Real Time Simulation of Stiff Dynamic Systems via Distributed Memory Parallel Processors

Michael C. Stanley
James E. Colgate
Department of Mechanical Engineering
Northwestern University
Evanston, IL 60208

Introduction

The context of this paper is the design of a kinesthetic interface to virtual environments. Several kinesthetic interfaces have previously been developed. Minsky[15]has developed a device for creating "virtual sandpaper". A powered joystick interfaced to a real-time computer simulation allows the user to "feel" textured surfaces. The simulation implements textures as a mapping from joystick position to actuator torque commands - essentially a two-dimensional force field. Researchers at UNC[17] have also developed a six degree-of-freedom device for simulating inter-molecular forces. Adelstein[2] has experimented with a two degree of freedom programmable joystick for studying human arm tremor.

Whereas these studies have considered an impressive array of virtual environments, most of these environments can be classified as either generalized force fields(defining a fixed force-displacement relationship) or as linear, time-invariant dynamic systems. Many potentially interesting virtual environments, however, are characterized by significant inertial dynamics, as well as nonlinear geometric constraints. A simple example would be placing a wrench onto a nut and then tightening the nut.

In this study, we are considering a broader class of dynamic systems consisting of constrained planar rigid bodies. This class of environments poses two significant challenges which we feel are fundamental to the development of advanced kinesthetic interfaces. First, the environment structure changes dynamically due to collisions between bodies. Thus, the equations of motion must be formulated and solved in real-time. Second, the equations are necessarily stiff due to rigid constraints in the system. Thus, real-time simulation will require extremely high update rates to be stable and present the operator with a realistic "feel" .

Dynamics Simulation

Various methods for formulating the dynamic equations of motion for mechanical systems have been presented in literature[4, 5, 6, 8, 13, 14, 19, 21, 22, 23]. Most are based on the creation of a library of joints between bodies in the system. A simulation environment is created by specifying objects and object properties and then connecting the objects with joints from the library. The objects and their connections are used to formulate a system of equations which is then solved for the accelerations of the bodies. These accelerations are numerically integrated to obtain object velocity and position at the next time step.

We plan to use a simplistic formulation of the system dynamics where each connection consists of a generalized spring and damper. Although more rigorous formulations are possible[4, 5, 6, 8, 13, 14, 19, 21, 22, 23], using springs and dampers to simulate connections decouples the system

0-7803-1363-1/93 $3.00 © 1993 IEEE

dynamic equations of motion. The acceleration of each object can then be computed independently. Thus, using springs and dampers leads to a very low level of parallelism in solving the equations of motion.

There are several disadvantages to this approach. In order to create a rigid connection between objects, very stiff springs must be used. The resulting equations must use a small integration time step in order to maintain numerical stability. Since the equations of motion must be solved during each time step, the small step size imposed by the integration routine requires high computational performance. However, since the equations of motion are decoupled, we can easily distribute tasks over a parallel architecture and thus achieve short execution times.

The choice of stiffness and damping parameters for a given connection is also difficult. A possible solution is to continuously adapt these parameters to maintain numerical stability of the integration routine. This may be done by monitoring the integration error at each time step and increasing or decreasing the joint stiffness based on the change in the error.

Collision Detection and Response

Simulation of interacting dynamic systems necessarily involves the detection of collisions between objects and response to these collisions. Algorithms employed in detecting collisions are specific to the method used to represent object boundaries. Object boundaries are commonly represented as polygons in two dimensions or as a set of polyhedra in three dimensions[10, 12, 18]. Other representations are possible but require a large amount of computation[7].

Algorithms for collision detection between arbitrary polygons in two dimensions are highly developed [3, 9, 11, 20]. These algorithms reduce the number of edges that must be checked for intersection by first determining those edges visible from a test point. The remaining visible edges are then ordered so that the search algorithm achieves better than linear performance in the number of vertices.

Collision response techniques have previously been used to generate realistic animation of colliding objects[7, 16]. These methods use a hybrid method for determining system response to collisions. Object velocities after a collision are determined analytically from the velocities before the collision and a model of collision dynamics. Springs are also used for mild collisions, such as an object resting on another object, so that the analytical solution does not have to be continuously computed.

In our scheme, the collision detection algorithm executes in parallel with the dynamics simulation. The collision detection algorithm determines the minimum distance between two objects in the environment. If this distance is less than a specified limit, a unilateral spring and damper is added between the two closest points on the bodies. The spring and damper have no effect until the distance between the two points in zero. However, by placing the spring and damper in the simulation before collision actually occurs, collision response will occur at the simulation update rate rather than at the slower collision detection rate.

