Dynamic Simulation of Interactive Robotic Environment

Paul U. Lee, Diego C. Ruspini, and Oussama Khatib
Robotics Laboratory
Computer Science Department
Stanford University, Stanford, CA 94305

Abstract

A dynamic simulation package, which can accurately model the interactions between robots and their environment, has been developed. This package creates a virtual environment where various controllers and workcells may be tested. The simulator is divided in two parts: local objects that compute their own dynamic equations of motion, and a global coordinator that resolves interactive forces between objects. This simulator builds upon previous work on dynamic simulation of simple rigid bodies and extends it to correctly model and efficiently compute the dynamics of multi-link robots.

1 Introduction

In recent years there have been many efforts to accurately simulate physical environments in both robotics and computer graphics. A physically accurate simulation gives important insights into the real-world behavior of a robotic environment.

The usefulness of robotic simulation is illustrated by a problem of the construction industry: the transfer of beams from one point to another (Figure 1). It is desired to accomplish this task in optimal time while limiting the swing of the load. Use of unstable controllers on a real crane, could result in permanent damage to the crane and/or its environment. Using simulation, we can test, change, and improve the controller design in an iterative fashion without fear of damaging the physical system. Computer graphics permit the visualization of results in a quick, intuitive manner, and is, therefore, a valuable tool to speed the design and analysis process.

We have developed a dynamic simulation package to simulate the interactions between robots and other objects in their environment. This simulation is based on dynamic equations of motion with accurate analytical solutions of the constraint equations [1]. Although

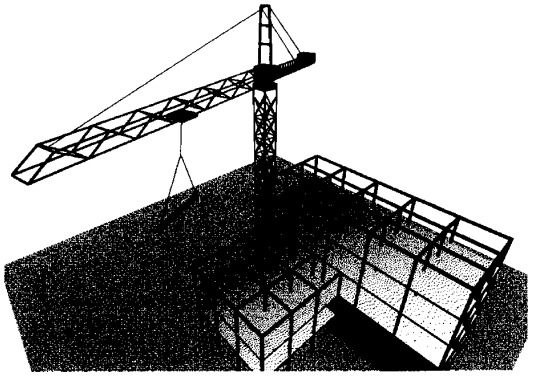

Figure 1: Simulation of a Crane

the dynamic equations of motion can be easily derived for each object, the interactions (i.e., collision and/or contact) between them, however, have proved to be more difficult to handle. In recent work, two general approaches have been proposed to address this problem.

The first approach resolves constraint forces, which keep objects from inter-penetrating, by using "spring-like" repulsion at the collision and contact points [11]. The advantage of this approach is its ease of implementation and its extendibility to nonrigid bodies. This method, however, does not guarantee a physically accurate solution for the dynamic behavior of the objects.

The second approach finds analytically correct solutions to the dynamic interactions between rigid bodies by explicitly solving the constraint equations. Baraff [1] uses this technique to find constraint forces that prevent inter-penetrations between rigid objects. Although these analytical solutions for rigid body contact yield accurate physical behavior, they are more

1050-4729/94 $03.00

difficult to implement and are not easily extended to nonrigid bodies.

A physically accurate solution is essential in a simulator that is designed to duplicate the motions of real world objects. Since the first approach does not guarantee a physically accurate solution, we have implemented the second method, which yields accurate analytical results [1]. Previous algorithms have been only developed for simple rigid-body objects. We have extended these algorithms to account for multiple link robots and other articulated objects.

2 Dynamic Simulator

The main objective in the design of our dynamic simulator was the development of an interactive package that could be run in tandem with a physical robotic environment. In an ideal system, robot controllers should be capable of being connected interchangeably to either real robots or to our dynamic simulator. This capability permits to preview the behavior of robots and their interactions with the environment. In this way, various tasks and controllers may be tested in a "virtual reality" environment before they are applied to physical systems.

