Hybrid Simulation: Combining Constraints and Impulses

Brian Mirtich *

Abstract

Impulse-based simulation has been shown to be a useful paradigm for rigid body simulation [12, 11], especially for systems which are collision intensive, or undergo many changes in contact configuration. In this paper we briefly describe the simulator Impulse, *and report some recent results in the domain of part feeding.*

Since the impulse- and constraint-based approaches work best for orthogonal situations, it is advantageous to use both methods simultaneously. After describing the range of impulse- to constraint-based contact interaction, we examine how the impulsive collision resolution may be extended to constrained systems. We also discuss some important open problems related to developing an efficient simulator that uses both contact interaction paradigms.

1 Introduction

Many simulators for simple physical systems employ constraint-based approaches [1, 3, 5, 15]. Constraints are used to describe the interactions between objects, which often occur only through physical contact. A large variety of contact interactions can be modeled efficiently and accurately by hard constraints, however the method is not well suited to situations like the one depicted in figure 1. Under constraint-based simulation,

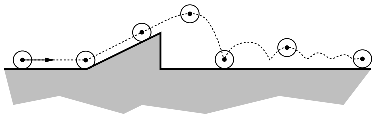

Figure 1: *A nightmare for constraint-based simulation.*

the constraints change as the ball begins traveling up the ramp, leaves the ramp, bounces for a while, perhaps sliding, and eventually settles into a roll along the ground. All of these occurrences must be detected and processed, and new equations of motion for the system must be derived at every transition. Next consider a vibratory part singulator (figure 2), which shakes small parts into recesses for automated assembly. This system has a very large number of degrees of freedom, and there are also many collisions and very transient contact

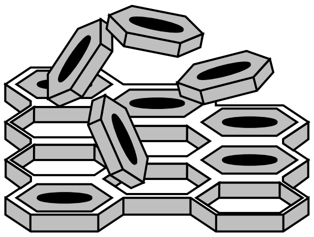

Figure 2: *High DOF, collision intensive, and transient contact configurations.*

configurations. All of these factors make it a difficult system to simulate via constraints.

Inspired by examples like these, we have developed a simulator based on a radically different approach to rigid body simulation. The simulator is called *Impulse*, and the main idea is that all contact interactions, whether colliding, rolling, sliding or resting, are affected via trains of collisions. There are no global constraints governing the motion of the objects. Instead, the correct macroscopic behavior results from processing individual collisions. The approach works well for systems like those shown in figures 1 and 2. It is very efficient, and produces physically valid simulations for a wide variety of problems [12]. However, impulse-based contact modeling is not always a good choice, especially in situations which involve prolonged, close contact. A challenge is to combine these two very different simulation paradigms, using each one when appropriate.

In section 2 we give an overview of the impulse based approach, focusing on the collision detection and resolution subsystems of *Impulse*. We also report some results on simulation speed and physical accuracy of the simulator. In section 3 we examine a spectrum of systems ranging from collision intensive to highly constrained, and explore the relationship between impulse- and constraint-based simulation. In section 4, we discuss how the two simulation paradigms might be combined, mentioning some solved problems and some open problems. We conclude in section 5.

2 Impulse-based simulation

The simulator *Impulse* demonstrates the feasibility of modeling contact with collisions. We briefly describe collision detection and response in this system, as well as some results pertaining to efficiency and accuracy. More

* *mirtich@cs.berkeley.edu*, Department of Computer Science, 387 Soda Hall, University of California, Berkeley, CA 94720. Supported in part by NSF grant #FD93-19412.

details on the impulse-based approach may be found in [12, 11].

2.1 Collision detection

Impulse-based dynamic simulation is inherently collision intensive, since collisions are used to affect all types of interaction between objects. Hahn found collision detection to be the bottleneck in dynamic simulation [7], and efficient data structures and algorithms are needed to make impulse-based simulation feasible.

