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 has been developed which can accurately model the interactions between robots and their environment. It creates a virtual environment in which various controllers and workcells can be tested. The simulator is divided into two parts: local objects that compute their 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 can be used to obtain insight into real-world behavior of a robotic environment.

To illustrate the usefulness of the robotic simulation, consider the construction industry, which uses cranes to transfer beams from one point to another (see Figure 1). Time optimal control of the crane which minimizes the swing of the load is desired for this task. If unstable controllers were implemented on a real crane, permanent damage to the crane and/or its environment could result. However, using simulation, we can test, change, and improve the controller design in an iterative fashion without the fear of damaging the physical system. Computer graphics provides a means for visualizing these 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 based on dynamic equations of motion with accurate analytical solutions of the constraint equations [1]. The dynamic equations of motion can be easily obtained for each object, but interactions (i.e. collision and/or contact) between objects have proved to be more difficult to handle. In recent work, two general approaches have been proposed to address this problem.

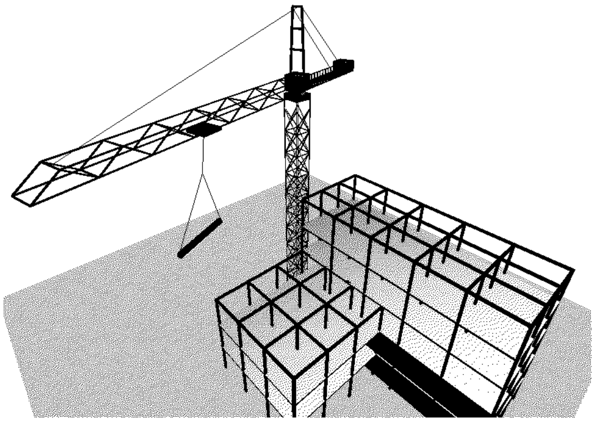

Figure 1: Simulation of a Crane

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

The other approach finds analytically correct solutions to the dynamic interactions between rigid bodies by explicitly solving the constraint equations. Baraff [1] uses this approach 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 difficult to implement and are not easily extended to non-rigid 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 approach, which yields accurate analytical solutions [1]. Previous algorithms have been developed only for simple rigid-body objects. We have extended these algorithms to account for multiple link robots and other articulated objects.

2 Dynamic Simulator

In the design of our dynamic simulator, the main objective is to create an interactive package that can be run in tandem with a physical robotic environment. In an ideal system, robot controllers can be connected interchangeably between real robots and our dynamic simulator. This capability will allow us to preview the behavior of robots and their interactions with the environment. In this way, various tasks and controllers can be tested in a "virtual reality" environment before they are applied to physical systems.

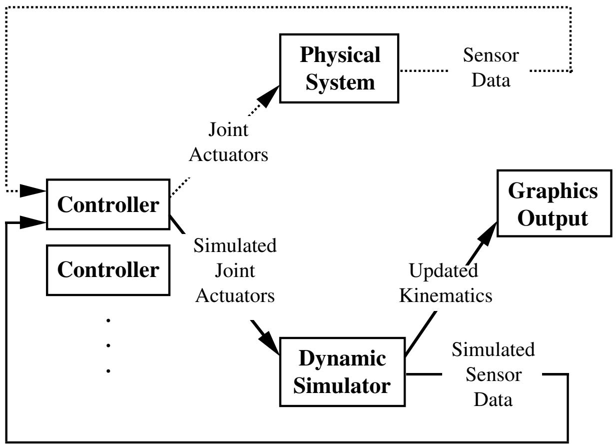

Figure 2: Overall Structure of the System

Figure 2 illustrates the overall structure of the system. The dynamic simulator takes joint actuator forces from a set of controllers as inputs. Using 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, are sent to a graphical output process, to update the display of the simulated environment.

3 Global/Local Architecture

Based on the following observations, the simulator has been separated into two parts: a global coordinator and a set of local objects with their corresponding equations of motion.

- Dynamic equations of motion can be determined and calculated independently for each object if there are no contact or collision between objects.
- If contact or collision occurs, there exist some constraint forces or impulses that impose correct dynamic behavior and prohibit inter-penetrations of objects. Determination of constraint forces and impulses requires knowledge of all objects that are colliding or in contact at a given time. Therefore, they must be resolved globally.

A local object consists of its equations of motion, an integrator to update its positions and velocities based on its dynamics, and a set of parameters necessary 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 to model the contacts [1].

4 Object Structure

The configuration of an n-DOF (degree-of-freedom) object is described by a set 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 can also be used. The equations of motion in terms of generalized coordinates are given by

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

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}.

$$\Gamma = \Gamma_{joint} + \Gamma_{ext} \tag{2}$$

To dynamically simulate an object using the its equations of motion, Eq. 1 is solved for $\ddot{q}$.