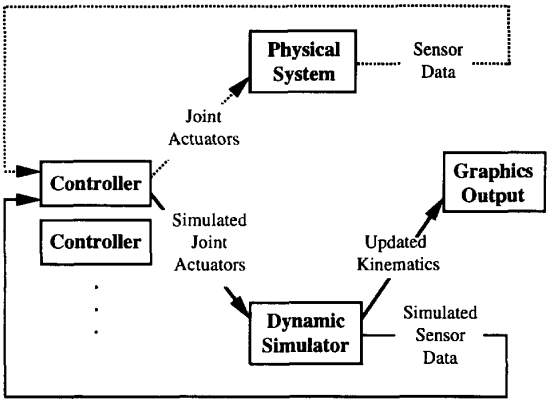

Figure 2: Overall Structure of the System

Figure 2 illustrates the overall structure of the system. The dynamic simulator uses as inputs joint actuator forces from a set of controllers. Employing the dynamic equations for each object in the environment, the simulator calculates and returns simulated sensor data to the controllers. Updated kinematic information, such as joint angles, is sent to a graphical output process to update the display of the simulated environment.

3 Global/Local Architecture

The simulator has been separated into two parts: a global coordinator and a set of local objects with their corresponding equations of motion. This functional separation is based on the following considerations:

- Dynamic equations of motion can be determined and independently calculated for each object as long as there is no contact or collision between objects.
- When contact or collision occurs, constraint forces or impulses impose correct dynamic behavior and prohibit inter-penetrations of objects. Determination of constraint forces and impulses requires knowledge of all objects that are interacting at a given time. This determination requires a global treatment.

A local object consists of its equations of motion, an integrator to update positions and velocities based on its dynamics, and a set of parameters required for the above calculations.

The global coordinator determines if interaction between objects has occurred and solves the constraint impulse/force equations at the collision/contact points. The collision is resolved by first calculating appropriate impulse forces that correctly model the collisions. If objects remain in contact after the collision is resolved, contact forces are calculated [1].

4 Object Structure

The configuration of an n-DOF (degree-of-freedom) object is described by a vector q of n independent generalized coordinates. Joint angles are generally used as generalized coordinates for robots, but other sets of parameters that satisfy the above constraints may also be utilized. The equations of motion in terms of generalized coordinates are given by

$$\Gamma = A(q)\ddot{q} + b(q,\dot{q}) + g(q)\,, \qquad (1)$$

where $A(q)$ designates the kinetic energy matrix; $b(q,\dot{q})$, the centrifugal and Coriolis force vector; $g(q)$, the gravity force vector; and Γ, the generalized torque vector of the object [4].

The generalized torque Γ can be divided into torques generated by robot actuators Γ_{joint} and torques from external constraint forces Γ_{ext}.

$$\Gamma = \Gamma_{joint} + \Gamma_{ext}\,. \qquad (2)$$

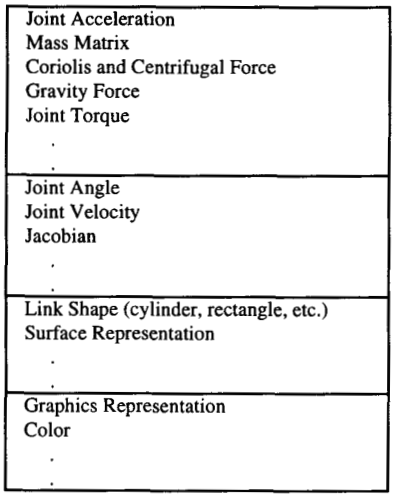

Figure 3: Robot Structure

Equation 1 is solved for $\ddot{q}$ to dynamically simulate the motion of an object

$$\ddot{q} = A^{-1}(q)[\Gamma - b(q, \dot{q}) - g(q)]\,. \qquad (3)$$

Given an initial position and velocity, updated positions and velocities can be obtained by numerical integration of Equation 3.

