



Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

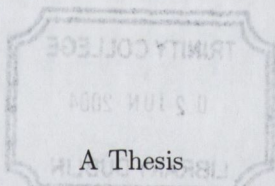
By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

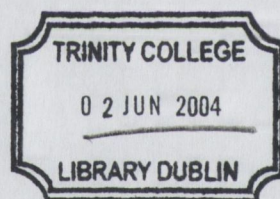
I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Time Adaptive Dynamic Simulation Of Rigid Bodies



Submitted to the Office of Graduate Studies
of
Trinity College Dublin
in Candidacy for the Degree of
Doctor of Philosophy

Thanh Giang
March 2004



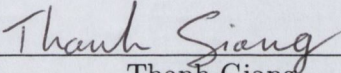
THESIS
7962
1501

© Copyright by Thanh Giang 2004
All Rights Reserved

Declaration

This thesis has not been submitted as an exercise for a degree at any other University.

Except where otherwise stated, the work described herein has been carried out by the author alone. This thesis may be borrowed or copied upon request with the permission of the Librarian, Trinity College, University of Dublin. The copyright belongs jointly to the University of Dublin and Thanh Giang.


Thanh Giang

Everything should be made as simple as possible, but not simpler.

- Albert Einstein

Acknowledgments

My sincerest thanks and gratitude goes to Carol O'Sullivan for all her guidance, faith and constant support throughout the duration of this PhD. I would also like to thank all the members of the ISG, past and present, for making this such a great environment to work in. Especially, a huge thanks must go out to Gareth Bradshaw for his sphere-tree generator and all his invaluable help and suggestions during his post-doctoral stay in the ISG. Without his help and brain storming sessions, there would be no time adaptive system.

I must also thank again, Carol and her mighty red pen, for taking the time to thoroughly proof read this thesis and confirming my beliefs that I am indeed a terrible speller with very bad grammar. Thanks must also be given to Niall and Nicky for finding the time in their busy schedules to further proof read the thesis when they didn't have to. I must also further extend my thanks to Al and Ronan for all their helpful grammar suggestions throughout the writing of this thesis.

To my siblings, Thai, Joanne and Annie, I'd like to say thanks for not forgetting me throughout these last few years when I haven't been around. I promise I'll pop in more often from now on.

Finally, this thesis is dedicated to the memory of my mother, Giang Xem Tien, whose subtle guidance and constant encouragement shall never be forgotten.

Abstract

Traditionally the maintenance of high constant frame rates for real-time performance relied on level of detail simplification, usually performed within the geometry domain. However, simplification can also be achieved for simulation if physical plausibility rather than physical accuracy is acceptable.

In this thesis, we present a novel framework for the production of a time adaptive dynamics system that aims to take advantage of an observer's preconceptions of physical behaviour to produce plausible physically based animations for real-time constant frame rates. We utilise sphere-tree bounding volume hierarchies and the concept of interruptible collision detection in order to allow us to arbitrarily conform to pre-imposed timing constraints in the collision handling phase of the simulation pipeline.

This thesis also addresses the many issues associated with time-critical simulation beyond that of interruptible collision detection. Specifically, we look at the issues involved in implementing an effective time adaptive simulation scheduling policy and consider the consequences of interrupting of collision detection, such as how to obtain good approximate contacting data that does not compromise overall simulation plausibility. The novel idea of *Closest Feature Maps* applied to refinable collision response is introduced to address the issues and problems associated with over approximation of contact information. Closest Feature Maps essentially utilise information for contact data approximation based on the underlying geometry of the colliding objects rather than through properties of the contacting spheres.

We also propose several novel algorithms for updating rigid body motion over time, based on *Verlet* integration, which aim to reduce the overall computation time but yet maintain a reasonably good level of accuracy and stability within the process. With less calculations per frame, these algorithms provide a viable alternative to the more popular Runge-Kutta algorithms for integration of rigid body motion, thus leaving more time for other important and computationally expensive stages of the simulation pipeline.

A metric to evaluate the visual fidelity of physically based animations is presented and the first steps towards applying this metric in a physically based system are described.

Contents

1	Introduction	1
1.1	Motivation	2
1.1.1	The Need For High and Constant Frame Rates	2
1.1.2	Variable Preconceptions Of Reality	3
1.2	Scope	4
1.3	Contributions	6
1.4	Summary of Chapters	7
2	Background and Related Work	9
2.1	Basics of Physically Based Modelling	10
2.1.1	Rigid Body Representation	10
2.1.2	Simulation Considerations	13
2.2	Collision Handling Issues	15
2.3	Approaches to Optimising Collision Handling	20
2.3.1	Broad Phase Approaches	21
2.3.2	Narrow Phase Approaches	23
2.4	Simulation Proxies	27
2.4.1	Time Critical Approaches	29
2.4.2	Other Approaches	31
2.5	Summary	33
3	Integration of Motion	34
3.1	Finite Difference Methods & the Taylor Series	35
3.2	Explicit Euler	37
3.2.1	Accuracy	38
3.2.2	Stability	39

3.3	The Midpoint Method	41
3.4	Runge-Kutta Method	42
3.5	Adaptive Step Sizes	45
3.6	Implicit Methods	46
3.7	Verlet Integration	50
3.7.1	The Basic Verlet Method	52
3.7.2	Leapfrog Verlet	53
3.7.3	Velocity Verlet	55
3.8	Tests and Results	57
3.9	Conclusions	62
4	Verlet for Rigid Body Dynamics	64
4.1	Rotational Properties Of Rigid Bodies	64
4.2	Dealing with Rotational Dynamics	68
4.2.1	Explicit Rotational Leapfrog	70
4.2.2	Fincham's Implicit Rotational Leapfrog	74
4.3	Collision Considerations	79
4.3.1	Adapting the Explicit Rotational Leapfrog	80
4.3.2	Adapting the Implicit Rotational Leapfrog	81
4.4	Hybrid Integration Solutions	85
4.5	Tests and Results	90
4.6	Conclusions	99
5	Time Adaptive Dynamics	101
5.1	Simulation Scheduling	102
5.1.1	Typical Simulation Event Scheduling	102
5.1.2	Time-Critical Scheduling	105
5.2	Progressive Collision Handling	109
5.2.1	Collision Detection with Sphere-trees	109
5.2.2	Traversal and Detection Scheduling	111
5.2.3	Optimising Broad Phase Detection	116
5.2.4	Adaptive Collision Response	117
5.3	Point Reduction	121
5.4	Tunnelling Problems	125
5.5	Further Discussions And Results	130

5.6	Conclusions	133
6	Closest Feature Maps	136
6.1	Approximating Contact information	137
6.1.1	Improved Approximations Using Closest Feature Maps	141
6.2	CFM Construction	142
6.2.1	Polar Sampling	143
6.2.2	Alternative Geodesic Sampling Approach	143
6.3	Determining Closest Feature Information from Sample Points	144
6.4	Efficient Nearest Sample Point Determination	147
6.4.1	Polar Coordinate Searching	147
6.4.2	Geodesic Coordinate Searching Alternative	150
6.5	Discussion and Results	150
6.6	Conclusions	157
7	The Perception of Physical Plausibility	158
7.1	Previous Work	159
7.1.1	Physical Plausibility Simulations	159
7.1.2	Studies In Perception	160
7.2	Recent Studies in Dynamic Distortion Thresholds	163
7.3	A Proposed Perceptual Metric	165
7.4	A Constrained Dynamics Case Study	168
7.5	The Plausibility Of CFMs and Point Reduction	170
7.5.1	Evaluation Of Visual Fidelity Probabilities	174
7.6	Pilot Studies In Time-Critical Plausibility	176
7.7	Conclusions	180
8	Conclusions And Future Work	182
8.1	Summary of Contributions	183
8.2	Future Work	184
A	Resolving Multiple Contacting Collisions	188
A.1	Basic collision impulse assumptions	188
A.2	Simultaneous collision points	191
	Bibliography	196

List of Figures

2.1	The Basic Simulation Pipeline	15
2.2	Examples of the fixed timestep weakness.	16
2.3	An iterative backtracking scheme	18
2.4	2D example of bounds from an AABB tree and an OBB tree	24
2.5	Approximating an object with a k-dop	26
2.6	Example of SSVs for a point, line and rectangle	27
2.7	Example Geometric Level Of Detail Based on Distance Heuristics	29
3.1	Accuracy of Euler's method at two time steps	38
3.2	Motion over time of a spring mass system and instability of Euler's method due to too large a step size	40
3.3	Path taken by the Midpoint method	42
3.4	Derivative steps of Fourth order Runge Kutta method	44
3.5	The basic Verlet integrator	52
3.6	Behaviour of velocity within the Leapfrog Verlet integrator	54
3.7	Velocity Verlet integrator	57
3.8	Mass spring test of accuracy and stability for Euler's method	59
3.9	Mass spring test of accuracy and stability for Midpoint method	59
3.10	Mass spring test of accuracy and stability for 4th Order Runge-Kutta method	60
3.11	Mass spring test of accuracy and stability for basic Verlet method	60
3.12	Mass spring test of accuracy and stability for Leapfrog Verlet method	61
3.13	Mass spring test of accuracy and stability for Velocity Verlet method	61
4.1	Energy dissipation in perfectly elastic collision situation due to using wrong pre collision velocities for collision calculations	80
4.2	Rigid body states representations within the time adaptive system	88

4.3	Rotational Accuracy test of Fincham's implicit rotational integrator algorithm compared to a Midpoint and 4th order Runge-Kutta scheme	91
4.4	Time lapse capture of scene used for accuracy tests	92
4.5	Results of integrator stability test for a three body constraint system. Graph shows max time step for each integration method before loss of stability . .	93
4.6	Time lapse capture of scene used for integrator stability tests	94
4.7	Integrator accuracy results from a dynamically constraint block set to fall at 90 degrees to the downward pointing y-axis with no damping	96
4.8	System integrator timing test for 4th order Runge-Kutta, RKVerlet and Revised Implicit Velocity Rotational Leapfrog methods	97
4.9	Average integrator timings over 30 second period for 1 to 10 objects	98
5.1	Typical simulation event scheduling over the simulation life time	104
5.2	Overview of simulation scheduling protocol allowing for more effective overall simulation timing control	106
5.3	Two timing possibilities during adaptive collision scheduling	108
5.4	Four levels of a sphere-tree representation of a shamrock model	110
5.5	Staircase traversal of sphere-trees adapted for interruptible collision detection	113
5.6	Breadth first collision detection scheduling over time for more than one colliding pair of objects. In this example we have numbered each object pair 1, 2 and 3 respectively.	114
5.7	Axis aligned bounding box as object moves through scene	116
5.8	2D example of dividing a point set based on greatest variance along an axis	122
5.9	Point reduction to four contacting points given a large set of contacts (only one object is shown)	123
5.10	Example of tunnelling problem at different levels of a sphere-tree. As can be seen in (c), the small sphere has almost completely passed through the board.	126
5.11	Unnecessary backtracking	127
5.12	Alleviating tunnelling problem through a lookahead prediction approach. .	128
5.13	Scenario for point reduction performance tests. Red dots are the points of contact considered. Yellow dots are points of contact found but not considered. Lines represent the normals of contact.	132
5.14	A scene and timing results from the time adaptive framework performance test	135

6.1	Approximating collision normal and point of contact between a colliding sphere pair during interruptible collision detection	137
6.2	Cases where we can and can't get away with over approximation of contact data	139
6.3	Closest Feature Map of four sphere-tree levels for Shamrock model	140
6.4	Difference in point distribution for polar and geodesic sampling	142
6.5	Different cases that can be encountered during closest point searching	145
6.6	2D example of voxel traversal for speed up of closest point testing on a polygonal mesh	146
6.7	Determining closest feature information during collisions from polar sampling	148
6.8	Bad θ indexing strategy during polar searching and rectified situation	149
6.9	Alternatively adopted geodesic search scheme	151
6.10	Average timings per frame per sphere for search algorithms	152
6.11	Snapshot of the simulation scene used for the search timing tests	152
6.12	Two example scenes illustrating Closest Feature Map approach rectifying anomalous contact response	156
7.1	Variables manipulated to study the perception of dynamic anomalies.	161
7.2	An ogive function	162
7.3	Angular Distortions. The red and blue lines indicate the trajectories of the centres of the striking and struck balls repectively	164
7.4	Results from psychophysical experiments	165
7.5	Gaussian probability functions for effects observed	166
7.6	Two Scenes from the constrained dynamics framework case study with graphs showing the deviation from normal response to collisions for each scene. . . .	168
7.7	Scenes and experimental results of CFM and point reduction plausibility tests	172
7.8	Varying levels of bounding sphere proxies for box objects used for visual fidelity probability test of CFMs	173
7.9	Results of the visual fidelity probabilities of CFMs in an animation	175
7.10	Results of preliminary tests on plausibility of time-critical approach and physical preconceptions	176
7.11	Scenes used in the plausibility experiments	177
A.1	Contacting points and normal directions during collision between two objects	189

Chapter 1

Introduction

Physically based simulation in computer graphics, at its current stage, can be considered a well studied research area. Traditionally, much attention by researchers has been given to the modelling of physically accurate behaviour for off-line proprietary systems. Inspirational research work conducted by the likes of Baraff [6, 3, 5, 7, 10], Mirtich [108, 110, 111], Lin [100], Gilbert et al [57] and Cameron [29] have paved the way for huge advances in realistic physically based modelling. More recently, with increasing computing power and the development of more robust and efficient solutions for specific problem areas in physically based modelling from such people, it has been possible for such simulations to migrate into the real-time domain.

As an offshoot of this, physically based simulation has made its way into many forms of digital entertainment media. Many computer and video games now incorporate some sort of physically based simulation within the virtual world in order to further enhance player involvement and immersion in the gaming environment. Big Hollywood movie productions all now, in some way or other, incorporate physical simulations into their productions for enhanced movie effects. In response to this growing demand, there are now companies in existence whose sole purpose is to provide fast and reliable solutions for the production of physically based simulations.

However, the performance of real-time systems is still hindered by over-complexity, causing a lower threshold limit to real-time performance than necessary. Fortunately, researchers have come to realise that the industry cannot rely solely upon increasing processing power and the development of improved algorithms dependant on the goal of physical accuracy. Therefore in recent times, the emphasis in physically based simulation research has shifted towards the advancement of better algorithms and models for real-time systems without

the stringent need for modelling *accurate* behaviour.

In this thesis, we present robust and efficient algorithms and solutions for the production of real-time physically plausible, rather than physically accurate, simulations, trading off simulation accuracy for speed whenever necessary, utilising various simulation level of detail techniques (what we refer to as *simulation proxies* in the thesis so as to distinguish between geometric level of detail techniques) in order to achieve this goal. In this way, we provide a guarantee of high and constant frame-rates for scenes whereby in the past, more traditional physical simulation systems may have failed.

1.1 Motivation

The principle concept that we adhere to in this thesis, is the need for plausibility over strict accuracy at all costs for interactive physical simulations. For many applications, the requirement for exact physical correctness (or at least as close to as possible) can only further hinder the achievement of desired real-time frame rates which may play an important role in the overall believability of the simulation. As such, if a reasonably high degree of plausibility is achieved during simulations, then such simulations can be considered physically realistic even if many gross over simplifications have been employed within the process.

1.1.1 The Need For High and Constant Frame Rates

The maintenance of constant real-time frame rates is intrinsically a prerequisite, if plausible motion synthesis is to be adhered to. It has been shown that variable frame rates if allowed to drop below a certain threshold, as well as impacting on a user's perception of plausibility in simulations, can also lead to motion sickness [140, 73].

Typically, real-time simulations conducted on computers usually try to achieve as high a frame rate as possible up to the refresh rate set for the computer monitor. The user will only notice visual and physical disparities in the motion of the simulations once frame rates drop significantly below the refresh rate of the computer monitor. Fortunately, computer simulations are less bound to video standard refresh rates than simulations that are displayed on the television screen. Such simulations are expected to update at a consistent fixed frame rate of 50 Hz (or 60 Hz for NTSC) as the hardware for running such simulations is tied into the refresh rate of the TV. If such simulations cannot keep up with the 50/60 Hz TV refresh rate, even missing the target by a small margin, the simulation will invariably drop to 25/30 frames per second on the next refresh update, with subsequent refreshes

further dropping the frame rate to half of what it currently is at, if simulator operation still cannot finish within the allotted time.

For physical simulation displayed on such devices as the TV screen, it is critically important that all processing be completed within the allotted frame time if physical plausibility is to be maintained to within a reasonably high degree. A level of detail approach must be adopted if we are to adhere to such stringent constraints. This then referring that we must rely upon the domain of plausibility rather than physical accuracy if we are to in any way achieve our desired goals.

1.1.2 Variable Preconceptions Of Reality

Encouragingly for us, past research [37, 127] has shown that, in general, people exhibit quite a high level of acceptance of anomalous behaviour within physical systems. Factors such as velocity, eccentricity, causality, separation and simulation distracters all play a part in the acceptance of plausibility within a dynamically animated scene. This high acceptance quotient can be attributed to the chaotic nature of physical events. In general, due to uncertainty in the world, there may be no single answer to a dynamic event but rather a *cloud* of answers that could be deemed correct. From this, many acceptable plausible outcomes to a dynamic event can be simulated provided that the outcomes fall within the viewer's cloud of predicted perceivable answers.

Within the realms of simulation, if we take a broader view, then we can conclude that there is no such thing as physically correct. Everything within a simulation is a model. In other words, a simplification of the world that tries to mimic as close as possible the chaos of dynamics. Indeed, it is very difficult or else acutely expensive to simulate real world events to an exceptionally high degree of accuracy.

For example, objects in the real world are never perfectly smooth or rigid. At the microscopic level, geometric imperfections are abundant, and during a collision event, no matter how *rigid* an object may be, there will still be slight deformations at the microscopic level. For a simulator to try to account for and simulate these micro-imperfections would require a lot of calculations per simulation cycle (providing of course that the mathematical complexity of modelling such events can first be overcome) and would generally be considered "overkill". Instead, in dynamic simulations, objects within the simulated world are always modelled as perfectly smooth surfaces and, if rigid, always considered to be perfectly rigid. This in itself is a gross but perfectly acceptable over-simplification. In fact, to accurately mimic the real world, we must account for many variables. So many, in fact, that solution

formulations are extremely difficult or, even if achievable, simulator running times become intractable. Instead, all simulations, even those that are classified as exhibiting a high degree of accuracy for critical applications, are merely mathematical models chosen to capture only the interesting and relevant properties of the objects and events being simulated.

In short, the real world never has one solution but many possible solutions to a physical event due to the chaotic nature of physics and the imperfections of objects. As such, all simulations ignore the parts of reality that are difficult or too expensive to model but are merely a demonstration of the consequences of our assumptions.

From the above discussion it can be concluded that, in general, physical simulations can strive to be too perfect when there is no need to be. People in general can be considered imperfect observers of reality and our *gullibility* and preconceptions of reality can be exploited in order to further simplify physical modelling and to achieve desired frame rates within interactive simulations. Indeed, some researchers have already questioned the idea of striving for physical accuracy but rather have adopted the idea of physical plausibility instead.

1.2 Scope

Much level of detail work has been concentrated in the geometric simplification domain [102]. However, there is also a lot to be gained through the research of simplification within the physical animation domain. The benefits of plausible rather than physically correct simulation have been, in the past, advocated by such researchers as Barzel et al [14]. Given that physical plausibility rather than physical accuracy is sought, there are many opportunities for simplification within physically based animations, which can be exploited for purposes of simulation speedup and efficiency.

Within the physically based animation domain, the area of collision handling¹ is one that can be easily taken advantage of and greatly optimised in terms of the exploitation of an observer's preconceptions of physical uncertainty. Physically based animations, in general, are greatly hindered by the bottleneck caused within the collision detection phase in the simulation pipeline. As a result, quite a bit of research conducted in the area of physically based simulation is aimed at deriving faster and more efficient methods of finding potential collisions during a simulation period. However, most of this research has been conducted with the goal of finding collision information based on the requirement of exact level of

¹In this thesis we refer to the term "collision handling" as incorporating both collision detection and resolution.

contact information for collision resolution. In this thesis, we argue that such information is not wholly necessary in order to convince the observer, especially if we are striving for plausible rather than accurate results during physical simulations. Many additional speedup procedures can be adopted providing that approximate contact information, rather than exact contact information, is appropriate for collision resolution.

Building on this belief, we adopt the idea of *interruptible collision detection* using sphere-tree bounding volume hierarchies, as first introduced by Hubbard [76], for a time-critical approach to collision detection. We also further build upon the work of Dingliana and O'Sullivan [48] for refinable collision handling (i.e. interruptible collision detection and the approximation of appropriate refinable response data). Specifically, we build upon the concept of refinable collision handling and the issues that arise when incorporating these ideas into a fully viable time adaptive dynamics framework. The aim of such a framework is to allow maintenance of high and constant real-time frame rates for physically based animations, by exploiting observer uncertainty and preconceptions of dynamic events without compromising overall physical plausibility.

In order to necessitate real-time constant frame rates during a dynamic simulation, a mechanism is needed to ensure that we do not violate the timing constraints imposed upon the system. This requires a system with strict time management protocols that also do not overtly hinder the plausibility of the end animation. The notion of interruptible collision detection allows us to arbitrarily stop collision detection processing to conform to timing restrictions. Collision data can then be arbitrarily approximated based on the level at which detection was interrupted, with collision approximation getting progressively more refined with increasing available time. For this to happen, a dynamics system are made time adaptive, such that all stages within the simulation pipeline must be efficiently managed for conformity to imposed system timing constraints. Systems that use interruptible collision detection and refinable collision modelling have been presented but, to date, such a fully time adaptive dynamics system has not been implemented. Many challenges must be faced if such a system is to be realised and this thesis presents solutions to many of these problems.

The area of refinable collision handling can be considered a relatively young area of research, with the focus to date being on simple cases of instantaneous collisions for objects freely moving towards each other in space. There are many issues that need to be considered for the incorporation of such schemas into a viable real-time dynamics system, that extend further than the simple case of objects in free flight. In particular:

1. how over-approximation of contact data can impact on cases of continuous contact,

2. how we are to deal with over-abundance of contacting data during collision detection due to the nature of the sphere-tree bounding volume hierarchy,
3. how to appropriately incorporate collision response processing to conform to timing constraints.

All these potential problems and many others must be addressed if a viable time adaptive dynamics system is to be made possible.

1.3 Contributions

We offer the following contributions in the work presented in this thesis:

- In order to further speed up updating of object motion so that more time can be allotted for other potentially more computationally expensive tasks during a simulation frame, the **Verlet family of integrators is investigated (pages 57-62)**. These integrators are traditionally designed to account for only motion in the linear domain. In this thesis, we present several algorithms that attempt to **adapt these integrators for use with rigid bodies (pages 85-90)**, that undergo both rotational and linear motion.
- To date, work on time-critical collision processing has not achieved incorporation of these principles into a full system. Scheduling strategies based on simple heuristics have been presented or else, work has been concentrated within the collision handling domain itself. We aim to address this issue by presenting a **fully comprehensive simulation scheduling schema (pages 105-109)** that adapts to demands for constant frame rate requirements within a time adaptive framework.
- The use of sphere-tree bounding volume hierarchies for time-critical collision detection can potentially produce an over-abundance of contact data, further hindering rather than helping real-time system performance during collision resolution. To date, this problem has been an open issue. We present in this thesis a **point reduction scheme (page 121-125)** for dealing with such cases to allow for better and more efficient collision resolution. Furthermore, we **incorporate collision response processing into the time-critical domain by making it time adaptive (pages 199-200)**. This is achieved by determining and limiting the number of contact points that the collision resolver can maintain within the time allotted to it. In general, there is

usually an over-abundance of contact data available to the collision resolver after the collision detection phase, much of which can be considered unnecessary information and can be discarded without affecting overall system integrity.

- The novel concept of *Closest Feature Maps* (CFMs) for better approximation of contact data for collision resolution (pages 141-150) with sphere-tree bounding volume hierarchies is introduced. CFMs serve to solve the problem of over-approximation of contacting data for cases of continuous contact, which could potentially lead to anomalous behaviour that would ultimately impact on the overall plausibility of the physical simulation. This is achieved by approximating contact information based on the underlying polygon level (i.e. the geometry of the object itself) rather than on the geometry of the contacting spheres as is the case with previous schemes.
- Given the importance of having good perceptual metrics in order to appropriately quantify the quality of plausible physical simulations, we contribute towards the first steps in using a viable perceptual metric (pages 168-171, 174-176) for the purposes of measuring the visual fidelity of physically based animations. We also offer a preliminary study into the plausibility of time-critical animations (pages 176-180).

1.4 Summary of Chapters

The remainder of this thesis is divided up into the following chapters:

Chapter 2 gives a detailed overview of related work conducted in the area of physically based simulation and previous level of detail simplification research in the animation domain.

Chapter 3 will provide a thorough discussion of the various common integration algorithms that are used for the advancement of rigid body states over time in physically-based simulations. This chapter also introduces the *Verlet* family of integrators, which can be considered as an additional speed up alternative over the traditional integrators for our proposed time adaptive system. Results from performance tests are provided in order to demonstrate the advantage that these integrators have over the commonly used integrators in physically based simulations.

Chapter 4 provides a study of existing methods that try to extend Verlet integrators into the rotational domain and also provides several novel algorithms that further improve upon these existing methods in order to better incorporate Verlet integration for full rigid body dynamics. Timing, accuracy and stability tests are provided at the end of this chapter in order to justify the performance of these new algorithms.

Chapter 5 presents the many issues that need to be considered within a full time adaptive dynamics system. In particular, we present these ideas within a novel viable time adaptive framework, addressing possible problems that may arise from using a sphere-tree hierarchy for such an approach. We expand upon the idea of interruptible collision detection and describe the traversal strategy employed for efficient collision scheduling. Also, the problems of comprehensive simulation scheduling, how to deal with over-abundance of point distribution during collision detection and the adaptation of collision response into a time adaptive module are all presented in this chapter. Finally, we demonstrate the advantages of employing interruptible collision detection for achieving target constant frame rates by presenting performance evaluations for time-critical collision detection with sphere-trees versus non-time-critical hierarchical bounding volume methods.

Chapter 6 investigates the suitability of current approximation schemes for obtaining approximate collision data for collision resolution and introduces the novel concept of *Closest Feature Maps* (CFMs) as a solution to the limitations of such schemes. The various strategies employed for creating and using CFMs is also discussed in this chapter. Various performance tests are also offered at the end of the chapter.

Chapter 7 examines at the concept of plausibility for physically based simulations and contributes towards this goal by implementing a viable perceptual metric for the evaluation of visual fidelity within physically based animations.

Chapter 8 concludes the discussion of the work presented in this thesis with a summary of contributions and suggestions for future improvements and possible directions for future research.

Chapter 2

Background and Related Work

Much work over the last few decades has been conducted in the area of physically based simulation, with early work concentrating on offline techniques for the production of realistic physical animations. Researchers such as Baraff [3, 4, 5, 6, 7, 8, 9] and Mirtich [112, 111, 106, 108, 107, 109, 110] have produced much excellent work addressing the major issues involved when attempting to simulate rigid body dynamics. With the increase in computing power over recent years coupled with the excellent research work conducted by these researchers and others in the field, we now have sufficient means of producing quite complex physical simulations within acceptable real-time bounds. However, there are still many issues to be resolved within physically based simulations as we shall see.

This chapter aims to provide a thorough study of the background and related work undertaken in the area of physically based modelling over recent years, focusing specifically on the research that relates to the topics that will be later discussed within this thesis. We first discuss the underlying principles of physically based modelling of rigid bodies as this provides the fundamental foundations for most of the later work. We shall examine the issues concerning discontinuities within a physically based system specifically referring to collisions. A thorough investigation to the many various techniques that are currently being utilised to resolve collisions shall be conducted. Finally a thorough discussion of the various level of detail techniques that have been employed to further improve upon simulation frame rates shall be presented.

Linear States	State Symbol	Angular States	State Symbol
<i>Linear velocity</i>	v	<i>Angular velocity</i>	ω
<i>Force</i>	F	<i>Torque</i>	τ
<i>Mass</i>	M	<i>Inertia Tensor</i>	I
<i>Linear momentum</i>	P	<i>Angular momentum</i>	L
<i>Center of mass</i>	x	<i>Orientation</i>	q/R

Table 2.1: Typical states information associated with a rigid body

2.1 Basics of Physically Based Modelling

Most of the recent research in physically based simulation has focused on the problems associated with rigid body dynamics as opposed to deformable body dynamics, which in itself is a much more complex problem area. In this thesis we too only concentrate on rigid body dynamics and leave soft body dynamics as an extension for future research.

The laws governing rigid body dynamics are well known and are described in many classical mechanics text books [61, 46]. We shall not delve too deeply into the texts of classical mechanics here but rather only present the fundamentals for rigid body representation and physical motion generation under Newton’s laws in physical simulations.

2.1.1 Rigid Body Representation

Goldstein [61] formally defines a rigid body as a “*system of particles in which the distances are fixed and cannot vary with time*”. In molecular physics, chemical bonds are most popularly viewed and modelled under this assumption. Rather than modelling a rigid body as a set of kinematically constrained particles in space, a better and more appropriate view of rigid bodies, and one that is adopted in computer graphics research, is that a rigid body may be represented in 3d space by a set of state variables with its position and orientation defined by its centre of mass and a rotation matrix respectively. The state variables associated with a rigid body not only describe positional information but also provide other information pertinent to the rigid body such as the mass, velocity and forces acting on the rigid body at that instant in time. The state variables, for clarity, can be further separated into two distinct categories, a linear component and an angular component.

Table 2.1 shows the typical states information associated with a rigid body along with the associated state symbols that we shall be adopting in this thesis whenever describing these states mathematically. The orientation of a rigid body as mentioned previously can be described by a rotation matrix R but due to the problem of numerical drift over long

	Linear Equations	Angular Equations
Velocity	$v = \frac{P}{M}$	$\omega = I^{-1}L$
Force/Torque	$F = Ma$	$\tau = (p - x) \times F$
Momentum	$P = Mv$	$L = I\omega$
Acceleration	$a = \frac{F}{M}$	$\dot{\omega} = I^{-1}(L \times \omega + \dot{L})$

Table 2.2: Rigid body equations of motion

simulation time periods it is common practice to represent the orientation as a quaternion q instead. We can trivially transform back to a rotation matrix R from a quaternion q whenever it is necessary to do so without any loss of generality. In general, it is not necessary to store the acceleration a of a rigid body as a state variable as this can be quite easily calculated from the known forces acting on the body at the time using the equation of motion from Newton’s second law (i.e. $F = Ma$). For completeness and general interest we present here Newton’s laws:

Law 1 A body will remain at rest or stay in uniform motion unless acted upon by an external force.

Law 2 The acceleration of a body is dependant on the force applied upon that body and its mass (i.e. $a = \frac{F}{M}$ or written in the more popular form $F = Ma$).

Law 3 For every action there is an equal and opposite reaction.

All objects within a physically based simulation must obey these laws in some way or another.

All other state terms associated with a rigid body as presented in Table 2.1 are trivial and if needed, can be easily derived from the equations of motion which can be easily obtained from any good classical mechanics text. We shall not discuss the derivation of these equations here, however, for completeness, we provide these rigid body equations of motion in Table 2.2. From Table 2.2, the p variable in the torque τ equation refers to any point on the body in which the force F is acting on. The derivative term of the angular momentum (i.e. $\dot{L}$) in the angular acceleration $\dot{\omega}$ equation is equivalent to the net torque acting on the body and can be substituted as such. This angular acceleration equation itself is unfortunately not as straight forward as its linear counterpart, however we can show it to be derived from taking the derivative of the angular velocity ω . All other terms in Table 2.2 should be fairly self explanatory.

Perhaps the only state term which may require more discussion is the *inertia tensor* term I . The inertia tensor, represented by a 3x3 matrix, can be considered as the rotational analog of the body's mass. It can be more generally viewed as a description of a rigid body's mass distribution. This dictates the overall rotational behavior of a rigid body under the influence of forces and torques over time. It is usual practice to represent the inertia tensor of a rigid body in terms of the body's local coordinate frame within the state descriptors. This is so that we do not have to recalculate the inertia tensor term in world space for every simulation time frame as the world space inertia tensor term will change as the body moves through space, which would be a very computationally expensive task. By representing the inertia tensor in local space coordinates within the state description, we can treat such a term as constant throughout all simulation time steps thus avoiding needless re-computation. The translation of the inertia tensor to the actual world space coordinate frame whenever needed can be done through a simple calculation involving the orientation of the rigid body:

$$I_{world} = R I_{body} R^T \quad (2.1)$$

The mass and the body space inertia tensor of the rigid body only need to be precalculated during system startup and are constant throughout the simulation.

For simplicity, it is usually assumed when calculating rigid body inertia tensors that a rigid body has a constant unit density. The calculation of a rigid body's inertia tensor then involves the integration over the volume of the body. However, this may not be very intuitive, especially for complex objects. In approximating the body space inertia tensor in the time adaptive system discussed in this thesis we use an approach developed by Mirtich [107]. This approach uses two passes of *Green's theorem* to reduce the overall volume integral complexity. For polygons in a plane, Green's theorem reduces the planar integration to a sum of line integrals over the edges of the polygons. Mirtich takes this theorem and applies it twice so that a volume integral is reduced to a set of line integrals in a plane while the values from these integrations are then propagated back into the values of the desired volume integration to eventually give the body's inertia tensor. This approach has proved to be simple and fast and has the added advantage that volume information is not required.

2.1.2 Simulation Considerations

During a simulation cycle, rigid body motion is produced through the integration of the continuous state changes of the rigid body over time based on Newton's equations of motion. Unfortunately, due to the discrete nature of computer clock cycles and floating point arithmetic, we can only approximate the true path of a rigid body at discrete time steps over the entire simulation time period. There are many different methods for integrating Newton's equations of motion over time, each one with their own advantages and disadvantages. The most popularly used methods within the computer graphics research community are the *Runge-Kutta* methods. These methods make an approximation of a body's true path over the simulation time interval by taking several successive sub-steps forward in time so as to give a better overall approximation of the state change over that time interval. More details of commonly used integration methods along with an emerging class of integration methods fast becoming quite popular within the graphics community known as *Verlet integrators* shall be discussed in Chapter 3. Although Verlet methods were designed to deal with particle dynamics, we present several possible solutions to expanding these methods to deal with rigid body dynamics in Chapter 4. Unfortunately, discretising time in this manner can potentially cause problems for the simulator. For collisions, since we are only looking at the simulation at discrete time intervals, it is possible that we may over-step and thus miss potential collisions or have very bad interpenetrations between dynamic objects in our scene. We shall discuss this further in the next section.

Rigid body motion without environmental interaction is not very interesting. However, things begin to get predominantly more problematic to simulate and interesting once user interactivity and environmental interaction and awareness is considered. When two or more rigid bodies travel through space in such a way that at some point their paths will cross, we do not expect them to "ghost" through each other but rather have some reasonable expectations about the resulting physical outcome. Similarly, if a rigid body were to fall onto level ground, we would expect it to come to an eventual state of rest after a few bounces. In a fully dynamic system, we must take into account the various behaviours of each rigid body within the system due to unexpected external events that may perturb the natural state of the system such as collisions and external user interaction. Even more so, we must have a mechanism in place that enables us to detect when such events have occurred and to deal with them as appropriate. We can therefore view physically based modelling as three distinct phases:

1. Representation and simulation of the physical attributes of the dynamic body over time: We must have a representation of the physical states of the objects within our system and a way to update the appropriate state variables over time such that the simulation outputs appropriate motion trajectories.
2. The second phase of the simulation loop is known under many names; “collision detection”, “clash detection”, “interference detection” being some of the more popular names. Environmental awareness of the simulated bodies to their system surroundings is important and a necessity of a complete dynamics simulator. This is more of a kinematics issue rather than a dynamics issue. However, it is extremely important that our rigid bodies be aware of their surroundings so that our dynamics system may take appropriate action in the event of any perturbations to the normal motion trajectory. It is this stage that causes most bottlenecks in dynamics systems and hence much work has been expended on finding efficient and accurate algorithms to solve such problems.
3. The determination and calculation of physically correct responses to interactions with external influencers on the dynamic bodies must lastly be considered. Once we have determined that the bodies in our simulation have been perturbed from their normal paths in some way we must be able to acquire the correct information so as to enable calculation of an appropriate response to account for the situation. Only when the previous phase reports a disturbance to the normal run of the simulation will this phase be invoked. Rigid body states should then be updated to account for any such changes so that during the next iteration through the simulation loop the appropriate rigid body dynamic behaviour is observed.

Figure 2.1 illustrates how the above phases slot into the overall simulation pipeline.

Collision detection is widely reported to account for a large amount of computation time in dynamic rigid body simulations [108]. For this reason, much research has been conducted in developing quicker and more efficient methods for performing collision detection. As this too constitutes a majority of the work and our main contributions in this thesis, we shall next review related research work conducted in collision handling and discuss the main issues involved.

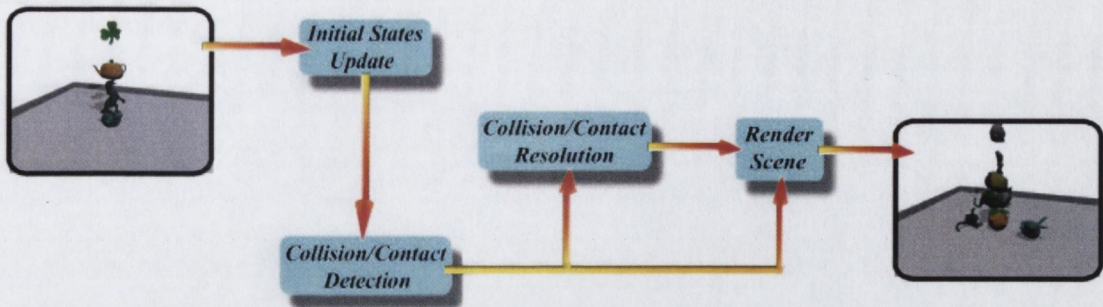


Figure 2.1: The Basic Simulation Pipeline

2.2 Collision Handling Issues

Most collision detection schemes for physically based simulation are performed up until the exact polygon level. It is not sufficient to know if two bodies are colliding or about to collide, as in robotics or automation, but rather it is a necessity that exact surface contact information is also known in order to resolve collisions. Exact collision algorithms aim to resolve collisions to the exact underlying mesh level so that such information is trivially obtained. Bounding Volume Hierarchies (BVHs) can be utilised to hone in on specific areas of interest so that exact collision detection may be performed more efficiently. It is at this stage that more precise and expensive techniques such as triangle to triangle intersection testing is employed. A good survey on various exact collision detection approaches can be found in Jiménez et al's [82] paper.