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

Joint Acceleration Mass Matrix Coriolis and Centrifugal Force Gravity Force Joint Torque . .	**Dynamic Model**
Joint Angle Joint Velocity Jacobian . .	**Kinematic Model**
Link Shape (cylinder, rectangle, etc.) Surface Representation . .	**Geometric Model**
Graphics Representation Color . .	**Graphical Model**

Figure 3: Robot Structure

Given an initial position, velocity, and Eq. 3, updated positions and velocities can be obtained using an ODE (ordinary differential equation) integrator.

The above representation (Eq. 3) 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, it solves only for the parameters which affect the dynamics of the system. In contrast, previous works [1, 3] treat multi-linked objects as series of simple objects. This approach requires solutions for all the forces, including the internal joint forces, which have no effect on the motion of the system. This unnecessary calculation results in an larger computation time.

To represent and solve the equations of motion, each object stores the necessary parameters 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 forces required to cause motion. It includes data such as inertial forces, Coriolis forces, and the kinetic energy matrix.

The kinematics model includes data such as joint angles, joint velocities, the jacobian, and transformation matrices, which can 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 and where contact between objects has occurred. The location of the contact points can then be used to resolve the constraint force equations.

A graphical model determines the visual image of the objects in a graphics output. Information such as shape, color, lighting, shading, etc. are included in the graphical model to determine the visual quality of the object.

5 Global Coordinator

5.1 Collision/Contact Detection

Collision and contact points are determined by first calculating object locations using the equations of motion while assuming no collisions or contact. If there are inter-penetrations between objects, collision and/or contact has occurred. Forces arising due to this collision/contact can then be determined by solving the constraint equations as described below in sections 5.2 and 5.3.

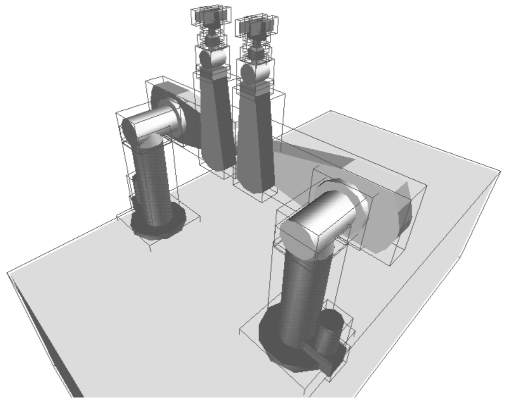

Figure 4: 6 DOF manipulator with Bounding Boxes

The algorithm which determines inter-penetration is divided into two parts. First, each object is enclosed in bounding boxes which can be checked quickly for intersections. If bounding boxes do not overlap, we can bypass further collision/contact checking. Figure 4 shows two 6 DOF arm robots with appropriate bounding boxes around their links.

A naive collision checking implementation, which checks all pairs of objects, produces $O(n^2)$ performance, but a sort and sweep algorithm can solve it in $O(n \log n)$ [2]. Subsequent steps can use an insertion sort for more efficiency.

Objects with overlapping bounding boxes require calculation of the minimum distance between them in order to determine inter-penetrations. Orlowski [12] uses an algorithm with $O(M \log M)$ performance where M is total number of vertices in the two objects, $M = M_1 + M_2$. Baraff uses an algorithm which has $O(M_1 M_2)$ performance initially, but uses temporal witnesses to reduce subsequent calculations to $O(M)$ performance [2]. We have implemented an algorithm developed by Gilbert [5], which is considered to be one of the most efficient algorithms for computing distance. Its computation times are $O(M)$ with a small coefficient of linear growth.

5.2 Impulse Force Resolution

A collision between two objects a, and b occurs when there exists points p_a and p_b such that at some collision time t_c, $p_a(t_c) = p_b(t_c)$ and their relative velocity just prior to collision is negative in the direction n_i normal to the contact region. Given m collision points and $1 \leq i \leq m$, 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. Given that n_i is defined to have a positive direction from b to a, v_i^- is formally defined as

$$v_i^- = n_i^T \dot{p}_{a_i}^- - n_i^T \dot{p}_{b_i}^- < 0. \tag{4}$$

where $\dot{p}_{a_i}^-$ and $\dot{p}_{b_i}^-$ are 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 $\dot{p}_{b_i}(t) = 0$, and Eq. 4 becomes

$$v_i^- = n_i^T \dot{p}_{a_i}^- < 0. \tag{5}$$

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 equations that are stiff, and therefore slow to integrate. We have instead chosen to model objects as rigid bodies, and calculate the impulse forces explicitly. An impulse causes an instantaneous change in the velocity of an object. Therefore the equations of motion are integrated to the time of collision, new values of the generalized velocities $\dot{q}$ calculated, and integration continued using the new state. By breaking up the integration at the velocity discontinuity, 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. Eq. 6 can be rewritten as

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

where

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

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 (Eq. 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

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

where μ_i is the generalized impulse caused by impulse j_i on contact point i. The total generalized impulse on the object is the sum of the individual impulses on the object mapped into the generalized coordinates of the object.

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

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

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

Using Eq. 13, 15, and 16, Eq. 9 can be rewritten as

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

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. Conditions of the constraint equations (Eq. 6, 7, 8) are such that they can be solved using quadratic programming techniques [1].