This approach finds an efficient solution for robots and other multi-linked objects by taking advantage of well-defined holonomic constraints at their joints. By taking these constraints into account, this method finds solutions only for the parameters that affect the dynamics of the system. In contrast, other works [1, 3] treat multi-linked objects as series of simple objects. These techniques require computation of all forces, including the internal joint forces, which have no effect on the motion of the system. These unnecessary calculations leads to slower simulations.

Each object stores the parameters needed to represent and solve the equations of motion in an object model. The description of the object model is divided into four categories: dynamic, kinematic, geometric, and graphical. Figure 3 shows an example of an object data structure for the robot manipulator.

The dynamic model contains information related to the forces required to cause motion including such data as inertial forces, Coriolis forces, and the kinetic energy matrix.

The kinematics model incorporates data such as joint angles, joint velocities, the Jacobian, and transformation matrices. This information may be used to calculate parameters such as the position and velocity of the end-effector.

The geometric model stores the geometric description of each object. This information is used to find if contact between objects has occurred and, if so, the location of that contact. This information can then be used to resolve the constraint force equations.

The graphical model determines the visual image of the objects in a graphical rendering of the scene representing such qualities as shape, color, lighting, and shading.

5 Global Coordinator

5.1 Collision/Contact Detection

Collision and contact points are determined by first calculating object locations using the equations of motion and assuming that no collisions or contacts occur. Inter-penetrations between objects indicate the presence of collisions and/or contacts. Forces arising from these collisions and contacts can then be computed solving the constraint equations as described below in sections 5.2 and 5.3.

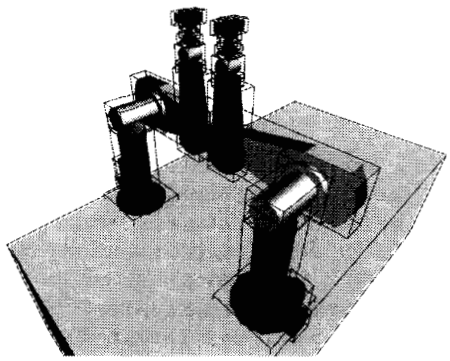

Figure 4: 6 DOF manipulator with Bounding Boxes

The algorithm that determines the existence of inter-penetrations is divided into two parts. First, each object is enclosed in bounding boxes that may be checked quickly for intersections. If the bounding boxes do not overlap, further checking of collisions or contacts for those objects is unnecessary. Figure 4 shows two 6 DOF arm robots with bounding boxes around their links.

A naive collision-checking implementation that checks all pairs of objects has a complexity of $O(n^2)$ but a sort and sweep algorithm requires only $O(n \log n)$ steps. [2]. In subsequent simulation cycles, an insertion sort may be used for additional efficiency.

If the bounding boxes of two objects overlap, the minimum distance between them is computed in order to determine inter-penetrations. For this purpose, Orlowski [12] relies on an algorithm with a complexity of $O(M \log M)$ where M is total number of vertices in the two objects, $M = M_1 + M_2$. Baraff employs an algorithm that has an initial complexity of $O(M_1 M_2)$ but that utilizes temporal witnesses to reduce that complexity to $O(M)$ [2]. We have implemented an algorithm developed by Gilbert [5], which is considered to be one of the most efficient to compute inter-object distances. It has a complexity of $O(M)$ with a small coefficient of linear growth.

5.2 Impulse Force Resolution

A collision between two objects a, and b occurs when there exist points p_a and p_b such that at some collision time t_c, $p_a(t_c) = p_b(t_c)$ and such that their relative velocity just prior to collision is negative in the direction n_i normal to the contact region. Given m collision points ($1 \leq i \leq m$), v_i^- and v_i^+ are defined as the component of the relative velocity parallel to the common normal at point i where "$-$" and "$+$" differentiate between the values before and after collision, respectively. The normal n_i is defined to have a positive direction from b to a. The relative velocity v_i^-, which has a negative value, is then formally defined as

$$v_i^- = n_i^T \dot{p}_{a_i}^- - n_i^T \dot{p}_{b_i}^- . \tag{4}$$

where $\dot{p}_{a_i}^-$ and $\dot{p}_{b_i}^-$ are the velocity vectors prior to the collision at i^{th} collision points on the objects a and b.