Implementation

Because of the computation performance required for real-time simulation, we are using a distributed memory parallel processing architecture based on the use of the Inmos T805 Transputer[1]. The T805 is a 32-bit microprocessor specifically designed for parallel processing applications. The processor runs at a 30 MHz clock speed and provides a sustained instruction rate of 15 MIPS (30 MIPS peak). The 64-bit floating point coprocessor provides a sustained rate of 3.3 MFLOPS (4.3 MFLOPS peak).

The T805 processor was specifically designed to support the execution of concurrent or parallel processes. Interprocessor communication is achieved using 4 high speed bi-directional serial links. Each link pair has an on chip DMA interface and communication support for the links is built into the microprocessor instruction set. The links operate at a rate of 20 Mbits/sec and can achieve a throughput of 2.4Mbytes/sec. Connections are established by directly connecting serial links between transputers. Alternatively, Inmos supplies a 32x32 solid state switch, called a configuration link switch, that connects transputer links under software control.

Simulation of the environment is achieved by assigning a transputer to each object in the environment. Since we are using generalized springs and dampers to simulate connections between objects, all external forces acting on the object depend only on the states of the object and the objects connected to it. Thus, by configuring the transputer network to mirror the topology of the actual physical system, the states of connected objects can be passed over the transputer's serial links. Then, each transputer can calculate the external forces acting on the object it represents. Once the external forces are determined, the object accelerations may be calculated and integrated to determine new object position and velocity.

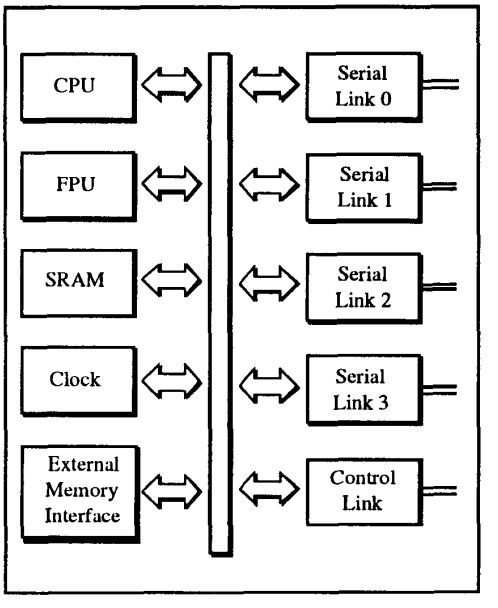

Figure 1: Transputer Chip

The number of connections to other objects is limited only by the number of serial links available to the transputer. The T805 has four serial links, one of which is used to transfer object information to the collision detection algorithm. Thus, each object may have up to three other objects connected to it. Mechanisms with kinematic loops may be treated identically to serial mechanisms. Any loops are automatically incorporated into the simulation when the transputer links are configured.

Data organization parallels the structure described in [23]. Object information is stored in the data structure shown in Figure 2. The object data structure contains information about the object (mass, moment of inertia, ...), the object states, and arrays of pointers to actions, forces, and torques that act on the object.

Each action data structure contains a vector from the center of mass to the point of application, parameters necessary for its evaluation, and pointers to functions that evaluate its value. Force and torque data structures are similar to the action structure. The object data structure provides a simple means for simulating the object. For each time step, the transputer indexes through the arrays of actions, forces, and torques, calculating and summing the forces and torques acting on the object. Once all forces are calculated, the object acceleration can be determined.

The object data structure also allows the addition of new actions due to collisions with other objects. This is accomplished by creating a new action data structure and adding another pointer to it to the object's action array.

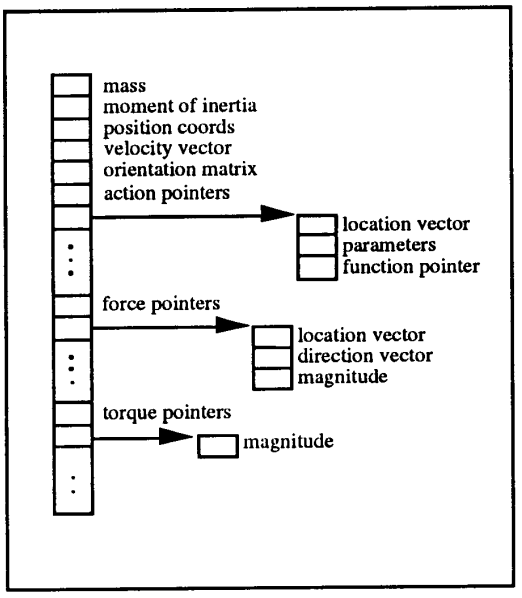

Figure 2: Object Data Structure

Preliminary Results

Preliminary results indicate that the architecture will be able to provide update rates in excess of 1kHz to the manipulandum for complex environments. Object simulation time depends on the number of actions in the data structure. The times for various number of actions is shown in Figure 4.