Once the impulse forces are found the new generalized velocity can be calculated.

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

While the equations above are for only a simplified environment they 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. When this occurs, 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 nonnegative, and zero if the contact points are separating $a_i > 0$. Given m number of contact force constraints, the constraint equations can 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 Eq. 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.

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

From Eq. 3, $\ddot{q}$ is expressed as

$$\ddot{q} = A^{-1}(q)[\Gamma - b(q, \dot{q}) - g(q)] \tag{24}$$

where Γ is

$$\Gamma = \Gamma_{joint} + \Gamma_{ext}. \tag{25}$$

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 as

$$\Gamma_{ext} = \sum_{k=1}^{m} J_i^T n_i f_i. \tag{26}$$

Eq. 19, 20, and 21 depend linearly on the unknown forces f_i (see Eq. 23, 24, 25, and 26) and have a form similar to the impulse constraint equations(Eq. 6, 7, and 8). Therefore, a similar method can be used to solve the contact constraints. Once the contact forces are found, Γ_{ext} can be computed and the dynamics can be updated to the next time cycle.

For a discussion on finding Γ_{ext} for redundant manipulators, and for contacts outside the end effector, see [10].

6 Implementation

Our dynamic simulator is implemented in C on a Silicon Graphics workstation. It uses Gilbert's distance algorithm to find the minimum distance between objects [5]. When interaction occurs, the constraint equations (Eq. 6, 7, 8 and 19, 20, 21) are solved using quadratic programming [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 off-line, symbolically using a program based on Kane's formulation [8].

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

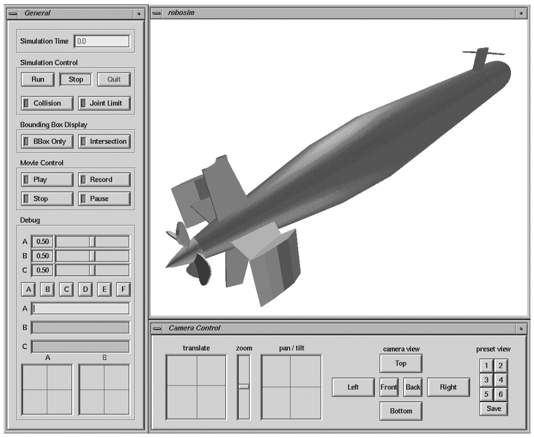

Figure 5: Graphical User Interface

We have also developed a graphical user interface (GUI) to make our simulator more accessible to the user. Figure 5 shows our GUI that controls camera views, collision detection, movie record/playback, and simulation.

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 determined the feasibility of different tasks given these specific conditions (Figure 1). Figure 4 shows a more traditional robotic environment which consists of two PUMA 560 manipulators. It has been used to test various robotic applications, including a singularity controller that smoothly moves in and out of singular configurations. Figure 5 shows a model of a submarine, implemented to show the feasibility of an autonomous underwater vehicle recovery system.

A simple robotic environment in simulation can demonstrate the dynamic interactions of robots. The setup shown in Figure 6 shows 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 observe the complex robot motions due to joint coupling.

Even for a simple environment as shown here, we need the dynamics of multi-linked objects and the resolution of interactive forces in order to produce a realistic simulation. In our simulator, we have incorporated these functionalities in a global/local structure that that creates efficient dynamic simulations of interactive robotic environments.

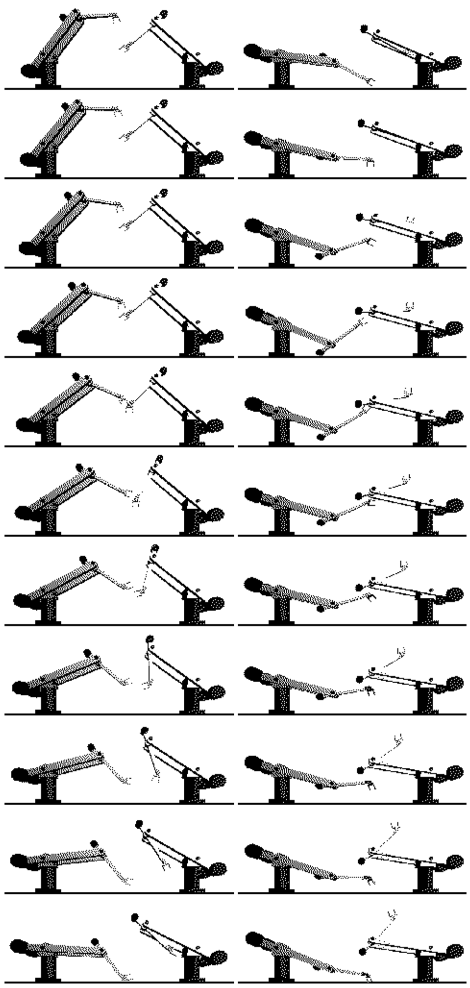

Figure 6: Sequence of Interactive Dynamic Simulation

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.