Because of space limitations we restrict our discussion to the collision of one generalized object against a fixed obstacle. This formulation can easily be extended to the generalized case. Using this simplifying assumption, then $\dot{p}_{b_i}(t) = 0$, and Equation 4 becomes

$$v_i^- = n_i^T \dot{p}_{a_i}^- . \tag{5}$$

When a collision occurs in a physical system, a local deformation is produced for a brief period of time that generates forces sufficient to prevent penetration. Modeling this deformation for hard objects produces stiff equations that are slow to integrate. We have instead chosen to model objects as rigid bodies and to calculate the impulse forces explicitly. An impulse causes an instantaneous change in the velocity of an object. The equations of motion are integrated, therefore, to the time of collision, new values of the generalized velocities $\dot{q}$ are then calculated, and the integration is continued using the new state. By breaking up the intergration at the velocity discontinuity the integration of a stiff set of equations can be avoided.

Baraff [1] uses three constraint equations for each contact point i to calculate the impulse forces on a body.

$$v_i^+ \geq -\epsilon_i v_i^- , \tag{6}$$

$$j_i \geq 0 , \tag{7}$$

$$j_i(v_i^+ + \epsilon_i v_i^-) = 0 , \tag{8}$$

where ϵ_i is the coefficient of restitution, and j_i is the magnitude of the impulse force in the direction n_i at the contact point. Equation 6 can be rewritten as

$$\Delta v_i \geq -(1 + \epsilon_i) v_i^- , \tag{9}$$

where

$$\Delta v_i = v_i^+ - v_i^- . \tag{10}$$

The quantity Δv_i is the instantaneous change in velocity between t_c^- and t_c^+ caused by the impulse forces j_i. The velocity v_i (Equation 5) can be expressed as

$$v_i = n_i^T J_i \dot{q} , \tag{11}$$

Since [4]

$$\dot{p}_i = J_i \dot{q} , \tag{12}$$

where J_i is the Jacobian that maps $\dot{q}$ to the velocity at the i^{th} collision point. Accordingly, Δv_i can be expressed as

$$\Delta v_i = n_i^T J_i \Delta \dot{q} . \tag{13}$$

The instantaneous change in velocity Δv_i is calculated by first mapping the unknown impulse forces into the generalized coordinate space of the object. For collisions at the end effector of a manipulator, this mapping is given by

$$\mu_i = J_i^T n_i j_i , \tag{14}$$

where μ_i is the generalized impulse caused by the collision impulse j_i on the contact point i. The total generalized impulse on the object is the sum of the individual impulses on the object. This vector is then given, in the generalized coordinates of the object, by the equation

$$\mu = \sum_{k=1}^{m} J_k^T n_k j_k . \tag{15}$$

The generalized impulse μ is then used to compute the instantaneous change of the generalized coordinates.

$$\Delta \dot{q} = A^{-1} \mu . \tag{16}$$

Using Equations 13, 15, and 16, Equation 9 can be rewritten as

$$n_i^T J_i A^{-1} \sum_{k=1}^{m} J_k^T n_k j_k \geq -(1 + \epsilon_i) J_i \dot{q}^- . \tag{17}$$

Since v_i^- is previously known and Δv_i depends linearly on the unknown impulse magnitudes j_i, the constraint equations can be solved simultaneously for the unknown impulse forces j_i. The constraint equations (Equations 6, 7, 8) are such that they can be solved using quadratic programming techniques [1].

Once the impulse forces have been found, the new generalized velocity can be calculated.

$$\dot{q}^+ = \dot{q}^- + \Delta\dot{q}\,. \tag{18}$$

The above equations can be easily extend to handle multiple, moving, and redundant objects [2, 10].