Other non BVH approaches in collision detection aim to either track closest features through the efficient use of *Voronoi* regions [100, 110] or to use sets of simplices for fast distance queries between potential colliding objects [57, 130, 29]. These schemes unfortunately do not handle non convex objects well. Instead, it is attributed that non convex objects be divided into subsets of convex parts for collision detection. We however do not consider such approaches in this thesis but prefer to concentrate on the potential simulation advantages in which BVHs may provide us.

Hahn [68], who conducted one of the earliest studies into full rigid body dynamics simulation, showed that collision detection took up most of the processing time during a typical simulation loop. Mirtich [108] states that collision detection accounts for over 95% of the computation time in a typical simulation loop. For one or two objects, real-time performance can still be achieved, so this is not too much of a problem. However, as scene complexity increases and the number of objects to be simulated and/or the complexity of

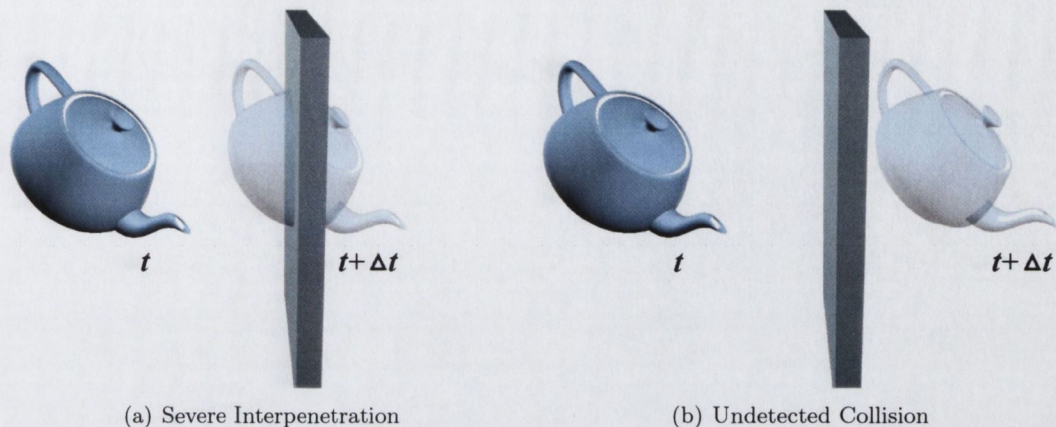


Figure 2.2: Examples of the fixed timestep weakness.

each object in the scene increases, collision detection increasingly becomes a bottleneck that will hinder the overall performance of a system. In this thesis we are primarily concerned with the production of real-time simulations of physically plausible events. As such, we wish to maintain as constant a frame rate as possible within real-time bounds. A reasonably high frame rate along with some degree of frame rate constancy is an important factor for the overall effectiveness of interactive applications and also in the reduction of a phenomenon that Hettinger and Riccio [73] and Uliano et al [140] refer to as *simulator sickness*, or otherwise known as *cybersickness* and *visually induced motion sickness*. The symptoms of this phenomenon are similar to that experienced in motion sickness and evidence suggests that it is attributed to low frame rates and high latency within a simulation. For off-line applications, the collision bottleneck may not concern us too much but if we wish to achieve reasonably consistent real-time performance we must aim to reduce the overall bottleneck caused by collision detection within the system.

Over the last few decades much work has been done in collision detection. Early collision detection work was mainly conducted for such applications as robot path planning and automation [26, 28, 20, 27, 129]. In this situation, it was merely sufficient to know if a collision was about to occur or not so all the collision resolver had to do was to return a yes or no answer. For graphical simulation of dynamic events, it is not enough to simply know if a collision has occurred or not but rather it is also a requirement that, given the detection of a collision event, we have sufficient means of returning useful information to the system about the collision so that we may deal with such events in an appropriately dynamic way. Information such as which objects have collided, points of contact, surface normals at

contact and surface material coefficients must all be made known to the collision resolver during a collision event in a dynamics system so that proper responses may be calculated. It is a requirement of a good real-time system to not only be able to detect a potential collision quickly but also to be able to return sufficient information from such collisions in a quick and efficient manner so that the system may deal with them as appropriate.

Hubbard [76] identifies three problems associated with achieving consistent high frame rates in an interactive dynamics systems. He classifies these problems as :

Fixed Time Step Weakness: This problem is the inability of an algorithm to deal with potential collisions due to the discrete nature of time steps during a simulation loop. As we only sample the objects' positions in space at discrete instants in time, objects travelling at high speeds or with sufficiently large time steps can potentially have their collisions go undetected or detected at too late a stage. This can result in severely interpenetrating objects and thus adversely affect the simulation (Figure 2.2).

All-pairs Weakness: This arises from the need of an algorithm to perform collision tests on all pairs of objects in the simulation. For n objects to be simulated in a scene, this constitutes a cost of $O(n^2)$ tests to be conducted. This is an extremely expensive process and particularly so as the number of objects to be simulated increases.

Pair-processing weakness: This refers to the potential weaknesses of algorithms at coping with certain situations during collision tests. For example, the original Lin-Canny closest feature algorithm [100] cannot handle interpenetrations and will go into a loop if such a situation occurs. Later implementations resolved this problem [124, 110].

There have been many solutions proposed to tackle the fixed time step weakness. The naïve solution would be to take smaller step sizes, but we can only take steps of decreasing magnitude to a certain point before the simulation becomes too slow for real-time performance. For real-time applications it is desirable to be able to take as large a time step as possible. Also, taking smaller steps does not necessarily guarantee that objects will not interpenetrate or miss each other. Another naïve solution to resolving cases of interpenetration is not to try to resolve the time of collision, but rather to simply kinematically transform the object back to the point where interpenetration does not occur. This does not, however, account for simulation states at collision time and may adversely affect the energy conservation in the system, causing objects to bounce or spin off with more energy than they should. Mirtich [111] categorises algorithms for coping with the fixed time-step weakness

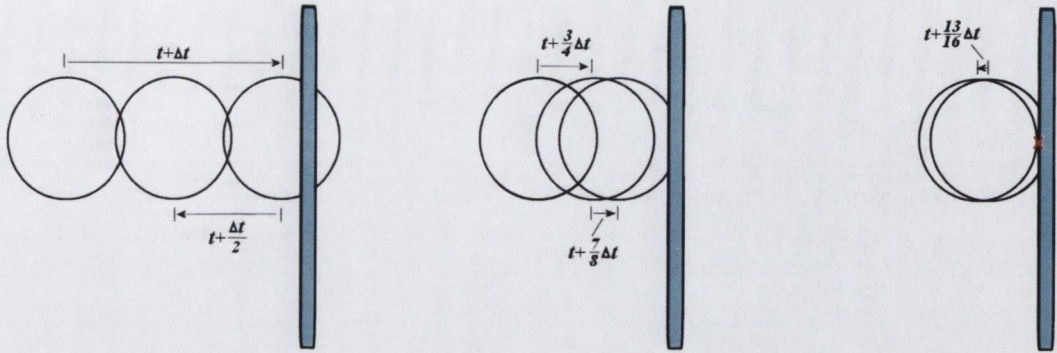


Figure 2.3: An iterative backtracking scheme

problem into two common approaches, called *Retroactive Detection*(RD) and *Conservative Advancement*(CA). Retroactive Detection refers to back stepping the simulation time to the time of first contact and then continuing the simulator at that time, dealing with collisions as appropriate. This strategy is not very feasible for large systems as all objects must be stepped back to time of first contact. It is also quite computationally expensive, as determining the point of first contact is not a trivial process. Conservative Advancement refers to strategies that try to predict times of collisions by finding lower bounds on the time between the previous simulation frame and the time of first contact. In effect, the simulation “creeps up” to the time of first contact. Such approaches are generally more robust than RA schemes as the chances of discovering possible missed collisions are reduced due to the (semi-)continuous nature of the algorithms. However, CA schemes tend to be excessively expensive.

In terms of RA techniques, Baraff [6] suggested an iterative backtracking approach to finding the first time of contact given detection of object interpenetration. In his approach, he iteratively takes an estimated time step t_e backwards and forwards in time until a valid threshold distance is obtained. Each time estimate is taken to be the half step bisection of the previous time step taken (see Figure 2.3). While this approach is conceptually easy to understand and implement it does not handle systems with many objects very well. To address the problems with dealing with large numbers of objects, Mirtich [111] presents a backtracking approach based on Jefferson’s [81] *Timewarp* algorithm. With this approach, objects are considered to be in different dependency groups which are backtracked within their own virtual time and collisions resolved, rather than affecting all objects within the system by augmenting the system time as a whole. The different times are reintegrated at intervals of *global virtual time* before any global events are processed. Although the

algorithm accounts for large amounts of objects in the simulation, large groups of objects can still cause the algorithm to be quite slow. For an avalanche simulation consisting of 250 convex objects running on a SGI Onyx, Mirtich found that it took 97 seconds on average to compute each frame.

In the category of CA methods, Hubbard [76] uses a 4D Conservative Advancement approach to tackling the fixed time-step weakness. He proposes a 4D occupancy bound known as a *space time bounds* to tackle the problem, where both time and space are bounded for an object in the simulation. Hubbard models these bounds as parabolic horns and hyper-trapezoids. The time steps of the simulation are adjusted depending on whether a potential collision has been detected through testing for intersections of the bounds. Cameron [28] also utilises a similar 4D approach of using both time and space bounds for collision detection. Another similar approach in tackling the fixed time step weakness problem was also used by Mirtich [108]. However, Mirtich's approach computes bounding boxes for the swept volumes of the objects in the simulation over a time interval. Overlap testing utilising a hierarchical hash table is then performed to determine any potential collisions. In a slightly different approach, Redon et al [132, 133] presented a scheme that utilised integration of *interval arithmetic*¹ to effectively compute the collision times between features of rigid polyhedral bodies in a scene. With such an approach, the computation of the first time of contact is inherently part of the collision detection algorithm such that the time of first contact is computed during the collision detection. Although Redon himself states in his paper that "*continuous methods are usually slower than discrete methods*", Redon's approach does seem quite interesting and merits further investigation. Unfortunately, due to timing constraints, in this thesis we leave this for future work.

The all-pairs weakness has been tackled by researchers by adopting, what Kitamura et al [87] refer to as, *hybrid collision detection* methods. This is a multi-phase approach to the collision detection problem that involves two distinct phases known as the *broad phase* and the *narrow phase*. The broad phase works by reducing the number of potential colliding pairs to be processed by performing an initial pass over the scene to be simulated and culling out those objects that are deemed to be definitely not colliding. The narrow phase follows from broad phase and further refines the remaining potentially colliding objects using either boundary or spatial localisation techniques to narrow in on the interesting collision regions, eventually performing exact polygonal level tests if necessary. The narrow phase can be refined to have another sub-level that will determine the exact contact information such

¹The basic idea behind interval arithmetic is that instead of computing with numbers we compute with intervals instead.

as point of contact and surface contact normal of the collision. Most researchers consider the narrow phase and this *exact phase* as one. However, some researchers view the narrow phase as two distinct levels. O'Sullivan [118] refers to the two distinct levels of the narrow phase collision detection as the *progressive refinement* and *exact* levels respectively. For the purposes of this thesis, we too view the narrow phase as containing two levels. We take advantage of the progressive refinement level to aid us in achieving a good time adaptive approach to producing relatively constant real-time frame rates in dynamic simulations, as demonstrated later. We shall discuss the different approaches adopted by researchers in implementing both broad and narrow phase algorithms in more detail in the next section.

2.3 Approaches to Optimising Collision Handling

Objects within a dynamic simulation can be modelled using a myriad of methods: implicit surfaces, polygonal meshes, parametric surfaces and constructive solid geometry. Algorithms for dealing with collision events vary depending on model representation and query type. Lin and Gottschalk [101] present a survey of different geometric model representations and discuss some ideas for collision testing of such representations. The most usual approach taken to model rigid bodies is to represent them as polygonal models and there are many methods available that deal with efficient collision detection between such models. As model complexity increases, so does the computational time of the collision algorithm. It is computationally expensive to test for every polygon on a rigid body model against all those of another corresponding model, especially if the polygon count is moderately high. To do so would exacerbate the all-pairs weakness problem. It is more practical to treat objects to be displayed within the scene for the purposes of collision detection by using a courser proxy representation. The chosen proxy objects must be quick and easy to simulate and approximate the object to be rendered as close as possible.

It is usual that we simplify the object for processing either using boundary or spatial occupancy representations. Boundary representations use simplified objects such as convex hulls to bound the object and collision tests are then conducted on the hulls. Spatial occupancy representations utilise bounding volumes to give a coarser approximation of the object for collision testing. In either case, if a collision is detected then the collision algorithm should be able to use the information gathered from the proxy objects to hone in on the desired collision manifold² and conduct more precise collision tests if needed to

²The spatial region of interest between two objects during a collision

determine the required collision information for processing purposes. The stage of collision detection that uses proxy objects belongs to the afore mentioned narrow phase. However, before we can proceed to detect collisions in this manner it is advantageous to cull all non-colliding object pairs within our scene for faster and more efficient collision testing. This is done in what is commonly referred to as the broad phase collision detection stage. We shall next examine the more common algorithms and proxy representations adopted within each of these phases.

2.3.1 Broad Phase Approaches

As previously mentioned, the broad phase of collision detection aims to optimise collision handling by quickly disqualifying all impossible colliding pairs of objects from the current collision detection pipeline. The usual basic idea behind a broad phase algorithm is to encapsulate the objects in the scene within some geometrically simple spatial bound whereby very quick yes-no collision tests may be performed. Any object pairs that are found not to be potentially colliding are thus discarded. The usual choice of bounds are bounding boxes or bounding spheres as these are geometrically very simple and collision tests may be performed on such geometric objects very quickly. We have already come across some broad phase algorithms in our previous discussions in this chapter, such as the 4D space-time bounds idea utilised by Hubbard [76] and Cameron [28] and Mirtich's [108] approach of using a bounding box over an object's volume swept through time.

Baraff [6] introduced an algorithm referred to as *Sweep and Sort* to determine potentially colliding pairs. In this algorithm he suggests using axis aligned bounding boxes to bound the objects within the scene. The interval, $[b, e]$ of each bounding box is then stored in an ordered list whereby overlap tests can be quickly conducted by a quick sweep through the sorted list. A separate active list is maintained for overlap testing. Whenever a value b_i is encountered in the ordered list, all intervals currently in the active list are reported as overlapping with that interval i . That interval b_i is then subsequently put onto the active list. Upon encountering a value e_i , the corresponding interval b_i is then removed from the active list. The time complexity of such an algorithm is given as $O(n \log n + k)$ whereby n is the number of objects in the scene and k is the number of pairwise overlaps. Baraff further improves upon this algorithm by exploiting temporal coherence and sorting the almost sorted interval list between frames using insertion sort. By doing so the cost of the algorithm is improved to $O(n + s)$ whereby s is the number of swaps during the insertion sort. To our knowledge, Baraff was one of the first researchers to exploit temporal

coherence between simulation frames to speed up collision testing. Cohen et al [38] utilises an almost identical approach in their I-COLLIDE collision system whereby they called the algorithm *Sweep and Prune*. Again, temporal coherence was exploited to reduce the overall algorithm complexity to give near constant time performance of $O(n + s)$. The Sweep and Prune method seems to be by and large the most popular method for broad phase collision detection.

Some other broad phase collision detection approaches employ an overlap table to determine potentially colliding objects in a scene. An example of such a scheme is described by Palmer and Grimsdale [123], who utilised a *global bounding volume table* to determine possible colliding objects. Bounding spheres were used for fast intersection testing. Similarly, Wilson et al [143] utilised a precomputed *overlap graph* to represent the proximity of objects within a scene. Collision determination was then conducted while traversing the graph. The method was used for scenes in which proximity queries on massive models had to be dealt with.

Dworkin and Zeltzer [49] have proposed an event based approach to allow for culling of the time that is needed for checking of collisions between successive steps during a simulation. The basic idea of this approach is to try and foresee future events so that the simulation time at which objects are likely to meet can be determined and so thus allowing us to ignore all times in between, before such an event is to happen. Only at a specific event times do we need to detect any potential collisions. All event times are maintained in a list. A bounding sphere is used as a rough bounding volume to be detected for an event and at the simplest level the linear path at a constant velocity is presumed when predicting object trajectories over time. The scheme was designed for optimising collision processing in sparse dynamic scenes and thus loses efficiency when dynamic scenes are densely populated.

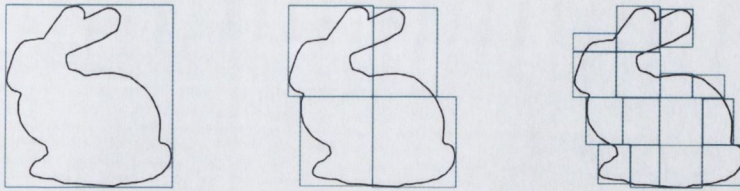
It is also common to use spatial partitioning strategies to further aid in the speedup of collision detection in the broad phase. Such schemes hail from ray tracing text [59] whereby spatial subdivision schemes are used to speed up ray-object intersection tests. Such schemes have also been incorporated into narrow phase collision detection as well. The basic idea of such strategies is to divide up space into regular grids or *voxels* (i.e. volume elements) whereby only objects that are found to reside in the same grid are checked for collisions. Probability of collisions can be further refined by refining the occupancy of objects inside a grid through various further subdivision strategies (e.g. quadtree or octree subdivision [135]). Mirtich [108] uses this approach in his broad phase collision detection but however further optimises the approach by utilising a hierarchical hash table to describe

object occupancy inside various space grids. This hashing approach reduces the overhead of storing cell occupancy information. The problem with using the space grids approach for broad phase collision detection is in choosing a good grid resolution so as not to have an object occupy too many grids and also not to have a grid so large that it bounds many objects that are not in contact.

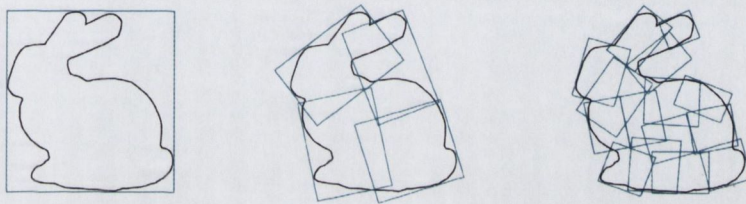
2.3.2 Narrow Phase Approaches

Given that a potential contact between a pair of objects has been determined by the broad phase algorithm, in order to have useful information that the collision resolver can work with, a spatial localisation of the potential collision manifold must be performed immediately afterwards. This is the main purpose of the narrow phase collision detection schemes. As was mentioned earlier, the narrow phase can be further viewed as two separate sub-stages; a *progressive refinement* level to hone in on the area of interest and an *exact* stage that determines at the polygon level important collision information such as points of contact and contact normal information. Most researchers do not distinguish between the two sub stages in the narrow phase in this way but rather view them as one step encapsulated within the narrow phase. In this thesis the two sub stage view of narrow phase collision detection is taken.

It is usual to employ spatial bounding volumes within a hierarchical structure, broadly known as *Bounding Volume Hierarchies* (BVHs), to obtain an efficient proxy representation of an object's spatial occupancy during narrow phase collision detection. BVHs usually consist of simple geometric primitives, such as boxes, spheres and cylinders arranged in a tree-like structure so that quick intersection tests may be conducted at each level of the hierarchy and irrelevant areas of the object culled away in a quick and efficient manner. BVHs allow for a progressively finer refinement of the area of interest of an object as the hierarchy is traversed from top to bottom. At the root of the hierarchy a course bounding volume is utilised to represent the object. As we traverse further down the hierarchy, each level provides bounding volumes of finer resolution such that a progressively tighter overall fit of the object is obtained. Sections of the object in question that are deemed to be not colliding as we traverse down the hierarchy are culled away. At the lowest level of the hierarchy, bounding volumes will contain information such as the relevant faces of the polyhedron bound by that volume so that exact collision tests may be performed. If intersection tests return negative results for all nodes of interest checked within a single level in the hierarchy, then a collision miss is reported. Positive collisions are only reported



(a) Top three levels of an AABB tree for a 2D bunny model



(b) Top three levels of an OBB tree for a 2D bunny model

Figure 2.4: 2D example of bounds from an AABB tree and an OBB tree

once we have a positive hit at the lowest level of the hierarchy. It is at this stage that more exact detection algorithms are employed.

Let us now discuss the various bounding volume structures utilised within progressive refinement of collision manifolds. Van den Bergen [18, 17] used an Axis Aligned Bounding Box (AABB) tree for collision detection, applying it to both rigid and deformable bodies. This data structure is so called because every face on the bounding box is always perpendicular to one of the world coordinate axes. The popularity of AABBs stem from the fact that they are geometrically simple and intersection tests between bounding boxes can be conducted very quickly. The only disadvantage with using AABBs is that they do not approximate rotating objects very well. As such, other researchers have adopted *Oriented Bounding Box* (OBBs) data structures that conform to the orientation of the objects better, thus providing a better fit. OBB trees have been adopted by researchers such as Gottschalk et al [62], Krishnan et al [92] and Gregory et al [64]. Figure 2.4 illustrates the difference between using an AABB tree and an OBB tree for bounding objects. As can be seen, OBB trees in general provide a better overall degree of fit to the underlying geometry of the object that they are bounding. Zachmann [146] and Barequet et al [11] have also used similar bounding box hierarchy data structures for collision detection which they have

dubbed *BoxTrees*.

A naïve approach to intersection testing between two oriented bounding boxes would require testing for intersection of every edge of one box with the faces of the other box and vice versa. This is quite an expensive process requiring $2 \times 12 \times 6 = 144$ face edge combinations to be tested. Fortunately, intersection tests between bounding boxes can be done much more efficiently using the *Separating Axis Theorem*³. This theorem, which was first formulated by Gottschalk et al [62] reduces testing to that of only fifteen axes for each box pair resulting in a significant speed up on box-box intersection testing.

A popular alternative to the bounding box as a bounding primitive is the bounding sphere. The rotational invariance of bounding spheres make them much simpler primitives to work with than bounding boxes. They are much faster to update than bounding boxes as their orientation does not need to be updated to that of the body it is bounding but rather a translation of the sphere's centre will suffice. Also, another big advantage that bounding spheres have over bounding boxes is that intersection testing is very easy involving only a quick difference test on the distance from the centre of two spheres and the sum of their radii. However, given thin or long objects, bounding spheres in general do not provide as good a bound as other bounding volumes such as bounding boxes. Researchers such as Quinlan [129], Palmer and Grimsdale [123], Hubbard [76, 78] and Dingliana and OSullivan [48, 47, 119, 118] and Bradshaw [21, 22, 23] have all utilised sphere-tree's for collision testing with great success. In this thesis, we too adopt a sphere-tree approach to collision handling.

An Octree spatial subdivision algorithm as described by Sammet and Webber [135] is the simplest way to construct a sphere-tree around an object. This approach however does not produce very tight fitting spheres due to the uniform grid like nature of the algorithm. Hubbard produced better fitting sphere-trees for objects by adopting *simulated annealing* and *medial axis* approaches [76, 78]. More recently Bradshaw [21, 22, 23] presented a series of algorithms based on an adaptive medial axis approach, called *Expand*, *Merge*, *Spawn*, *Burst* and *Combined* that further improved upon the algorithms presented by Hubbard to provide even better fitting sphere-trees. In constructing the sphere-trees for this thesis, we utilised Bradshaw's sphere-tree construction algorithms.

Klowsowski et al [90] borrow from an idea used in ray tracing that was first introduced by Kay and Kajiya [86], and use a bounding volume structure called a "*Discrete Oriented*

³The *Separating Axis* theorem states that if two boxes are not in contact, then there exists a separating axis $\vec{L} = \vec{V} \times \vec{W}$, where $\vec{V}$ and $\vec{W}$ are distinct vectors from the six box axes. Tests on fifteen of these axes (i.e. the 6 box axes and their 9 cross products) is enough to determine if two boxes overlap.

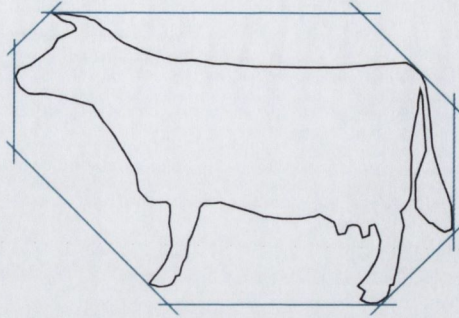


Figure 2.5: Approximating an object with a k-dop

Polytope” (DOP) for narrow phase collision detection. These data structures are bounding convex polytopes consisting of faces/planes whose outward pointing normals are determined from a fixed set of k orientations. Within a DOP we always have a face pair whose normals are collinear but opposite to each other. This then means that any k-DOP will always bound an object in $k/2$ directions. From the above description of a k-DOP, an AABB can thus be classified as a 6-DOP. Figure 2.5 illustrates a 2D example of a rigid body bounded by a k-DOP bounding volume. This example illustrates an 8-DOP with 45° orientations.

Intersection testing with k-DOPs involves performing $k/2$ elementary interval overlap tests. Kłowsowski et al adopt a k-DOP bounding volume tree for progressive refinement within the narrow phase collision detection. Furthermore, they precompute k-DOPs for objects at start up and dynamically create new bounds around the object by utilising the initial precomputed k-DOP bounds whenever objects rotate. It is quite an expensive process to have to recompute new k-DOP boundary planes for a given object whenever its orientation changed in space, thus necessitating the adoption of this looser approximation.

More recently, Mezger et al [103] has used k-DOP hierarchies to aid in speeding up collision detection for cloth simulations. In their work, an 18-DOP hierarchy is utilised for speeding up of self collision tests as well as collision detection with other objects in the scene before an adapted GJK [57] algorithm is applied.

Another bounding volume approach known as *Swept Sphere Volumes* (SSVs) was proposed by Larsen et al [96, 95]. These bounding volumes consist of a sphere that is swept across some core primitive (Figure 2.6), for example a point, line or rectangle. Collision testing involves testing of the primitive that the SSV represents. For example, a SSV on a point primitive is just a simple sphere and collision testing involves an elementary distance/radius comparison. Larsen et al utilise BVHs of swept spheres for fast distance

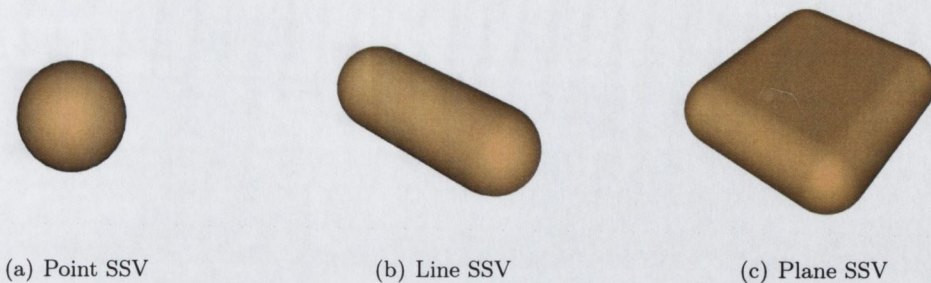


Figure 2.6: Example of SSVs for a point, line and rectangle

queries. Dingliana [47] has proposed the use of SSV trees for collision detection for articulated figures.

Other Bounding Volumes Hierarchies that have been proposed have been *Spherical Shells* by Krishnan et al [92, 93]. A spherical shell is the volume enclosed between two concentric spheres and within a solid angle of some orientation vector. It is stated to be about two to three times slower than an OBB in terms of overlap testing but makes up for it by providing a tighter fit to the model it bounds. He [70] proposed the use of *QuOSPO trees* (i.e. Quantised Orientation Slabs with Primary Orientation) which basically combines OBBs with k-DOPs. Kitamura et al [87] and Smith et al [136] proposed a spatial partitioning scheme using a voxel based structure for collision detection. Exact collision tests are conducted on triangles of objects that are found to occupy the same part of a voxel grid that is subdivided using an octree subdivision of the environment's boundary. He and Kaufman [71] represented voxels as OBBs or spheres to avoid having to work in voxel space. Dingliana [47] proposed a *Voxel Tree* BVH which he referred to as simply a *VoxTree*. This data structure is similar to the AABB tree and BoxTree BVHs previously discussed except that each sibling within each level of the tree always has the same orientation and dimension.

2.4 Simulation Proxies

As scenes get progressively more complicated we eventually will hit an upper limit whereby our animations will fail to achieve required real-time rates regardless of algorithm efficiency or hardware computational power. For such situations, we must turn to geometric and animation simplification techniques to further aid animation speedup. These *level of detail* (LOD) techniques aim to guarantee better frame rates whenever the usual algorithms and hardware dependencies no longer suffice through exploiting a person's perceptual naïvety

of expected results. In such cases, there is a fine trade off between required speed and perceived accuracy.

A popular and obvious LOD approach concerns the simplification of model representation to speed up processing for the likes of image rendering and animation. These techniques are commonly referred to as geometric LOD techniques. Advantages of such a scheme can be seen from Figure 2.7 which gives an example of geometric LODs applied to an object based on distance from the viewer. Funkhouser and Sequin [52] describe a framework for adaptively switching between geometric LODs in order to achieve the best constant frame rates for real time walkthroughs of architectural buildings. Redon et al [134] utilised the sphere-tree data structure within collision detection to perform conservative back-face culling of objects in constant time in order to further speed collision detection within dynamic animations. Aubel et al [2] utilised *animated imposters* (i.e. animated 2D billboarded representations of a 3D virtual human) in order to speed up animation of large crowds of virtual humans. For a further thorough and insightful discussion on the subject matter we refer the interested reader to Luebke et al [102].

Further within the simulation domain, simplification of objects outside the view frustum through the use of simplified proxy representations, along with scene and object culling techniques, have been employed to achieve faster frame rates. Approaches for physically based animations include the use of time critical techniques and simplification through proxy dynamics. The work presented in this thesis is primarily concerned with the achievement of good constant frame rates of physically based simulations through the manipulation of proxy object representations within a time critical approach, whereby physical plausibility rather than exact physical realism is the primary goal. To distinguish between geometric and animation LOD approaches, we refer to simulation simplification LOD techniques in this thesis as *simulation proxies*.

It has been exhibited that many people have preconceptions on what to expect given a physically based scenario. In a study conducted on physics students, Clement [37] showed that many of the students had false expectations of the final outcome of dynamic events, even to the point where conformity to Newtonian mechanics was lost. Profitt and Gilden [127] showed that people in general only use one dimension of information when making judgements involving dynamic events. This suggests that people generally are not as susceptible to anomalous physical behaviour when presented with visual information beyond one dimension. Reddy [131] suggested that in 3D graphics, people are presented with far more

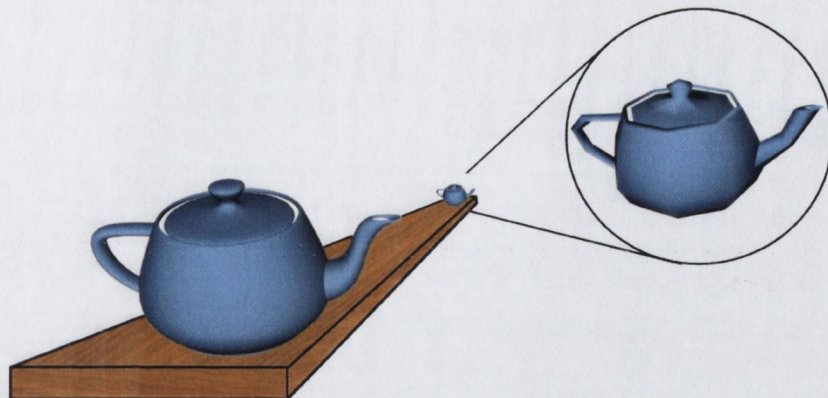


Figure 2.7: Example Geometric Level Of Detail Based on Distance Heuristics

detail than they can actually perceive. These results further imply that perfect physical behaviour within a dynamics system is not entirely necessary when exact simulation accuracy is not of utmost importance in the likes of computer games. Thus, it is possible to exploit people's preconceptions and uncertainties within a dynamics system to produce animations at more constant frame rates during increased scene complexity. Barzel et al [14] recommended such a push for simulation plausibility over accuracy through the exploitation of uncertainty to save upon processing cycles.

2.4.1 Time Critical Approaches

Hubbard [77, 76, 78] first introduced the idea of interruptible collision detection, whereby collision handling accuracy is allowed to be traded off for more efficient speed requirements. A time-critical scheme was presented whereby collision detection was allowed to be interrupted once a predetermined time quota had been exceeded and collisions processed with whatever information available at the time of interruption. Hubbard realised that a sphere-tree data structure could be effectively utilised in such a way as to allow processing to be arbitrarily interrupted at any level within the tree thus giving a progressively refined collision approximation the further down the tree we progress. As such, sphere-trees were utilised by Hubbard to perform interruptible collision detection accounting for time constraints. In this manner, a refinable level of detail collision detection is achieved whereby it is possible for the simulation to fall back to a lower level approximation of collisions whenever time constraints prohibit full collision processing. Such a process has been referred to as *graceful degradation* of collision handling by Dingliana and O'Sullivan [48]. In Hubbard's algorithm, sphere-trees were traversed in a breadth first manner and collision response was

very roughly approximated through an elementary reversal of the colliding object's velocities during collisions. No reasonable contact modelling was considered but rather the algorithm was just concerned with an elementary collision detection scheme. We adopt Hubbard's time critical approach in the work described in this thesis, integrating it into a fully time adaptive dynamics framework and further refining the overall approximation of the method.

O'Sullivan [122, 118, 120] used a perceptual model approach to speed up animations in terms of simplifying collision handling. O'Sullivan and Dingliana [119] have adapted Hubbard's work to use perceptual metrics and return information for collision response. Prioritised scheduling algorithms based on perceptual heuristics were proposed to replace Hubbard's round-robin breadth-first traversal of sphere-trees to allow better utilisation of simulation time. In this approach, specific areas of the environment are prioritised into areas of perceptual importance whereby more time is allocated to collision handling for salient objects. O'Sullivan and Dingliana also adapted Palmer and Grimsdale's *stair case* traversal algorithm within a pair of potential colliding sphere-trees for interruptible collision detection.

Dingliana and O'Sullivan [48, 47] have also proposed more appropriate collision response solutions to Hubbard's elementary response model and have addressed to some extent the issues of collision response in an interruptible system. In their discussions on this issue, the problem of multiple contact responses and the issue of continuous contact are identified as open issues. We shall discuss the idea of interruptible collision detection and the work of Hubbard, O'Sullivan and Dingliana in more detail later in this thesis, along with proposing viable solutions to the issue of multiple collision and continuous contact modelling within an interruptible system.

Most recently Klein and Zachmann [88, 89] have also explored the idea of time-critical collision detection. However, their approach does not utilise sphere-trees (but they do claim that it can be incorporated for any arbitrary hierarchy) but instead they refer to their hierarchies as *ADB-trees* (*Augmented DistriButed trees*). The ADB-tree is just a hierarchical bounding box tree but rather, Klein and Zachmann refer to any hierarchy adopted to use their *average case approach* to be an ADB-tree. Klein and Zachmann's approach differ in that the traversal of the tree is guided and aborted based on the probability that a pair of bounding volumes contain intersecting polygons. The polygons are not explicitly stored within the bounding volume but rather the characteristics about the average distribution of the sets of polygons for that bounding volume node within the tree is stored. Given an intersection between two nodes, A and B, the intersecting volumes are further partitioned

into regular grids. Any cell that contains enough polygons of one bounding volume is considered a *possible collision cell*. If the cell is a possible collision cell with respect to both nodes A and B then the cell is upgraded to a *collision cell*. This information is then used to estimate the probability that polygons from A and B intersect. Klein and Zachmann do not consider the case of collision response but only that of collision detection but however state that the approximate response model proposed by Dingliana and O'Sullivan [48] can be incorporated into their approach.

2.4.2 Other Approaches

Chenney et al [32, 33, 34, 36] used a *dynamics culling* approach to achieve desired real time frame rates in his simulations. Whenever objects moved away from view, Chenney adopted a cheaper simulation approach through the use of proxy dynamic calculations on objects in order to advance them in simulation time, thus producing physically plausible outcomes. Chenney also stressed the importance of paying attention to the interdependencies between dynamic states of objects inside and outside of the viewable region when performing such culling, so that simulation consistency is not effected. Such events as causality⁴ and simulation completeness must be accounted for.

Carlson and Hodgins [30] also utilised a dynamic simplification approach to achieve desired frame rates. In their simulation consisting of a scene of one-legged hopping robots, each robot's motions were culled depending on its importance to the game. This importance was dictated by whether the robot was visible in the scene and if so, how far away it was from the viewer. The simulation levels of detail employed by Carlson and Hodgins ranged from pure dynamic simulation to precalculated kinematics to simple point mass behaviour.

Grzeszczuk et al [65] presented a neural network approximation of dynamics instead of more accurate dynamic motion synthesis. A neural network was trained to approximately imitate the physical motions of the original system, whereby the actual physical motions could be eventually replaced by these quicker neural network approximates.

Giang et al [56] proposed a framework for the unification of both geometric and animation levels of detail for virtual human animations. It was proposed that distance heuristics along with determination of viewable areas, would dictate not only the geometric resolution of the virtual human but also the manner in which it was to be animated. Virtual human simulations were changed from a dynamic approach to kinematic or scripted animations,

⁴Causality refers to one event causing another. Michotte [104] showed that a person's impression of causality is very strong and Leslie and Keeble [98] further provided evidence that such abilities are firmly established in early life.

depending on scene complexity and viewer distance heuristics.

