, R., Lederman, S. and Reed, C., 1989, "Haptic Integration of Object Properties: Texture, Hardness, and Planar Contour," *J. of Exp. Psychology: Human Perception and Performance*, Vol. 15, No. 1, pp. 45-57.

Lin, M. C., 1993, "Efficient Collision Detection for Animation and Robotics," PhD, University of California, Berkeley.

Lin, M. C., Manocha, D. and Canny, J., 1994, "Fast Contact Determination in Dynamic Environment," *IEEE*, , pp. pp. 602-608.

Massie, T. H. and Salisbury, J. K., 1994, "The PHANToM Haptic Interface: A Device for Probing Virtual Objects," *International Mechanical Engineering Exposition and Congress*, C. J. Radcliffe, ed., ASME, Chicago, Vol. DSC 55-1, pp. 295-302.

Mirtich, B. and Canny, J., 1994, "Impulse-based Dynamic Simulation," *Workshop on Algorithmic Foundations of Robotics*, K. Goldberg, P. Halperin, J. C. Latombe and R. Wilson, ed., A.K. Peters, Boston, MA.

Routh, E. J., 1905, *Dynamics of a System of Rigid Bodies*, Dover Publications, New York.

Salcudean, S. E. and Vlaar, T. D., 1994, "On the Emulation of Stiff Walls and Static Friction with a Magnetically Levitated Input/Output Device," *International Mechanical Engineering Exposition and Congress*, C. J. Radcliffe, ed., ASME, Chicago, IL, Vol. DSC 55-1, pp. 303-310.

Simo, J. C. and Wong, K. K., 1991, "Unconditionally Stable Algorithms for Rigid Body Dynamics that Exactly Preserve Energy and Momentum," *International Journal for Numerical Methods in Engineering*, Vol. 31, pp. 19-52.

Wang, Y. and Mason, M. T., 1992, "Two-Dimensional Rigid-Body Collisions with Friction," , Vol. 59, pp. 635-642.