5.3 Contact Force Resolution

If $v_i^+ = 0$, inter-penetration may still occur if $a_i = \dot{v}_i^+$ is negative. In this case, the objects are said to be in resting contact. When objects are in resting contact, forces of magnitude f_i must be generated at the contact points in the direction normal to the contact surfaces. The magnitude of the force must be non-negative, and zero when the contact points are separating, i.e., $a_i > 0$. Given m contact force constraints, the constraint equations may be written as

$$a_i \geq 0\,, \tag{19}$$

$$f_i \geq 0\,, \tag{20}$$

$$f_i a_i = 0\,. \tag{21}$$

where $1 \leq i \leq m$. Note these constraint equations are similar in form to Equations 6, 7, and 8. Given that

$$v_i^+ = n_i^T J_i \dot{q}^+\,, \tag{22}$$

we can differentiate v_i^+ to obtain a_i.

$$a_i = n_i^T[J_i\ddot{q} + \dot{J}_i\dot{q}] + \dot{n}_i^T J_i\dot{q}\,. \tag{23}$$

Recalling now Equation 3

$$\ddot{q} = A^{-1}(q)[\Gamma - b(q,\dot{q}) - g(q)]$$

where $\Gamma = \Gamma_{joint} + \Gamma_{ext}$, and Γ_{ext} is the generalized torque caused by the sum of the forces f_i applied at the contact points in the direction n_i. The mapping between Γ_{ext} and the applied forces at the contact points [4] is given by

$$\Gamma_{ext} = \sum_{k=1}^{m} J_i^T n_i f_i. \tag{24}$$

Equations 19, 20, and 21 depend linearly on the unknown forces f_i (see Equations 23, 2, 3, and 24) and have a form similar to the impulse constraint equations(Equations 6, 7, and 8). A similar method may be used, therefore, to solve the contact constraints. Once the contact forces are found, Γ_{ext} may be computed and the dynamics are updated to the next time cycle.

A discussion on determining Γ_{ext} for redundant manipulators and for contacts outside the end effector has been given by Khatib [10].

6 Implementation

Our dynamic simulator is implemented in C on a Silicon Graphics workstation. Gilbert's distance algorithm is used to find the minimum distance between objects [5]. When interaction occurs, the constraint equations (Equations 6, 7, 8 and 19, 20, 21) are solved by means of quadratic programming techniques [6]. When the objects are not in contact with each other, they are governed solely by their rigid body dynamics. For a given body, these dynamic equations are predetermined and calculated symbolically off-line by a program based on Kane's formulation [8].

These equations are integrated to obtain position and velocity information using an Adams-Bashforth adaptive step-size predictor-corrector integrator [7].

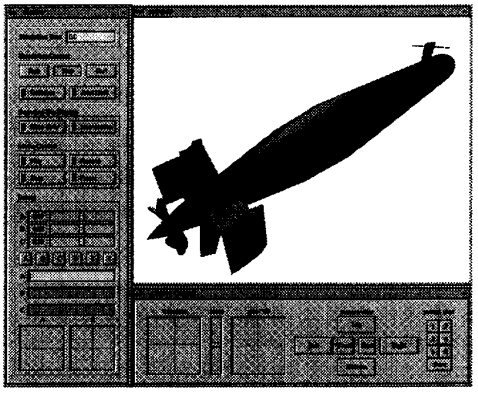

Figure 5: Graphical User Interface

We have also developed a graphical user interface to make our simulator more accessible to the user. This interface, which controls camera views, collision detection, movie record/playback, and simulation is shown on Figure 5.

7 Results

We have developed a modeling system to dynamically simulate interactive robotic environments. Using this simulator, a wide range of robotic environments can be represented and simulated.