O'Brien et al [115] presented an approach to simplifying the animation of large particle systems through clustering particles into simulation LOD (SLOD) groups. The algorithm then approximates each cluster of particles as one whole weighted particle during its force computations and then applies the same results to all particles in the cluster. Given the desired minimum frame rate and requirements for specific regions within the animation, an appropriate hierarchy of motion models is generated.

Recent work in dynamic hair animation, conducted by Bertails et al [19], utilised an *Adaptive Wisp Tree* (AWT) data structure, which is essentially an acyclic oriented graph of wisp segments, to efficiently mimic and adapt to the clustering and splitting behaviour of hair. The geometric complexity of hair in Bertails et al's approach was further separated from the motion complexity of the hair, which was controlled by the AWT, by a separate multi-resolution model, which was essentially a hierarchy of cylinders. These generalised cylinders allowed the progressive construction and refinement of hair strands as well as being utilised for collision detection.

For completeness we also mention here the existence of work on various adaptive and level of detail schemas for deformable bodies. Terzopoulos et al [138] was one of the first people to model deformable bodies, utilising *finite element* methods in doing so. Hutchinson et al [79] later presented a local adaptive mesh refinement scheme for cloth modelling. The refinement heuristic was based on the angle between two springs and its joining mass. Gascuel et al [53] and Desbrun and Cani-Gascuel [44] presented a refinement technique for highly deformable bodies such as mud and lava. This technique was based on *Smooth Particle Hydrodynamics* where particles discretise the material, splitting and merging based on an energy criterion. Debunne et al [41] proposed an approach based on local discretisation of local elasticity laws for the multiresolution animation of deformable objects. They later improved upon this approach by proposing the use of a mixed finite element method to quickly derive local discrete differential operators over irregular grids with tight error bounds [42]. More recently Debunne et al [43] presented a paper on using space and time adaptive level of detail techniques in combination with a large displacement strain tensor formulation for adaptively animating deformable objects. A multiresolution hierarchy of tetrahedral meshes is used to partition the object in question.

2.5 Summary

In this chapter a thorough review of the area of physically based modelling has been presented. Specifically the area of collision handling has been discussed in some detail as this is the most computationally intensive part of any physical simulation system, accounting for the biggest overall system bottleneck. The basics of representing a rigid body within a dynamic animation system were also discussed providing a general idea of the issues and the underlying data structure of a rigid body representation. We have reviewed the general considerations that must be accounted for during physically based simulations and presented many alternative approaches utilised by researchers when tackling such issues. We have firmly distinguished between the two phases of collision detection and have further defined the narrow phase as having two finer grained detection stages.

Further to the discussion on physically based simulations, a discussion on simulation level of details, referred to within this thesis as *simulation proxies* was also presented to the reader. Such techniques complement the already utilised dynamics algorithms to aid better constant frame rates during physically based simulations when the underlying algorithm and hardware efficiency no longer suffice. In this discussion, we have given a broad overview of the most significant research conducted to date in the implementation of simulation proxies.

In the following chapters, we shall discuss our efforts towards the production of solutions for plausible dynamic simulations that try to conform to timing constraints through various novel simulation proxy approaches. Some of the issues that have been highlighted in this chapter shall also be addressed in doing so. More specifically, we build and expand upon the work of Hubbard and Dingliana and O'Sullivan, further examining the idea of refinable collision handling and the problems associated with expanding this concept for the incorporation into a full time adaptive dynamics system.

Chapter 3

Integration of Motion

In a dynamic system, the simulation of the physical trajectories of any object acting under Newton's equations of motion involves the integration of the continuous state changes over the simulation time period of the various physical state parameters. Readers that are familiar with classical mechanics text will know that according to Newton's laws of motion, the motion of any rigid object can be reduced to a set of second order Ordinary Differential Equations (ODEs) in which time is the common independent variable. It is thus desirable to describe the evolution of these state changes, $\dot{\mathbf{x}}$, as a function of time $f(\mathbf{x}, t)$ (i.e. $\dot{\mathbf{x}} = f(\mathbf{x}, t)$). Thus, solving Newton's equations of motion requires a numerical solution of a set of ODEs. In terms of graphical animation of a physical system, this then means any physical simulation of the underlying dynamics of the continuous system thus resolves down to a series of discrete approximations over time of the solution of ODEs. It is often undesirable to have to resolve second order ODEs as the formulation or solution is often not intuitive. Fortunately, problems involving ODEs can always be rewritten as sets of first order ODEs [126].

In terms of implementation and functionality within our system, we require our ODE solver to act like a black box into which we simply pass the appropriate state variables of our system and get back the updated values. As far as our dynamics system is concerned, it does not care what goes on inside the black box as long as appropriate updated state variables are returned, thus advancing the system. Our concern, of course, when implementing this black box solver is to ensure that it can solve large sets of ODEs within a reasonable time period, returning to the calling system a reasonably accurate and stable set of solutions. For example, for a rigid body simulation, we pass the state variables of the rigid body into the ODE solver and expect an updated position of the center of mass of the rigid body to

be returned along with updated body orientation, updated linear and angular velocity and momentum and any other appropriate state variables.

The literature on solving ODEs is quite vast [54, 94, 126, 97, 31, 144]. A simple search on the web will bring up an abundance of integration algorithms from many various disciplines for solving from first order ODE problems up to some n th order. Many are beyond the scope of this thesis and fortunately do not directly concern us. Here we are only concerned with a class of ODE problems classified as *initial value problems*. A description of an initial value problem can be stated as the determination of a set of values at some final point $\mathbf{x}_{final}$ or over some discrete list of points (e.g. time intervals) from a given starting point $\mathbf{x}_{start}$ with known initial values. The basic idea behind all algorithms for resolving initial value problems is to find the new set of values by discretely *stepping* towards the final solution using the derivatives of the values over some finite number of steps.

This chapter investigates some of the more common and not so common integration algorithms used within the computer graphics research community for the resolution of ODEs that facilitate object locomotion. This chapter also delves into a previously almost untouched family of integration algorithms within the graphics community known as *Verlet* algorithms. As these algorithms are increasingly being considered by more and more graphics researchers over the last few years it is only right to conduct some sort of investigation into them. We shall first examine some of the more widely known and used explicit integrators used by graphics researchers. We shall then present to the reader methods that we may use to improve accuracy and stability of such schemes over time. Following on from this, we shall describe some of the basic and more common Verlet integrators. These have been used extensively by molecular dynamic researchers over the last few years and have increasingly been adopted by researchers within the graphical community due to their *Symplectic* nature (explained later). We conclude the chapter with a discussion on the Velocity Verlet method.

3.1 Finite Difference Methods & the Taylor Series

Numeric integration of Newton's equations of motion are typically done using methods known as *finite difference* methods. The goal of finite difference methods is to determine the state values $\mathbf{x}_{t+\Delta t}$ at time $t + \Delta t$. Information available at time t is usually used to predict the updated state values at $t + \Delta t$. These methods are typically explicit so the time step interval Δt must be chosen so that the integration method will generate a sufficiently

stable solution. If the system is conservative, Δt must be sufficiently small so that total energy is conserved. The nature of many of these finite difference integration methods is best understood if we consider the Taylor expansion of states $\mathbf{x}$ at time $t + \Delta t$. The Taylor expansion of states $\mathbf{x}$ is essentially the expression of the infinite sum involving the states variable $\mathbf{x}$ and its derivatives.

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \frac{\delta \mathbf{x}_t}{\delta t} + \frac{(\Delta t)^2}{2!} \frac{\delta^2 \mathbf{x}_t}{\delta t^2} + \frac{(\Delta t)^3}{3!} \frac{\delta^3 \mathbf{x}_t}{\delta t^3} + \dots + \frac{(\Delta t)^n}{n!} \frac{\delta^n \mathbf{x}_t}{\delta t^n} + \dots \quad (3.1)$$

Different integration algorithms vary in the way in which they implement this expansion. The entire series is never taken into account but rather a truncation of the series up to some n th order is considered instead. Accuracy of the integration algorithm is dominated by the right hand side leading term of the truncation from the series. The error that inevitably creeps into our integration over time is essentially caused by the difference between the solution from the truncated integration step and the exact solution from the full untruncated Taylor series. If we consider a very elementary linear integration scheme that only uses the first two terms of the right hand side of the expansion (i.e. the Explicit Euler method that will be discussed later) then we can say that such a scheme is of *first order with accuracy* or a *global error of $O(\Delta t^2)$* . Different integration schemes treat the basic Taylor series in slightly different ways in order to achieve a higher order of integration accuracy.

There is no one "perfect" method of integration as such but rather each algorithm has its advantages and disadvantages in terms of simplicity, stability and accuracy. Sometimes, the need for simplicity may supersede the need for accuracy or stability and/or vice versa so it is not uncommon for researchers to sometimes choose different integration schemes depending on the task at hand. For example, Turner and Gobbetti [139] implemented a library of various ODE solvers ranging from an elementary explicit *Euler* integrator to a slightly more advanced 2nd order *Verlet* integrator (this will be discussed in more detail later) for their interactive deformable 3D character system. These schemes were made switchable at run time in their system as a compromise was needed to be found for two conflicting design criteria (i.e. the need for high simulation rates vs. high interaction rates). Bridson et al [24] used a hybrid explicit and implicit integration approach for updating the positions and velocities respectively of their cloth animations. The final positions of the cloth particles were updated using an approach analogous to the explicit *Leapfrog Verlet* method. The half step velocity for this update was adjusted to include repulsion impulses, friction and collisions from the linear trajectories of particles depending on the proximity

of each particle at time t . The determination of the final velocity at time $t + \Delta t$ from the cloth system was resolved implicitly whenever repulsions or collisions appeared.

3.2 Explicit Euler

The explicit Euler method, invented by the 18th century Swiss mathematician Leonhard Euler, is the simplest integration algorithm of all the finite difference methods for solving ODEs. The method is a literal implementation of the basic idea for solving initial value problems. Here, in order to find the solution for $\mathbf{x}_{t+\Delta t}$, we compute the derivative of the variable at time t and then take a single step directly in the derivative direction. The step size is specified by the time interval Δt .

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \dot{\mathbf{x}}_t \quad (3.2)$$

As one can see, this is equivalent to truncating the Taylor series (Equation 3.1) so that we take only the first two terms of the right hand side of the series into consideration. As can be seen directly from Equation 3.2, this integrator is of first order with a global error of $O(\Delta t^2)$. Extending this to the integration of Newton's equations of motion, if we consider a single particle (with no rotational values for simplicity), we may write the new position of the particle, x , and its linear velocity, v , at time $t + \Delta t$ as:

$$x_{t+\Delta t} = x_t + \Delta t v_t$$

$$v_{t+\Delta t} = v_t + \Delta t a_t$$

In the above equation, a_t is simply the acceleration of the particle at time t , the derivative of the particle's velocity v . This is easily found, as the force acting on the particle (or any rigid object within our simulation) at the given time t should be known to us¹. Angular state variables such as angular velocity, angular momentum and orientation that are present when simulating rigid bodies are updated in the same manner. Again, angular force components (i.e. torque) at time t should be known as they would have been calculated previously.

Although Euler's method is algorithmically very easy to understand and thus very easy to implement, it unfortunately is not very accurate and can be quite unstable. In fact,

¹We could have easily expressed all second positional derivative terms in our equations as an expression of force over mass, $\frac{F}{M}$, instead of acceleration a . We chose to use a here so as to lessen the number of terms in the equation for clarity purposes.

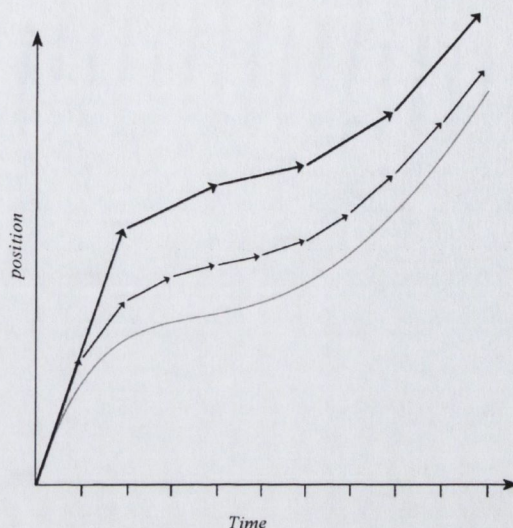


Figure 3.1: Accuracy of Euler's method at two time steps

many integration texts warn against using it unless the ODE problem at hand is so complex, such that using more advanced integration methods would be very difficult or not feasible. However, as an integration algorithm, Euler's method is conceptually important as all integrators in one way or another derive from the same basic idea (i.e. discretely advance a function over time by adding to it derivatives of that function in small increments). For those who are beginning their journey into the vast world of ODE solvers, Euler's method lays a sound foundation from which to start.

3.2.1 Accuracy

The inaccuracy in Euler's method stems from the fact that we are only considering derivatives at the start of each time interval when advancing our variables, rather than the entire set of derivatives through the interval from time t to $t + \Delta t$. As a result, Euler's method is asymmetrical, leading to inaccuracies per time step. Figure 3.1 shows a plot of the path taken by Euler's method over time for two different time steps compared to the true integral curve (i.e. the light gray curve). The bigger the time step taken by the method the greater the deviation from the *true* path (i.e. the integration error is proportional to the step length) as is illustrated by Figure 3.1. It would then seem logical to only take really small time steps in order to maintain overall accuracy for the algorithm. While this is fine in theory, it is unfortunately not very practical for several reasons :

- As time steps get smaller and smaller our animations will get slower and slower (since

the distance covered per step per animation frame is smaller) until finally it will seem that the entire simulation has ground to a halt.

- As the number of iterations through an interval increases, this means that the number of derivative calculations are also increased for the entire interval. This in turn makes Euler's method extremely inefficient as more advanced methods (even though there are more computations per step) may only need one or two iterations through the algorithm to cover the interval with the same accuracy using perhaps only one or two larger step sizes.
- Decreasing the step size will not completely eliminate inaccuracies but rather it will only reduce the deviation from the intended path.

Accuracy may be further improved by adopting more advanced integration methods such as those that consider multiple samples over the interval per step as we shall see later on in this chapter.

3.2.2 Stability

In order to understand how instability can creep in under Euler's method (and under all finite difference methods for that matter) it is probably advantageous in this instance to consider a simple example. Let us consider a 1D *monotonically decreasing* function of x , $f(x, t) = -kx$, such that the function will eventually drive x to zero (note that we mean f in this case to represent a function rather than the force variable). The Euler integration for x will then be:

$$x_{t+\Delta t} = x_t - \Delta t k x_t = (1 - \Delta t k) x_t$$

For relatively small time steps, Δt , this integration will give reasonable results. However, if we increase Δt such that $\Delta t > \frac{2}{k}$ the numerical solution becomes a *monotonically increasing* oscillatory function of x (i.e. $|x_{t+\Delta t}| > |x_t|$). The solution never reaches zero but rather oscillates about zero with greater and greater divergence per step until eventually the entire system blows up. Figure 3.2 illustrates this type of behaviour for a path taken using Euler's method with too large a step size within a spring mass system. The ideal path in the spring mass system as it gradually oscillates to its final rest position is traced out in gray. As can be seen, instability creeps in during Euler's method due to using too large a step size causing the Euler path to never reach the desired solution path. Instead,

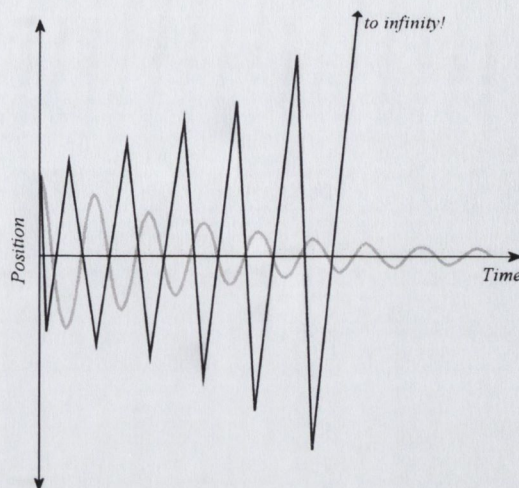


Figure 3.2: Motion over time of a spring mass system and instability of Euler's method due to too large a step size

the error per step accumulates over time and slowly causes the system move further and further away from the true solution until eventually flying off into infinity. If the spring is made too rigid, this will also eventually cause the same instability problems even for step sizes that previously did not cause instability within the system. So such problems like “stiff” springs usually cause great stability problems for lower order explicit integrators. Problems that cause instability issues for our explicit integrators are said to exhibit *stiff* sets of equations².

We should at this point make clear that accuracy and stability are two different things. Even though the previous example may suggest so, improved accuracy does not necessarily mean improved stability and vice versa. If a system is sufficiently stiff, then it doesn't matter how accurate your integration method is given large time steps. If the method is not sufficiently robust enough to account for the stiff problem over time then the system will eventually blow up. The same criterion applies for accuracy. An integration method may be very stable with respect to the problem that it needs to handle but may be quite inaccurate due to the integration algorithm used and the type of system being solved for. Going back to the springs example, this could mean that your mass spring system may not blow up if the integrator is given a very large time step but rather it will cause your mass spring system to continue to oscillate back and forth over the min and max path lengths rather than

²For a formal definition of what makes a problem stiff we refer the interested reader to chapter 16.6 of Press et al [126] and to chapter 6 of Lambert [94] for an excellent discussion on stiffness theory.

eventually coming to rest. Alternatively, if it is meant to continue oscillating due to lack of damping, it could eventually come to rest. Of course, any system given sufficiently large time steps will eventually become unstable due to too large an accumulation of truncation errors over time.

We may tackle problems that exhibit stiff ODEs in several ways. One possibility is to reduce our step size so that the numerical solution is within some acceptable tolerance level (i.e. it doesn't cause the system to diverge). The only problem in doing this is that we may need to use a time step of such small magnitude to stabilise the system that the simulation will look like it has ground to a complete halt. Higher order more advanced integration methods will afford us larger step sizes but only to within a certain limit. The ideal solution to tackling stiff ODEs is to reformulate the problem to avoid solving a stiff ODE. Failing that, the next best solution is to use implicit integration methods. This may be undesirable as we shall discover later on.

3.3 The Midpoint Method

One way to improve the accuracy of the Euler integration algorithm is to take more samples over the interval rather than to just consider the derivative at the start of that interval. With the Midpoint method this is exactly what we do. The method itself is just a small extension to the already discussed Euler method. Here we take a single *trial* step into the middle of the interval and then derive the midpoint derivative that we use to help us take our *real* step forwards,

$$k_1 = \Delta t f(\mathbf{x}_t, t) \quad (3.3)$$

$$k_2 = \Delta t f\left(\mathbf{x}_t + \frac{k_1}{2}, t + \frac{\Delta t}{2}\right) \quad (3.4)$$

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + k_2 \quad (3.5)$$

In essence, we are taking two Euler steps (i.e. two evaluations of the function f are needed) in order to find the final solution through our interval. Figure 3.3 illustrates how the Midpoint method uses the half step derivative to enable it to make a better guess at where to go. The dotted arrows denote the half step derivative direction obtained from the trial step while the thicker arrows denote the path taken by the Midpoint method over that specific time interval. Note how the Midpoint method is closer to the true solution path for

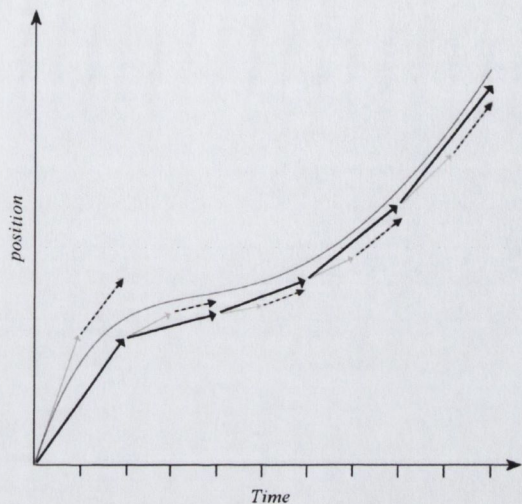


Figure 3.3: Path taken by the Midpoint method

bigger step sizes compared to the previously discussed Euler method.

The method itself is a second order integration method with a global error or accuracy of $O(\Delta t^3)$. This can be easily shown to be equivalent to expanding the truncated Taylor series of the Euler method by an extra term. We shall not derive the formulation here but rather refer the reader to Witkin et al [144] for a derivation. Of course, we do not need to stop at taking just the one extra trial step forward in time. It is possible to take as many trial steps of different magnitudes as we want forward in time (i.e. continue evaluating the function f from higher order derivatives to agree to first order) to improve accuracy. However, for each extra trial step we take we are adding an extra evaluation of the function f per time step into our algorithm thus increasing the overall computation time of the algorithm. The most common of these “multi-step” algorithms is the Runge-Kutta algorithm which we shall discuss next. It is worth pointing out that although the explicit Midpoint method is an improvement on the overall accuracy (and stability) of Euler’s method, it is often passed over for more advanced algorithms as researchers favour slightly higher order precision integrators to one of $O(\Delta t^3)$. However, it is advantageous to understand it as it gives us the basic idea behind multi-step integration methods.

3.4 Runge-Kutta Method

As previously mentioned, it is possible to continue evaluating our function f such that we continue to eliminate higher and higher orders of derivatives thus improving the overall

accuracy of our integration algorithm. This is the basic idea behind Runge-Kutta methods. In fact, the Midpoint method previously discussed is a second order Runge-Kutta method and the Euler method can be considered a first order Runge-Kutta method. In this section we discuss the fourth order Runge Kutta method which has a global error of $O(\Delta t^5)$. This method achieves much higher accuracy and stability per step than the two previous methods discussed by the fact that it uses four evaluations per step. The derivative is evaluated once at the beginning of the interval (Equation 3.6), twice at trial midpoints (Equations 3.7 and 3.8) and once at a trial endpoint (Equation 3.9). Figure 3.4 illustrates this over one integration time step.

$$k_1 = \Delta t f(\mathbf{x}_t, t) \quad (3.6)$$

$$k_2 = \Delta t f\left(\mathbf{x}_t + \frac{k_1}{2}, t + \frac{\Delta t}{2}\right) \quad (3.7)$$

$$k_3 = \Delta t f\left(\mathbf{x}_t + \frac{k_2}{2}, t + \frac{\Delta t}{2}\right) \quad (3.8)$$

$$k_4 = \Delta t f(\mathbf{x}_t + k_3, t + \Delta t) \quad (3.9)$$

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \quad (3.10)$$

Although fourth order Runge-Kutta takes four times as many calculations per time step compared to the explicit Euler method, this is offset by the fact that to traverse the same time interval with the same accuracy, Euler's method would need to take about 16 times more steps.

Runge-Kutta methods (up to some n th order) are probably the most favoured and widely used integration methods within graphics research [60, 68, 111, 137, 16, 3, 142] when any kind of ODE initial value problems are to be resolved. Out of all the n th order Runge Kutta methods, according to literature, the fourth order Runge-Kutta method discussed here is the most popular as many researchers find that this method gives sufficient accuracy and stability along with a moderate computation time. To quote from Press et al [126]:

“For many scientific users, fourth order Runge-Kutta is not just the first word on ODE integrators, but the last word as well. In fact, you can get pretty far on this old workhorse, especially if you combine it with an adaptive stepsize algorithm.”

Of course, the fourth order Runge-Kutta methods still suffer from instability if the step size is made too large. This isn't as bad as it sounds if you consider that all integration

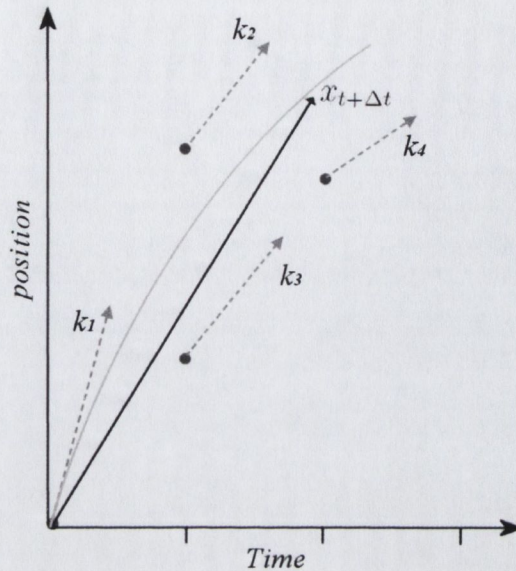


Figure 3.4: Derivative steps of Fourth order Runge Kutta method

algorithms will eventually become unstable if the step size used is made sufficiently large. It is just one of the problems of trying to solve a continuous system using discrete methods.

So it would seem that a fourth order Runge-Kutta method is ideal as far as computation time vs accuracy and stability tradeoff is concerned for solving all our rigid body dynamics simulations. We could accept this fact to be so and like many other graphics researchers before us continue using Runge-Kutta schemes in some shape or form for solving our ODEs. However, in our time adaptive dynamics framework, any savings on the integration time means more time will be available to us when doing collision handling. So, it would be advantageous to us if we could find or come up with another integration algorithm that will give us as close a stability and accuracy advantage as the fourth order Runge-Kutta method but with less evaluations per step. We shall later look at a family of integrators used extensively in the field of molecular dynamics for its high stability and accuracy over long integration time periods known as *Verlet* methods. It may be possible to adapt such integration algorithms for the purposes of rigid body simulation in computer graphics, providing us with a reasonably high degree of stability and accuracy over reasonably large time steps and with lesser computational cost per step than the higher order integrators previously discussed such as fourth order Runge-Kutta. We shall discuss the three most common Verlet integrators but will leave the discussion on adapting these integration schemes for rigid body dynamics to the next chapter. But before we do so, let us explore how one might

go about further improving integration accuracy and stability of the previously described schemes.

3.5 Adaptive Step Sizes

Up until now we have only been considering fixed step sizes within our integration algorithms. The problem with using fixed step sizes is that we are confined to the worst case assumption of our chosen integration algorithm. This means, we may only proceed forward in time as fast as the “worst section” of $\mathbf{x}_t$ will allow us to. Ideally, we want to take as large a step size as possible without compromising accuracy and stability within our simulations. In other words, we would like to choose as large a Δt as possible per integration step. One method of improving the worst case problem of using fixed step sizing, thus enabling us to choose relatively large Δt 's, is to use adaptive step sizes instead.

With adaptive step sizing, the time step per simulation iteration is not fixed but rather is variable. We attempt to take as large a time step as possible while maintaining a reasonable level of accuracy and stability so that global error per step is relatively constant. For each step we take within our integration loop, we must somehow determine the measure of the potential truncation error of the step. One possibility of measuring the error of the step is to take two separate steps to $\mathbf{x}_{t+\Delta t}$. Press et al [126] refer to this as *step doubling*. For example, take one step by Δt to $\mathbf{x}_{t+\Delta t}$ then starting back at $\mathbf{x}_t$, take two smaller more accurate $\frac{1}{2}\Delta t$ steps to $\mathbf{x}_{t+\Delta t}$. We then take the difference between the two steps for error measurement. Another possibility would be to take a step to $\mathbf{x}_{t+\Delta t}$ using a quicker but less accurate integration scheme and compare that to the slower but more accurate scheme. For example, we could take a step to $\mathbf{x}_{t+\Delta t}$ using a fourth order Runge-Kutta integrator and we do the same step again using a first order explicit Euler integrator and then take the difference as a measure of accuracy. If the step error is within a pre-accepted tolerance then advance to $\mathbf{x}_{t+\Delta t}$ using that time step, otherwise reduce the time step to within an acceptable level.

Taking adaptive step sizes has the advantage of providing better problem handling within our system as it is self monitoring process. If we find that the system will become unstable due to the time step or the bounds set by the step, adaptive step sizing will allow us to automatically “slow down” the system so as to adjust the potential instability within the system. Due to the self checking mechanism of adaptive steps, this also means that bad errors are less likely to creep in over time. However, there are disadvantages to employing

an adaptive step sizing scheme as well as the potential advantages mentioned. One of the more obvious disadvantages is that it is more computationally expensive than just taking fixed steps per iteration. This is due to the extra evaluations and adjustments per step. For a time dependent system, such as the one proposed within this thesis, this time could be better spent doing more important things such as better evaluations of collisions and responses. Another potential disadvantage that adaptive step sizing has is that, as each step size is fluctuating, the amount of computation time per simulation frame continually alters. This can cause fluctuations in the velocity of objects within our animation and as a result objects may appear to sporadically speed up or slow down per frame if frame and time dependence is not accounted for.

However, despite the obvious increase in computational complexity, adaptive step sizing combined with reasonably high order explicit schemes have been adopted by quite a number of researchers within the graphics community over the years. Mirtich [111] in his Timewarp paper uses a fifth order Runge-Kutta scheme, employing adaptive steps for improved accuracy and stability in his simulations. Baraff [3] in one of his earlier papers on analytical methods for dynamic simulation states that he uses a fourth order Runge-Kutta method with adaptive steps to advance his system. Turner and Gobbetti [139] again, also utilise adaptive steps within their deformable character system. They interestingly determine the step error by comparing the difference between a Verlet scheme (very popular within the molecular dynamics community) and the lower order explicit Euler scheme for the step size. In the world of cloth animation where simulation stability is a big issue, Bridson et al [24], Baraff and Witkin [10] and Volino et al [142] all use adaptive step sizing to supplement the integration schemes within their systems for better overall stability.

3.6 Implicit Methods

Up until this point all integration algorithms that we have been discussing have been defined explicitly. Recall that we mentioned explicit methods are not very good at solving for problems that exhibit stiff ODEs. One can reduce the time step taken to increase stability but that itself can lead to undesirable results if one is forced to reduce the time step to too small a value. For such situations it may be more appropriate to adopt an implicit integration scheme instead. For the purposes of completeness we shall very briefly discuss the ideas behind implicit integration methods and for simplicity shall introduce the implicit Euler method as an example.

Implicit integrators exhibit great stability characteristics when solving for stiff sets of equations. However, the formulation and solution of implicit equations is not usually very intuitive as we shall see. With explicit integration methods we attempt to move forwards in time using the information available to us at that present time. However, with implicit methods, we attempt to move forwards in time by evaluating the function at the time that we wish to proceed to. This idea is best viewed as going to the place we want to be first, then looking backwards to see the path we must take and then going back and taking that path. It is like travelling forwards by travelling backwards! For large step sizes over time, we may lose accuracy but not stability. For such situations, the implicit method will eventually drive the state $\mathbf{x}_{t+\Delta t}$ to an equilibrium solution of zero. This is in fact the correct solution for differential equations. Unfortunately, this only holds true for linear systems. In general we cannot solve for $\mathbf{x}_{t+\Delta t}$ unless the derivative function of $\mathbf{x}$, $f(\mathbf{x})$ (i.e. $\dot{\mathbf{x}} = f(\mathbf{x})$), is a linear function. We can get around this by expressing the function $f(\mathbf{x})$ as a linear approximation based on the Taylor series.

A simple example will probably best make what we have discussed clearer. Let us take a look at the implicit Euler method. This method is also known as the *Backward Euler* method and as the name suggests we are looking backwards from where we want to go in order to go there. The implicit Euler method can be expressed as:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta t \dot{\mathbf{x}}_{t+\Delta t} \quad (3.11)$$

where the differential $\dot{\mathbf{x}}_{t+\Delta t} = f(\mathbf{x}_{t+\Delta t})$. At first glance, this may seem identical to the explicit Euler method as described by Equation 3.2 but upon closer inspection one can see that we are dependent on knowing the state variable derivative at time $t + \Delta t$ rather than at time t to advance our state $\mathbf{x}$ forward in time. One can immediately see a problem here. We do not know what the state derivative at time $t + \Delta t$ is until we first move there but we need to know this derivative in order to move there! In order to resolve Equation 3.11, let us say that the state change, $\Delta \mathbf{x}$, that causes the state $\mathbf{x}_t$ to move to $\mathbf{x}_{t+\Delta t}$ is $\Delta \mathbf{x} = \mathbf{x}_{t+\Delta t} - \mathbf{x}_t$ or rearranging $\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \Delta \mathbf{x}$. So we may write Equation 3.11 as

$$\mathbf{x}_t + \Delta \mathbf{x} = \mathbf{x}_t + \Delta t f(\mathbf{x}_t + \Delta \mathbf{x}) \quad (3.12)$$

From the previous equation it can be seen that $\Delta \mathbf{x} = \Delta t f(\mathbf{x}_t + \Delta \mathbf{x})$ and we have simply replaced the derivative state term $\dot{\mathbf{x}}_{t+\Delta t}$ with the function representation $f(\mathbf{x}_t + \Delta \mathbf{x})$. If we can solve for $\Delta \mathbf{x}$ then we can resolve Equation 3.11. In order to solve for $\Delta \mathbf{x}$ it is necessary

to approximate $f(\mathbf{x}_t + \Delta\mathbf{x})$ based on its Taylor expansion as stated earlier. So,

$$\Delta\mathbf{x} = \Delta t f(\mathbf{x}_t + \Delta\mathbf{x}) \quad (3.13)$$

$$= \Delta t (f(\mathbf{x}_t) + f'(\mathbf{x}_t)\Delta\mathbf{x}) \quad (3.14)$$

where the approximation of $f(\mathbf{x}_t + \Delta\mathbf{x})$ by Taylor expansion is $f(\mathbf{x}_t) + f'(\mathbf{x}_t)\Delta\mathbf{x}$. One thing to note here is that if the function $f(\mathbf{x}_t)$ is a vector (which it more than likely is in almost all cases), the derivative $f'(\mathbf{x}_t)$ is thus a matrix. So, rearranging the above formulation we end up with a linear equation of the form:

$$\left(\frac{1}{\Delta t}\mathbf{I} - f'(\mathbf{x}_t)\right)\Delta\mathbf{x} = f(\mathbf{x}_t) \quad (3.15)$$

in which we can easily solve for $\Delta\mathbf{x}$ using any linear equation solver. In the above equation the $\mathbf{I}$ term is simply the identity matrix. In practice, the matrix involved in the linear equation is usually quite sparse so linear equation solvers such as the Conjugate Gradient method are usually utilised by graphics researchers to exploit this sparsity. After we have solved for $\Delta\mathbf{x}$, resolving $\mathbf{x}_{t+\Delta t}$ is but a trivial matter of substitution.

As one can see, trying to resolve ODEs implicitly is quite an involved process. We end up having to resolve a linear system of equations per step which adds to the overall computational overhead of the algorithm, especially when there are a lot of variables to consider. On top of this, the mathematical formulation of the linear system is not very intuitive even for those with a firm mathematical background. This is especially true when the system to be integrated is quite a complex one.

Let us consider a simple example of integrating for Newton's equations of motion, here only considering a single particle with no rotational terms for simplicity, using the Backward Euler scheme previously discussed. Then integrating for the particle's position x and velocity v over time using the Backward Euler scheme we may express the integration equations as:

$$x_{t+\Delta t} = x_t + \Delta t v_{t+\Delta t} \quad (3.16)$$

$$v_{t+\Delta t} = v_t + \Delta t f(x_{t+\Delta t}, v_{t+\Delta t}) \quad (3.17)$$

In order to resolve this we must rewrite these two equations by defining $\Delta x = x_{t+\Delta t} - x_t$ and $\Delta v = v_{t+\Delta t} - v_t$. Here we first solve for Δv and then we can get a solution for Δx by

simple substitution. Working through the Taylor expansion of the function f as before we finally come up with the linear equation

$$\left(I - \Delta t \frac{\delta f}{\delta v} - \Delta t^2 \frac{\delta f}{\delta x}\right) \Delta v = \Delta t \left(F_t + \Delta t \frac{\delta f}{\delta x} v_t\right) \quad (3.18)$$

in which we solve for Δv using our favorite linear system solver. All terms in the above equation are as they were before. F_t is simply the force acting on the particle at time t . We have neglected the derivation here but rather point the interested reader to [144, 10, 126] for a more comprehensive treatment of what was discussed in this section along with full derivations of the basic implicit Euler integrator (i.e. Backward Euler) and particle example presented here. As one can see, the implicit formulation gets less intuitive as the problem at hand becomes more complex. To our knowledge, no one to date has tried directly using or deriving any implicit formulations for resolving rigid body motions in a dynamics system by the Backward Euler method or similar. Reasons for this could be that rotational variables are quite difficult to formulate into an implicit form and solve for. Another strong possibility is that the increased computational costs of implicit integrators do not justify using them in rigid body simulations. The already existing high order explicit solvers previously discussed, be they supplemented with adaptive step sizing or some other modification or not, are perfectly fine for the job as far as stability and accuracy issues are concerned. In general, if at all possible, researchers would like to avoid having to solve linear system of equations which is one of the major drawbacks of implicit methods.

Early work using implicit methods was conducted on deformable models by Terzopoulos et al [138]. In this paper, an implicit approach was taken to advance the equations governing a deformable body's motion over time. The deformable body's equation of motion was expressed in Lagrangian form. The first and second derivative terms of this equation were approximated interestingly (though the paper does not mention this, either because the authors were not aware of this or neglected to do so) from a reformulation of the basic *Verlet* equations. This then gave a linear system in which the updated position in time could be solved for. Terzopoulos et al stated that they used Choleski decomposition and a Gauss-Seidel relaxation method to solve the linear system. More recently, implicit methods have been used in the implementation of stable cloth dynamic simulations by Baraff and Witkin [10]. A Backward Euler method very much the same as that which was previously described in this section was utilised by Baraff and Witkin and the linear system produced was solved using a modified Conjugate Gradient method. In the following year after the publication of Baraff and Witkin's paper, Desbrun et al [45] proposed solving such an

implicit system using a predictor-corrector approach coupled with an inverse dynamics relaxation technique, in order to combat having to resolve the large linear systems when using such implicit methods. This approach was shown by the authors to be about ten times faster than the Conjugate Gradient approach as used by Baraff and Witkin. More recently, some researchers in cloth simulation such as Bridson et al [24, 25] have been adopting an explicit/implicit approach.

Next we shall have a look at a class of integration schemes, which to a large extent has been either ignored or overlooked by the graphics community up until very recently. These class of integrators, called *Verlet* integrators have been shown to exhibit reasonably good stability and accuracy over long simulation time periods without the need for multiple steps over each integration time interval and are thus widely used in the chemistry, molecular dynamics and celestial mechanics communities where stability and accuracy over long simulation time periods are a requirement.