All objects in *Impulse* are geometrically modeled as convex polyhedra or compositions of them. A collision is declared when the distance between two such objects falls below some threshold. At the heart of the collision detection system is the Lin-Canny closest features algorithm [10]. This efficient algorithm exploits coherence to return the closest pair of features (faces, edges, or vertices) between two polyhedra in near constant time. From this information, the intervening distance is easily computed.

Collision checks are prioritized in a heap (figure 3). Whenever collision detection is performed on a pair of

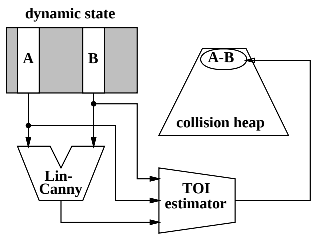

Figure 3: *Prioritizing collisions in a heap.*

objects, a time of impact (TOI) estimator also computes a lower bound on the time of collision of these two objects. This can be determined since the objects execute ballistic trajectories between collisions, however if an object experiences a collision, its TOI with all other objects must be recomputed. The objects pairs are kept in a heap sorted on the TOI field. In this way, a dynamic integration step may proceed safely to the time given by the TOI field of the top heap element. At this point, a collision check need only be performed between this top pair, possibly causing it to drop in the heap, and then dynamic integration may resume.

To further cull collision checks, a spatial partitioning scheme is employed. Object pairs are only kept in the heap if the objects are "close." Closeness can be rapidly determined by storing the positions of all objects in a large hash table, based on a cubical tiling of physical space, as described in [13]. This scheme tremendously reduces the number of collision checks and TOI computations that must be performed, since objects are usually in the vicinity of only a small subset of the set of all objects.

2.2 Computing collision impulses

When a collision is detected, the collision response subsystem computes the appropriate impulses to apply to the two objects. The points of application for these equal and opposite impulses are determined by the collision detection subsystem. The impulses must prevent interpenetration, and also obey certain other physical laws relating to friction and restitution. Central to our analysis are the assumptions of: infinitesimal collision time, Poisson's hypothesis, and the Coulomb friction model.

Since the frictional force is dependent on the relative sliding velocity of the bodies in contact, and this velocity is not constant during a collision, the dynamics of the object must be analyzed during the collision. Let $\mathbf{u}$ be the relative velocity between the two bodies at the contact point, and $\Delta\mathbf{u}$ be the change in this quantity over the course of the collision. If $\mathbf{p}$ is the collision impulse imparted by one body on the other, one can show [11] that

$$\Delta\mathbf{u}(\gamma) = M\,\mathbf{p}(\gamma). \tag{1}$$

Here M is a 3×3 matrix dependent only upon the masses and mass matrices of the colliding bodies, and the locations of the contact points in the body frames. Note that M is constant for a given collision. We compute $\mathbf{p}$ by tracking $\mathbf{u}$ during the collision, and then inverting equation (1).

Using the Coulomb sliding friction law, one can derive the following differential equation for $\mathbf{u}$:

$$\begin{bmatrix} u'_x \\ u'_y \\ u'_z \end{bmatrix} = M \begin{bmatrix} -\mu\frac{u_x}{\sqrt{u_x^2+u_y^2}} \\ -\mu\frac{u_y}{\sqrt{u_x^2+u_y^2}} \\ 1 \end{bmatrix}, \tag{2}$$

where differentiation is with respect to the normal component of impulse. These equations are integrated by the collision response subsystem to track the evolution of $\mathbf{u}$ during a collision. If sticking occurs during this integration ($u_x = u_y = 0$), the model changes and a simpler set of differential governs the evolution of $\mathbf{u}$.