For example, a crane has been simulated to study the load swing under various conditions and to determine the feasibility of different tasks under those conditions (Figure 1). We have also simulated (Figure 4) a more traditional robotic environment consisting of two PUMA 560 manipulators to test various robotic applications including a singularity controller that smoothly moves in and out of singular configurations. We have also used our approach to model a submarine (Figure 5) to determine the feasibility of an autonomous underwater vehicle recovery system.

A simple simulation can demonstrate the dynamic interactions of robots. Figure 6 illustrates collisions between two 6 DOF robots. The left robot is falling under the effect of gravity while the right robot has gravity compensation. The up/down sequence shows the two robots colliding at frame 5 and again at frame 16 (right of frame 5). In this sequence, we can also observe the complex robot motions due to joint coupling. Even for such a simple environment, the dynamics of multi-linked objects and the resolution of interactive forces must be simulated in order to produce realistic results. We have incorporated these functionalities into our package by means of a global/local structure that creates efficient dynamic simulations of interactive robotic environments.

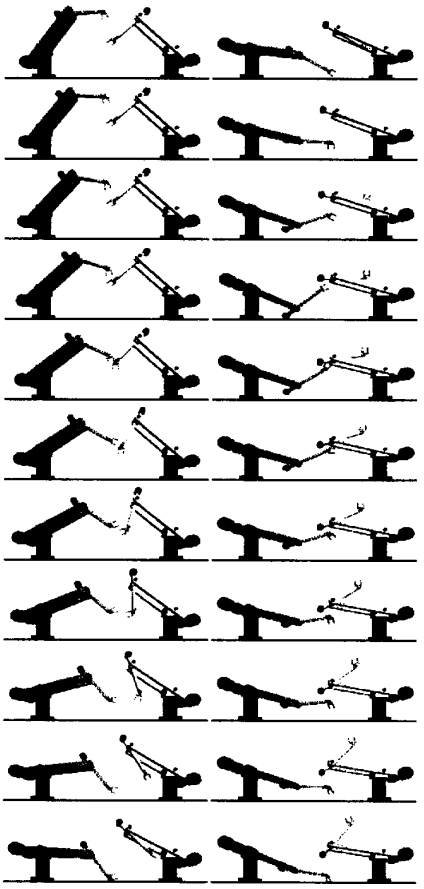

Figure 6: Sequence of Interactive Dynamic Simulation

References

[1] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," *Computer Graphics 23*, (August 1989), 223-232.

[2] D. Baraff, "Rigid Body Simulation", *Lecture Notes for SIGGRAPH '92 Course*, (1992).

[3] R. Barzel and A. H. Barr, "A modeling system based on dynamic constraints," *Computer Graphics 22*, 4 (August 1988), 179-188.

[4] J. Craig, "Introduction to Robotics Mechanics and Control," *Addison-Wesley Pub. Co.*, 1989.

[5] E. G. Gilbert, et. al., "A Fast Procedure for Computing the Distance between Complex Objects in Three-Dimensional Space," *IEEE J. of Robotics and Automation*, Vol.4, No. 2, April 1988.

[6] P. Gill, S. Hammarling, W. Murray, M. Saunders and M. Wright, "User's guide to LLSOL," *Stanford University Technical Report SOL 86-1*, (January 1996).

[7] D. Kahaner, et. al., "Numerical Methods and Software," *Prentice Hall*, Englewood Cliffs, NJ, 1989.

[8] T. Kane, *Dynamics: Theory and Applications*, McGraw-Hill, 1985.

[9] O. Khatib, "A Unified Approach to Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE J. on Robotics and Automation*, Vol. 3, No. 1, 1987.

[10] O. Khatib, "Object Manipulation in a Multi-Effector Robot System," *Int. Sym. of Robotics Research*, Santa Cruz, CA, Aug. 1987.

[11] M. Moore and J. Wilhelms, "Collision detection and response for computer animation," *Computer Graphics 22*, (1988), 289-298.

[12] M. Orlowski, "The computation of the distance between polyhedra in 3-space," *SIAM Conf. on Geometric Modeling and Robotics*, Albany, NY, July 1985.