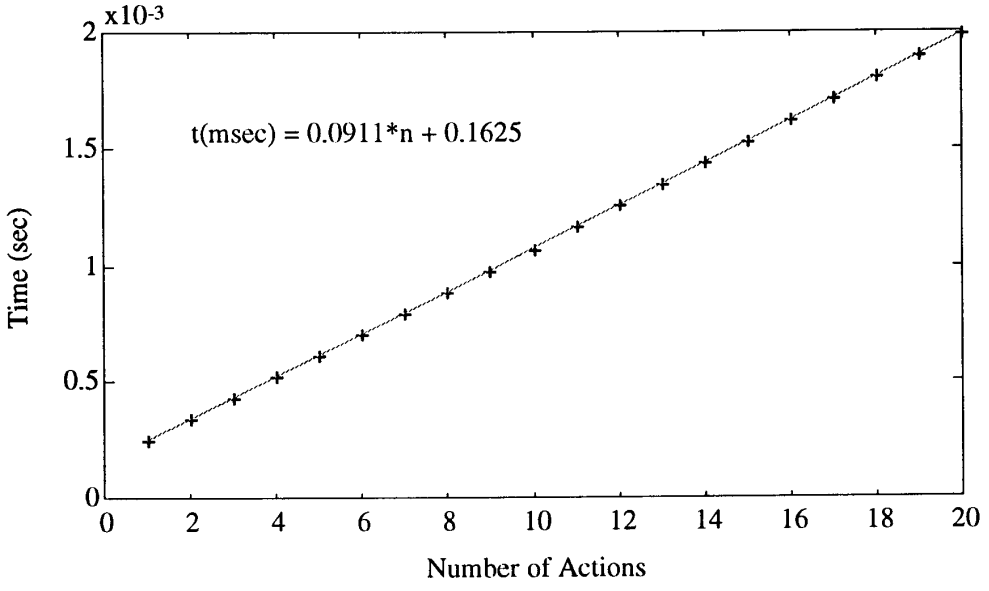

Figure 4: Object Simulation Times

Collision detection times for a simple algorithm that compares every edge of one object against every edge of another object are shown in Figure 5. The high execution time for this collision detection algorithm illustrates the need to separate the collision detection algorithm for simulation.

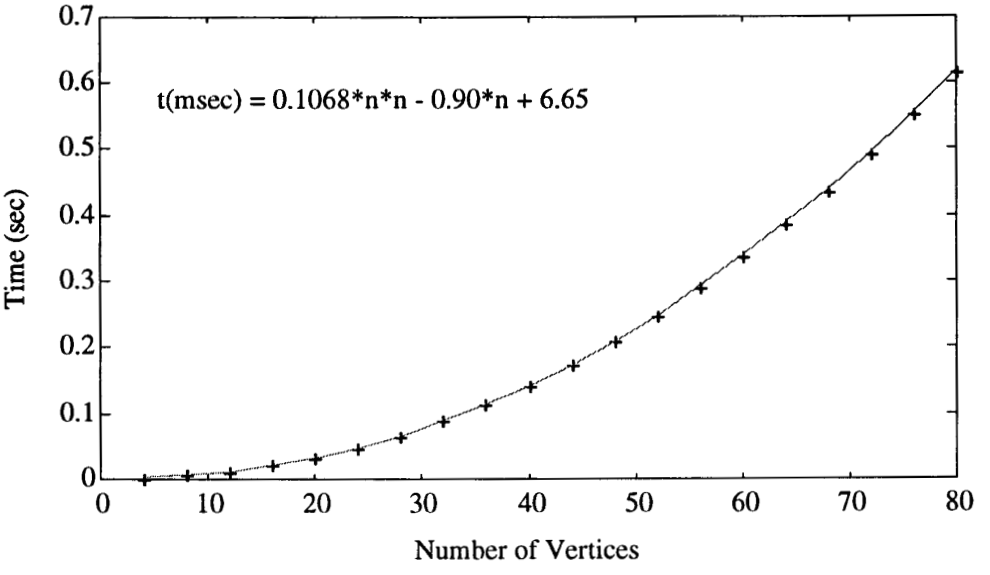

Figure 5: Collision Detection Times

Conclusions

A system for the real-time simulation of dynamic mechanical systems has been described. This system will be used to investigate dynamic interaction between a human operator and a usefully complex virtual environment. Computational performance and psychophysical results will be reported in future publications.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation, Grant #MSS-9022513.