2.3 Computational efficiency

We have tested *Impulse* on a wide variety of rigid body simulation problems, some of which exhibit quite interesting physics. We have generally worked with small scale physical systems: on the order of five to ten moving objects, and a similar number of fixed objects. Nonconvex objects have been used in some simulations. Simulation times for a battery of tests are shown in table 1. *Virtual time* is the length of time which passed in the simulation, *real time* is the actual time needed to compute the simulation[1], and *slowdown* is the ratio of

[1] Real times were computed by averaging over several trials. All simulations were performed on an *SGI Indigo 2* (R4400 CPU).

simulation	virtual time (s)	real time (s)	slow-down
dominos	1.2	9.0	7.50
chain of balls	2.5	3.7	1.5
coins	3.6	17	4.7
pool break	3.0	44	14.7
bowling a strike	5.0	78	15.6
ball on platter	40	65	1.6
balls in dish	7.8	22	2.8
rattleback tops	5.0	18	3.6
part feeding	5.0	66	13.2

Table 1: *Simulation times for experiments.*

the latter to the former (a 1.0 slowdown corresponds to real time simulation).

The slowdowns for these experiments range from 1.5 to 15.6. *Impulse* is roughly an order of magnitude off real time simulation for systems of this size.

2.4 Physical accuracy

Physical accuracy is of paramount importance for a dynamic simulator. We have tested the accuracy *Impulse* with several experiments, many of which are detailed in [12]. A more recent "real world" problem solved with *Impulse* was the estimation of the stable pose distribution for a small part dropped onto a flat surface. This is an important problem in designing production lines and parts for automated assembly [4]. Such knowledge is useful in estimating feeder throughput as well as in modifying part designs to optimize assembly speed or success rates. For the experiment, we used the small, plastic test part shown in figure 4. This

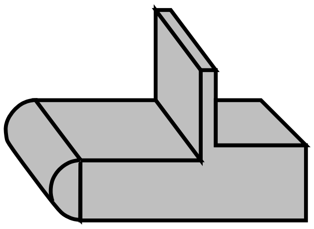

Figure 4: *Test part for stable pose distribution test.*

part has six stable poses, illustrated in the histogram of figure 5 (two stable poses have been grouped together).

Experimental results were obtained by performing over 1000 physical drop tests with this part, under controlled conditions. Two predictions for the stable pose distribution were also obtained, one from a quasi-static algorithm [14], and the other from a series of over 2000 drop tests using *Impulse*, performed under similar conditions to the actual drop tests. As can be seen by the

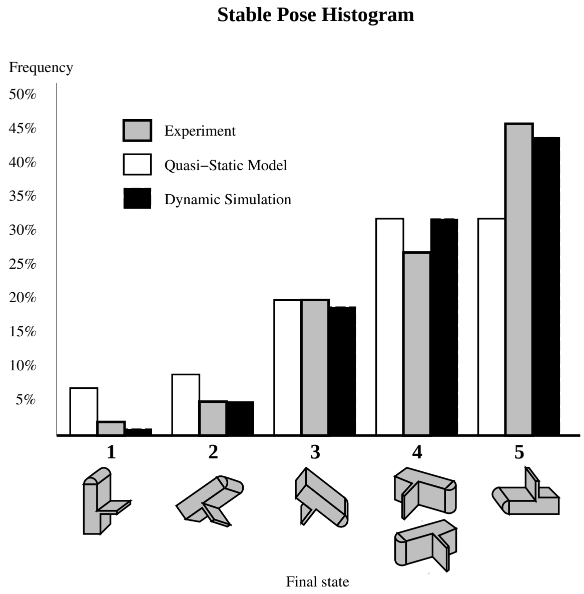

Figure 5: *Results from the stable pose distribution test.*

histograms of figure 5, the results from dynamic simulation are quite good; the maximum deviation from the experimental data is 4%. The drop tests took less than 45 minutes to perform using *Impulse*. Such results are encouraging, but more testing is needed to verify that impulse-based simulation is a valuable tool for this kind of problem.

3 A spectrum of systems

While there is a large range of contact interactions that can be modeled via trains of impulses, there are also limitations of this approach. Constraints often play important roles in many mechanisms common in our everyday lives. Consider one of the most simple constrained mechanisms: a hinge joint (figure 6). In prin-

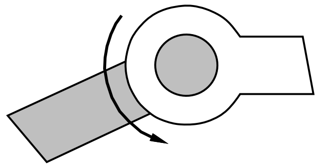

Figure 6: *A nightmare for impulse-based simulation.*

ciple, one could model the joint in an impulse-based way, enforcing the hinge constraint through collisions between the hinge pin and sheath. However, due to the enormous amount of collision detection and resolution that would be necessary to model this contact, impulse-based simulation would be far too slow; constraint-based dynamics should be used instead.