3.7 Verlet Integration

As previously mentioned, in the fields of molecular dynamics and celestial mechanics where high accuracy and stability over long integration time periods is essential for good simulations, the previously discussed Euler and Runge-Kutta integration methods are rarely used or even considered when solving ODE problems. Instead, for years, a different family of integrators known as *Verlet* integrators have been the unquestionable method of choice for most researchers in these fields (this is particularly true of molecular physicists).

Verlet integrators, as the name suggests are commonly attributed to Verlet [141] due to his work in molecular dynamics in the 1960s. However, Verlet was not the first person to have come up with the basic underlying algorithm. Historically, the algorithm was first used by Delambre in 1791 and has been used many times since then in one form or another. Störmer used it to study the motion of electrical particles in a magnetic field in 1907 and Cowell and Crommelin again used it in 1909 to compute the orbit of Halley's comet. Furthermore, Verlet methods are known in numerical analysis literature as *explicit central difference methods*. In fact, Bridson et al [24] in their cloth dynamics paper refer to their integration method as central differencing. If one looks closely at the algorithm, one will notice that this is a hybrid explicit implicit Verlet integration algorithm. It is explicit in the position update and implicit in the velocity update stages. The choice of name used in the paper may be attributed to the fact that Ronald Fedkiw, who is one of the co-authors

of the paper, was a numerical analysis. For consistency (and a bit of sanity), we shall refer to all such families of integration algorithms in this thesis as Verlet integrators.

The big advantage with Verlet integration schemes over other integration schemes such as Euler and Runge-Kutta is that they present the following attributes:

- They generally provide very good stability and accuracy for large time steps compared to the number of sub steps needed.
- They in general conserve energy and momentum. The previously discussed explicit schemes tend to numerically drift over time. Generally these schemes are considered drift free.
- They are what is known as *Symplectic* integrators. Simply, what we mean by Symplectic is that the integration preserves area in phase space thus making the algorithm time reversible.

The last property of Verlet methods is an important property if we consider that Newton's equations are themselves time reversible. What we mean by time reversible is that at any given time, it is possible to trace a trajectory back to its origin. The previous explicit integration methods discussed unfortunately were not time reversible due to the asymmetric nature of the algorithms. Thus, Verlet methods give a more accurate description of the underlying nature of Newton's equations. Of course, numerical errors over time will augment this reversibility but for short time periods, a stable integration algorithm should exhibit some apparent temporal reversibility.

With all the apparent advantages that Verlet methods have over the previously discussed integration methods it is somewhat surprising to discover that very few graphics researchers have adopted this scheme but rather prefer to consider Euler and Runge-Kutta in their simulations. However, in saying this, over the last couple of years an increasing number of graphics researchers have begun to realise the potential of Verlet methods and have adopted the use of these methods in some shape or form [24, 84, 80, 139, 69, 128]. In fact Haut et al [69] acknowledges the accuracy advantage of Verlet integrators over the other more common explicit integrators and Provot [128] has also shown that Verlet integrators offer a good accuracy efficiency compromise. Here we shall discuss the three most commonly used Verlet methods in the molecular dynamics community and later we will discuss the potential problems with adapting Verlet schemes for rigid body dynamic simulations and present some possible solutions to these problems.

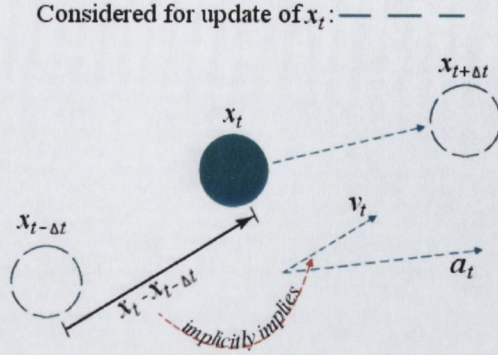


Figure 3.5: The basic Verlet integrator

3.7.1 The Basic Verlet Method

The simplest form of all the Verlet integration methods is the basic Verlet method. This is sometimes referred to as the Störmer integration method and the formulation looks like thus:

$$x_{t+\Delta t} = 2x_t - x_{t-\Delta t} + a_t \Delta t^2 \quad (3.19)$$

The above equation expresses the advancement from a position x at time t to time $t + \Delta t$. Again, all variable meanings are as before with variable a denoting acceleration. Here, we cannot generalise the formulation to all state variables when expressing the integration formulation as we have done with the first three explicit integrators that we have previously seen, as position and velocity variables are dealt with differently by Verlet integrators. From Equation 3.19 we can see that, unlike the Euler and Runge-Kutta schemes, velocity plays no part in the update of the position from one time step to another. Instead, velocity is implicitly implied within the integration step. This is illustrated in Figure 3.5. The velocity at time t if needed may be approximated by the following formula,

$$v_t = \frac{(x_{t+\Delta t} - x_{t-\Delta t})}{2\Delta t} \quad (3.20)$$

This is a very rough approximation at best and can cause undesirable energy dissipation within the system.

The basic Verlet method can be easily derived by taking two 3rd order Taylor expansions of the position variable of our system:

$$\begin{aligned}
x_{t+\Delta t} &= x_t + \Delta t \dot{x}_t + \frac{1}{2} \Delta t^2 \ddot{x}_t + \dots \\
x_{t-\Delta t} &= x_t - \Delta t \dot{x}_t + \frac{1}{2} \Delta t^2 \ddot{x}_t - \dots
\end{aligned}$$

one step forward in time and one step backward in time and summing them. The eventual summation of these two expansions will eliminate the first derivative terms (i.e. the velocity terms) in the expansions, thus giving us Equation 3.19. It is easy to see, from how the expansions $x_{t+\Delta t}$ and $x_{t-\Delta t}$ are used in the derivation, that the basic Verlet method is symmetric with respect to time and hence is time reversible. Also, from the derivation, it can be concluded that the method gives positional accuracy of $O(\Delta t^4)$. However, the computed velocity at time t is only correct to an order of Δt^2 .

As the reader may have concluded by now, the basic Verlet can be a good and stable integrator. However, it is not completely perfect. Like all integration algorithms, there are several problems that need to be addressed. The first being, we are adding variables of $O(\Delta t^2)$ directly to variables of $O(\Delta t^0)$ which leads to higher numerical imprecision in the final solution. Higher accuracy may be achieved if the derivative gap between the numbers being added was not so large. Furthermore, because the basic Verlet scheme does not use velocity explicitly, this may potentially cause several problems for us. For example, forces that depend on velocity like drag or friction cannot be easily incorporated into the integration formulation. One other potential disadvantage of the basic Verlet method is that we need to keep information on the previous position to where we are at all times in order to advance forwards in time. To improve overall numerical accuracy and stability and to tackle the more obvious problems mentioned, more advanced and improved formulations of the basic Verlet algorithm mentioned here have been developed. The most common and popular of these algorithms are the *Leapfrog* and the *Velocity* Verlet algorithms. Both of which we shall discuss next.

3.7.2 Leapfrog Verlet

A variation on the basic Verlet algorithm previously discussed is the *Leapfrog Verlet* method. It is a second order Symplectic integrator designed to improve the overall accuracy of approximation of the velocity evaluations. The Leapfrog method derives its name from the way it handles velocities, as will become apparent. Numerical inaccuracies during integration are further improved with the Leapfrog algorithm by virtue of the fact that here

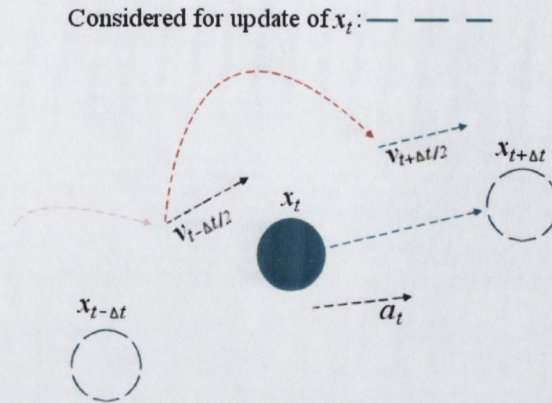


Figure 3.6: Behaviour of velocity within the Leapfrog Verlet integrator

we are adding variables of $O(\Delta t^1)$ to variables of $O(\Delta t^0)$. The overall algorithm can be described as:

$$v_{t+\frac{\Delta t}{2}} = v_{t-\frac{\Delta t}{2}} + a_t \Delta t \quad (3.21)$$

$$x_{t+\Delta t} = x_t + v_{t+\frac{\Delta t}{2}} \Delta t \quad (3.22)$$

As can be seen, with the Leapfrog algorithm, the velocities are evaluated at the half step (Equation 3.21). The force at time t is used to advance the velocity (i.e. from Newton's second law, $F = Ma$) which is half a time step behind the current time to one half a time step forward from the current time. In this way, it appears that the velocities are *leapfrogging* over the current time step and hence the name of the algorithm. The half time step velocity is then used for the positional update (Equation 3.22) so that our new position is based on the half step velocity update. We illustrate this behaviour of the Leapfrog algorithm in Figure 3.6.

Obviously, dealing with velocities in this manner means that the velocity states in our system will always be one half step behind where we actually are in time. Once again, the velocity at time t can be got if needed from a simple calculation:

$$v_t = \frac{1}{2}(v_{t+\frac{\Delta t}{2}} + v_{t-\frac{\Delta t}{2}}) \quad (3.23)$$

or alternatively

$$v_t = v_{t-\frac{\Delta t}{2}} + \frac{1}{2}a_t \quad (3.24)$$

Whereby in the basic Verlet method the position variables were used to approximate the velocity at time t , in the Leapfrog method the half step velocities before and after time t are used instead in the approximation. This as a result gives an overall better evaluation of velocities than before. The approximation of v_t , like the basic Verlet velocity approximation, is still only first order accurate but however is still a better approximation than the basic Verlet. With both the Leapfrog and basic Verlet methods, velocity at time t is not known until we do our time step update. This may be undesirable since this means that we are required to take an extra step ahead of time when we wish to know velocity at time $t + \Delta t$ (e.g. for when calculating collision impulses).

Another potentially undesirable attribute of both Verlet algorithms discussed so far is that both methods are not *self starting*. We need past information in order to move forward in time. The problem with this is that, when our system first starts up, past information about position and/or velocity from initial data may not be available to us. This is not too much of a problem to tackle as there are several methods that we may employ to approximate past information for startup purposes. For example, since the Verlet algorithms are time reversible, it is possible to make a reasonable approximation of past variable data at time $t - \Delta t$ from initial data at starting time t . In the next subsection we shall have a look at the *Velocity Verlet* algorithm, which addresses the previous two problems mentioned along with giving us a better velocity approximation.

3.7.3 Velocity Verlet

The Velocity Verlet algorithm is a further improvement upon the Leapfrog algorithm in terms of velocity approximation. It is considered the most complete form of the explicit Verlet methods discussed since it approximates both position and velocity at the same instant in time. Positional update with the Velocity Verlet algorithm does not depend on any past data, so the algorithm, unlike the previous two Verlet algorithms discussed, is self starting.

In the general form of the Velocity Verlet method, the positional update is dependent on both velocity and acceleration at time t . The velocity update to time $t + \Delta t$ however is dependent on the acceleration at both time t and $t + \Delta t$. We use the positional data at time $t + \Delta t$ in order to calculate the forces at that time step, thus enabling us to calculate

the acceleration at $t + \Delta t$ for the velocity update step. The general form equations for the Velocity Verlet method can be expressed as:

$$x_{t+\Delta t} = x_t + v_t \Delta t + \frac{1}{2} a_t \Delta t^2 \quad (3.25)$$

$$v_{t+\Delta t} = v_t + \left(\frac{a_t + a_{t+\Delta t}}{2} \right) \Delta t \quad (3.26)$$

However, in practice, we generally do not integrate directly as described by Equations 3.25 and 3.26. Rather, these two equations are further split up into three equations of the following form:

$$v_{t+\frac{\Delta t}{2}} = v_t + \frac{a_t}{2} \Delta t \quad (3.27)$$

$$x_{t+\Delta t} = x_t + v_{t+\frac{\Delta t}{2}} \Delta t \quad (3.28)$$

$$v_{t+\Delta t} = v_{t+\frac{\Delta t}{2}} + \left(\frac{a_{t+\Delta t}}{2} \right) \Delta t \quad (3.29)$$

The half step velocity is first calculated and then this in turn is used for the full positional update. The update of the velocity to time $t + \Delta t$ still requires the acceleration at time $t + \Delta t$, which can be obtained from the force calculation at the new position, $x_{t+\Delta t}$. However, this time the half step velocity is utilised in both the position and velocity update to time $t + \Delta t$. In general it is adequate enough to assume that the force at time step $t + \Delta t$ within our dynamics system consists of just gravity and any other external forces such as wind, for the purposes of velocity approximation at time $t + \Delta t$. If there are any collisions or contacts during the time step then the collision and/or contact resolvers will adjust the forces and velocities as appropriate to the situation so that at the next simulation iteration we should have the appropriate states data before passing them into the ODE solver again. The only time that such an assumption may not hold up is when there are internal constraint forces acting on the system (e.g., an internal spring force or internal constraint force that holds a joint together). In such instances we must account for such forces during the force calculation at $t + \Delta t$. Figure 3.7 illustrates the steps involved in the Velocity Verlet integration scheme.

We can easily show the equivalence of the Velocity Verlet method to the basic Verlet method if we remove the velocity terms from the general formulation. It should be mentioned here that both the Velocity and Leapfrog Verlet methods have the same order of positional global error as the basic Verlet method (i.e. $O(\Delta t^4)$). The Leapfrog and Velocity Verlet algorithms were designed to improve on the overall velocity approximations as the

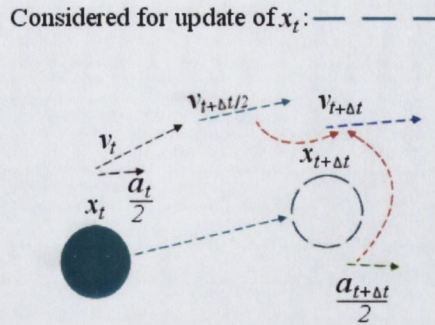


Figure 3.7: Velocity Verlet integrator

basic Verlet velocity calculations are too loose an approximation for a lot of useful energy calculations necessary in Molecular Dynamics. Also by virtue of the fact that the algorithms add an $O(\Delta t^0)$ term to an $O(\Delta t^1)$ term instead of an $O(\Delta t^0)$ term to an $O(\Delta t^2)$ term, as in the basic Verlet algorithm, this has the advantage of reducing the overall numerical drift per step. Obviously, the number of terms to be evaluated per step in each algorithm increases as the complexity and accuracy of each algorithm increases. There are more terms to be evaluated with the Velocity Verlet method than in the Leapfrog Verlet method and both methods require more evaluations per step than the basic Verlet method. This is a small concession for greater accuracy and stability. Of course, accuracy and stability can be further improved upon in the Verlet schemes discussed if one was to adopt adaptive step sizes or an implicit formulation of the schemes.

In the next chapter we shall discuss how we might go about using Verlet methods in computer graphics research to simulate rigid body dynamics. We shall investigate the pros and cons of these methods for such situations and offer potential solutions to any such problems that may arise from using such a scheme with rigid body dynamics.

3.8 Tests and Results

In order to determine the algorithmic efficiency and advantages that Verlet algorithms may potentially have over the basic popular algorithms utilised by graphics researchers, some simple tests were set up. We conducted the popular springs test to determine overall accuracy and stability of each of the Verlet algorithms discussed in this chapter, comparing the results to the Euler, Midpoint and fourth order Runge-Kutta algorithms that were

discussed earlier.

In order to avoid excess stiffness in the system, we simulated the spring with a spring coefficient of 10 and a spring dampening coefficient of 1. This meant that the spring should have been quite springy in nature and should not have exhibited problems associated with stiff ODEs for reasonably large time steps. The spring mass system was suspended from the origin and had a final rest length of one unit and was released at half a unit from the origin. This meant that it had a deviation of half a unit from its final rest length at the beginning of the simulation. It was chosen that tests were to be conducted at three different time steps with each time step getting successively larger. The time steps of .05, .5 and 1 respectively were chosen (these being actually quite large time steps). Tests were conducted over a one minute time interval whereby the position of the point mass on the spring system was then taken for every second of the simulation and the deviation from the final rest length recorded. These results were then plotted on a graph against to the true path of the spring mass system for that second within the simulation time period.

As can be seen from Figure 3.8, the Euler method gave reasonably accurate results for the .05 time step but after that, inaccuracies caused by the increase in the time step caused the point mass on the spring system to over shoot at each step and eventually the system exploded. At a time step of one, the Euler method (Figure 3.8(c)) caused the system to explode almost immediately. The Midpoint method (Figure 3.9) showed very good signs of accuracy and stability up until the last time step increase (Figure 3.9(c)). At this stage, inaccuracies crept in causing the algorithm to overshoot its mark at each step eventually causing the system to explode. As is evident from Figure 3.10, by far the most accurate and stable method for the tests conducted was the fourth order Runge-Kutta method. This method maintained extremely good accuracy and stability for all simulation time steps.

Next up were the Verlet methods. Both the basic Verlet and Leapfrog Verlet methods (Figure 3.11 and 3.12) performed similarly well. They both maintained very good accuracy for the first two time steps. At the third time step trial (Figure 3.11(c) and 3.12(c)), the accuracy of the algorithms started to wane a little bit but however still maintained system stability. If we were to let the simulation run beyond the one minute mark it can be seen that both of these Verlet methods would eventually bring the system to its equilibrium state as is suppose to happen. It was found that the velocity Verlet method, as can be seen from Figure 3.13, performed almost as well as the fourth order Runge-Kutta method. Accuracy was ever so slightly off in comparison to the expected true path of the mass spring system for the last time step test but this was not noticeable at all during the simulation.

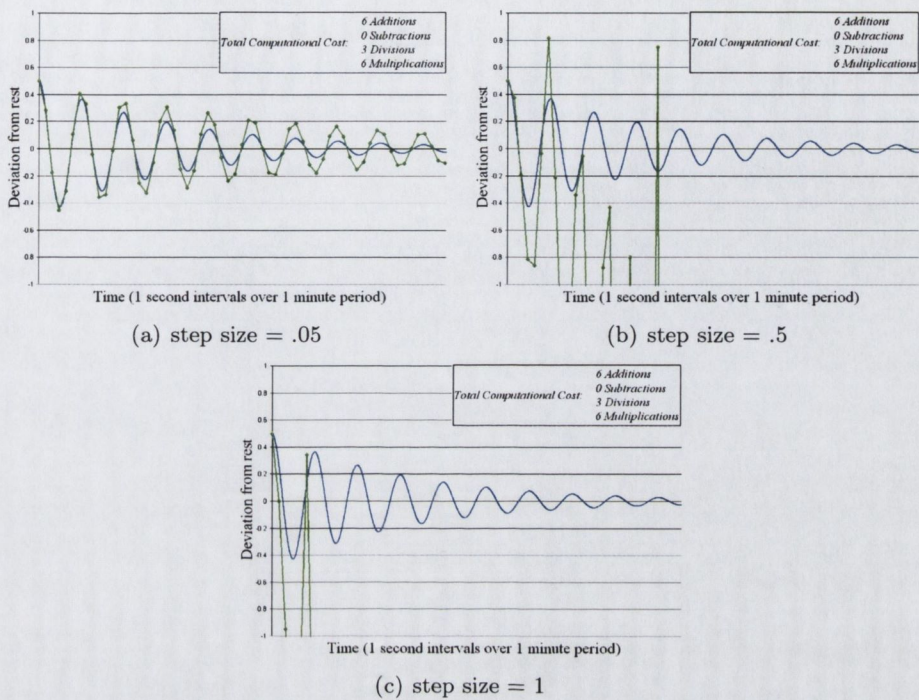


Figure 3.8: Mass spring test of accuracy and stability for Euler's method

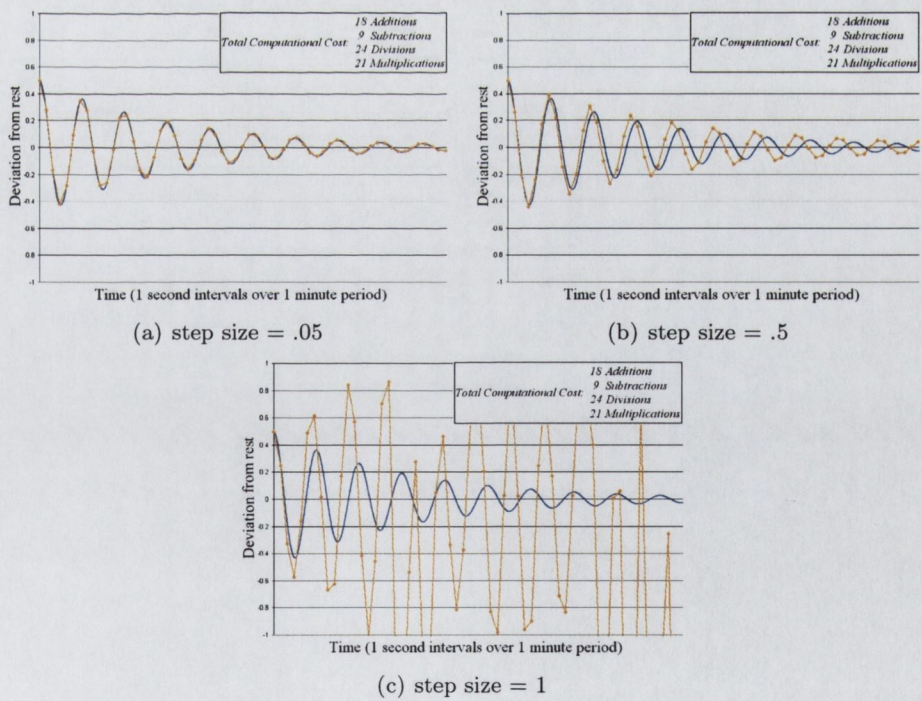


Figure 3.9: Mass spring test of accuracy and stability for Midpoint method

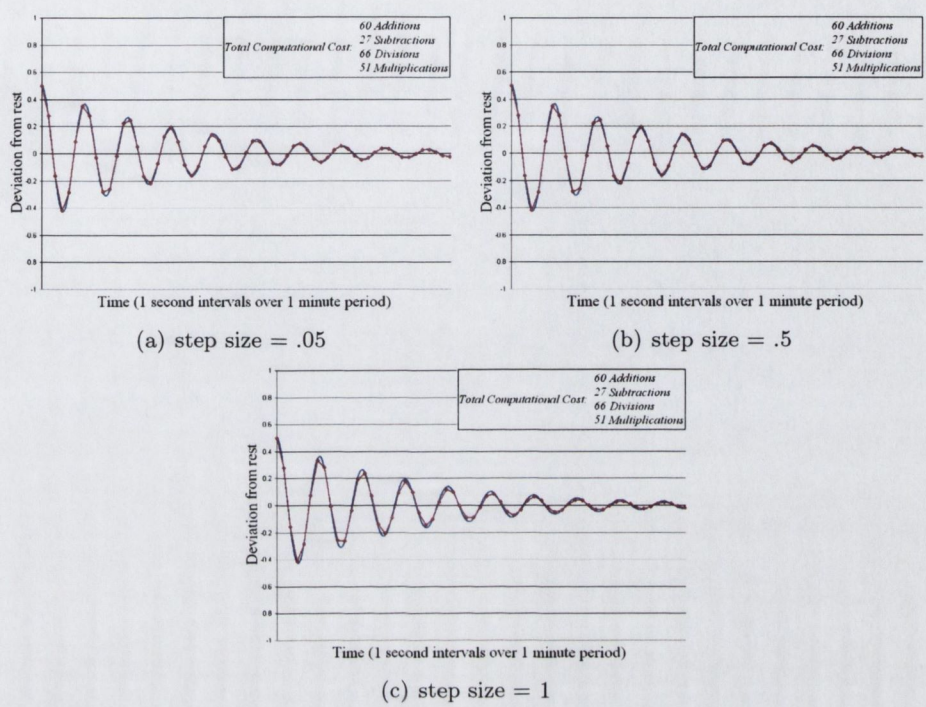


Figure 3.10: Mass spring test of accuracy and stability for 4th Order Runge-Kutta method

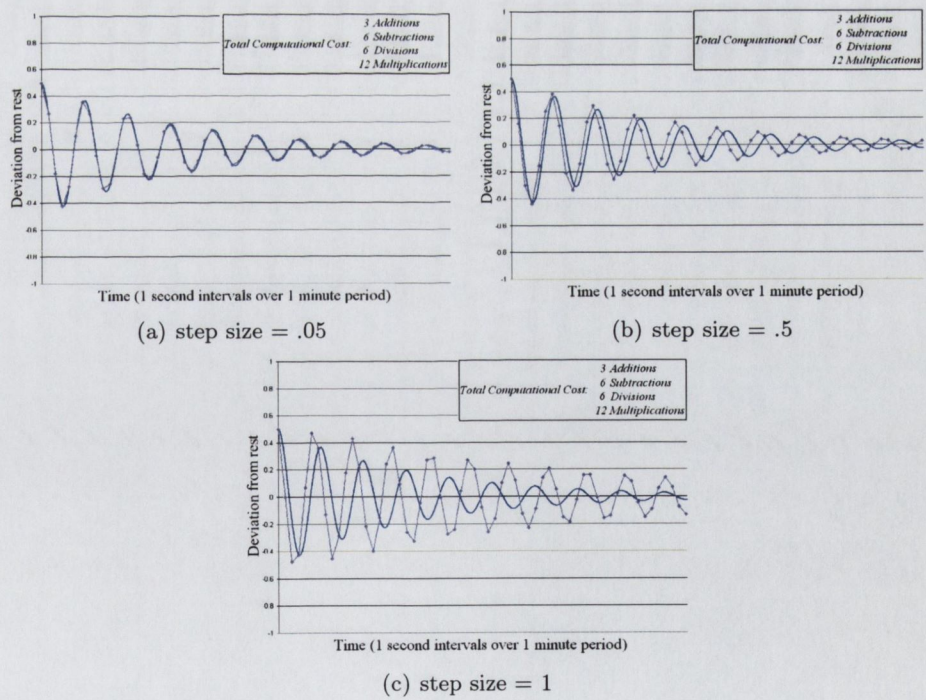


Figure 3.11: Mass spring test of accuracy and stability for basic Verlet method

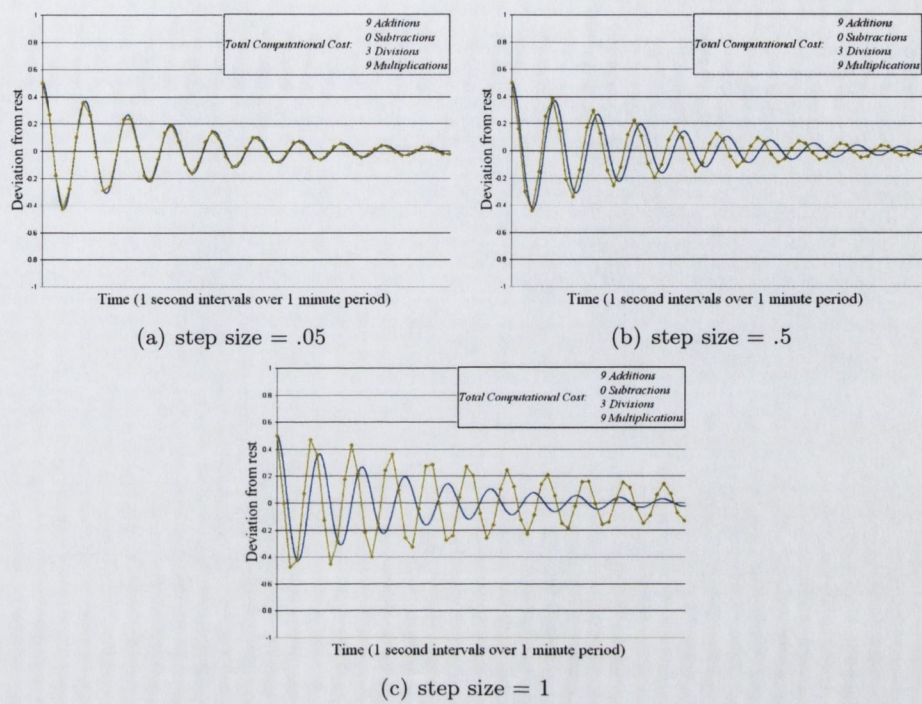


Figure 3.12: Mass spring test of accuracy and stability for Leapfrog Verlet method

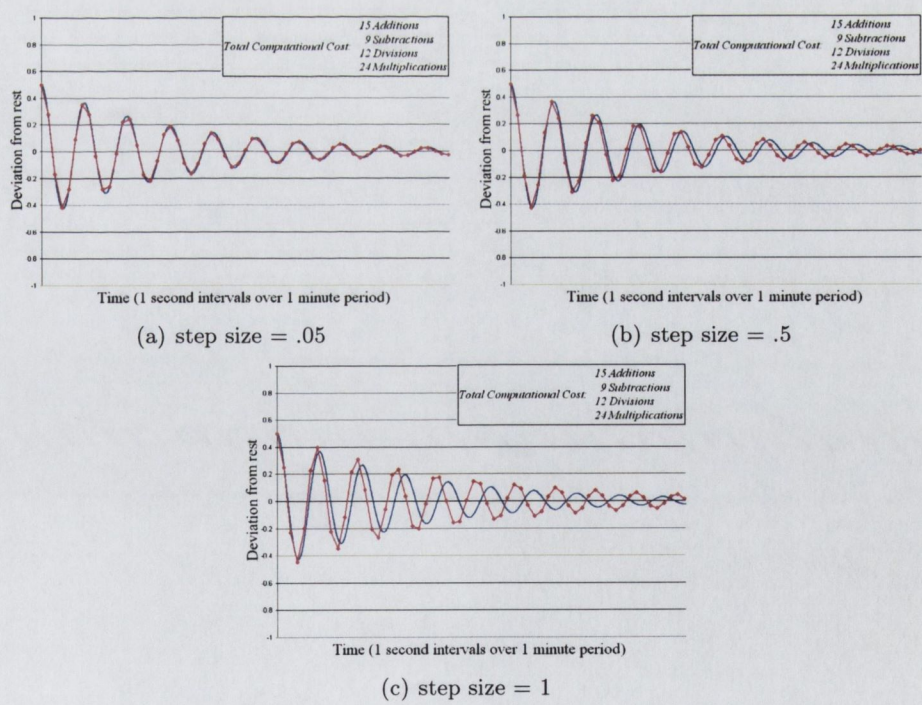


Figure 3.13: Mass spring test of accuracy and stability for Velocity Verlet method

Bearing in mind that the best performance of the common explicit integration methods used in computer graphics previously discussed (i.e. fourth order Runge-Kutta method) requires four sub step passes to achieve its final results at each simulation time loop, the Verlet methods in comparison have shown to be very strong contenders. Given that the best of the Verlet methods (i.e. the velocity Verlet method) gave results almost comparable to the fourth order Runge-Kutta method at only about a quarter of the computational cost this qualifies Verlet methods to be worthy candidates for further investigation to expanding such schemes for dealing with rigid body dynamics within a dynamics system.

3.9 Conclusions

In this chapter we have introduced various integration methods used for resolving initial value problems. These problems are a family of ordinary differential equation problems frequently encountered when simulating dynamic system states based on Newton's equations of motion over time. We have shown how such initial value problems may be solved using ideas based on the Taylor series. Of the techniques within this chapter, we have seen three of the most well known and popular explicit techniques used by the graphics community, these being the Euler, Midpoint and fourth order Runge-Kutta methods. The various advantages and disadvantages of each of the methods were presented to the reader. Methods such as adaptive step sizing and implicit formulation of such integration schemes were discussed as an implementation alternative to the basic algorithms for improving overall accuracy and stability within simulations and examples of their utilisation within the graphics community were given.

As mentioned at the start of the chapter, methods for resolving ODEs are vast. We have only scratched the surface of what is available for solving ODEs. One can literally write an entire book (and many have) covering the subject. There are many other algorithms and different approaches available out there for resolving initial value problem ODEs that we have not discussed. These range from *predictor corrector* methods, the Bulirsch-Stoer method, and Rosenbrock integrators. Each of these methods have varying success as far as stability, accuracy and computational efficiency is concerned. Recall that there is no such thing as a "perfect" integrator. We refer the interested reader to the cited texts within this chapter, in particular to Press et al [126] and Lambert [94] which both provide very comprehensive studies on solving ODE problems.

In our discussions on the Verlet schemes we have neglected to consider rotational terms

in our integrations. The basic Verlet schemes described in this chapter do not take into account such terms as they were designed with the idea of particle motion in mind whereby rotation and orientation are not a concern. Those researchers in the graphics community who have utilised Verlet schemes have done so in systems that can be modelled ignoring rotational dynamic terms. Those that have attempted to deal with rigid bodies have done so through alternative representations that do not require the integration of rotational states. We shall discuss such systems in the next chapter. We shall also consider the Verlet schemes discussed in this chapter for integrating rigid body motion whereby orientational terms must also be accounted for.

Chapter 4

Verlet for Rigid Body Dynamics

So far, in our treatment of Verlet algorithms we have only considered the linear terms of Newton's equations of motion. This is so because, in molecular dynamics, we are integrating for particles that we assume for simplicity do not have rotational properties. The Verlet methods were designed without consideration for rotational dynamics but rather for the optimisation of integration of translational motion only. However, if we wish to simulate rigid bodies, we must also take orientational information into account. Additional modifications to the previously discussed Verlet methods must be considered if we are to use them for rigid body dynamics. It would be most advantageous to be able to maintain the benefits provided to us by the Verlet schemes in the translational motion domain, in the rotational domain also.

4.1 Rotational Properties Of Rigid Bodies

Within the states description of a rigid body, for every linear component there exists an angular equivalent. We have already seen these terms in Table 2.1 at the beginning of Chapter 2, however we did not go into any detail in describing them. We give a brief overview here of the angular components associated with rigid body states as this will prove to be advantageous for later discussions within this chapter.

As previously mentioned at the beginning of Chapter 2, rigid bodies have an orientation as well as a position in space described by the position of its centre of mass x . This orientation can be described through a 3x3 orientation matrix R . This orientation matrix R is essentially a rotation matrix that describes the rotation of any point on the rigid body about the body's centre of mass. However, as we progress in time through our simulation,

numerical drift will eventually creep into this matrix R causing it to become less and less of a rotation matrix. To account for this we have to ensure that orthogonality is maintained at all times for R throughout the simulation. A better solution and one that is employed by most researchers is to represent the orientation of a rigid body in terms of a unit quaternion q instead. A quaternion only has four parameters in contrast to the nine parameters within a matrix. As such, there are less redundant variables to worry about with a quaternion when describing rotations than there is with matrices and hence there is far less numerical drift associated with a quaternion. If any numerical drift does creep into the quaternion, a simple re-normalisation is all that is required to correct the problem. In our work, we describe all rigid body orientations within our simulations with a unit quaternion, converting back to matrix form whenever appropriate.

Now, let us consider the time derivatives of the states of a rigid body. We can equivalently consider the linear velocity v of a rigid body to be the derivative of its linear position x over time (i.e. $\dot{x}_t = v_t$). Similarly, we may consider the linear acceleration a (or written in another way $\frac{F}{M}$ where M is the mass) to be the derivative of the linear velocity v over time (i.e. $\ddot{x}_t = \dot{v}_t = \frac{F_t}{M}$). For the linear and angular momentums, the force F and torque τ respectively describe the time derivatives of these terms. However, when it comes to describing the differential equation for orientation over time, things get a little more complicated as there is no simple direct relationship as in the linear case. If describing orientation with a unit quaternion q , the rate of change of orientation over time (i.e. $\dot{q}_t$) is expressed as:

$$\dot{q}_t = \frac{1}{2}\omega_t q_t \quad (4.1)$$

This above equation relates the angular velocity term ω_t to the orientation term q_t . We will not provide a derivation for this equation here but rather point the reader to Appendix B of [144] and Chapter 10, Section 10.6 of [50] for two well explained derivations. In the above equation, we view ω_t not as a vector (which is the normal case) but rather as a quaternion with a scalar value of zero (i.e. $[0, \omega_t]$). It should be noted here that the ω_t term in Equation 4.1 is not a unit quaternion and so neither is the result of the resultant quaternion multiplication, $\dot{q}_t$. As such, it must be made clear that the $\dot{q}_t$ term in this instance does not represent rotation but rather the rate of change of the body's orientation over time. So when updating our rigid body states over time, we cannot directly multiply $\dot{q}_t$ with q_t as if composing rotations as we will not end up with a new unit quaternion describing the new orientation of our rigid body.

Let us take an example of states updating through the use of a simple explicit Euler integration scheme. Following from the explicit Euler integration structure as described by Equation 3.2 in Chapter 3 we may express the update to time $t + \Delta t$ of both the linear and angular states P , L , x and q as:

$$P_{t+\Delta t} = P_t + \Delta t \dot{P}_t = P_t + \Delta t F_t \quad (4.2)$$

$$L_{t+\Delta t} = L_t + \Delta t \dot{L}_t = L_t + \Delta t \tau_t \quad (4.3)$$

$$x_{t+\Delta t} = x_t + \Delta t \dot{x}_t = x_t + \Delta t v_t \quad (4.4)$$

$$q_{t+\Delta t} = q_t + \Delta t \dot{q}_t = q_t + \Delta t \frac{1}{2} \omega_t q_t \quad (4.5)$$

Note that for updating the orientation of the rigid body we abide by the truncated Taylor series structure of the Euler integration scheme and sum q_t to the $\dot{q}_t$ term. Most introductory text in physically base modelling of rigid bodies, such as [72] for example, will present the updating of orientation over time in this format as it is the most straight forward way of doing so. However, the result of updating orientations in this way can make the quaternion leave the unit sphere for relatively large time steps. As a result we must re-normalise $q_{t+\Delta t}$ to bring the quaternion back onto the unit sphere again. If we had calculated $q_{t+\Delta t}$ using products as if composing rotations in the following manner:

$$q_{t+\Delta t} = q_t \dot{q}_t \Delta t$$

an incorrect result would be obtained for the reasons mentioned earlier. If the above result was applied to our animations, visually what we would see is the object suddenly disappearing as it is sent to some unknown spot off screen.