References

1. *The Transputer Databook.* Inmos Ltd. (1989)

2. Adelstein, B. D. *A Virtual Environment System for the Study of Human Arm Tremort.* PhD, Massachusetts Institute of Technology (1989)

3. Avis, D. and G. T. Toussiant. *An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge.* IEEE Transactions on Computers **Vol. C-30**(No. 12)(1981)

4. Bae, D. and E. J. Haug. *A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems.* Mechanical Structures and Machines **Vol. 15**(No. 3):359-382 (1987)

5. Bae, D. and E. J. Haug. *A Recursive Formulation for Constrained Mechanical System Dynamics: Part II. Closed Loop Systems.* Mechanical Structures and Machines **Vol. 15**(No. 4):481-506 (1988)

6. Bae, D., J. Kuhl and E. J. Haug. *A Recursive Formulation for Constrained Mechanical System Dynamics: Part III. Parallel Processor Implementation.* Mechanical Structures and Machines **Vol. 16**(No. 2):249-269 (1988)

7. Baraff, D. *Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation.* Computer Graphics **Vol. 24**(No. 4)(1990)

8. Barzel, R. and A. Barr. *A Modeling System Based on Dynamic Constraints.* Computer Graphics **Vol. 22**(No. 4)(1988)

9. Chin, F. and C. A. Wang. *Optimal Algorithms for the Intersection and Minimum Distance Problems Between Planar Polygons.* IEEE Transactions on Computers **Vol. C-32**(No. 12)(1983)

10. Cyrus, M. and J. Beck. *Generalized Two- and Three- Dimensional Clipping.* Computers and Graphics **Vol. 3**:pp. 23-28 (1978)

11. Dobkin, D. P. and D. G. Kirkpatrick. *A Linear Algorithm for Determining the Seperation of Convex Polyhedra.* J. Algorithms **Vol. 6**:pp 381-392 (1985)

12. Griffiths, J. G. *Bibliography of Hidden-line and Hidden-surface Algorithms.* Computer Aided Design **Vol. 10**(No. 3):pp. 203-206 (1978)

13. Haug, E. J. *Elements and Methods of Computational Dynamics.* Computer Aided Analysis and Optimization of Mechanical System Dynamics. Haug ed. Springer-Verlag. (1984)

14. Luh, J., M. Walker and R. Paul. *On-Line Computational Scheme for Mechanical Manipulators.* Journal of Dynamic Systems, Measurement, and Control **Vol. 102**(1980)

15. Minsky, M. and e. al. *Feeling and Seeing: Issues in Force Display.* Computer Graphics **Vol. 24**(No. 2)(1990)

16. Moore, M. and J. Wilhelms. *Collision Detection and Response for Computer Animation.* Computer Graphics **Vol. 22**(No. 4)(1988)

17. O. Ming, D. V. B., F.P. Brooks, Jr. *Force Display Performs Better than visual display in a simple 6-D docking task.* Proceedings of the IEEE International Conference on Robotics and Automation :1462-66 (1989)

18. Rogers, D. *Procedural Elements for Computer Graphics.* McGraw-Hill Book Company. New York (1985)

19. Schroder, P. and D. Zeltzer. *The Virtual Erector Set: Dynamic Simulation with Linear Recursive Constraint Propagation.* Computer Graphics **Vol. 24**(No. 2)(1990)

20. Schwartz, J. T. *Finding the Minimum Distance Between Two Convex Polygons.* Information Processing Letters **Vol. 13**(No. 4-5)(1981)

21. Walker, M. and D. Orin. *Efficient Dynamic Computer Simulation of Robotic Mechanisms.* Journal of Dynamic Systems, Measurement, and Control **Vol. 104**(1982)

22. Wang, L. and B. Ravani. *Recursive Computations of Kinematic and Dynamic Equations for Mechanical Manipulators.* IEEE Journal of Robotics and Automation **Vol. RA-1**(No. 3)(1985)

23. Witkin, A., M. Gleicher and W. Welch. *Interactive Dynamics.* Computer Graphics **Vol. 24**(No. 2)(1990)