Consider the spectrum of physical systems depicted in figure 7. On the right most end are jointed manip-

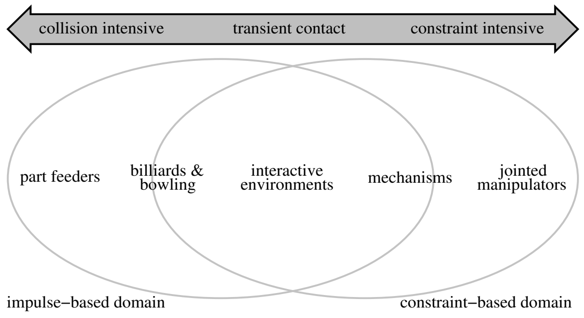

Figure 7: *A spectrum of physical systems.*

ulators, a class of systems for which a constraint-based simulation approach is clearly the appropriate tool. The motion of these systems of rigid links is governed by explicit, well-understood, and *permanent* constraints, such as those imposed by revolute or prismatic joints. Computing the forward dynamics of these systems is a classical problem of robotics, and a variety of methods exist for doing so [6, 9]. Note that the interaction between the manipulator and the environment (typically occurring at the end effector) is not so clearly a constraint-based interaction, since often this contact is transient or not as accurately modeled by a hard constraint. The contact can still be modeled this way, and closed-loop methods exist for computing the dynamics of a manipulator in contact with its environment.

Moving toward the left on the spectrum, we encounter mechanisms, a class of systems which are also highly constrained. Here the constraints are not always permanent; mechanisms often have multiple modes during a complete cycle of operation. Nonetheless, there are usually relatively large ranges of motion over which the contact configuration does not change, and constraint-based simulation works well. Baraff's *blockfeeder* and *double-action jack* are excellent examples of constrained simulation of 2D mechanisms [2].

On the other end of the spectrum are part feeders and similar systems. For these collision intensive systems, impulse-based dynamics is a natural choice for simulation. Moving to the right, we encounter billiards and bowling. In these systems, collisions still play a major role in determining the dynamics, and the contact modes change frequently. The rolling and sliding of objects along horizontal surfaces can also be easily modeled with impulses.

The middle ground of the spectrum is thus far the least studied, however there are many applications which fall in this category. Some important examples are interactive environments. Collisions provide much of the interaction between agent and environment, especially if the agent is constrained by walls and other fixed objects or has the ability to move or throw objects around the environment. Many of the contact modes are transient as the agent grabs, pushes, or pulls objects in the environment. On the other hand, there are several nicely constrained objects as well: hinged doors between rooms, simple mechanical structures like see-saws, etc. We believe that the fastest route to real time simulation of interesting interactive environments will involve combining both simulation paradigms.

4 Hybrid simulation

We now discuss some of the issues involved in combining impulse- and constraint-based simulation.

4.1 Colliding constrained bodies

Consider a constrained body coming into contact with part of its environment (figure 8). Prolonged contact may suggest establishing a constraint between the body and environment, but at least initially the contact should be handled via impulses. The goal is to

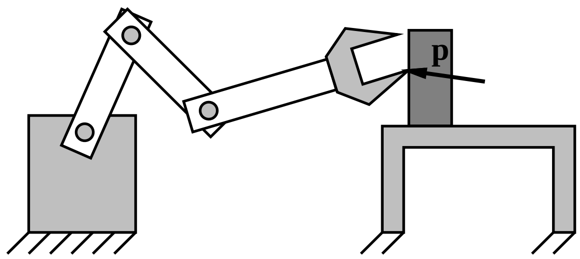

Figure 8: *The constrained end effector collides with its environment.*

determine the collision impulse $\mathbf{p}$ which should be applied to the manipulator. Analogous to the situation for colliding (unconstrained) rigid bodies, this is done by relating the change in relative contact velocity to the collision impulse.