It is however possible to keep the $q_{t+\Delta t}$ result on the unit sphere and to avoid re-normalising if we utilise an exponential mapping [63] of the angular velocity vector ω_t instead of the $\dot{q}_t$ term, and then re-implementing Equation 4.5 using products instead. Such a formulation is thus expressed in the following form:

$$q_{t+\Delta t} = e^{\omega_t \Delta t} q_t \quad (4.6)$$

Johnson [83] refers to this method of integrating for the quaternion q over time as *Intrinsic Euler integration*. Equation 4.6 is equivalent to that of Equation 4.5 and is thus still only first order accurate. We have only expressed the Euler integration step, in this instance, in

another form¹. Johnson gives a somewhat not so straight forward derivation of the above equation through the use of *Lie* algebra in the Appendix C of his PhD thesis. For updating the quaternion orientation state q in our work, we do not use this *intrinsic* form as it is only still first order accurate since it is essentially an Euler update. We chose instead to pursue another course and try to adapt the structure of Verlet algorithms to account for both linear and rotational motion instead.

We should mention here why we did not present the equation for angular acceleration in our discussions within this section. Normally it is more advantageous to work in terms of momentums, especially angular momentum as this remains constant through time given that no torque is acting on the rigid body. This however is not the case with angular velocity. When updating our rigid body states, we may treat the velocity terms as auxiliary variables so that we do not need to update them through integration. Instead we can update the velocity terms to the next time step once we have obtained the updated momentum terms via our preferred integration algorithm. Once the updated momentum terms have been obtained, all we need to do is to apply the following equations :

$$v = \frac{P}{M} \tag{4.7}$$

$$\omega = I^{-1}L \tag{4.8}$$

to obtain our required velocity terms. The equations for linear velocity (i.e. Equation 4.7) and angular velocity (i.e. Equation 4.8) given above are simply derived from the linear and angular equations of momentum respectively. From this, in general, there is no need for us to know the acceleration term (either in the linear or angular domain) at any stage during our states update unless we wish to update the velocity terms through an integration step. The I variable in Equation 4.8 is simply the rotational equivalence of the mass of the rigid body, known as the *inertia tensor*. It describes the mass distribution of the rigid body and affects how a body will spin in space. We have already discussed this term in Chapter 2.

Let us next present some of the pass works done by both computer graphics and molecular dynamics researchers in adapting Verlet algorithms for rigid body motion.

¹Indeed, we have conducted an elementary angular accuracy test using a kinematically constrained rigid rod similar to the angular accuracy tests discussed at the end of this chapter and have found that Equation 4.6 exhibited about the same angular accuracy as if we were to integrate using Equation 4.5. This equivalency may not be true for stability, however, we did not test for this case.

4.2 Dealing with Rotational Dynamics

Most graphics researchers that have adapted Verlet methods have done so for problems that do not require them to integrate for rotational dynamic terms. There have, however, been a few who have tried to adapt Verlet methods for rigid body simulations but mostly they have bypassed having to deal with the rotational properties of rigid bodies by using some sort of alternative representation. For example, Jakobsen [80] has utilised the basic Verlet scheme to simulate cloth, rag dolls and rigid bodies. However, he treats the rigid bodies as a set of constrained points (used for modelling a tetrahedron proxy within the rigid structure) that are updated individually without consideration for the bodies' rotational dynamic properties. Rather the algorithm used for the enforcement of the positional constraints through successive kinematic adjustment for the points representation of the rigid body is sufficient to produce believable rotational rigid body behavior. However, Jakobsen states himself that the point-based tetrahedron representation approach for rigid bodies still needs some work. Another example within the graphics community of Verlet integrators being used in dynamics systems that incorporate rigid body motion in some shape or form is a system presented by Kacic-Alesic et al [84]. This system, used by ILM in its film productions, is built on a mass springs systems capable of representing rigid bodies through stiff spring constraints. Kacic-Alesic et al describe their system as using Verlet integration for advancing the state of the spring system. They state that the rotational properties of rigid bodies may also be incorporated into the solver and go on to present rotational equivalent formulations for resolving rigid bodies in the system. However, the paper does not give us any stability, accuracy or timing results for these formulations. In fact, we are not even sure if these formulations are actually used within the system as Kacic-Alesic et al later describe their system as building a mass spring system out of the shape of a given dynamic object, then dealing with them as such.

Within the molecular dynamics community, if the need to simulate molecular bonds arises, usually such bonds is represented as a bunch of distance constrained molecules with no orientational information. The motion of the molecular bonds are integrated forward in time by integrating each "molecular particle" in the bond using either the popular Leapfrog or Velocity Verlet methods. The distance constraints on each molecular particle within the molecular bond are then usually enforced using either the popular SHAKE [67] or the velocity version, RATTLE [1] algorithms or some variant of such. These are iterative algorithms that ensure that if molecular particles break their distance constraint from their neighbours then they are iteratively adjusted to a position (and/or velocity) that enforces

the distance constraint using some sort of successive relaxation method ².

However, simulating molecular bonds in this fashion can get quite computationally expensive and it has been suggested by some molecular dynamics researchers to treat molecular bonds as rigid bodies with orientational properties rather than as a bunch of constrained molecular particles. In this way, each molecular particle in the bond does not have to be accounted for individually and explicitly constrained during each integration loop but rather considered as a whole, as part of a rigid body. Their position in space can then be easily obtained if needed by simple calculation using the rigid body's center of mass and orientation in the space that they are associated with. This representation is similar to how graphics researchers view rigid bodies in dynamic simulations. Rather than viewing them as a collection of individual atoms or particles constrained to some particular rigid form in space, instead rigid bodies are viewed as having positional information described by a center of mass and orientational information described by an orientation matrix. It is very common for reasons of numerical drift when using a matrix representation for orientation, to represent the rigid body orientation using a quaternion instead. We later convert back to matrix form whenever necessary. This is also very common practice in molecular dynamics. In our implementation of rigid body states in our time adaptive system, we represent all orientational states of a rigid body as a quaternion, referring back to a matrix representation whenever it is necessary to do so.

Early work with orientational dynamics in molecular dynamics simulation research utilised high order Gear predictor corrector [54] algorithms for integration of rigid body angular states. These methods were shown to be accurate at very small time steps and became quickly unstable at larger steps. Also they were not time reversible as is the case with Verlet methods. These methods were later coupled with Verlet methods such that translational motion was dealt with using Verlet integration and rotational motion with Gear predictor methods (as the former did not extend to rotational terms). In recent times, one of the most popular algorithms used in molecular dynamics to integrate rigid body motion whenever Verlet integration is considered is an *implicit rotational Leapfrog* algorithm attributed to Fincham [51]. Fincham was one of the first researchers in molecular dynamics to try to adapt the Verlet schemes to incorporate rotational motion terms. Other researchers since Fincham, such as Kol et al [91] and Omelyan [116, 117], have proposed numerous

²It is of interest to note here that Jakobson [80] also recently adopted a very similar approach within the graphics community to simulate rigid links for rag dolls, cloth and to some extent rigid bodies for the computer game *Hitman*. It was probably his paper in GDC 2001 that demonstrated to many graphics researchers the potential advantages of using Verlet methods for physical simulation as an alternative to Euler or Runge-Kutta methods.

alternative strategies for incorporating Verlet algorithms into the rotational domain. These algorithms are all based either on the reformulation or reordering of the angular variables during integration to better conform to the Verlet integration algorithm or through further modification of the SHAKE and/or RATTLE algorithms to account for angular properties.

In our work, we chose to further investigate Fincham's methods not only due to its popularity within the molecular dynamics community but also because of its algorithmic simplicity and the fact that we felt it could be more easily adapted to our work with rigid bodies. As such, in this chapter, we shall present Fincham's explicit and implicit *Rotational Leapfrog Verlet* algorithm for the integration of rigid body motion. We shall present the underlying formulations for dealing with orientation and angular velocity terms along with a discussion of these algorithms. We shall present an investigation of the advantages and disadvantages of Fincham's algorithms and discuss various solutions to adapting the Verlet algorithms for dealing with rigid body simulations, especially within our system.

4.2.1 Explicit Rotational Leapfrog

For linear and angular momentum, the other state variables that we may wish to update in a rigid body simulator, it is possible to update these using the same formulation as for linear velocity in both the Leapfrog and Velocity Verlet methods without the need for any additional intermediate computational steps. As before, we treat the linear position, x , of the rigid body as the position of the center of mass of the body when we do the positional update. The only difficulty that arises is when we wish to integrate for orientation and angular velocity using the previous Leapfrog or Velocity Verlet formulations. The problem lies in the fact that angular velocity is dependent on the body's orientation through the inertia tensor I , which cannot be handled intuitively with the Leapfrog or Velocity Verlet algorithms. So this makes it quite difficult to use any of the Verlet algorithms directly to calculate the angular velocity and orientation of a rigid body without having to resort to using intermediate steps. It is not necessary to utilise any of the Verlet formulations previously discussed in order to resolve the new angular velocity update but rather we can calculate the angular velocity as an auxiliary variable as soon as we have the orientation of the rigid body. Let us consider the Leapfrog algorithm for advancing linear position and apply this to rigid body orientation represented as a quaternion, q . Conforming to the Leapfrog structure for updating position, we express the update of orientation as:

$$q_{t+\Delta t} = q_t + \dot{q}_{t+\frac{\Delta t}{2}} \Delta t \quad (4.9)$$

As before, updating in this fashion requires us to re-normalise to ensure that the updated orientation quaternion $q_{t+\Delta t}$ remains on the unit circle. Again, it would be incorrect for us to directly use the products, as if composing rotations, rather than summing in this instance for the same reasons as discussed in Section 4.1. We could reformulate the above equation using a similar exponential mapping approach to that previously described (i.e. Equation 4.6), for the explicit Euler integration, however we choose here in our discussion of this explicit rotational Leapfrog algorithm to not stray too much from the familiar basic Leapfrog Verlet structure.

In the above equation, let us expand the derivative term to include the angular velocity, ω , of the rigid body by the following formulation:

$$\dot{q}_{t+\frac{\Delta t}{2}} = \frac{1}{2}\omega_{t+\frac{\Delta t}{2}}q_{t+\frac{\Delta t}{2}} \quad (4.10)$$

So rewriting equation 4.9 we get

$$q_{t+\Delta t} = q_t + \frac{1}{2}\omega_{t+\frac{\Delta t}{2}}q_{t+\frac{\Delta t}{2}}\Delta t \quad (4.11)$$

Immediately we can see a potential problem. If one recalls, with the Leapfrog (and Velocity) Verlet algorithms we first calculate the positional derivative term (i.e. the velocity) at time $t + \frac{\Delta t}{2}$ and then use this term to advance our positional term. This was not a problem since the advancement of the velocity is force dependent. It was possible to approximate the updated velocity term based on the known force acting on the body at time t . The same idea applies here by the virtue that we are using the same formulation but if we look closer at equation 4.11 we will see that it is dependent on the half step orientation as well as the half step angular velocity. However, we also need to know the half step orientation in order to calculate the half step angular velocity! This dictates the necessity to approximate the half step orientation and angular velocity as auxiliary variables before we can obtain the full step orientation and angular velocity which is computationally wasteful. Let us write down the half step auxiliary calculations:

$$q_{t+\frac{\Delta t}{2}} = q_t + \frac{1}{2}\left(\frac{1}{2}\omega_t q_t \Delta t\right) \quad (4.12)$$

$$I_{t+\frac{\Delta t}{2}}^{-1} = R_{t+\frac{\Delta t}{2}} I_{body}^{-1} R_{t+\frac{\Delta t}{2}}^T \quad (4.13)$$

$$\omega_{t+\frac{\Delta t}{2}} = I_{t+\frac{\Delta t}{2}}^{-1} L_{t+\frac{\Delta t}{2}} \quad (4.14)$$

In the above, I^{-1} is the inverse world space inertia tensor of the rigid body, I_{body}^{-1} is the body

space inverse inertia tensor. All other terms are as before. R is easily got by an elementary quaternion to Euler conversion. Once the half step orientation and angular velocity have been determined we may use equation 4.11 to work out $q_{t+\Delta t}$ and from there, if we so wish, we may approximate the full step angular velocity $\omega_{t+\Delta t}$ by the following equations:

$$I_{t+\Delta t}^{-1} = R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T \quad (4.15)$$

$$\omega_{t+\Delta t} = I_{t+\Delta t}^{-1} L_{t+\frac{\Delta t}{2}} \quad (4.16)$$

If we do not approximate $\omega_{t+\Delta t}$ at this time step then during the next iteration step, providing we are keeping the angular velocity as a state variable, we will be one half time step behind (as is the nature of the Leapfrog Verlet algorithm) and will need to calculate L_t at that time step in order to obtain ω_t for the half step orientation calculation of Equation 4.12. The reader might have noticed that, because we are using the Leapfrog algorithm to update angular momentum, we are always half a time step behind the current time before the update step and half a time step ahead of the current time after the update step (this also holds true for linear momentum and velocity). So we updated the angular momentum to time $t + \frac{\Delta t}{2}$ and used this approximation to approximate $\omega_{t+\Delta t}$ in Equation 4.16.

Algorithm 1 describes this Leapfrog integrator as applied to all essential state variables that might be used to represent a rigid body. This algorithm encapsulates all the ideas presented in this subsection to date in the proper execution order. It is of interest to note that with Algorithm 1, all momentum and linear velocity terms are a half step behind the spatial terms, except for angular velocity which is kept at the full time step.

The explicit rotational Leapfrog algorithm just described is analogous to the explicit rotational Leapfrog algorithm described by Fincham [51]. The main differences between Fincham's algorithm and the algorithm that we have just described is that Fincham suggests that we first start by advancing the angular momentum $L_{t-\frac{\Delta t}{2}}$ (which is half a time step behind the previous update) to time t . The angular momentum at time t is then rotated to body space, whereby the angular velocity in body space at time t is calculated. The reason for this body space coordinate conversion is so that the world space inertia tensor, which is fixed in body space, never needs to be known. The body space angular velocity at time t is then used in an alternate body space formulation of the orientational derivative term of Equation 4.10 to advance the orientation q (in world space) to its half time step location using a similar formulation to Equation 4.11. After this is done, the half step angular momentum and velocity is calculated using the same methods (again converted to

Algorithm 1 Full Explicit Rotational Leapfrog Verlet. Integrates for full set of state variables of a rigid body.

Input : List of rigid body states, Number of rigid bodies to update n , Timestep Δt

Output : Updated list of rigid body states

EXPROTLEAPFROG($BodyStates, n, \Delta t$)

for all rigid bodies i up to n **do**

$$v_{t+\frac{\Delta t}{2}} \leftarrow v_{t-\frac{\Delta t}{2}} + \frac{F_t}{M} \Delta t \quad \{\text{linear velocity update}\}$$

$$P_{t+\frac{\Delta t}{2}} \leftarrow P_{t-\frac{\Delta t}{2}} + F_t \Delta t \quad \{\text{linear momentum update}\}$$

$$L_{t+\frac{\Delta t}{2}} \leftarrow L_{t-\frac{\Delta t}{2}} + \tau_t \Delta t \quad \{\text{angular momentum update}\}$$

$$x_{t+\Delta t} \leftarrow x_t + v_{t+\frac{\Delta t}{2}} \Delta t \quad \{\text{center of mass position update}\}$$

 {temporary auxiliary variable calculations}

$$q_{t+\frac{\Delta t}{2}} \leftarrow q_t + \left(\frac{1}{2}\right) \frac{1}{2} \omega_t q_t \Delta t \quad \{\text{half step orientation}\}$$

$$I_{t+\frac{\Delta t}{2}}^{-1} \leftarrow R_{t+\frac{\Delta t}{2}} I_{body}^{-1} R_{t+\frac{\Delta t}{2}}^T \quad \{\text{half step world space inverse inertia tensor}\}$$

$$\omega_{t+\frac{\Delta t}{2}} \leftarrow I_{t+\frac{\Delta t}{2}}^{-1} L_{t+\frac{\Delta t}{2}} \quad \{\text{half step angular velocity}\}$$

 {end of temporary auxiliary variable calculations}

$$q_{t+\Delta t} \leftarrow q_t + \frac{1}{2} \omega_{t+\frac{\Delta t}{2}} q_{t+\frac{\Delta t}{2}} \Delta t \quad \{\text{orientation update}\}$$

$$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$$

$$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\frac{\Delta t}{2}} \quad \{\text{angular velocity update}\}$$

end for

body space) and finally, the orientation $q_{t+\Delta t}$ at time $t + \Delta t$ is calculated. The full step angular velocity $\omega_{t+\Delta t}$ is never calculated but always left on the half step, conforming to the idea of the Leapfrog Verlet algorithm. Linear momentum is never discussed in Fincham's algorithm description but this is a trivial matter. If necessary we can either calculate the linear momentum (which will be at the half step) directly by using the known velocity term (i.e. $P = Mv$) or else by integration as in Algorithm 1.

One final thing to note about the treatment of terms in Algorithm 1 is that, instead of using the acceleration term a directly as we have been doing in our equation formulations to date, we have replaced it with the force over mass (i.e. $\frac{F}{M}$) equational equivalence. We do this as it is more common for a rigid body's state descriptions to contain the force acting on the body at that time with the acceleration term easily derived from Newton's second law whenever needed. We keep this convention for all proceeding algorithms.

4.2.2 Fincham's Implicit Rotational Leapfrog

Fincham also presents an implicit approach to the rotational Leapfrog problem whereby velocities and momentum are approximated at the full step as temporary auxiliary terms in the integration step and then used along with the time t velocity approximation to obtain the linear positional update. With the implicit approach, the need for mid-step calculations of the orientation or angular velocity terms, as in the explicit approach, is eradicated. Instead, the orientation is calculated implicitly through an iterative refinement process with angular velocity easily approximated from this result.

In Fincham's implicit rotational Leapfrog algorithm, he again translates the angular velocity into body space coordinates and uses a body space equivalent to the derivative term in the $q_{t+\Delta t}$ formulation as he had done in the explicit rotational Leapfrog scheme. In our description of this algorithm we shall approximate all terms in world space coordinates as we did in the previous explicit Leapfrog algorithm. We do this to keep conformity with the classical description of rigid body states in physical simulations for computer graphics. The underlying idea of the approach and overall description of the algorithm is not affected by this slight change in any way.

The translational terms in Fincham's implicit rotational Leapfrog are integrated slightly differently to the explicit Leapfrog algorithm. However, it is not hard to see the equivalence of this new formulation to the original Leapfrog algorithm. The velocity terms are calculated at all three time steps, t , $t + \frac{\Delta t}{2}$ and $t + \Delta t$ and then the linear position is updated using the velocities at time t and $t + \Delta t$ instead of the half step velocity directly:

$$v_t = v_{t-\frac{\Delta t}{2}} + \frac{1}{2}a_t\Delta t \quad (4.17)$$

$$v_{t+\frac{\Delta t}{2}} = v_{t-\frac{\Delta t}{2}} + a_t\Delta t \quad (4.18)$$

$$v_{t+\Delta t} = v_t + a_t\Delta t \quad (4.19)$$

$$x_{t+\Delta t} = x_t + \frac{1}{2}(v_t + v_{t+\Delta t})\Delta t \quad (4.20)$$

Only the half step velocity $v_{t+\frac{\Delta t}{2}}$ term is kept between steps. The other two velocities v_t and $v_{t+\Delta t}$ are considered temporary variables and are not saved between time steps. Fincham does not mention the integration of the linear momentum term in his description of his algorithm but if needed, the linear momentum can easily be incorporated into the algorithm in the same way we handle the linear velocity and angular momentum terms. In fact, in describing Fincham's implicit rotational algorithm in Algorithm 2, we include the linear momentum term for completeness. For angular momentum, L , this is treated in the same way as the linear velocity term:

$$L_t = L_{t-\frac{\Delta t}{2}} + \frac{1}{2}\tau_t\Delta t \quad (4.21)$$

$$L_{t+\frac{\Delta t}{2}} = L_{t-\frac{\Delta t}{2}} + \tau_t\Delta t \quad (4.22)$$

$$L_{t+\Delta t} = L_t + \tau_t\Delta t \quad (4.23)$$

with τ_t denoting the torque at time t . Again, the steps are analogous to that of the explicit rotational Leapfrog method. The only difference here is that we calculate all three time steps for processing later, again only keeping the half step information between steps as in the same manner as the linear velocity term. It is possible for us to calculate the angular velocity ω_t at time t with the information so far through Equation 4.14 (of course replacing the $t + \frac{\Delta t}{2}$ terms with the t equivalent terms). Fincham suggests the we first rotate the angular momentum to its body space equivalent, do the calculations in body space and then rotates back to world space again. This extra step is unnecessary and is computationally inefficient. Thus we remain in world space at all times possible. In order to calculate $\omega_{t+\Delta t}$ we will have to utilise the following equation,

$$\omega_{t+\Delta t} = I_{t+\Delta t}^{-1}L_{t+\Delta t}$$

This time in calculating the angular velocity term $\omega_{t+\Delta t}$ we use the full time step for all

terms in the equation. In order to resolve the above equation it is necessary to know the world space inverse inertia tensor term $I_{t+\Delta t}^{-1}$ beforehand. Unfortunately for us, this term is dependent on the orientation of the body at time $t + \Delta t$. So our next step is to calculate the orientation at time $t + \Delta t$. In this instance, the update equation for our orientation term (represented as a quaternion) is

$$q_{t+\Delta t} = q_t + \frac{1}{2} \left(\frac{1}{2} \omega_t q_t + \frac{1}{2} \omega_{t+\Delta t} q_{t+\Delta t} \right) \Delta t \quad (4.24)$$

Looking at equation 4.24 we can see that this is an implicit equation for $q_{t+\Delta t}$. At this point in time we do not know what the solutions to $\omega_{t+\Delta t}$ and $q_{t+\Delta t}$ are, which appear on the right hand side of the equation. Fincham suggests solving for $q_{t+\Delta t}$ by successive relaxation. By this, it is meant that we iterate through the solution for $q_{t+\Delta t}$, represented by Equation 4.24 taking

$$q_{t+\Delta t} = q_t + \frac{1}{2} \omega_t q_t \Delta t$$

as an initial guess until a reasonably acceptable solution is reached. We can then also use this information to take a guess at the angular velocity $\omega_{t+\Delta t}$ at each iteration, which is needed for the overall approximation of $q_{t+\Delta t}$. Convergence to an “acceptable”³ solution is fast and usually takes only about three to four iterations. We present an ever so slightly modified version of Fincham’s implicit rotational Leapfrog algorithm in Algorithm 2. The changes to this algorithm are those that we have just discussed. Before entering the function described by Algorithm 2 it is assumed that the following full state set is known and kept for the rigid body representation:

³we say acceptable here because it is not possible to ever get the true solution to an integration problem no matter what integration method we use, due to the discrete nature of the problem formulations, but rather as close an approximation to the true solution as possible.

State Variables

$v_{t-\frac{\Delta t}{2}}$	linear velocity
ω_t	angular velocity
$P_{t-\frac{\Delta t}{2}}$	linear momentum
$L_{t-\frac{\Delta t}{2}}$	angular momentum
x_t	center of mass position
q_t	orientation

Constant Variables

I_{body}^{-1}	inverse body space inertia tensor
M	mass of rigid body

Computed Variables

F_t	force on the rigid body
τ_t	torque on the rigid body

The time that each variable is at on entering the ODE solver is also represented for each variable above. Of the state set listed, the mass of the body, M and the inverse body space inertia tensor term I_{body}^{-1} are constant and known from the start of the simulation and need not be recalculated at each iteration. The calculated force F_t and torque τ_t at time t are the net force and torque acting on the rigid body so that the states $v_{t-\frac{\Delta t}{2}}$, ω_t , $P_{t-\frac{\Delta t}{2}}$, $L_{t-\frac{\Delta t}{2}}$, x_t and q_t are advanced forward in time with respect to these forces. We can see from the list that, due to the nature of the algorithm, the momentum terms and the linear velocity term are always half a time step behind so that they appear to leapfrog over the current time at each update iteration. All other state variables are evaluated from time t . It may not be very desirable to have the state variables at different times from each other like this, especially when we need to resolve collisions and contacts. Also, having to calculate the velocity and momenta at three time intervals at t , $t + \frac{\Delta t}{2}$ and $t + \Delta t$ is not very computationally efficient.

According to Fincham's tests on molecular liquids using both the explicit and implicit versions of his algorithms, he found that the implicit method far outperformed the explicit variant for the rotational motion of the water molecules. His tests showed that, although the implicit rotational Leapfrog required an iterative refinement approach to obtain an approximation to the orientational values, the approach converged very quickly to a solution. His tests also showed that the explicit approach gave very high erroneous results for relatively large time steps as compared to the implicit approach which handled these large time steps relatively well. The stability and accuracy of Fincham's implicit rotational Leapfrog

Algorithm 2 Fincham's implicit rotational Leapfrog algorithm. Slightly modified to use world space angular velocity calculations rather than body space.

Input : List of rigid body states, Number of rigid bodies to update n , Timestep Δt

Output : Updated list of rigid body states

FINCHIMPROTLEAP($BodyStates, n, \Delta t$)

for all rigid bodies i up to n **do**

 {calculate initial temporary terms}

$v_t \leftarrow v_{t-\frac{\Delta t}{2}} + \frac{F_t}{M} \Delta t$ **for** $j=1$ to $MaxIterations$ **do**

$L_t \leftarrow L_{t-\frac{\Delta t}{2}} \quad q_{t+\Delta t} \leftarrow q_t + \frac{1}{2} \left(\frac{1}{2} \omega_t q_t + \frac{1}{2} \omega_{t+\Delta t} q_{t+\Delta t} \right) \Delta t$

$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$

$v_{t+\frac{\Delta t}{2}} \leftarrow v \quad \omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t}$ {half step linear velocity update}

$P_{t+\frac{\Delta t}{2}} \leftarrow P_{t-\frac{\Delta t}{2}} + F_t \Delta t$ {half step linear momentum update}

$L_{t+\frac{\Delta t}{2}} \leftarrow L_{t-\frac{\Delta t}{2}} + \tau_t \Delta t$ {half step angular momentum update}

 {calculate full time step temporary terms}

$v_{t+\Delta t} \leftarrow v_t + \frac{F_t}{M} \Delta t$

$L_{t+\Delta t} \leftarrow L_t + \tau_t \Delta t$

$x_{t+\Delta t} \leftarrow x_t + \frac{1}{2} (v_t + v_{t+\Delta t}) \Delta t$ {full step position update}

 {initial guess at orientation and angular velocity}

$q_{t+\Delta t} \leftarrow q_t + \frac{1}{2} \omega_t q_t \Delta t$

$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$

$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t}$

$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$ **for** $j=1$ to $MaxIterations$ **do**

$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t} \quad q_{t+\Delta t} \leftarrow q_t + \frac{1}{2} \omega_t q_t + \frac{1}{2} \omega_{t+\Delta t} q_{t+\Delta t} \Delta t$

$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$

$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t}$

end for

end for

{MaxIterations can be either 3 or 4}

algorithm over the previously used Gear predictor corrector methods has attracted many researchers in the molecular dynamics community to utilise this algorithm whenever a rigid body approach is decided upon for simulating bonded molecules. This algorithm, however, is not without its faults, as can be seen from the results of tests described in this chapter. Furthermore, we must account for collisions and contacts that occur in our system. It is not sufficient to calculate for such instances using half step approximations as collisions and contacts must be dealt with at the collision time if they are to be dealt with in a proper manner. The next section shall explore the potential problems of these algorithms and shall give suggestions about possible solutions to these problems.

4.3 Collision Considerations

There are many improvements and considerations for the previously discussed algorithms. For example, it would be nice if we did not have to calculate the mid step orientation and angular velocity in order to get to the final solution. Furthermore, it may be somewhat undesirable to have both linear and angular velocity at different time steps from each other as we have done in Algorithm 1. Unfortunately, this was a necessity of the algorithm so as to avoid unnecessary recalculation of ω_t at the next time step iteration. For the purposes of consistency it is desirable to have all state variables at the same time step. It is also most desirable to reduce the number of temporary variables to be calculated at each iteration for algorithmic speed considerations.

Let us first consider the problems of dealing with collisions in our simulations when employing the previously described rotational Leapfrog algorithms. For the sake of simplicity, let us assume that the time of collision is at $t + \Delta t$ (i.e. no back tracking scheme or any other similar schemes are employed). This conforms to the notion of a classical simulation loop whereby collisions are detected and dealt with after the system states have been updated to the next simulation time frame. In the simulation loop after the detection of a collision, we would then proceed to resolve these collisions, producing an updated set of momenta and velocities for the colliding bodies based on calculated impulses⁴ so as to prevent the bodies from travelling through each other but rather causes them to bounce off each other during the next simulation update. Our collision resolver needs to know the pre collision velocities (in this case the velocities approximated from the integrator at time $t + \Delta t$) in order to resolve the new post collision velocities, as the collision formulation requires such

⁴For now we can view an impulse as a large force that acts instantaneously over time to prevent two bodies from interpenetrating. We shall discuss this in more detail in later chapters.

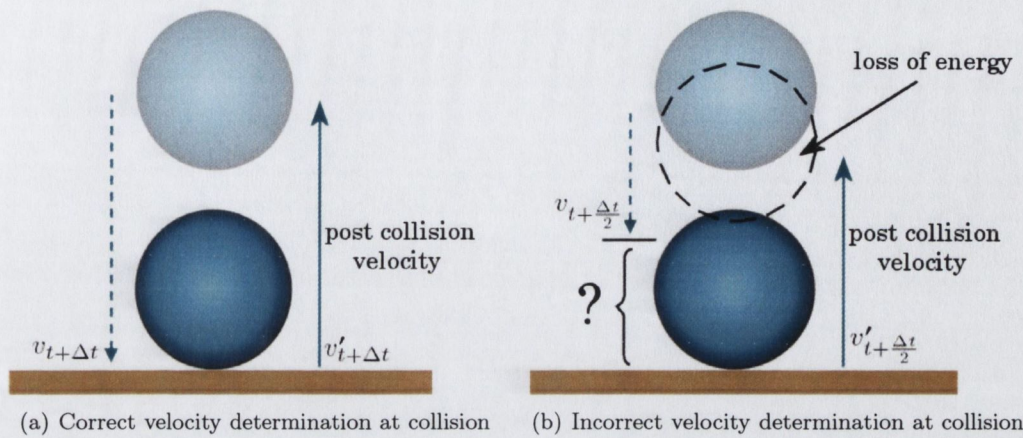


Figure 4.1: Energy dissipation in perfectly elastic collision situation due to using wrong pre collision velocities for collision calculations

information.

4.3.1 Adapting the Explicit Rotational Leapfrog

Maintaining velocities in the half step domain as we do in the Leapfrog like algorithms can potentially be problematic when resolving collisions. If we use the half step velocities, we are then potentially telling the collision resolver that the collision occurred at the half step rather than at the time $t + \Delta t$. However, with respect to position, we are at time $t + \Delta t$ but the resolver does not know this. The resolver will take into account the given velocity and collision points whose positions are specified at time $t + \Delta t$ and calculate an impulse in accordance with this half step velocity and full step position. The impulse will prevent interpenetration and make objects appear to bounce off each other in the simulation but, however, some energy may be lost (if the object was accelerating before collision) or energy gained (if the object was decelerating before collision) due to the fact that we took the velocity that was half a time step behind the true pre collision velocity for our calculations. Figure 4.1 illustrates this problem for an idealised, perfectly elastic, collision (of course there is no such thing in the real world) whereby we expect the rigid body to bounce back to its original starting position each time given that we don't rotate away from our linear path.

We would like to avoid such a situation by giving the collision resolver the right velocity information with respect to the time frame in which the rigid body is positioned in. We could, on finding a collision calculate the velocity at $t + \Delta t$ by taking another temporary

simulation update step, behaving as if the collision never occurred, and then use Equation 3.23 as described in the previous chapter to approximate the pre-collision velocity and momentum. However, the half step velocity during the next simulation time frame update can still potentially cause us problems. The newly updated post collision velocities and momenta are for time $t + \Delta t$, which in the next simulation frame time will be considered to be at time t . We can at this point just consider the post collision velocities to be the new half step velocities used to update the positional terms in the algorithm but, if we are to treat our ODE solver as a black box system, then the solver will want to update the velocity step based on the previous half step at time $t - \frac{\Delta t}{2}$. Either way, our velocity approximations due to the collision will be out by half a time step and, as a result, our simulations will either exhibit an unconditional increase or loss of energy due to the collision. If we are not too concerned with such a situation, as this will be probably be almost unnoticeable for all but perfectly elastic collisions, then such a solution to the problem is perfectly acceptable. Although we keep the angular velocity in the explicit rotational Leapfrog at the full time step, we may still experience such a problem in the rotational domain as we are still dependent on the angular momentum, which is kept at the half step for both the orientation and angular velocity updates.

4.3.2 Adapting the Implicit Rotational Leapfrog

Perhaps a better solution can be found in the implicit rotational Leapfrog algorithm. In this algorithm, the velocities are available to us (be it temporarily) at all three time steps, t , $t + \frac{\Delta t}{2}$ and $t + \Delta t$ and the spatial updates are dependent on the average sum of the velocities at time t and $t + \Delta t$, in effect removing the half step dependence problem as previously described. Instead of treating the full step update on the velocity and momentum terms as temporary variables only to be discarded when done, as we would normally do in the algorithm, it is possible for us to hang onto these state calculations and utilise them in our collision resolution phase for a more appropriate response calculation during collisions. However, if we look closer at the algorithm we will notice that the linear velocity and momentum terms at time t are calculated from the half step terms. So, if we obtain the updated velocities and momenta during a collision and not save them as the half step states but rather keeping them as the full step state variables, these collision updated terms will be overwritten during the time t calculations in the algorithm during the next iteration in the simulation update loop. This would result in objects moving right through each other as if the collision was never detected in the first place. If, on the other hand, we keep

the collision updated variables as the half step updated variables we suffer from the same problem as previously.

One possible solution to resolving this situation is to rewrite Algorithm 2 so that we do not treat the velocity and momentum variables at time t as temporary state variables to be calculated using the half step information and then discarded after we are done with them. Rather, we keep a record of both state variable changes at the full step as well as the half step for each iteration of the simulation loop. If there are no collisions, we go ahead and calculate the update of the state variables as before. However, on detection of a collision we should set some sort of a flag in the rigid body states to indicate to the ODE solver that the rigid body velocities and momenta have been additionally changed outside of the solver due to a collision event. A different approximation for the half step variable updates should then be used and the state variables at time t should not be calculated so as not to overwrite the newly updated states. We present this possible alternative solution in Algorithm 3.

It is possible to further remodify Fincham's implicit rotational Leapfrog algorithm so as to optimise the number of calculations per step and also to maintain timing consistency between state variables for better approximations during collisions and contact situations. We present these suggested modifications below:

```

for all rigid bodies  $i$  up to  $n$  do
  {full step state variable updates}
   $P_{t+\Delta t} \leftarrow P_t + F_t \Delta t$ 
   $L_{t+\Delta t} \leftarrow L_t + \tau_t \Delta t$ 
   $x_{t+\Delta t} \leftarrow x_t + \frac{1}{2}(v_t + \frac{P_{t+\Delta t}}{M})\Delta t$ 

   $v_{t+\Delta t} \leftarrow \frac{P_{t+\Delta t}}{M}$ 

  {now deal with rotational domain implicitly as before}
end for

```

Here, we describe the state of the rigid body system by its position and momentum terms only and leave the velocity terms as auxiliary variables which can be easily calculated without having to go through an integration step once we have obtained the updated momentum and spatial information. We keep all state variables at time t and update them to time $t + \Delta t$ without ever having to consider going through a half time step as we did in the

Algorithm 3 Adapting the Implicit Rotational Leapfrog algorithm for dealing with collisions. Rigid body states contain a *Colliding* flag to indicate whether or not rigid body is involved in a collision

Input : List of rigid body states, Number of rigid bodies to update n , Timestep Δt

Output : Updated list of rigid body states

COLIMPROTLEAP(*BodyStates*, n , Δt)

 for all rigid bodies i up to n do

 if *Colliding* then

 {Don't calculate v_t , L_t or P_t but rather use the collision resolved variables}

$$v_{t+\Delta t} \leftarrow v_t + \frac{F_t}{M} \Delta t$$

$$L_{t+\Delta t} \leftarrow L_t + \tau_t \Delta t$$

$$P_{t+\Delta t} \leftarrow P_t + F_t \Delta t$$

 {Half step approximations}

$$v_{t+\frac{\Delta t}{2}} \leftarrow \frac{1}{2}(v_t + v_{t+\Delta t})\Delta t$$

$$P_{t+\frac{\Delta t}{2}} \leftarrow \frac{1}{2}(P_t + P_{t+\Delta t})\Delta t$$

$$L_{t+\frac{\Delta t}{2}} \leftarrow \frac{1}{2}(L_t + L_{t+\Delta t})\Delta t$$

 else

$$v_t \leftarrow v_{t-\frac{\Delta t}{2}} + \frac{F_t}{2M} \Delta t$$

$$L_t \leftarrow L_{t-\frac{\Delta t}{2}} + \frac{1}{2}\tau_t \Delta t$$

$$P_t \leftarrow P_{t-\frac{\Delta t}{2}} + \frac{1}{2}F_t \Delta t$$

$$v_{t+\frac{\Delta t}{2}} \leftarrow v_{t-\frac{\Delta t}{2}} + \frac{F_t}{M} \Delta t$$

$$P_{t+\frac{\Delta t}{2}} \leftarrow P_{t-\frac{\Delta t}{2}} + F_t \Delta t$$

$$L_{t+\frac{\Delta t}{2}} \leftarrow L_{t-\frac{\Delta t}{2}} + \tau_t \Delta t$$

$$v_{t+\Delta t} \leftarrow v_t + \frac{F_t}{M} \Delta t$$

$$L_{t+\Delta t} \leftarrow L_t + \tau_t \Delta t$$

$$P_{t+\Delta t} \leftarrow P_t + F_t \Delta t$$

 end if

$$x_{t+\Delta t} \leftarrow x_t + \frac{1}{2}(v_t + v_{t+\Delta t})\Delta t \quad \text{\textit{\{full step position update\}}}$$

 Calculate orientation $q_{t+\Delta t}$ and angular velocity $\omega_{t+\Delta t}$ as before

end for

previous algorithms. In this way, velocity and momentum terms are never half a time step behind the spatial terms as they are in the Leapfrog algorithms previously discussed. We do not need the extra steps of having to calculate for variables at time t at each iteration, as we take this information from the previous step, so no further approximation is required. Recall that we said that the Leapfrog algorithm was unfortunately not self starting and thus potentially problematic for the inclusion of velocity dependent forces such as drag and friction, due to the fact that we are always half a time step behind the spatial time. This also holds true for the previously discussed Fincham's algorithms as they are derived from the original Leapfrog algorithm. In representing all momentum and velocity variables at the same time step as the spatial variables, we have alleviated these disadvantages from this modified rotational Leapfrog algorithm.