Let $\mathbf{q}$ be the vector of manipulator joint positions. The dynamics of the manipulator are given by [9]:

$$\ddot{\mathbf{q}} = H^{-1}(\mathbf{q}) \; \left[\tau - C(\mathbf{q}, \dot{\mathbf{q}}) \; \dot{\mathbf{q}} - G(\mathbf{q}) + J^T(\mathbf{q}) \; \mathbf{f}\right], \quad (3)$$

where τ is the vector of joint torques (or forces), $\mathbf{q}$ is the vector of joint positions, H is the joint space inertia matrix, C is the matrix of Coriolis and centripetal force terms, G is the vector of gravitational forces, J is the manipulator Jacobian, and f is the vector of external forces and moments exerted on the last link.

Integrating (3) through time results in an expression for the change in joint velocities $\Delta\dot{\mathbf{q}}$. Consider integrating this equation over the course of a collision. Since the Coriolis, centripetal, gravitational, and actuated joint forces are all of finite magnitude, over an infinitesimal time interval the contributions of these forces to $\Delta\dot{\mathbf{q}}$ are negligible. Another way of thinking of this is that the impulsive impact forces dominate all other forces during a collision; the identical assumption is made in rigid body collision analysis when gravity is neglected during a collision. As a result, we have:

$$\Delta\dot{\mathbf{q}} = \int_0^{t_c} H^{-1}(\mathbf{q}(t))\ J^T(\mathbf{q}(t))\ \mathbf{f}(t)\ dt, \tag{4}$$

where t_c is the duration of the collision. As t_c approaches zero $\mathbf{q}$ can be treated as a constant, thus

$$\Delta\dot{\mathbf{q}} = H^{-1}(\mathbf{q})\ J^T(\mathbf{q})\ \mathbf{p}, \tag{5}$$

where $\mathbf{p}$ is the collision impulse. If the object contacting the manipulator is constrained to be motionless, the relative velocity at the contact point is given by

$$\mathbf{u} = J(\mathbf{q})\ \dot{\mathbf{q}}. \tag{6}$$

From (5) and (6) we obtain the desired relation between change in relative contact velocity and applied impulse:

$$\Delta\mathbf{u} = J(\mathbf{q})\ H^{-1}(\mathbf{q})\ J^T(\mathbf{q})\ \mathbf{p}. \tag{7}$$

Compare this result to equation (1). The key feature is that $\Delta\mathbf{u}$ and $\mathbf{p}$ are related by a constant matrix. Incidentally, this constant matrix is exactly the operational space inertia matrix $\Lambda(\mathbf{x})$ developed by Khatib [8]. The assumptions that the object the manipulator contacts is fixed, and that the collision occurs at the end effector can both be relaxed. Collision impulses can be computed using the same approach which worked for simple rigid body collisions.

4.2 Switching between constraints and impulses

For a given simulation, one could predetermine which types of contact interaction would be processed by impulses and which by constraints, for all time. For example, the interaction between an agent and a door could be modeled by impulses, while the door's interaction with the doorjamb was modeled as a hinge constraint. While this is more powerful than either approach used alone, a more flexible approach is better. The interactive environment example motivates the use of "paradigm shifts" in contact modeling.

Consider an agent arranging objects in a room. As he places objects on tables or shelves, impulse-based simulation provides very realistic settling behavior during the transition from the initial contact to a resting state. At some point, however, processing the collisions to maintain this contact seems a bit ridiculous, since much work is being done to make an object sit at rest on a surface. The solution is to simply *constrain* the object to sit on the surface, and stop processing collisions at this contact. Impulse to constraint shifts can also be more complicated. In the example of figure 1, one could change from an impulse to a constrained model once the ball had stopped bouncing, stopped slipping, and was rolling along the floor.