We cannot continue to refer to this modified algorithm as Leapfrog, but rather perhaps a more correct name would be to call it semi-Leapfrog as we still need to utilise the half step velocities in our integration approximation in the positional updates. So it still appears that the velocities are leapfrogging over the current positional time. The half step velocity approximations for the positional update step is done in the same manner as in Algorithm 2 for both linear position and orientation. Rotational motion is still integrated in the same manner as in the implicit rotational Leapfrog algorithm. Here, we have simply optimised the way in which we store the state variables and calculate the half step velocities. It is thus possible to keep all state variables along with auxiliary variables at time t within the rigid body states description so that they are never out of synch with each other.

The only disadvantage with this approach (and it is probably quite a large disadvantage as well) is that we lose a lot of the accuracy and stability in the velocity domain that the half step approximation had previously given us. The reader may have noticed that the momentum updates are done in an identical manner to an explicit Euler update. This approach produces only marginally more stable and accurate results than using a full Euler update step in the linear motion and positional domain. Since the angular momentum is updated in such a way, it is only intuitive to conclude that the errors will also propagate over to the rotational motion domain, as the angular momentum is utilised to approximate angular velocity used for the orientation update. Though the accuracy and stability of this modified algorithm is better than the explicit Euler algorithm, is less computationally expensive by virtue of the fact that it requires less approximations per step and is better suited to dealing with collision events than the original implicit Rotational Verlet algorithm it cannot warrant a first choice when choosing integration algorithms for long simulation

times for a dynamics system due to its unfortunate large accuracy compromise. Next we shall explore further alternative solutions to optimising our integration algorithm and adapting the Verlet methods to rigid body dynamics by using a hybrid approach.

4.4 Hybrid Integration Solutions

So it would seem that we need to maintain the structure of Verlet methods in some shape or form in order to maintain the accuracy and stability vs computation efficiency advantage that such methods provide. It would also seem that we need to know the velocity at time t without having to depend on a half step approximation, as we do in Fincham's implicit rotational Verlet method, if we are to deal with collisions effectively within our dynamics system.

Up until now we have only considered changing the terms but maintaining the Leapfrog structure of the algorithm. Our problems, would seem to stem from this Leapfrog structure, so possibly a better approach would be to integrate the linear motion using a Velocity Verlet approach, leaving the rotational part of the algorithm alone as it does not cause us any immediate problems when dealing with collision events. This idea is presented in Algorithm 4. In this algorithm we have replaced the calculations of linear velocity, angular momentum and linear position to a Velocity Verlet approach and have left the approximation of orientation and angular velocity in the same form as described by Fincham. With the linear momentum term we could have integrated it using the same Velocity Verlet approach as we had done with the linear velocity and angular momentum variables but instead here, we chose to treat it as an auxiliary variable derived from the linear velocity term. In this way we cut down on the number of calculations and the need to keep track of a half step linear momentum variable as well.

For the calculations of the forces and torques acting at time $t+\Delta t$ for the full step velocity and angular momentum updates, it is sufficient to presume that the known continuous external forces such as gravity or continuous contact forces, calculated as acting on the rigid body at this time frame will remain to do so at the next time frame as well. If any event such as a collision or making or breaking of a continuous contact case occurs in the next simulation time step, this can be accounted for then and velocities and momenta appropriately adjusted to fit the situation. If there are internal constraint forces acting on the rigid body (i.e. such as those keeping a joint between two rigid links together) as well, then these must be taken into account when calculating the forces and torques at time

Algorithm 4 Velocity Verlet adaptation of Fincham's implicit rotational Verlet algorithm. This version resolves previous collision handling issues.

Input : List of rigid body states, Number of rigid bodies to update n , Timestep Δt

Output : Updated list of rigid body states

VELIMPROTLEAP($BodyStates, n, \Delta t$)

for all rigid bodies i up to n do

$$v_{t+\frac{\Delta t}{2}} \leftarrow v_t + \frac{1}{2} \frac{F_t}{M} \Delta t$$

$$L_{t+\frac{\Delta t}{2}} \leftarrow L_t + \frac{1}{2} \tau_t \Delta t$$

$$v_{t+\Delta t} \leftarrow v_{t+\frac{\Delta t}{2}} + \frac{1}{2} \frac{F_{t+\Delta t}}{M} \Delta t \quad \{it \text{ is sufficient to assume just external forces here}\}$$

$$L_{t+\Delta t} \leftarrow L_{t+\frac{\Delta t}{2}} + \frac{1}{2} \tau_{t+\Delta t} \Delta t \quad \{it \text{ is sufficient to assume just external forces here}\}$$

$$P_{t+\Delta t} \leftarrow v_{t+\Delta t} M$$

$$x_{t+\Delta t} \leftarrow x_t + v_{t+\frac{\Delta t}{2}} \Delta t$$

{initial guess at orientation and angular velocity}

$$q_{t+\Delta t} \leftarrow q_t + \frac{1}{2} \omega_t q_t \Delta t$$

$$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$$

$$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t}$$

for $j=1$ to $MaxIterations$ do *{Again, MaxIterations can be either 3 or 4}*

$$q_{t+\Delta t} \leftarrow q_t + \frac{1}{2} \left(\frac{1}{2} \omega_t q_t + \frac{1}{2} \omega_{t+\Delta t} q_{t+\Delta t} \right) \Delta t$$

$$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$$

$$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t}$$

end for

end for

$t + \Delta t$ when updating the linear velocity and angular momentum terms if one is to maintain overall stability and accuracy. Our tests have shown (Figure 3.13) that the Velocity Verlet approach is in general more accurate and stable than the Leapfrog approach in the linear domain, so modifying Algorithm 2 in this way should be an overall improvement in accuracy and stability, if not providing a better overall velocity approximation. Another advantage of using the Velocity Verlet formulation in this way is that, while previously the likes of Algorithms 2 and 1 were not self starting due to the Leapfrog nature of the algorithms, this version is self starting. We have used the above algorithm in our time adaptive system with great success. Rigid bodies within the simulation acted with no apparent energy loss or gain during collisions, even for the simulations of the very unlikely cases of perfectly elastic collisions, and empirically the rigid body motions acted in a physically plausible manner during free flight motions.

Our tests (which will be discussed later) on Fincham's rotational algorithms showed that, as long as a rigid body was not subjected to constant internal constraint forces, the algorithm's stability and accuracy was sufficiently good. On inclusion of continuous internal forces acting on a rigid body (e.g. internal constraint forces keeping two linked rigid bodies together at a point) it was discovered that the stability of Fincham's algorithms (in the rotational domain) wasn't any better than using a second order Runge-Kutta scheme. Stability and accuracy is maintained within high order limits in the linear domain as was expected, as the Verlet formulations were not altered in any way in the linear domain, but did not translate over to the rotational domain. As long as the forces acting on the rigid body remain external in nature and are relatively constant (e.g. gravity, wind, non-interpenetration forces), the rotational approximation from using Fincham's algorithm was found to work reasonably well. Our studies [121] have shown that people in general are more accepting of angular distortions than they are of linear distortions, especially when the object in question is sufficiently complex in shape. This may explain why, if there was indeed any significant increase in inaccuracy in the rotational domain of Fincham's algorithms, that our simulations still looked physically plausible. We shall discuss these findings further later on in the plausible motion chapter. There has been work conducted in the molecular dynamics community in the last few years to improve the overall energy conservation and stability within a constrained system of Fincham's algorithms [116, 117, 91]. However, the research done has either focused on an alternate atomic constraint approach, a principal axis representation approach or an approach such that we cannot directly apply to our rigid body dynamics problems without some major changes and structural rethinking.

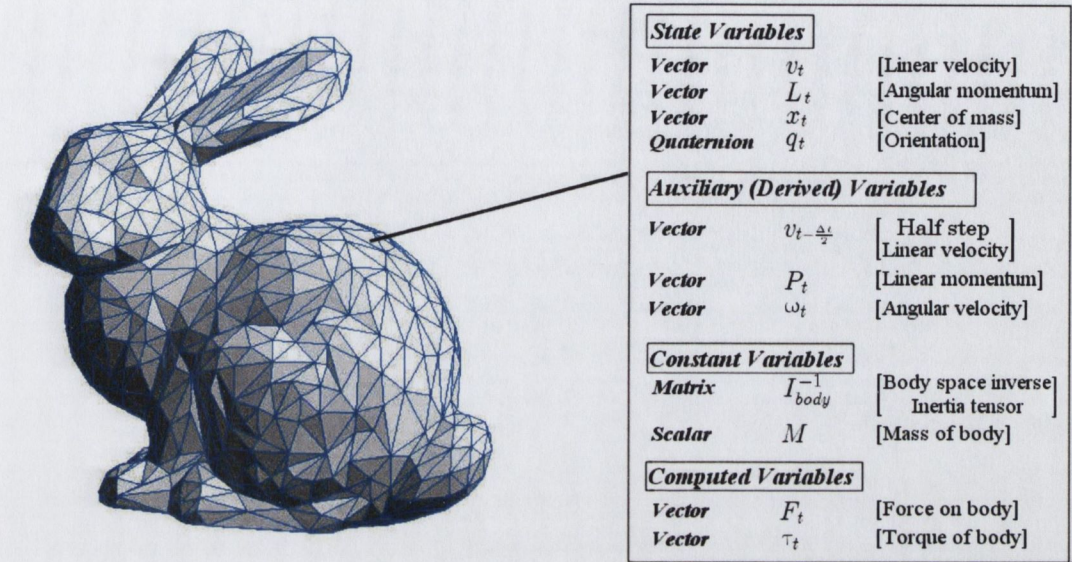


Figure 4.2: Rigid body states representations within the time adaptive system

To maintain a higher order of stability and accuracy in both the linear and rotational domains, while maintaining as much computational efficiency as possible during the integration steps in the dynamics system, we have derived a hybrid Velocity Verlet Runge-Kutta scheme. With this scheme, we apply the Velocity Verlet method to the linear motion variables and a fourth order Runge-Kutta method to the rotational motion variables. Algorithm 5 describes this hybrid algorithm. We treat the linear momentum P and angular velocity ω terms as auxiliary variables that are derived from the updated state variables after integration. Here, we need to keep the linear velocity state variable at two different time steps, one term at the current time t and the other half a time step behind. The requirement of having to keep an additional linear velocity term is a disadvantage of using the Velocity Verlet approach but is however compensated for by the reduction in the number of variable calculations that we must perform per iteration through the simulation loop.

It is advantageous to represent the rotational state variables of the rigid body in terms of orientation and angular momentum as we do here rather than orientation and angular velocity. Representing the rotational state variables in this way simplifies our treatment of velocities. How so? Angular momentum is “conserved in nature” which means that a rigid body’s angular momentum remains constant if moving through space with no torque acting on it. This assumption is however not true for angular velocity. Even though the angular momentum in a body is constant, the angular velocity of the body may still undergo changes

Algorithm 5 A hybrid fourth order Runge-Kutta Velocity Verlet integrator. Utilises Velocity Verlet integration in the linear domain and fourth order Runge-Kutta integration in the rotational domain.

Input : List of rigid body states, Number of rigid bodies to update n , Timestep Δt

Output : Updated list of rigid body states

RKVERLET(*BodyStates*, n , Δt)

for all rigid bodies i up to n **do**

 {deal with linear motion - Velocity Verlet integration}

$$v_{t+\frac{\Delta t}{2}} \leftarrow v_t + \frac{1}{2} \frac{F_t}{M} \Delta t$$

$$v_{t+\Delta t} \leftarrow v_{t+\frac{\Delta t}{2}} + \frac{1}{2} \frac{F_{t+\Delta t}}{M} \Delta t \quad \{it is sufficient to assume just external forces here\}$$

$$P_{t+\Delta t} \leftarrow v_{t+\Delta t} M$$

$$x_{t+\Delta t} \leftarrow x_t + v_{t+\frac{\Delta t}{2}} \Delta t$$

 {deal with rotational motion - 4th order Runge-Kutta integration}

$$InitVals \leftarrow \begin{Bmatrix} q_t \\ L_t \end{Bmatrix} \quad \{remember q and L at time t\}$$

$$X \leftarrow \begin{Bmatrix} q_t \\ L_t \end{Bmatrix} \quad \{the current state of q and L at current sub-time\}$$

for $i=1$ to 4 **do**

$$DxDt \leftarrow f(X) \quad \{derivatives of q and L terms at current sub-time\}$$

$$k[i] \leftarrow DxDt \Delta t$$

if i equals 4 **then**

$$FinalStates \leftarrow InitVals + \frac{k[4]}{6} + \frac{k[3]}{3} + \frac{k[2]}{3} + \frac{k[1]}{6}$$

else if i equals 3 **then**

$$FinalStates \leftarrow InitVals + k[i]$$

else

$$FinalStates \leftarrow InitVals + \frac{k[i]}{2}$$

end if

if i NOT equals 4 **then**

$$X \leftarrow FinalStates \quad \{the current state of q and L at current sub-time\}$$

end if

end for

$$\left. \begin{matrix} q_{t+\Delta t} \\ L_{t+\Delta t} \end{matrix} \right\} \leftarrow FinalStates$$

$$I_{t+\Delta t}^{-1} \leftarrow R_{t+\Delta t} I_{body}^{-1} R_{t+\Delta t}^T$$

$$\omega_{t+\Delta t} \leftarrow I_{t+\Delta t}^{-1} L_{t+\Delta t}$$

end for

even when there are no forces acting on the body. This makes it quite troublesome to be modelling rigid bodies using orientation and velocity terms as state variables. Thus, it is more intuitive and easier to consider orientation and momentum terms as state variables instead, as it simplifies how we deal with our equations.

Figure 4.2 illustrates what our data structure looks like for a rigid body representation within our dynamics system. As can be seen, all state and auxiliary variables are maintained in the same simulation time frame with perhaps the exception of the linear velocity terms that also have the additional half step information for processing purposes. This makes the algorithm self starting and influential to velocity dependent forces such as drag and friction. This also means that it is not problematic for the algorithm to deal with collisions as Fincham's algorithms were. For each iteration through our simulation loop, we pass a list of rigid bodies with the following states structure into our ODE solver and we get back an updated list of all state and auxiliary variables, thus advancing our rigid bodies through our simulation. On detection of a collision, the collision handler calculates the correct impulses to apply to the appropriate colliding bodies and updates the rigid body states as appropriate before going into the next simulation iteration. No additional condition checks or adjustments need to be done to the state representations at the next simulation loop. The ODE solver should be able to handle whatever information it is given at that time, providing that all previous calculations and approximations were handled correctly. The algorithm is $O(\Delta t^4)$ accurate in the linear positional domain and $O(\Delta t^5)$ accurate in the rotational domain. Even though the property does not directly affect us here it is still nice to know that because we are utilising a Verlet integrator, the algorithm is also time reversible in the linear domain. From tests conducted (which we shall discuss next) it can be seen that we have traded a little bit of accuracy for more computational speed. This we deem an acceptable trade off, as visual plausibility and overall system stability is still maintained.

4.5 Tests and Results

In order to examine the rotational accuracy and stability of Fincham's rotational algorithms, several tests were conducted. The purpose here was not only to gauge the overall accuracy of Fincham's rotational algorithms within the rotational domain but also the overall performance of our proposed algorithms in comparison to the popularly used fourth order Runge-kutta algorithm. Tests were conducted on the proposed algorithms presented

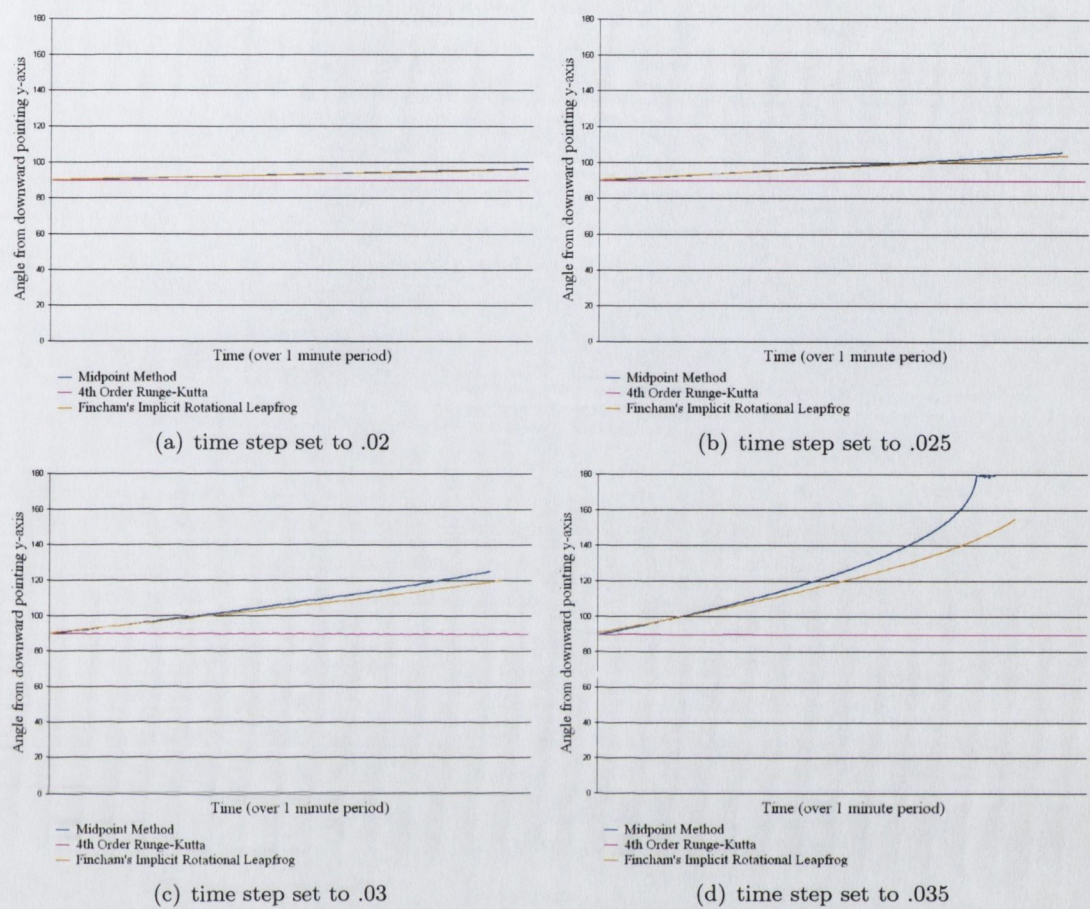


Figure 4.3: Rotational Accuracy test of Fincham’s implicit rotational integrator algorithm compared to a Midpoint and 4th order Runge-Kutta scheme

in this chapter on simulations whereby over and under constrained situations can also be problematic. We expected our proposed algorithms to out perform the Midpoint method as the Verlet algorithms did during the tests within the linear domain and to come fairly close to those of the fourth order Runge-Kutta method.

Accuracy Tests on Fincham’s Rotational Algorithm

A simulated rigid block with a constant downward force applied to one end was used to test the rotational accuracy of Fincham’s rotational algorithm. We only considered Fincham’s implicit rotational Verlet algorithm for this test and compared it with the Midpoint method and the fourth order Runge-Kutta method. The reason for neglecting Fincham’s explicit rotational algorithm was that the implicit algorithm was stated by Fincham to

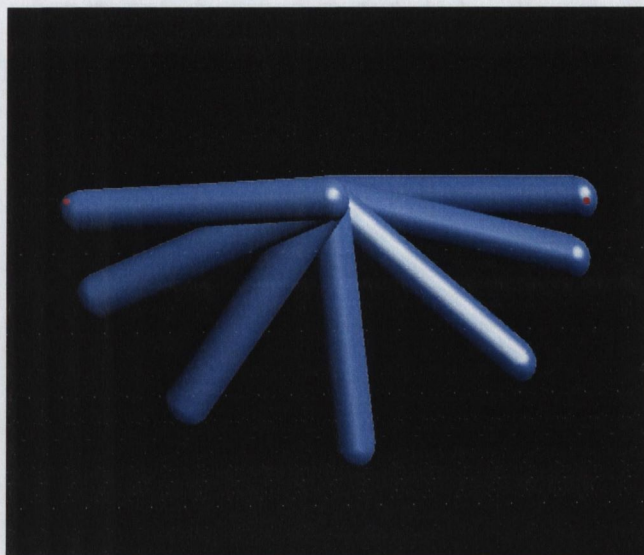


Figure 4.4: Time lapse capture of scene used for accuracy tests

vastly outperform the explicit algorithm, so we were only interested in knowing the rotational accuracy of the better algorithm. We kinematically constrained the top of the block to the origin such that only rotational motion was accounted for.

The rigid block was started off at a ninety degree angle from the y-axis and then a constant force was applied to a bottom point on the block for a one minute period. In doing so, it was expected that the block should exhibit behaviour similar to a simple pendulum. Since no damping was applied, we expected a return to a ninety degree deviation from the y-axis at the peak of each swing.

In the test, we measured the angle from the y-axis at the peak of each swing to determine any erroneous energy gain or loss within the system. Four time steps were taken for measurement purposes, starting with a moderate step size of .02 and then increasing the step size to .025, to .03 and finally to .035. Figure 4.3 shows the results from these tests and Figure 4.4 shows a time lapse capture of the test scene. As can be seen, at a time step of .02 all three algorithms performed well. However as the time steps increased, so did the deviation away from ninety degrees at the peak of each swing with the Midpoint method and Fincham's algorithm. The fourth order Runge-Kutta algorithm stayed consistently within the bounds of accuracy for all four time steps. It was found that the Midpoint method performed the worst out of the three, with Fincham's implicit rotational Verlet coming in second. The truncation error from the integration step crept into Fincham's implicit

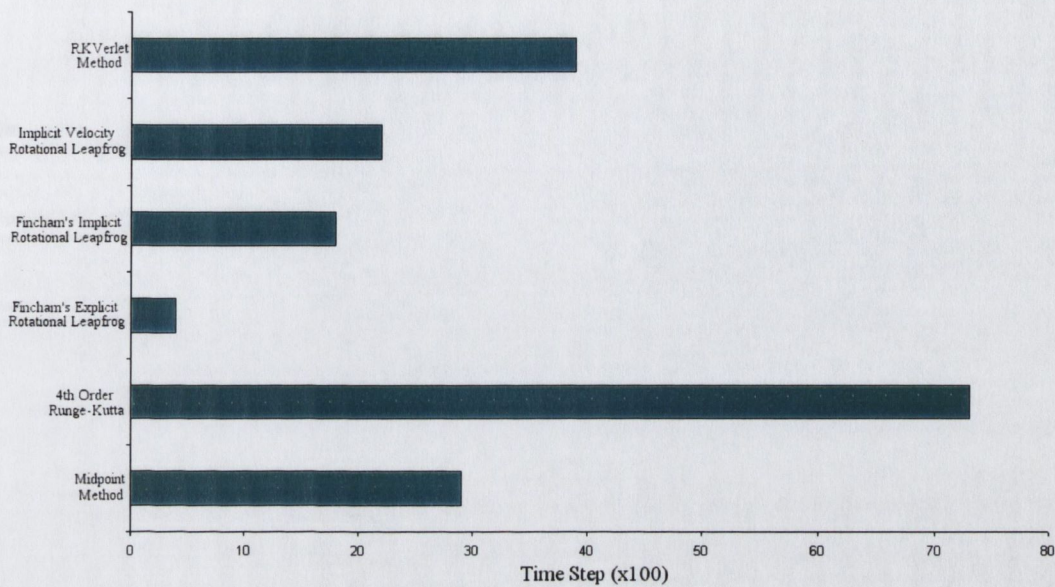


Figure 4.5: Results of integrator stability test for a three body constraint system. Graph shows max time step for each integration method before loss of stability

rotational Verlet algorithm at a more gradual rate at large time steps than the Midpoint method.

Algorithm Stability Tests

For a second test, we wanted to determine the stability of our proposed hybrid algorithms compared to Fincham’s algorithms and the Midpoint and fourth order Runge-Kutta method. For this, we used a dynamic constraint simulation whereby internal constraint forces as well as external forces affected the overall rigid body motion. Using a dynamic constraint technique based on the work conducted by Barzel and Barr [13], we built a dynamic chain out of three rigid bodies which were just rigid blocks with the top of the chain dynamically constrained to the origin. This approach differs from the kinematic approach described for the previous test whereby we naïvely translated the top back to the origin whenever the constraint was broken. Here we calculated the necessary internal constraint force acting on the rigid body to maintain a point constraint and then worked out the necessary net force (i.e. internal constraint forces plus external forces) acting on each body during the system update. To account for numerical inaccuracies that invariably creep in during linear equation solving due to the discrete nature of computers, we used *Baumgarte stabilization* [15] on our dynamic constraint linear equations.

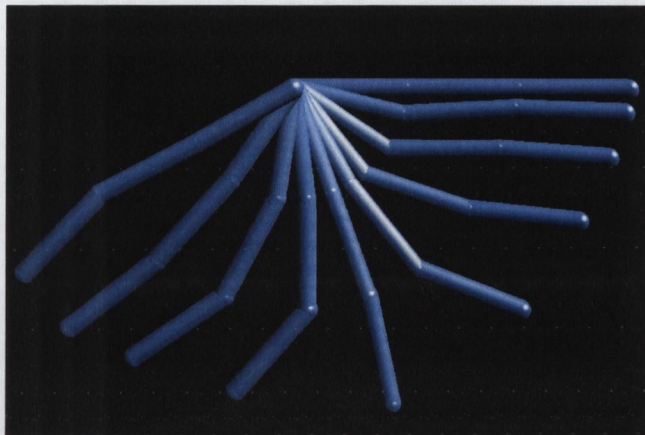


Figure 4.6: Time lapse capture of scene used for integrator stability tests

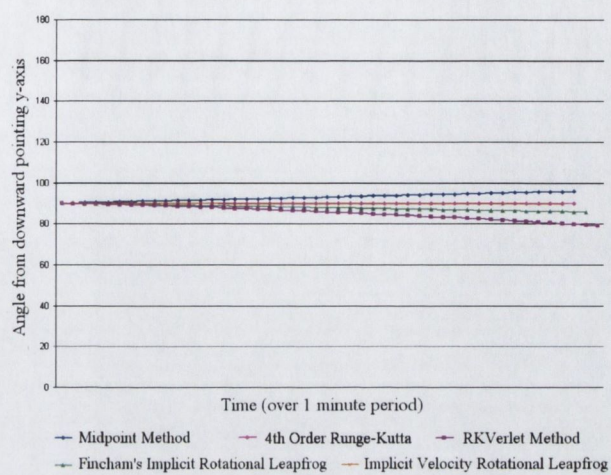
We set the chain at initial orientation of ninety degrees to the x axis and then started the simulation. A gravitational force was applied to the chain and a damping was also applied so that when the simulation started the chain would start to swing under gravity and eventually come to rest. Figure 4.6 illustrates this setup. We gradually increased the time step for each integration method tested, restarting at the initial states for each time step increase and took note of the maximum time step which caused the system to explode. Figure 4.5 illustrates the results from these tests clearly showing the max time step that caused each integration method to explode. As forces in this instance are not constant but are instead allowed to both instantly push and pull to maintain constraints, system stability can become an issue if the integrator used cannot appropriately handle such situations. As can be seen from Figure 4.5, both of Fincham's algorithms performed the worst in terms of stability, being beaten by all other four integration methods tested. The fourth order Runge-Kutta method proved to be the most stable of the integration methods tested followed by our RK Verlet hybrid method. Our hybrid implicit velocity rotational Leapfrog method was only just ever so slightly outperformed by the Midpoint method.

Algorithm Accuracy Tests

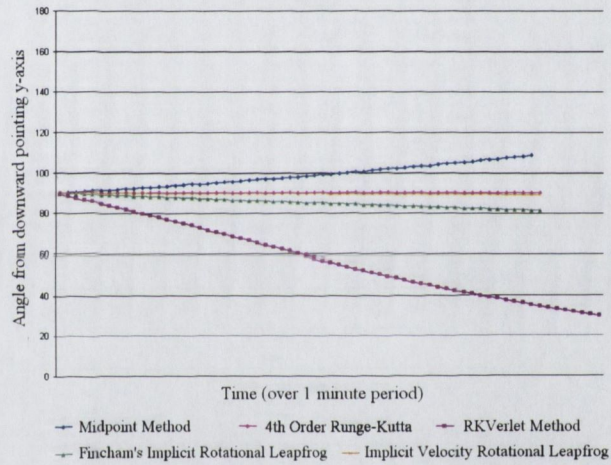
For a third test, we wanted to gauge the overall accuracy of the proposed integration methods compared to Fincham's algorithms and the Midpoint and fourth order Runge-Kutta methods. In this instance we kept the dynamic constraint approach of test two but used the same testing procedure as that of the first test. Figure 4.7 illustrates the results from these tests. In this instance three time steps of .01, .013 and .016 were tested.

Accuracy was maintained by the fourth order Runge-Kutta method as of test one, with the hybrid implicit velocity rotational Verlet method performing almost equally as well. The Midpoint method showed a gradual gain in energy at each swing with the rate of energy gain proportional to the time step size used. This means that, if the simulation was left to run for a long period of time past the one minute period, this energy gain would eventually lead the system to undesirably explode. Fincham's implicit rotational Leapfrog algorithm showed very close to accurate trajectories displaying a very gradual loss of energy over time. The hybrid RK Verlet method displayed quite a substantial amount of energy loss over time as can be seen from the results of Figure 4.7.

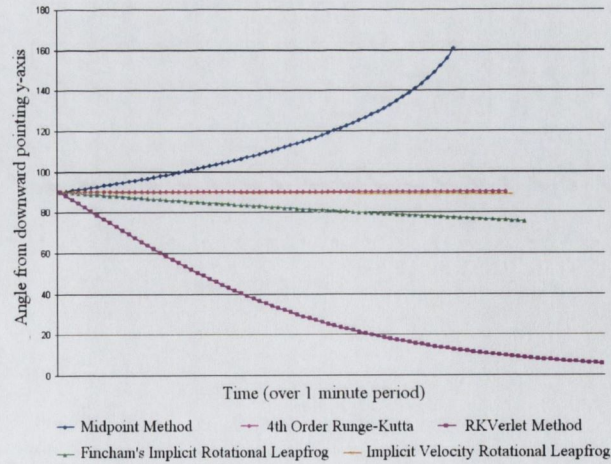
At first it were thought that this was due to the inadequacies of the overall algorithm but upon further inspection the result was put down to the manner in which the constraint forces was being calculated given the information provided. If we examine the fourth order Runge-Kutta method a little closer, we will realise that for every sub-step taken in the integrator, the constraint force at that sub-step must also be recalculated to maintain system consistency. As a result, we have at least four constraint force calculations per simulation iteration. The position, external forces and velocities within both the linear and rotational domain of the rigid bodies in question are all taken into account for the constraint force calculation at that sub-time. Now, let us examine how our RK Verlet algorithm works. Within this algorithm, we take a step in the linear domain using one Verlet approximation but use four rotational sub-steps to approximate the motion in the rotational domain. This then means that we must recalculate the constraint force for each sub-step we take in the rotational domain. However, the linear information that we use within the constraint force calculation does not conform to what the information should be at each sub-step but rather to that of the end of the integration step. So during each sub-step constraint force calculation within each simulation iteration, the linear and rotational information will be displaced from each other. An appropriate constraint force will be calculated to maintain the imposed constraints on the system but some energy will be lost in the calculation, as the linear and rotational domain information used to calculate the constraint force does not conform to the same simulation time period. Even though this is undesirable, we still prefer an unwanted energy loss that will eventually bring the constraint system to rest over that of unwanted energy gain that will eventually cause the constraint system to explode.



(a) time step set to .01



(b) time step set to .013



(c) time step set to .016

Figure 4.7: Integrator accuracy results from a dynamically constraint block set to fall at 90 degrees to the downward pointing y-axis with no damping

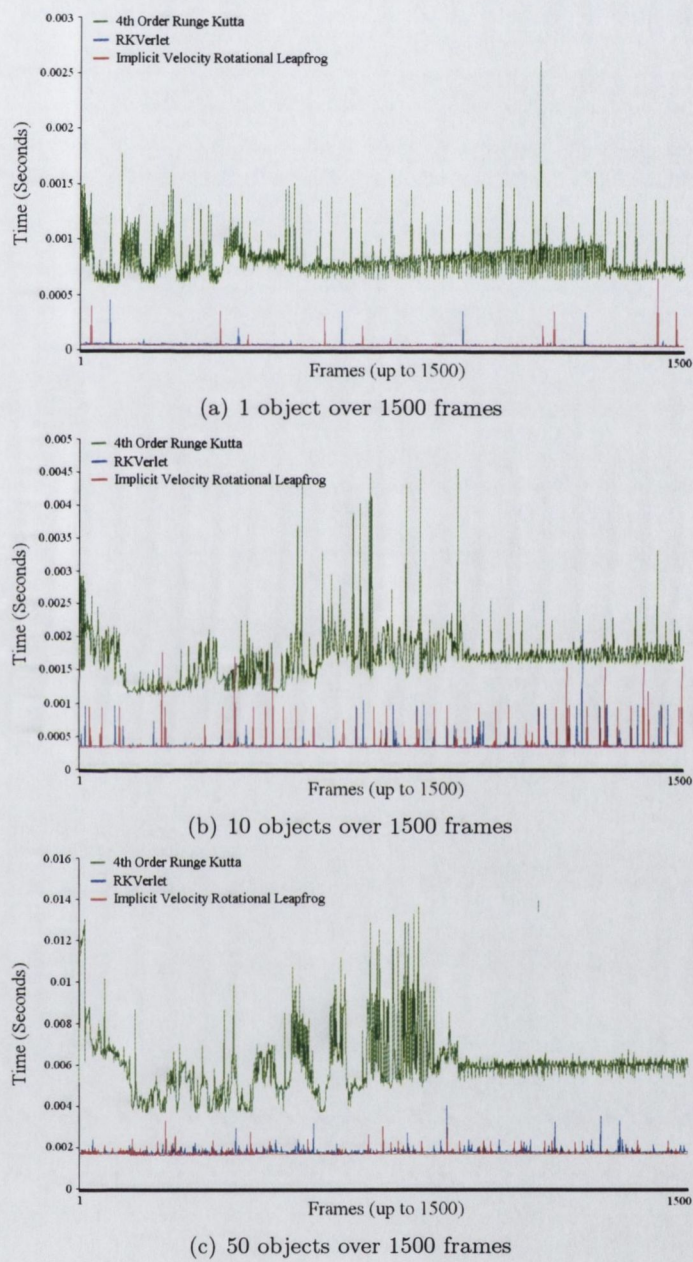


Figure 4.8: System integrator timing test for 4th order Runge-Kutta, RKVerlet and Revised Implicit Velocity Rotational Leapfrog methods

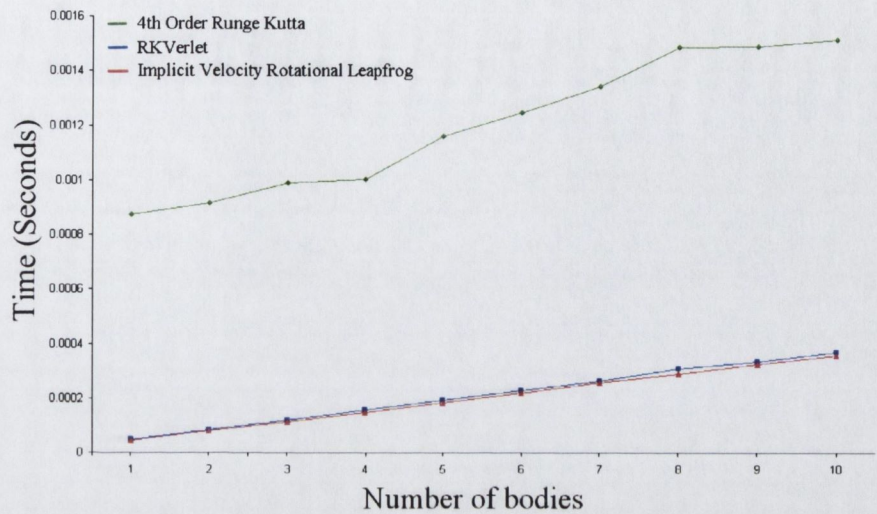


Figure 4.9: Average integrator timings over 30 second period for 1 to 10 objects

System Performance Tests

To gauge the overall speed advantages of our proposed algorithms to those of the more popular fourth order Runge-Kutta algorithm, a number of tests involving several unconstrained rigid bodies reacting under gravity with collisions between themselves was taken. We did not measure collision time in this instance, as this does not affect our overall integration methods, but only accounted for the underlying time the ODE solver took to resolve initial state changes at each simulation time frame. A 1Ghz AMD Athlon with 500megs of RAM running a GForce 3 graphics card was used for conducting the timing tests. Figures 4.8 and 4.9 illustrate the results from these test. Since our proposed hybrid integration methods were designed to have less calculations per step, it was predicted that both the proposed RKVerlet and implicit rotational velocity Leapfrog methods were to out perform the 4th order Runge method in all the timing tests. For all tests conducted, the integrator time step was set at .01.

The results from the tests clearly show overall savings on system processing time for the proposed hybrids methods over that of the Runge Kutta method, as was expected. In all of the experiments conducted for Figures 4.8 and 4.9, the rotational calculations within the implicit rotational velocity Leapfrog method consisted of four iterations before termination. The results show that both the proposed RKVerlet method and the implicit rotational velocity Leapfrog method are very close in terms of performance timings. The somewhat erratic looking initial performance measurements in Figure 4.8 is put down to

processor caching. This theory is supported by the fact that the graphs start to level out a bit as time progresses. An averaging of performance timings was also taken, as is illustrated in Figure 4.9, for a progression of bodies over thirty second time intervals. This test was aimed to demonstrate what sort of timings, on average, we could expect for such methods during a typical simulation time period of thirty seconds for different number of objects.

Observations

From the tests conducted, it was shown that the proposed hybrid algorithms do not compromise accuracy and stability to too much of a degree. Though they did not beat the fourth order Runge-Kutta method, they did give comparably good results and outperformed the Midpoint method. Thus they have been shown to be a viable means of replacing the fourth order Runge-Kutta method for the integration of rigid body dynamic states if we do not mind trading off a little bit of accuracy for speed. However, for simulations involving internal constraints on the rigid bodies such as rag doll simulations, we recommend the use of Runge-Kutta methods over our proposed hybrid methods. Alternatively, Jakobsen [80] presented a very viable and fast alternative approach to simulating constrained rigid bodies such as rag dolls using Verlet methods whereby the rotational domain can be ignored.

4.6 Conclusions

In this chapter, we have investigated the problems involved in utilising Verlet algorithms for solving full rigid body dynamics as opposed to just point masses. Fincham's rotational Leapfrog algorithms were introduced. These algorithms come in two flavours, an explicit and an implicit version. It was shown that the implicit version far outperformed the explicit version in terms of stability and accuracy. As a result of Fincham's tests and our own tests we conclude that the explicit version of Fincham's rotational Leapfrog algorithm is not viable as a consideration for any serious rigid body dynamic simulations.

The problems of collision response with Leapfrog-like integration algorithms (such as Fincham's) were investigated and several issues were tackled. We presented several alternative integration approaches building on Fincham's algorithms to address the issue of collisions and Verlet integrators. Out of these alternative approaches we found that the hybrid RK Verlet method presented the best results.

Several tests were conducted to determine the accuracy of Fincham's algorithms in the rotational domain as compared to the Midpoint and popular fourth order Runge-Kutta

methods. Tests were also conducted on the proposed hybrid algorithms, comparing the performance of such to the Midpoint, fourth order Runge-Kutta and Fincham's methods. The results and a discussion of these tests were presented. We can conclude from the tests that our hybrid algorithms provide a viable alternative to integration methods for rigid body dynamics. From the results it can be seen that the proposed hybrid algorithms come fairly close to competing with the fourth order Runge-Kutta method. They presented overall better performance results than the Midpoint and both of Fincham's algorithms. Timing tests using the proposed Implicit velocity rotational Leapfrog and RK Verlet method compared to the fourth order Runge-Kutta method within a simulation of many free falling rigid bodies showed both methods required less time to update as then the Runge-Kutta method. Here we traded off a bit of accuracy for improved speed but not to such an extent that overall system stability was compromised. However, our tests further showed that for simulations involving internal constraints on rigid bodies, such as joint constraints, it is more favourable to use the fourth order Runge-Kutta method in such cases.

Chapter 5

Time Adaptive Dynamics

Current dynamic simulations are hindered by many factors such as the limitations of current collision handling techniques and inflexible simulation scheduling policies. Despite advances in collision handling algorithms and the current increase in the computation power of processors, an upper limit will ultimately be reached whereby the achievement of required real-time frame rates becomes problematic. Unnecessary complexity of algorithms and simulation rules within a dynamics system can work to hinder further simulation frame rate improvements. Various level of detail techniques are thus employed once the current algorithm and processor efficiency no longer suffice in providing real-time frame rates. The common level of detail geometry and animation techniques such as occlusion culling, adaptive mesh simplification and adaptive simulation simplification heuristics of occluded objects can all be considered.

This chapter proposes various novel approaches that can be used within physically based simulations that further address the issues of desired timing constraints. We present these proposed methods in a novel framework for simulating dynamics in a viable time adaptive fashion. Various *Simulation Proxies* are utilised, sacrificing accuracy for speed whenever necessary, but also maintaining a good level of plausibility in the process. The ideas and work presented in this chapter, combined with the contributions presented earlier, culminate in a viable and comprehensive time adaptive dynamics system.

We follow on from the work of Hubbard [76, 77, 78], O'Sullivan [118, 119] and Dingliana [47, 48], adopting the idea of interruptible collision detection and the use of sphere-tree BVHs. We further consider the issue of efficient simulation scheduling and approximation of collisions within a time-critical system in more detail in this chapter and offer alternative strategies to approximating collision data in the next chapter. We adapt some of the ideas

for time-critical collision handling from the work of Hubbard, O'Sullivan and Dingliana into a fully comprehensive time adaptive framework and address the issues that arise from such approaches and present novel solutions for implementation within a fully time adaptive dynamics system.

5.1 Simulation Scheduling

As we mentioned in Chapter 1, the maintenance of constant frame rates is an important criterion if one is to ensure physical plausibility within a simulation. Variable or “jittery” frame rates can hinder the overall perception of velocity within a dynamic simulation, or cause too long a pause during collision response such that objects no longer are perceived as reacting appropriately to the dynamic event.

For applications such as interactive digital entertainment displayed on television screens, it is important that frame rate criterion are met or else incur strict performance penalties due to the fact that the hardware for running such media are intrinsically locked with the television refresh rate. This means that a failure to meet the 50/60 Hz refresh target for any part of the simulation, by even a small amount results in an immediate drop to 25/30 frames per second.

Due to the requirement for good constant frame rates, it is advantageous to have strict simulation scheduling protocols that will ensure that all processing in the simulation pipeline will finish up within the allotted time. In this section, we shall present a simulation scheduling strategy that aims to maintain required constant frame rates throughout a simulation's lifetime.

5.1.1 Typical Simulation Event Scheduling

In a typical simulation, we continuously iterate through the simulation loop, conducting all necessary update calculations on bodies in the simulation and finally rendering the new animation frame based on the updated states information. As the number of rigid bodies in a scene increases, so too does the overall calculation time taken between frames. Of these calculations, collision detection has proven to be the biggest bottleneck. In Chapter 2 we presented many approaches that have been developed to speed up collision handling. These efficient collision handling algorithms go a long way towards alleviating the overall time spent within each simulation iteration but there is still no guarantee of an acceptable constant frame rate. An unrestricted animation frame will typically take as long as necessary

to go through the following basic stages of the simulation pipeline:

1. Initial iteration update of all rigid body states in the scene so as to advance all rigid bodies to the next simulation sub-time. The time taken for calculations at this stage is dependent on the iteration algorithm used to resolve Newton's equations of motion as well as the number of rigid bodies involved. We have discussed several of the more common integration algorithms utilised by graphics researchers in physically based simulations in Chapter 3. The most commonly used algorithm here is the fourth order Runge-Kutta method for its accuracy and stability. Chapter 4 also presented several viable alternative integration schemes. Any of these may be used at this stage, bearing in mind the speed versus stability and accuracy costs.
2. Any simulation discontinuities (i.e. collisions) must next be detected and then resolved so as to maintain system state consistency. This stage can be viewed as two distinct processes. The first is kinematic in nature, whereby we are concerned with the validity of the spatial occupancy information about each object in the scene. It is the job of the collision detection algorithm to determine such information in an efficient manner. Once all contacts and collisions have been determined and relevant contact/collision information has been found, in order for the system to respond to such discontinuities appropriately, we must re-update the rigid body states as appropriate to reflect the consequences of the collisions on the system by resolving the events in a physically correct or plausible manner. It is this stage that takes the most time, but however is also the most exploitable for improvements, within the simulation loop.
3. Lastly, after all rigid body states have been updated and discontinuities resolved, the next simulation frame is then rendered. Here, the total rendering time is dependent on the number of objects that are visible in the scene and the complexity of each object within the scene.

Figure 5.1 illustrates the time scheduling of the previously described events within a typical simulation loop. It is obvious that, in order to maintain acceptable constant real-time frame rates, we cannot arbitrarily allow each simulation frame to take as long as it wants to process all events within the simulation loop. We must enforce strict upper limit timings upon each simulation frame, allowing processing within each stage only as long as a predetermined timing allocation will permit. A heuristic is required to correctly handle cases where events are stopped prematurely from completing their tasks.

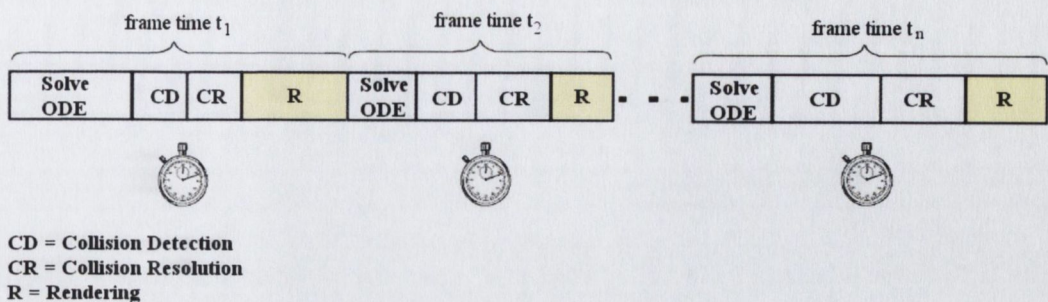


Figure 5.1: Typical simulation event scheduling over the simulation life time

We assume in this thesis, for the sake of simplicity, that the rendering stage in the simulation loop cannot be interrupted and must be allowed to run to its entirety. As such, other stages must then be adapted to the remaining times after the rendering and all other compulsory events have completed, sacrificing some accuracy in the process if necessary. Of course, it is possible for the rendering stage to be made to conform to required time constraints. Work in the field of level of detail rendering has indeed demonstrated the idea of interruptible rendering, that tries to conform to constant frame rate requirements. Holloway's Viper system [75] is an early example of such work. This brute force scheme simply stopped rendering when the allotted time was exhausted. As a result, it was possible to see gaps appearing in objects or objects simply disappearing completely from the scene. A more sophisticated example of work in interruptible rendering can be found in the most recent published research conducted by Woolley et al [145]. In Woolley et al's paper, a progressive rendering framework for the unification of spatial and temporal error is proposed. Within their framework, a coarse image is rendered into a back buffer and continuously refined through monitoring of temporal error. Once temporal error exceeds that of spatial error caused by coarse rendering, the image is displayed. In this thesis, we do not consider the notion of interruptible rendering schemas but rather concentrate our efforts in the physical simulation domain. We leave the incorporation of such interruptible rendering schemas for future work.

Within the simulation scheduling domain, Hubbard [76, 77] treated each simulation frame as having a predetermined fixed time. However, unlike the simulation scheduling heuristic that we shall discuss shortly, Hubbard only considered the simulation time within each simulation iteration in two basic stages, a rendering and collision detection stage. Hubbard proposed using an average of some number of previous frame times to predict the time needed for rendering. Any remaining time from the predetermined simulation

frame time is then allotted to collision detection. Hubbard only considered time-critical processing within the detection stage of collision handling and did not handle collision response problems, instead opting to deal with such processing in a naïve way by simply reversing the velocity of the colliding objects.

Whereby Hubbard only concentrated on the issue of collision detection scheduling presuming a predetermined fixed time frame rate, Dingliana and O’Sullivan [48] expanded upon this and considered the problems associated with scheduling within a full collision handling (i.e. collision detection and response modelling) schema.

We aim to further expand the notion of time critical and time adaptive processing by here considering and applying stricter timing protocols to the entire simulation loop and determining some important rules for managing events in such a loop. Taking the entire simulation loop (i.e. Figure 5.1) into account, a novel overall simulation scheduling solution which incorporates the nature of a fully time adaptive system will be presented.

5.1.2 Time-Critical Scheduling

Let us now examine this novel scheduling protocol in the context of a proposed framework for the simulation of time adaptive dynamics that imposes strict rules on how and when certain events within each simulation time frame can proceed. Within this proposed framework, for the purposes of processing clarity, each simulation frame is viewed as four distinct timing segments or stages: a rendering stage, an initial states update stage (i.e. initial ODE solving), a collision detection stage and a collision resolution stage respectively. For the entire simulation, a strict maximum frame time must be chosen and adhered to at all times if possible.

In this thesis, we consider the premature disruption of the rendering time, t_{Render} , as not an option and the initial states update time, $t_{ODEsolve}$, must be allowed to finish if all objects within the scene are to be appropriately advanced within the simulation. The best we can do for each of these phases is to try and resolve them as quickly and efficiently as possible so that they do not chew up too much of our allotted simulation frame time. Advanced scene culling heuristics and geometric LOD techniques may be applied to speed up rendering and faster integration schemes utilised within our ODE solver to speed up initial states processing per frame. Ideally each frame time must be large enough to incorporate frame rendering, initial state updating and still have some time left over to account for collision processing to some degree.

Unlike Hubbard, a guess of average frame rendering time is not taken but, rather more

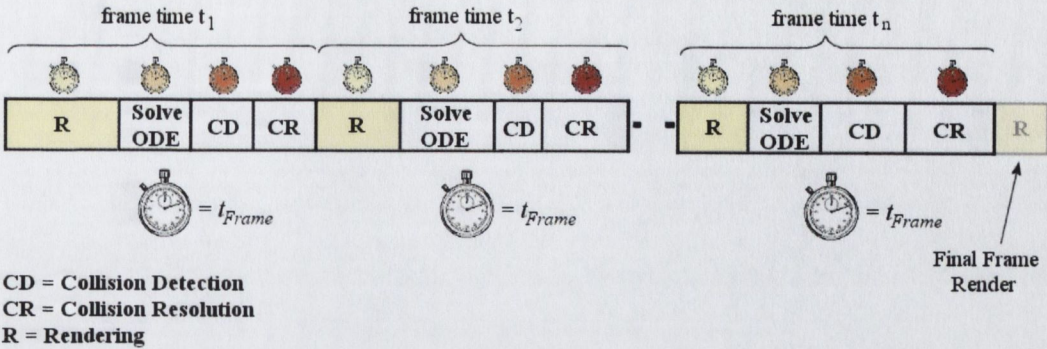


Figure 5.2: Overview of simulation scheduling protocol allowing for more effective overall simulation timing control

precisely, the scheduler will allot timings for collision detection and resolution based on the remaining time after frame rendering and initial ODE solving is complete. This scheduling protocol may at first seem somewhat counter-intuitive. It is obvious that typical scene rendering of a current simulation frame can only be done after all simulation object states (i.e. object update and collision and contact handling) have first been calculated and dealt with appropriately. However, in our scheduling protocol we are dictating that collision handling be allotted a slice of simulation time based on the remaining time *after* all scene rendering is done! Obviously we cannot do one without first acquiring appropriate information from the other. To account for this situation, we adopt a *Past Predictor* heuristic for predicting currently available collision handling times. This approach is somewhat similar to what Luebke et al [102] refer to as a *reactive* approach for level of detail (LOD) fixed frame rate rendering. With this approach, the rendering time from the previous frame is taken and the level of detail for the current frame rendering increased or decreased depending on whether or not the previous frame exceeded a predetermined deadline. However, in our proposed approach, it is not level of detail rendering that we aim to adjust through the timing information gathered from the previous rendering time, but rather, physical simulation detail.

Using this proposed *Past Predictor* heuristic, the rendering and collision handling processes can effectively be viewed in reverse order without loss of generality. The heuristic achieves this by proposing that the available collision handling times be dependent on the previous rendering frame time rather than the yet unknown current frame rendering time. Each simulation frame time is still maintained within a constant time state adhering to strict timing rules, but rather it has now been *phase shifted*. Let t_{Frame} be our total frame

time, then we may now express this time as comprising of the following sub-event times:

$$t_{Frame} = t_{Render} + t_{ODEsolve} + t_{ColDetect} + t_{ColResolve} \quad (5.1)$$

Figure 5.2 illustrates this new perception of the overall simulation scheduling within the time adaptive framework. The cumulative result of this scheduling strategy globally maintains coherent timings and thus the overall simulation executes as is expected without any unwanted temporal artifacts.

The total amount of time allocated to collision handling (i.e. both collision detection and response) is calculated from the remaining time after (the previous) scene rendering time and the initial simulation states update time has been taken into account. It should now be quite clear why it was advantageous for us to investigate alternative methods to the popular Runge-Kutta method so as to have as quick and efficient an ODE solver as possible. We only have so much simulation time allotted to us for each frame that we wish to *waste* as little of this time as possible before the collision handling phase, so that we are given enough of an opportunity to handle discontinuities that arise in the simulation to within an acceptable degree of plausibility. In the actual system implementation, we utilised the proposed RKVerlet method from Chapter 4 in the ODE solver as this method has been shown in tests to provide a moderate speed advantage over the popular Runge-Kutta method, giving reasonable accuracy and stability to within acceptable bounds.

From this remaining simulation frame time, the scheduling protocol exerts an initial fair timesharing policy, whereby both collision detection and response stages get allocated an equal share of whatever frame time is remaining. However, the final amount of time that is actually allocated to the collision response stage is wholly dependent on the time taken during the collision detection stage. Figure 5.3 illustrates two timing allocation possibilities for the collision resolution stage during this collision scheduling. If all collision detection completes before all its allotted time has been spent, this remaining portion is thus offered to the response processing stage. In doing so, the time allotted to dealing with collision response is further expanded. This then indicates that, if we so wish, we may consider more points of contact for collision resolution since time now allows us to. However, if the collision detection phase exceeds its allocated time, processing is hence interrupted at the maximum allowable time for this frame. The collision detection algorithm must then deal with this interruption in some way, returning enough useful information so that our collision resolver may resolve any potential collisions in a plausible manner that will not compromise overall system integrity. Thus, the time allotted to dealing with collision response is the initial time

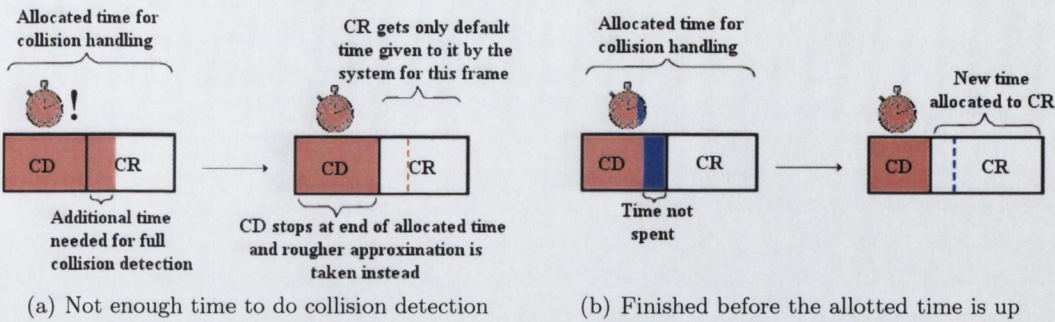


Figure 5.3: Two timing possibilities during adaptive collision scheduling

that was agreed upon before any collision handling had begun. Once the collision response stage completes its task, the simulation frame timer is thus reset and the next *phase shifted* simulation frame is processed.

To allow for more flexibility in the scheduling of collision handling, it is possible to adaptively assign percentage time slices to collision detection and response in accordance with importance of processing. In the previous paragraph it was stated that both collision detection and response are allocated an equal share of the remaining simulation frame time. This of course is not a strict rule and can be augmented to suit simulation needs. If it turns out that it is more important to have more accurate collision detection or that collision detection is taking a lot more time than collision response processing, then we could allocate say, 75% of the remaining simulation frame time to the collision detection stage and the remaining 25% to the collision response stage or vice versa. The allocation of simulation frame time for each collision handling stage thus reflects the percentage allocation for each of the stages. Such a heuristic can be adapted for use with a prioritisation scheme such as that described by O’Sullivan [118]. In O’Sullivan’s scheme, objects are prioritised given their perceptual importance to the viewer depending on their position on the screen and other heuristics. Collision processing is thus given more or less attention based on the events’ perceptual importance to the viewer.

Depending on the resolution technique used, an over-abundance of collision variables given to the resolver can potentially cause the time taken to fully resolve all collisions to become as expensive as the collision detection phase itself (as will be discussed later, a Linear Complementarity Problem is solved for in our system, which can get very computationally expensive with increasing variables passed into the system). The previously discussed equal time share in the collision handling phase was thus so chosen to account for such cases. Although we place an initial equal importance for both collision detection and resolution,

in practice, we have allowed the user the flexibility within the time adaptive dynamics system to increase or decrease the percentage time share of the collision detection phase over the collision resolution phase. A reduction strategy for collision variables to adapt for timing constraints is discussed in Section 5.3.

5.2 Progressive Collision Handling

The collision handling phase within the simulation loop is the most important phase in terms of large computational overhead considerations. It is true that collision detection is the major bottleneck in physically based simulation but it is also an area that presents significant opportunity for time accuracy exploitation. Many collision detection algorithms work to accurately define the necessary collision data in order to resolve collisions. It is feasible to forgo such accuracy for a more approximate approach to collision handling so that further simulation speedup is achieved. In light of this observation, we propose a time adaptive approach to collision detection as this provides many advantages. We adopt and build upon the time-critical approaches of Hubbard, O'Sullivan and Dingliana and propose an improved adaptive adjustment of the data we give to our response resolver so as to conform to available frame time.

For time-critical collision detection, our framework adopts the ideas presented by Hubbard and Bradshaw [21, 22, 23] in terms of the sphere-tree BVH representation and from O'Sullivan and Dingliana for cases of sphere-tree node traversal. We leave an examination of the proposed collision resolution approximation strategy presented by Dingliana and O'Sullivan and the issues that may arise for certain situations within a dynamic simulation for the next chapter. There, we shall also present alternative novel solutions to such problems that may arise. Let us first discuss further the issue of time-critical collision detection within our proposed framework.

5.2.1 Collision Detection with Sphere-trees

Adopting a sphere-tree BVH for time-critical collision detection has many advantages, one being that spheres are rotationally invariant. This means that we do not have to consider the orientation of spheres during updates or collision detection, the only information we need is the position of the sphere's centre in relation to the body (or part of the body) that it bounds during simulation update. This makes for very efficient proxy modelling of a rigid body's spatial occupancy.

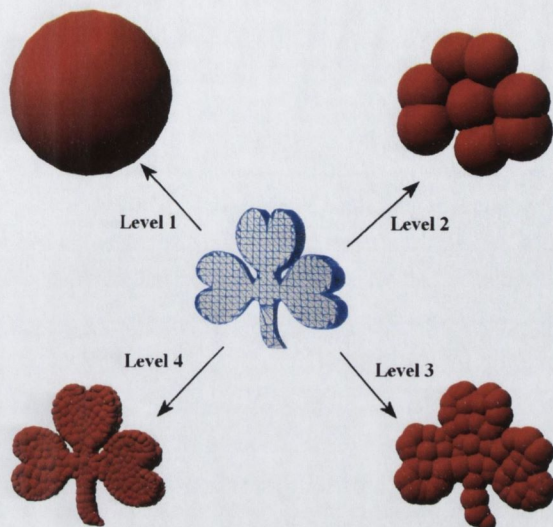


Figure 5.4: Four levels of a sphere-tree representation of a shamrock model

For the generation of sphere-trees in our time adaptive system we adopt Bradshaw's [21] and Bradshaw and O'Sullivan's [22, 23] sphere-tree generation algorithms. These algorithms follow the same basic idea as Hubbard's medial-axis algorithm for generating sphere-trees for arbitrary closed polyhedra. However, through the use of an adaptive medial-axis approximation scheme, in general, Bradshaw's algorithms are a vast improvement in terms of tight and efficient spatial occupancy approximation versus the number of spheres required. Bradshaw presented five variant algorithms that utilised an adaptive medial-axis approximation when constructing sphere-tree bounding volumes. He called these different algorithms, *Expand*, *Merge*, *Burst*, *Spawn* and *Combined*. Figure 5.4 illustrates four levels of a sphere-tree BVH approximation for a model created using Bradshaw's *Combined* algorithm. In terms of sphere-tree construction algorithms we offer nothing new in this area. As such, we shall not further discuss Bradshaw's algorithms here but refer the interested reader to Bradshaw's PhD thesis [21] and accompanying papers [22, 23].

Sphere-tree BVHs also offer the advantage of a very efficient means of performing collision detection, requiring only an elementary distance check. If needs be, checks for other basic primitives against spheres (e.g. boxes, cones etc.) can also be conducted in a similar efficient manner. During collision detection between two sphere-trees, any pair of spheres can be efficiently checked for possible collisions by comparing the magnitude of the distance between the center of each of the spheres and the sum of their radii. A collision is thus detected if

$$|C_1 - C_2| \leq (r_1 + r_2)$$

where C denotes the center coordinates of the sphere within the sphere-tree being checked with r denoting its radius.

It is usual to check for sphere overlaps starting with the root nodes of each sphere-tree, progressing further down the tree only if positive collision results have been recorded at the parent nodes. In most collision detection approaches that use hierarchical BVHs, a successful collision hit is only recorded once the algorithm has navigated down to the leaf level of the sphere-tree or whatever the BVH might be. The leaf nodes will further hold information about which polygons they encapsulate, so an exact polygon to polygon intersection test or similar will be then be utilised to recover information that is useable for collision response.

In a time-critical approach to collision detection, collision processing may be interrupted at any stage regardless of whether we have completed processing the entire tree or not. In such cases we cannot simply “throw away” these tests and regard the bodies as non-colliding. If we were to do so, bodies that are in contact with each other may drift into or pass through one another causing unsightly simulation inconsistencies. So, in time-critical collision detection, unless an entire level of a sphere-tree tests negative for collisions before detection is interrupted, we assume that objects are in contact regardless of sphere-tree level at process interruption time.

5.2.2 Traversal and Detection Scheduling

Sphere-tree data structures provide an excellent means for us to progressively refine the approximation of collisions over time during narrow phase collision detection. The hierarchical structure allows for very course approximations initially during collision detection but facilitates progressive refinement the further we traverse down the tree. From the previous discussions, it is of obvious importance for us to have an efficient sphere-tree traversal algorithm that will also guarantee to give us good approximations given an interruption to processing.

Within the time adaptive framework, we chose to adopt Palmer and Grimsdale’s *staircase* sphere-tree traversal method [123], utilising O’Sullivan’s adaptation for interruptible collision detection. We chose to deal with traversal of sphere-trees in this manner as it is the

most efficient approach to dealing with pairs of potentially colliding sphere-trees in an interruptible manner. Hubbard traversed sphere-tree pairs by simultaneously going down the levels of each tree and comparing pairs of children at each level for overlap. For a tree with n children, such a method would constitute a worse case requirement of $O(n^2)$ overlap tests. The *staircase* method on the other hand greatly simplifies this to $O(n)$ necessary overlap comparisons by simply traversing down a level of one sphere-tree at a time. Figure 5.5 illustrates the process of utilising a *staircase* traversal approach for a pair of sphere-trees during collision detection adapted for time-interruption.

For ease of discussion, let us assume that we have two potentially colliding bodies, with sphere-trees A and B for which we begin to check for collisions at the narrow phase as shown in Figure 5.5. Adopting O'Sullivan's approach to managing potential collisions, the narrow phase algorithm in the time adaptive framework maintains a *collision queue* which is utilised to store possible collision sphere pairs. This collision queue has two main purposes. It enables us to efficiently schedule collision detection for a list of potentially colliding bodies in a round-robin, breadth-first manner and it also manages efficient staircase-like traversal of sphere-trees during collision detection. Working from the given example of Figure 5.5 the algorithm proceeds as follows:

1. The collision queue is initialised with root spheres of potential colliding body pairs (Figure 5.5 (a)).
2. While there is still time *and* there are still sphere pairs in the collision queue, then take the sphere pair at the head of the queue and refine down to the next child level of either sphere in a staircase fashion (Figure 5.5 (b) to (f)).
3. Place all new lower level contacting sphere pairs found that are not both leaf nodes of the sphere-tree at the *end* of the collision queue and remove current sphere pair from *head* of the collision queue. Ignore all non colliding spheres at the new level and their sub-trees.
4. If contacting sphere pair are both leaves of the sphere-tree hierarchy then place them into a positive collision list.
5. If time for collision detection has expired, then interrupt collision detection process. Treat all sphere pairs in the collision queue as actual collisions and move them into the positive collision list¹.

¹For example, if collision detection was interrupted at the level represented by Figure 5.5 (d), then the

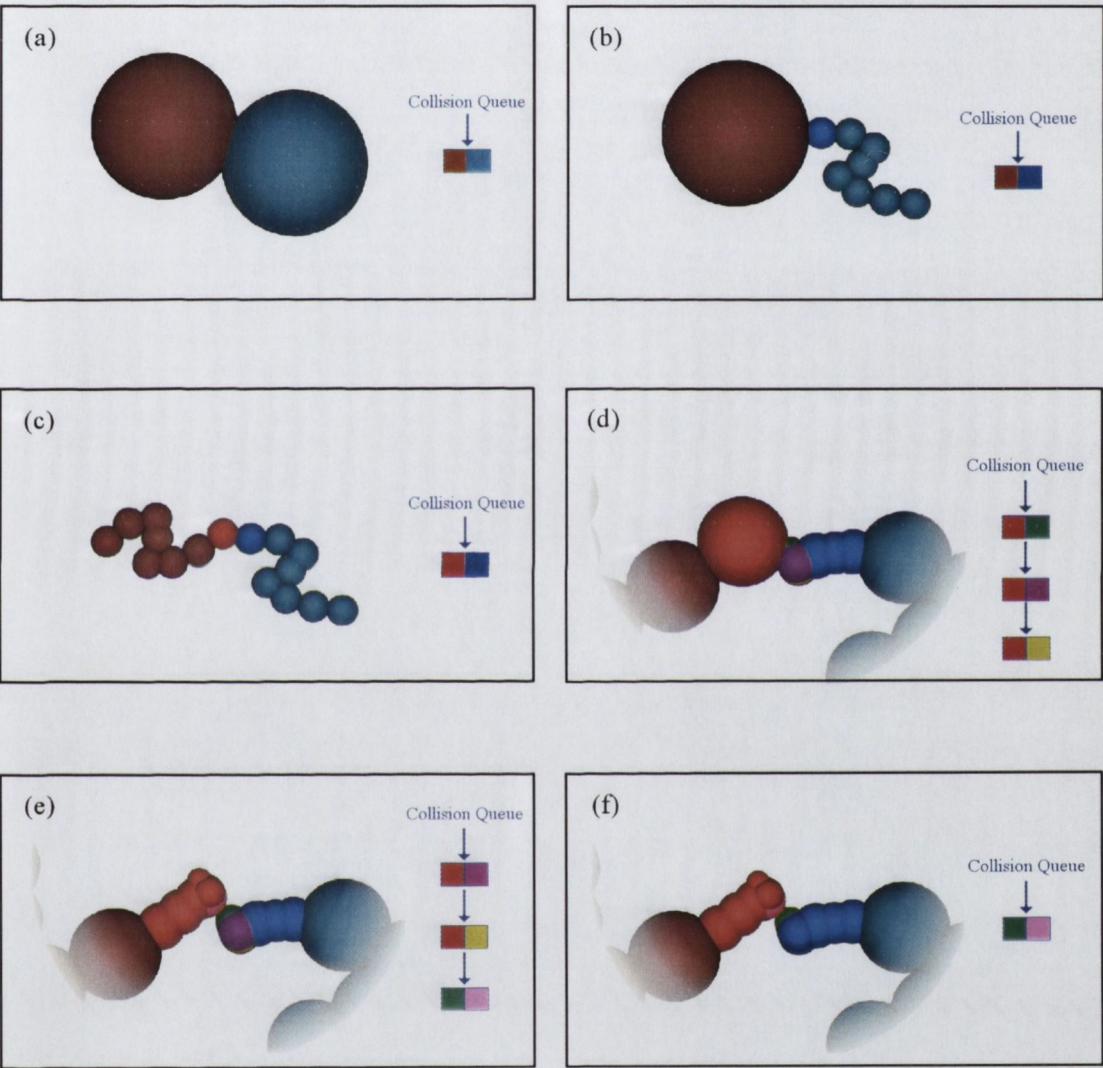


Figure 5.5: Staircase traversal of sphere-trees adapted for interruptible collision detection

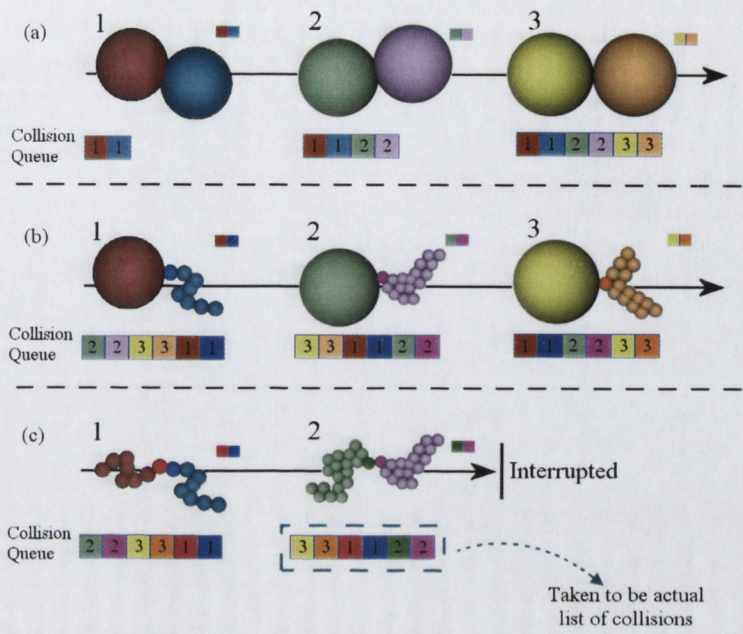


Figure 5.6: Breadth first collision detection scheduling over time for more than one colliding pair of objects. In this example we have numbered each object pair 1, 2 and 3 respectively.

6. Otherwise, go back to step 2 and repeat.

The choice of which sphere to be refined to its children’s level for further testing is determined by whether the sphere from tree A is at a higher level than the sphere from tree B or vice versa. Subsequently, if both spheres are at the same level within their respective sphere-trees then we choose to test the children of tree B’s sphere with tree A’s sphere.

Recall that the collision queue that we adopted from O’Sullivan was utilised not only for efficient interruptible *staircase* traversal of colliding pairs of bodies but also it enables us to efficiently implement a round-robin, breadth-first scheduling approach to collision detection. Figure 5.6 illustrates how the collision queue is utilised in such a manner, given more than one pair of colliding bodies. As can be seen, for each potential colliding pair of bodies, their sphere pairs are put into the collision queue following the previously highlighted algorithm. The collision queue, however, also acts in this case to schedule the traversal of sphere-trees in such a way that all collision pairs get an even turn per sphere-tree level in a round-robin, breadth-first manner. In this way, we maintain a fair detection scheduling policy for all potentially colliding body pairs.

red-green, red-purple and red-yellow contacting spheres are moved into the positive collision list and treated as the BVH collisions found during collision detection

Breadth First Vs. Depth First Scheduling

A breadth first scheduling approach was decided upon as it was deemed the fairest detection scheduling policy. A depth first scheduling policy can be very easily adapted to the above traversal algorithm if so desired instead. The advantage of adopting a depth first scheduling strategy would be that we may at least be able to refine collisions for one or more colliding pairs to the lowest level of the sphere-tree before interruption, resulting in two or three objects being resolved very accurately while all the other objects are resolved only to the highest sphere-tree level. However, we do not desire such an outcome as the results of such a scheduling scheme may further enhance unwanted visual biasing and thus further detract from the overall plausibility of the simulation.

To further explain the above statement, let us take an overly simplified example. Consider a simulation consisting of several objects, all geometrically similar, placed beside each other (in a copied and pasted type manner) such that they are all equal height above a perfectly smooth plane. Now, if we start the simulation such that all the objects are dropped at the same time onto the plane, we expect that all collision response for each object act in an identical manner as all collision properties should be identical due to the nature of the simulation. Assume depth-first scheduling was employed and that only two or three objects were resolved before interruption. Visually, we would see in the animation, two or three of the objects bouncing off the plane in a *pleasing* manner, while all other objects will exhibit a noticeably *rougher* response since they have been resolved at the root node. We tend to notice these inaccuracies more as the two or three *well refined* objects give us a point of comparison for the collision response and also the difference between sphere levels during collision resolution is at its maximum in this case (i.e. lowest level versus highest level spheres). However, assuming a breadth-first scheduling strategy, any differences from higher level approximations may be harder to judge as all objects are being resolved at the same level, and if not at the same level, the difference between sphere levels is not as great as that of the depth-first scheme, ranging from a difference of one level at the most. This in turn, leaving less of a comparison for the observer to judge disparities in collision response.

O'Sullivan [118] proposed using perceptual metrics to prioritise collision detection scheduling whereby those events that are deemed to be perceptually more important are put at the head of the queue. As such, they will then be dealt with first leaving the perceptually unimportant collisions towards the end whereby very coarse approximation to collisions is more accepting. Such a scheme has its merits but is very hard to enforce and relies heavily on a good perceptual metric and current scene configuration. Another approach, involving

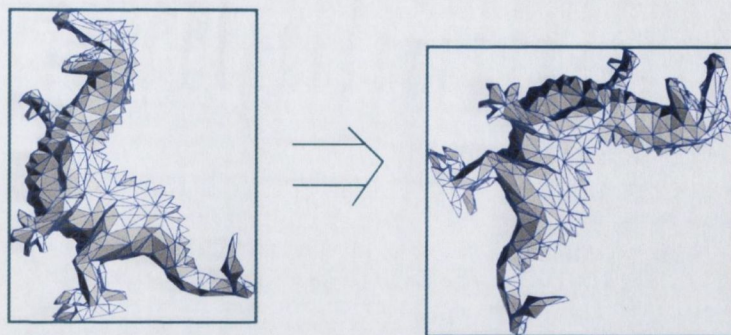


Figure 5.7: Axis aligned bounding box as object moves through scene

a bin sort of collisions into a number of priority lists, within which round-robin scheduling occurred, was found to give the best tradeoff.