Paradigm shifts can go the other way as well. Basically any time the agent interacts with objects previously following some constrained trajectories (e.g. kicking a rolling ball or knocking down a tower of blocks), a switch to the impulse-based model is appropriate. The agent need not even be involved; constrained objects can suddenly begin to interact in very complicated, collision intensive ways, such as when a ball strikes a set of bowling pins initially at rest.

How does one determine exactly when a paradigm shift is necessary? Simple criteria such as constraining an object to be fixed when its velocity reaches zero do not always work; when such a rule was placed into *Impulse*, objects began to freeze in mid-air at the top of their parabolic trajectories. Nonetheless, we believe good rules governing the switches between contact models can be found, and should be implemented.

4.3 Collision detection

Another important challenge to meshing the two simulation paradigms concerns collision detection. Impulse-based simulation gains a lot from the fact that objects follow ballistic trajectories between collisions. From this knowledge, reasonably tight lower bounds on times of impact can usually be found, and collisions are never missed. When objects are constrained to follow other trajectories, new methods are needed to predict collision times. Or perhaps the current scheme based on completely conservative time of impact bounds should be significantly altered to handle constrained objects. Since tight bounds on the motion of the end effector of a manipulator are hard to obtain, it might be better to predict the time of impact non-conservatively, and recover gracefully if the prediction proves poor. We are currently exploring the collision detection problem for hybrid simulation.

5 Conclusion

Constraint-based simulation has a long and successful history for many types of problems, and impulse-based simulation has more recently proved its utility for an orthogonal class of problems. A new challenge is to unite these methods, combining the strengths of each. Interactive environments are natural test problems for using the two methods in tandem.

Rigid body collision resolution can be straightforwardly extended to constrained systems; the approach based on tracking the relative contact velocity during collision will work. How to extend *Impulse's* efficient collision detection to handle constrained bodies is less clear. It is also important to determine when to make paradigm shifts in contact modeling, so that both the impulse- and constraint-based methods are used at the appropriate places and times. Solving these problems should lead to a very powerful simulation capability.

References

[1] David Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[2] David Baraff. Fast contact force compuation for nonpenetrating rigid bodies. In *SIGGRAPH*. ACM Press, 1994.

[3] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988.

[4] Brian Carlisle, Ken Goldberg, Anil Rao, and Jeff Wiegley. A pivoting gripper for feeding industrial parts. In *International Conference on Robotics and Automation*. IEEE, 1994.

[5] James F. Cremer and A. James Stewart. The architecture of newton, a general-purpose dynamics simulator. In *International Conference on Robotics and Automation*, pages 1806–1811. IEEE, May 1989.

[6] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *International Journal of Robotics Research*, 2(1):13–30, 1983.

[7] James K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988.

[8] Oussama Khatib. A unified approach for the motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, RA-3(1):43–53, February 1987.

[9] Kathryn W. Lilly. *Efficient Dynamic Simulation of Robotic Mechanisms*. Kluwer Academic Publishers, Norwell, 1993.

[10] Ming C. Lin and John F. Canny. A fast algorithm for incremental distance calculation. In *International Conference on Robotics and Automation*, pages 1008–1014. IEEE, May 1991.

[11] Brian Mirtich and John Canny. Impulse-based dynamic simulation. In K. Goldberg, D. Halperin, J.C. Latombe, and R. Wilson, editors, *The Algorithmic Foundations of Robotics*. A. K. Peters, Boston, MA, 1995. Proceedings from the workshop held in February, 1994.

[12] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, New York, 1995. ACM Press.

[13] M. Overmars. Point location in fat subdivisions. *Information Processing Letters*, 44:261–265, 1992.

[14] Jeff Wiegley, Anil Rao, and Ken Goldberg. Computing a statistical distribution of stable poses for a polyhedron. In *30th Annual Allerton Conference on Communications, Control, and Computing*, 1992.

[15] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *Computer Graphics*, 24(2):11–22, March 1990.