5.2.3 Optimising Broad Phase Detection

Like O'Sullivan and Dingliana, to further improve the efficiency of collision detection we too chose to utilise Baraff [6] and Cohen et al's [38] *Sweep and Prune* algorithm for the broad phase collision detection within the framework. This scheme is conducted using axes aligned bounding boxes to encapsulate the objects within the scene. The extents of each of the bounding boxes within the x, y and z axis are then put into sorted lists for efficiency, whereby overlap testing can be quickly conducted through the sorted list structure. Temporal coherence can be exploited to further enhance processing if the sorted lists are maintained over to the next simulation frame and resorted using the *insertion* sort method. A potential collision between objects is only reported if overlap tests within all three axes report back as positive.

As an object moves across the scene it may rotate, thus causing its bounding box to expand or contract to accommodate the changes. As such, for each simulation frame we must recalculate the axis aligned bounding boxes of each object (Figure 5.7). We may simplify this calculation if we don't mind over-approximating a little bit and assume a fixed size bounding box. We can integrate the sphere-tree hierarchy even more into the collision detection phase by utilising the top level sphere that bounds a body to aid us in approximating the extents of the axis aligned bounding box in the *Sweep and Prune* algorithm. The extents in the x, y and z axis can be simply calculated by utilising the root sphere's world centre and radius information.

5.2.4 Adaptive Collision Response

If we are to model the physical behaviour of rigid bodies well in a dynamics system, then given that there is a collision there must also be an appropriate response. While collision detection is only kinematically aware of the environment, we must also be physically aware so that any discovered collisions are resolved as appropriately as possible. Without appropriate collision response, objects that are deemed to be colliding will ghost through each other, leaving undesired gaps in the perceptual believability of the simulation. Similarly, a poor collision response estimation from inaccurate assumptions or inaccurate collision data will cause visual impurities, leading to an overall degradation of the desired physical plausibility within the dynamics system.

In the situation of collision response resolution, it is easiest to calculate and deal with impulses for a single point of contact. This may explain why it is easier to find literature in collision handling that considers cases of a single contact point for collision response than it is to find literature that touches on the subject of multiple contact collision response. Given multiple contacting points, it may be the case that ad hoc solutions are employed by researchers to deal with such outcomes.

One of the earliest complete treatments of the subject of collision detection and response for physically based simulations was conducted by Hahn [68] and Moore and Wilhelms [113]. Both papers give extensive consideration to the problem of dealing with response for single point collisions and both very briefly suggest solutions to the problem of multiple point collisions. Moore and Wilhelms tackle this problem by *inventing* a series of individual non-related impulses that act exclusively on each point of contact which is then incorporated into the matrix of 15 linear equations to be solved for. Hahn on the other hand suggested the solution of obtaining an impulse for each point individually and then summing up the resultant impulses that acted on the same body. This, however, was only suggested for situations whereby a body was in simultaneous contact with multiple other bodies at a single point per contact and the idea was not extended to account for multiple contact points for each pair of contacting bodies.

The most comprehensive treatment to date on the issue of tackling the tricky situation of response impulses for multiple point collisions can be found in work conducted by Baraff [3]. Baraff models simultaneous collisions on each colliding body as a collection of impulses acting at each colliding point, each having an effect on how the other behaves. This method is based very much on the same non-penetration techniques presented for resolving contacting forces to prevent interpenetration in resting contact situations, which is

the main body of Baraff's paper. As such, the bulk of derivative details and assumptions for the simultaneous collision impulse section in the paper are omitted and left to the readers' own devices. This is, however, a far from trivial procedure.

Before we proceed any further, it must be made clear that we are not talking about discovering the penetration of objects but rather how to avoid object penetration. In order to do this we must first find the point of contact for each colliding object (i.e. the point at which the objects touch/hit each other) and it is then the job of the collision/contact resolver to ensure that given these points of contact, each object be treated in such a way as to thus avoid any possible further interpenetration. Part of the overall job of any collision handling system is the prevention of objects from *over travelling* (due to the discrete nature of computer simulations), such that they are already interpenetrating when the collision is finally detected. We have already mentioned various schemes that aim to avoid such situations in Chapter 2 and we will further discuss this problem in Section 5.4. For the purposes of discussion heres, let us assume that the collision detection algorithm has found the desired points of contact at the earliest possible time of collision so as any undesirable interpenetration caused by the nature of discrete time stepping in computer simulations is avoided.

In dealing with collision response, we wished to maintain as much physical plausibility as possible and so we chose to deal with multiple contacting points in a simultaneous manner, analogous to the approach briefly described by Baraff. Given a set of all colliding bodies in the scene with appropriate collision information (i.e. points of contact, normals of contact etc.) we formulate and solve a *Linear Complementarity Problem* (LCP) of the form:

$$v_{rel}^+ + \epsilon v_{rel}^- \geq 0 \quad (5.2)$$

$$f \geq 0 \quad (5.3)$$

$$f(v_{rel}^+ + \epsilon v_{rel}^-) = 0 \quad (5.4)$$

accounting for all contacting bodies in a simultaneous manner. In the above formulation, v_{rel}^- and v_{rel}^+ are the relative velocities in the normal direction of the colliding bodies before and after the collision respectively. The terms f refer to the impulse magnitude that we wish to solve for and ϵ is the coefficient of restitution, which is a term that determines how *bouncy* a collision is. We will not go into a full discussion, derivation or expansion of the above equations here. Instead, a comprehensive discussion and treatment on dealing with multiple points of contact with full derivations from first principles based on Baraff's

suggestions is presented in Appendix A. We further refer the interested reader to Giang et al [55] for further discussions on the subject matter.

In both cases of single or multiple contact point considerations, the computation time for resolving collisions grows as the number of contact points given to the resolver grows. Like the collision detection phase, the collision response phase is only allotted a certain amount of time to resolve collisions. However, unlike the collision detection stage, it is not possible for us to arbitrarily interrupt processing if we run out of time. Once the collision resolver has begun to resolve collisions it must be allowed to complete. If prematurely interrupted, the resolver may return highly erroneous results that will cause severe disparity with the expected responses of the colliding objects. These responses may range from object interpenetration to an exponential increase in the energy of the system, causing it to explode. Therefore, within the time adaptive framework, given a time constraint for collision resolution, it is necessary to be able to predict how many points of contact our collision resolver can deal with within that time limit. From the results of the collision detection stage, it is possible for us to gauge whether our resolver can handle the amount of data obtained from collision detection within the allotted time or not. If it is found that too much data is being given to the resolver, then a point reduction scheme must be put into place to reduce the amount of data that the resolver has to deal with.

To account for resolver timing restrictions, an approximate response solver timing table for the collision response phase is predetermined at system start and set up as a preprocessing step. We later use this timing table in order to try to predict how many points of contact we may deal with within the allotted response time, given the present contacting point set from the collision detection phase. The timing table is determined by doing an initial set of collision resolution calculations based on a given set of random data at the start. The number of points to be resolved for is gradually increased until calculation exceeds a specific predetermined maximum allowable frame calculation time. At this point we have found the upper limit to our response resolver given the current machine processor specifications and hence our timing table is also complete. In this way, the framework adapts not only to the time allowable for collision response but also to the computing power of a machine, allowing more points to be dealt with during the resolution stage for machines with greater processing capabilities.

During the collision resolution phase of a simulation time frame, the system conducts a quick lookup on the simulation response timing table. The number of contacting points from the collision detection phase is used to quickly determine from the approximate response

timing table if we can handle that much data within the available time. If it finds that it does not have enough time to process all desired contacting points, the number of points are therefore reduced through a point reduction scheme until our resolver timing table predicts that they can be processed within the remaining frame time. In doing so, the collision response phase adapts to the remaining frame processing time given to it and thus implicitly also adapts to the speed of the available maths processor on the machine on which the simulation is being run.

One potential problem that we may encounter with such a scheme is that, in general, the collision resolver is not guaranteed to be always constant in terms of time taken to calculate results for any given number of points. This is highly dependent on the collision resolution scheme that is being used by the resolver. If the collision resolution scheme utilises something along the lines of an iterative or simplex-based algorithm, whereby variables are constantly changing and saltating around the system in order to eventually converge to an answer that will satisfy system constraints, such methods cannot guarantee a constant resolution time for fixed numbers of variables. Time of convergence to an eventual answer is dependent on the initial values given to the system to be solved. For such cases, pre-processing of the timing table may dictate that we can resolve, for example, ten contacting points in the space of three nanoseconds but in reality this time may fluctuate depending on the initial contacting data given to the resolver during the time of contact. We may tackle such a potential problem in two ways. The first is to simply make sure that we choose a *LCP* or *Linear Equation* (or whatever our impulse formulation may be) resolution method that is guaranteed to be fairly constant in its calculation times, regardless of initial system variable data, or one that can be interrupted but still give us approximately reasonable answers. The second solution is to build up an average approximate timing for the number of points within the timing table by accumulating and then averaging timing results during simulations over a period of time. For each run of a simulation, the response timing table continually gets more refined to give us a better approximation of the average processing time required for a given number of contacting points. If such a solution were to be adopted the response timing table must be kept in a file after system shutdown, to be reloaded and further altered during the next and each further simulation run.

5.3 Point Reduction

Due to the nature of hierarchical bounding sphere-trees, there may be many collision points to consider after collision detection has completed. Since we deal with points of contact in a simultaneous approach it is sometimes necessary (not to mention advantageous) to reduce the set of established contact points to avoid too much computational expense and to conform to given timing restrictions. As previously discussed, within the collision resolution phase it may be determined that we do not have enough time to properly solve for all presented contacting points and thus must perform some sort of reduction.

The reduction scheme that we adopt to resolve overly abundant points of contact is a simple if not somewhat naïve one but nevertheless works quite well. For each pair of contacting objects, their set of contacting points is taken and the normals of each point summed and then averaged to give a single normal. The averaged normal is utilised to determine a virtual plane (which is perpendicular to the normal) for the purposes of our point reduction algorithm. Working in the plane's body space, we proceed to project the set of contacting points onto this virtual plane, further treating them as a 2D point set on a planar grid. A bounding rectangle is then constructed around the set of projected points. The min and max u and v coordinates obtained from the set of projected points is used for creating the bounding rectangle. These u and v coordinates are used to describe the bounding rectangle's min and max coordinates. Once the min and max of the rectangle is known to us, the construction of the bounding rectangle is a trivial matter.

This 2D bounding rectangle schema accounts for both concave as well as convex bodies. In either case, the schema tries to choose and return the widest spread of reduced contacting points as possible. The purpose for this is to ensure that we have a stable base for cases of continuous contact so that we can avoid having objects unnecessarily toppling over when they shouldn't do.

In practice it is not necessary to consider the entire region of a contacting face of an object in order to suitably model the desired response of colliding/contacting objects to prevent interpenetration. Take for example a cube resting on a plane. In reality, all the underside of the cube is touching the plane. We however, when simulating the cube, do not require to model contact for the entire region of the underside of the cube (this would not be very practical or feasible), but simply, we can use the vertices of the cube (four in this case) that are contacting the plane for our calculations to prevent interpenetration. This is the underlying assumption that the points reduction algorithm works under. For disjoint bodies, the reduction algorithm, unless requested to consider more points, will usually ignore

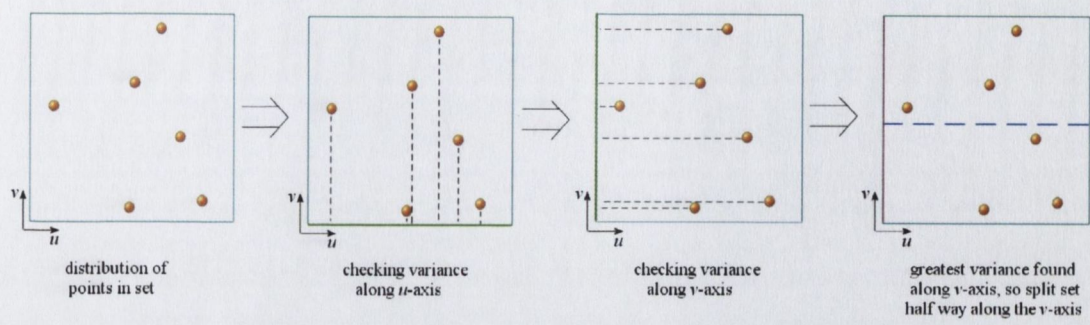


Figure 5.8: 2D example of dividing a point set based on greatest variance along an axis

all inner contacting points of the points spread, considering only the outer points for the reasons previously mentioned. However, as more points are considered, the subdivision scheme employed (discussed later on) will act as such to reconsider various points within the points spread.

The question arises as to the validity of the direction of the orientation of the bounding rectangle. A differently orientated bounding rectangle may give rise to the possibility of different reduced points being calculated. Although we present an approach involving the averaging of normals, other methods such as using the *principal axes of inertia* or utilising an objects *convex envelop* may be further considered. However, even with these other suggested approaches, it is still not certain whether the eventual orientation of the produced bounding rectangle will in turn produce an optimal point reduced set or even if such a thing exists.

Depending on the number of required reduced points for a given set of contact points, the reduction scheme may either further subdivide this bounding rectangle into smaller sub-bounding rectangles, separating the set of projected contact points into smaller sub-regions if necessary. The division into sub-regions is determined by the greatest variance along the bounding u or v axis of the bounding (sub)rectangle for that current set of points (Figure 5.8). For each set or subset of points projected onto the plane and encapsulated within a bounding rectangle, the set of considered reduced points is defined to be never more than four. This is because we determine the reduced set of points to be considered for processing as those closest to the corner of the bounding rectangle that encapsulates it. Figure 5.9 illustrates a simple example of this point reduction scheme. The example shows a reduction to four contacting points given a relatively large set of contacts. The yellow dots represent the discovered contact points during collision detection before point reduction. The points circled in red are the actual points considered after point reduction has been conducted on the set of contacting points. All other points not considered after

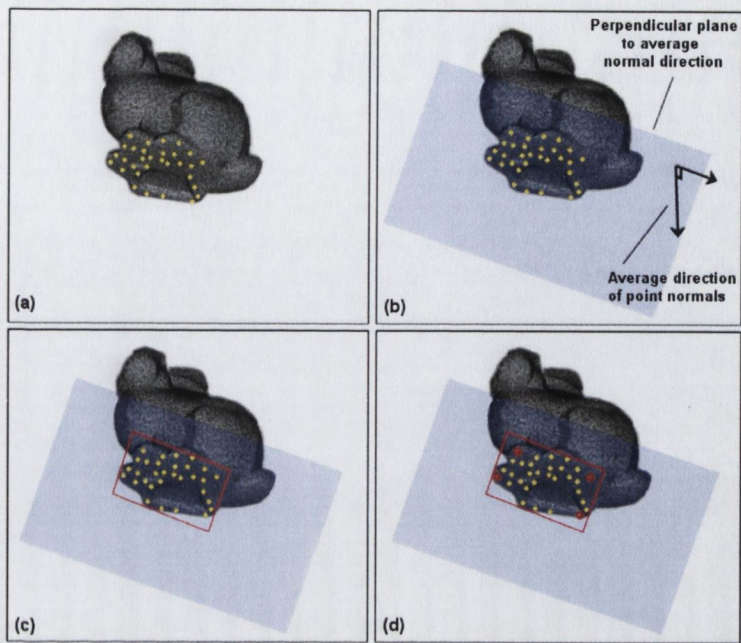


Figure 5.9: Point reduction to four contacting points given a large set of contacts (only one object is shown)

point reduction are thus removed from the points list that is eventually given to the resolver.

After all reduced points have been determined, this new compact list is translated back to its original world coordinates and is processed by the collision resolver. If point reduction involves reduction of any number of points below four, the point set is further subdivided into either two or three subsets and the reduction point is determined as the nearest point to the average of each subset of points within each divided subsection. Of course if there are less than four points within the bounding or sub-bounding rectangle and we require a reduction of four points for each bound, we simply take the points that are present within that bound.

During point reduction between a colliding pair of bodies when adapting to required time constraints for collision resolution, the number of points to be handled between each colliding pair can be considered in two ways. The first scheme is to consider as many points on each pair of colliding bodies as the collision resolver can solve within the allotted time. The second scheme that we can consider is based on the hypothesis that, in order for plausible response to be calculated from the collision resolver, we only need a maximum of four contact points per colliding body pair. Any more than this could be computationally wasteful. With sphere-trees, in general we end up with a lot more points of contacts than we

need. Many of these can be ignored and we reason that only a maximum of four contacting points is needed for collision resolution in many instances. The reason for the choice of this number is that, for a reasonable representation of a stable base during collisions, four conservatively spaced contacting points is enough to model such a situation. Consider again the example of a polygonal cube model falling onto a flat plane, if utilising a sphere-tree to bound the cube model and to determine contact with the plane, we end up with numerous contacting points along the face of the cube if allowing for progression to the lowest sphere-tree level during collision detection. It would be wasteful to take all these points into consideration during collision resolution as only the points at or nearest the vertices of the faces of the cube in contact with the plane (i.e. four points) is in general enough to give a reasonable approximation to collision response.

If adopting the first scheme, given an initial list of contact points sets for each colliding body pair within the collision list, we must have a strategy in place that will allow us to determine which sets we can allocate more points to during the reduction step given the time constraint. In order to adopt a fair point distribution scheme across all colliding body pairs, we first assume that every point set within the contact list will be reduced to a maximum of four contact points whenever possible. In order to allow less of a reduction and hence consideration for more contacting points per colliding body pair, we further separate the chosen contact points set of a colliding body pair into two (or more) subsets to be treated separately by the contact reducer. We base the decision on which contact set (or contact subset) to subdivide by choosing the set within the current contact set list with the greatest variance along the x , y or z axis. We subsequently then divide the chosen contact points set along that axis to give us two separate contact points subsets that the points reducer will treat as a separate set of points to be reduced (See Figure 5.8 for a 2D example). Splitting the sets in this way ensures that a well spread point distribution per subset is left. We can potentially do several scans over the contact point sets list, further subdividing appropriate sets into further subsets until we reach the maximum number of points that the timing table says our resolver can resolve given the remaining time to do so. This set subdivision scheme is just a 3D extension of the subdivision scheme employed by the contact points reducer if more than a maximum of four contact points is specified during reduction.

It must be made clear the difference between the previous 2D and this just discussed 3D subdivision algorithm. The previously discussed 2D bounding rectangle algorithm was used to reduce a set of given contacting points. The algorithm will reduce all given contact sets (i.e. we will have one contact points set for each colliding/contacting body pair) to

four points but however this reduction value can be further increased by employing the 2D subdivision algorithm discussed. The 3D algorithm just discussed however, is used to subdivide sets of contacting points before the 2D reduction algorithm is employed on these sets so that reduction is done in a fair manner when employing the first proposed scheme (i.e. keeping as many points as possible). Let us imagine several disjoint bodies colliding. The 3D algorithm will first try to divide the contact sets for each colliding/contacting pair into further separate sets choosing the most disjoint set (i.e. set with greatest variance) to split first, as we prefer to reduce fine clusters of points rather than points that are already fairly far away from each other. The more we subdivide into separate sets before reducing with the 2D rectangle algorithm, the more points that will be considered for collision response processing at the end, as the 2D algorithm will then have less of a chance of getting rid of as many points per set.

Let us assume that we adopt the second proposed scheme, whereby we only allow a maximum of four contact points per colliding body pair to be processed. If such a scheme were to be adopted then it may be possible for us to consider the likelihood of a more adaptive scheduling scheme that allows further prediction and adaptation of allocation of processing time during collision scheduling. We could utilise the response timing table to adaptively choose the percentage of the remaining simulation frame processing time to allocate for collision detection and response. Since we know that we can only ever have at most four contacting points per colliding object, we can make a quick calculation based on the number of objects in the scene and take a worse to average case collision timing scenario from the response timing table. We can thus approximate the time it will take for collision resolution from this information and hence adjust the percentage allocation for remaining simulation frame time so that collision resolution gets allocated a percentage time that will allow for resolution of the predicted number of contacts, leaving collision detection with the remainder.

5.4 Tunnelling Problems

One of the problems in taking discrete time steps during simulations is that we can only ever *sample* bodies at discrete locations in space. Given fairly large time steps or relatively large velocities, bodies in motion can potentially pass through each other without being detected or else be severely interpenetrating when a collision has been determined. Such events are commonly referred to as *tunnelling*. With sphere-trees, the tunnelling problem

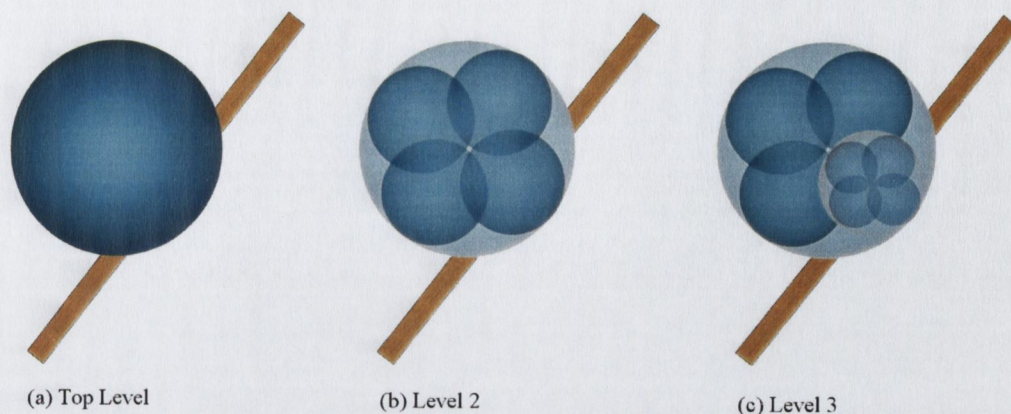


Figure 5.10: Example of tunnelling problem at different levels of a sphere-tree. As can be seen in (c), the small sphere has almost completely passed through the board.

can potentially get progressively worse the further down the tree we go. This is so because spheres get progressively smaller to give a tighter fit around the body as we further refine the collision information, as is illustrated by Figure 5.10. If this were to happen, it could lead to bad estimation of contacting points and hinder good contact normal determination.

To avoid the potential problem of tunnelling as we further refine collision approximations if using sphere-trees in interruptible collision detection, Dingliana [47] proposed the notion of *recursion capping*. The idea behind recursion capping is to stop traversing further down the sphere-tree once it is determined that any further refinement will cause excessive tunnelling of contacting spheres and thus bad contact approximations. Dingliana suggests that the determination of whether or not to stop traversing further down the sphere-tree be based on the criterion that the sphere's radius within the current refinement level be not larger than the magnitude of the velocity of the object just before the collision. This is a very naïve approach at best, as we cannot say that, just because the velocity of an object is greater than a given sphere's radius, there is a severe interpenetration problem. This is not usually the case. It could be that we have a reasonably good contact situation for all levels of the sphere-tree. Such a scheme could hinder any contact refinement when it is perfectly reasonable to proceed to do so.

Of course, we could somewhat alleviate the tunnelling problem by taking smaller time steps so that we do not have too much of a gap between each simulation frame. This would ensure a lower likelihood of missed collisions and less severe interpenetration during collisions. However, such a solution is not an ideal one. Generally speaking, it is advantageous to have as large a time step as possible in interactive simulations. Also, our simulations

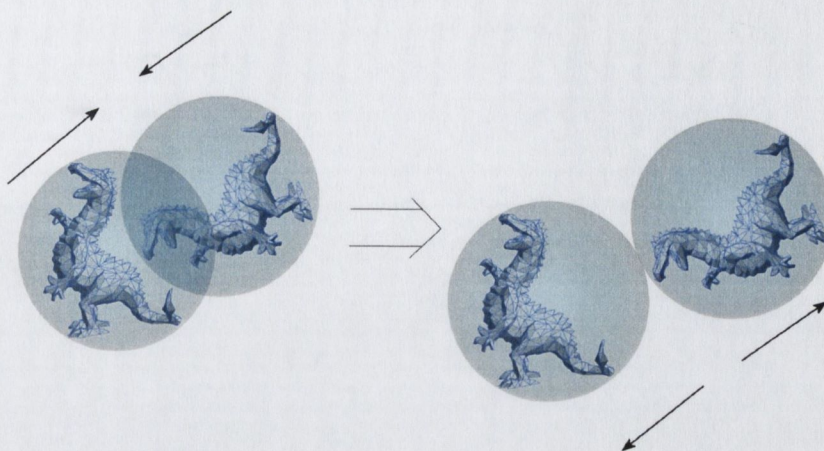


Figure 5.11: Unnecessary backtracking

will get progressively slower as the time step gets smaller due to the fact that each simulation frame is not displaying as large a change in motion over time. We can only hope to reduce the time step to some minimum before the lack of desired animation speed becomes detrimental to the overall simulation.

In Chapter 2 we briefly discussed various methods employed in tackling the problem of tunnelling during collision detection. *Conservative Advancement* (CA) approaches such as those utilised by Hubbard [76], Redon [133] and Cameron [28], tried to predict time of collision, adapting the time steps to creep up to the time of collision. *Retroactive Detection* (RD) approaches used by Baraff [6] and Mirtich [111] employed backtracking given cases of detected interpenetration. For implementation within time adaptive dynamic situations, whereby the accuracy of collision approximations are allowed to change in order to adapt to required timing constraints, such RA schemes may not be very feasible or very intuitive to incorporate as will be explained later. However, although CA schemes are probably less efficient than RA schemes, it would have been interesting to further test and evaluate the CA schemes for use within the time critical dynamics framework. Unfortunately, due to timing constraints we have not to date done so but rather leave this for future work.

As just mentioned, considering the situation within the proposed time adaptive framework whereby interruptible collision detection is utilised, it is not feasible to employ a backtracking scheme during collision detection. This is because, in general, it is not very intuitive to backtrack to the first time of collision, considering the time-critical refinement schema of the collision detection algorithm. It is difficult to determine the level of refinement that would have been proposed by the system between colliding objects based on

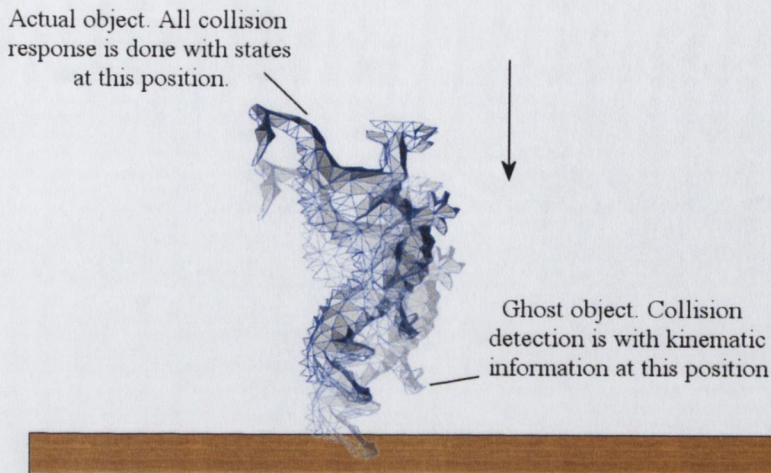


Figure 5.12: Alleviating tunnelling problem through a lookahead prediction approach.

the yet unknown time availability during first time of contact. Also, we cannot intuitively backtrack, presuming the same amount of allotted simulation time, without destroying the strict timing constraints we have placed upon each simulation frame. Furthermore, detected collisions are only initially very rough approximations of contact, especially so if only the top refinement level can be considered during a collision detection phase. As such, it may be that we are unnecessarily backtracking. If our algorithm was allowed to further refine the collisions, we may discover that indeed the objects have not yet collided as illustrated in Figure 5.11. Within a refinable collision detection scheme it may be sometimes advantageous, if not necessary, to allow BVHs to interpenetrate at the higher echelons of their hierarchies because of the unnecessary backtracking problem previously described, if we wish to be able to further refine our collision data.

As an alternative, we propose a more simplistic and efficient *lookahead prediction* approach. In such an approach we propose temporarily updating a *ghost* copy of the rigid body after each initial ODE solver update and using this *ghost* copy for collision determination. We illustrate this in Figure 5.12. A similar prediction approach was also used by Milenkovic and Schmidl [105] and Guendelman et al [66].

Differing from these authors, we do not conduct a full update step and only consider the linear domain during the prediction update. After the initial body states update, before we check for collisions and utilising a *ghost* copy of the rigid body, we conduct another quick update using only half the time step so that it would seem that the body has moved further than it actually has. Algorithm 6 gives a general overview of our proposed *lookahead scheme*

within a typical simulation cycle.

Algorithm 6 Lookahead scheme employed in a simulation cycle

```

while Simulating do
  {Update all Bodies to current time}
  {Copy updated Bodies to GhostBodies}
  {Update all GhostBodies to  $\frac{1}{2}$  time step ahead}
  {Do Collision Detection on GhostBodies}
  if Collision Detected then
    {Resolve Collisions using states information of non ghost Bodies}
  else
    {Move onto next simulation time and repeat from start again}
  end if
end while

```

Since we utilise a *Verlet* update approach within the linear domain, the second update is a quick and accurate prediction. We chose to only use a prediction update step of half of what the actual time step is, as we do not wish to overshoot or have too much of a potential simulation gap during collisions for bodies that display relatively fast motion.

In the collision resolution phase we use the states of the bodies at the first update time rather than at the prediction update time. To consider the body states at the prediction update time during collision resolution would be incorrect, adding extra energy to the system as the bodies in collision did not actually move that far. Such a scheme does not guarantee that we will avoid all undetected collisions for objects moving with high velocities or with large time steps, but does go some way towards preventing missed collisions in such cases. Also, the scheme addresses the problem of severe tunnelling at lower levels of our sphere-tree hierarchy by virtue of the fact that it tries to deal with cases of severe interpenetration before they happen by trying to predict such circumstances and disallowing the bodies to go further than they should.

It is presumed in the algorithm that fast rotating objects without any or little linear motion is rare. If an object is spinning quite rapidly with its centre of mass in the one spot or moving very slowly in the linear domain (i.e. such that it looks like it is fixed to a pivot) then the algorithm will not account for this. If we wish to account for such situations or similar then we must also consider updating the rotational domain of the ghost bodies as well. We do this at a slight increase in overall computational cost. As this update is only meant to be a approximation to predict any impending interpenetration problems a fast integration algorithm such a simple *Euler* update in the rotational domain can be used.

It could be argued that this lookahead prediction scheme can potentially produce large visual gaps between objects during collisions. This may only be true for objects moving at high speeds because the distance travelled during the prediction step is not as great for objects moving at lower speeds as the potential for missing a collision or severely tunnelling during a collision is reduced for such circumstances. The nature of the prediction step takes this into account. For bodies travelling with relatively large velocities, we can exploit a person's visual uncertainty for judging reactions during high velocity motions [120]. This visual uncertainty can only be hindered if we allow simulation frame rates to drop below acceptable real-time performance levels. Of course, detecting collisions in this way does not guarantee physically accurate results but neither does our time adaptive progressive refinement schema. Our goal here is not the achievement of perfect physical modelling, but rather the production of simulations that will pass as both physically and visually plausible to the viewer.

5.5 Further Discussions And Results

We present here further discussions on the approaches proposed in this chapter along with some results of tests conducted.

A Complete Time Adaptive Simulation Cycle

Although in our discussions in this section, we only considered the collision handling phase for time adaptive computation, it is however completely feasible to make all phases within a given simulation frame time entirely time adaptive. As was previously mentioned, the rendering phase can indeed be made interruptible, and there already exists various techniques for interruptible rendering. We did not concern ourselves with this area in our work but rather opted to concentrate on the area of dynamic simulation leaving this for future work.

Similarly, it is also entirely feasible to consider the inclusion of a time adaptive ODE solver within the simulation time frame as well. If we were to consider a time adaptive ODE solver for each simulation frame time, the protocols for considering appropriate allowable times for states updating would be in a similar manner to that of the collision handling timing protocols previously discussed. Unfortunately, it is not feasible to make the integration step interruptible because for an object to advance forward in time, the ODE solver must be allowed to complete. Since the typical computation time of an integration step for one object will be known to us, it may be possible to predict which ODE solver algorithms we

may use within a simulation time frame given an allotted ODE solver time quota.

In our work, there was consideration in making the ODE solution phase time adaptive as well but it was concluded that the collision handling phase was the most critical phase within the simulation loop. So instead, it was opted that we try to reduce overall integration times as much as possible (of course trying to maintain as much accuracy and stability within the process), thus giving as much remaining simulation time to collision handling as possible.

Why Sphere-trees?

Our choice of using sphere-trees for collision detection was due to two reasons. The first being that we found collision detection with spheres in general were more computationally efficient than any of the other bounding volumes. It is however possible to utilise other bounding volume hierarchies such as a box hierarchy within the narrow phase collision detection. Our scheduling and traversal algorithms will remain unchanged. In fact, if such a hierarchy was to be utilised, the broad phase collision detection as mentioned in subsection 5.2.3 could simply use the root node for the bounding box without the need to recalculate the bounding box again for the broad phase collision detection. However, as is the case with collision detection, the expense is within the narrow phase as testing collision between boxes require a series of overlap tests as opposed to spheres which only require a quick distance check. However, different BVHs may better approximate certain shaped objects which in the long run may give us better overall coverage thus reducing the number of collision pairs that need to be checked and solved for. We leave the comparison of different BVHs for tightness of fit versus potential optimisation of collision checks for future work.

The second reason for choosing sphere-trees was fortunately out of convenience, as methods for generating good tight bounding sphere-tree BHVs was readily available to us from previous research work conducted by Bradshaw [21, 22, 23] within the group.

Larger Polygons Than Bounding Spheres

It is possible that a sphere bounding volume be such that it is smaller than the polygon that it bounds. In such a case it may be that several small spheres bound this one polygon. In such a situation it may be more advantageous to just do collision detection using the polygons instead. This sort of situation is dependant on the bounding volume generation algorithm (in our case the sphere-tree generation algorithm). A good algorithm will ensure that we have as large an area covered by the bounding volume as possible in as tight a

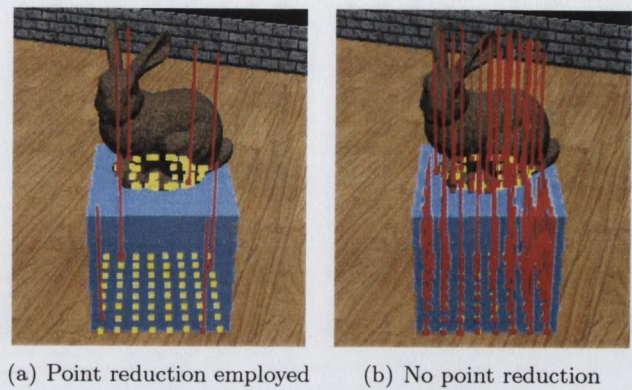


Figure 5.13: Scenario for point reduction performance tests. Red dots are the points of contact considered. Yellow dots are points of contact found but not considered. Lines represent the normals of contact.

fitting manner as possible to avoid cases of several small bounding volumes bounding the same polygon. If we wanted, we could as a pre-processing step determine if any spheres are smaller than the polygons they bound and if so mark them as such. Then during collision detection we can simply conduct a quick check to see if this is the case and if so stop refining down to the corresponding sphere level as it is not necessary, or else simply conduct a collision test on the polygons in question.

Some Results Of Conducted Tests

A scenario involving a box and a bunny model dropped one on top of the other onto a horizontal plane in such a manner that the underside faces of each object were perfectly horizontal to the plane was set up to test the effectiveness of our point reduction schema. This configuration was so chosen so that the faces would produce an optimum contact surface giving an overabundant spread of contacting points for a worse case scenario. Two simulations were run; one with point reduction and another without. The average response processing times over 1000 frames were then taken for these simulations. Figure 5.13 shows a snapshot of the two scenarios from the tests, along with the contacting points considered and not considered for each test.

For this setup, the resultant animation from both scenes were the same. For the first scenario (Figure 5.13(a)) we recorded an average response performance timing of 0.001122 seconds with a maximum of 8 contact points ever considered. For the second scenario (Figure 5.13(b)), this performance timing increased to an average of 2.170422 seconds with a

recorded maximum of 279 contacting points. The increased computational time is partially attributed to the make up of the matrix within the final equation to be solved for as well as the number of points considered. The large amount of contacting points from the two contacting objects produces quite a dense matrix (due to the co-dependence of each contacting point to each other), which in turn is also quite large (i.e. 279×279).

To test our overall time adaptive framework we set up an animation whereby 30 bunny models were dropped onto a ramp in such a way that they start to bunch up over time before the velocity of the incoming bunnies free up the bunching. Again, two tests were conducted; one with the time adaptive framework in place and another whereby timing constraints were not considered. We present the results of these tests in Figure 5.14. As can be seen from Figure 5.14(a), the frame performance timing starts to rise as the bunny models start to bunch up in the gap between the ramps with the non-time adaptive approach, whereby with our time adaptive framework in place the frame performance timings remain fairly consistent throughout the simulation.

5.6 Conclusions

In this chapter we have investigated some of the major issues to be considered in a fully time adaptive dynamics system. Although the idea of time-critical processing has been proposed in the past, its incorporation into a viable working system has not been extended. This chapter has presented a thorough investigation into the full incorporation of time-critical procedures into a viable time adaptive dynamics framework.

We have incorporated the work of Hubbard and Dingliana and O'Sullivan in time-critical collision detection into a novel time critical simulation scheduling schema. The rendering phase in the pipeline was not considered to be time critical as we preferred to concentrate on overall collision handling. Similarly we preferred to have as fast and accurate an ODE solver as possible rather than using an adaptive ODE solver solution. As a result, the overall view and timing considerations of the simulation pipeline within the schema was adjusted to account for this.

While sphere-tree bounding volume hierarchies are advantageous for fast collision detection and incorporation into time-critical schemas, there remain many issues with the approach such as the overabundance of contact points found during lower levels of collision detection. Apart from the work presented in this chapter, to our knowledge such an issue has never been considered in previous works. This is surprising, due to the fact that

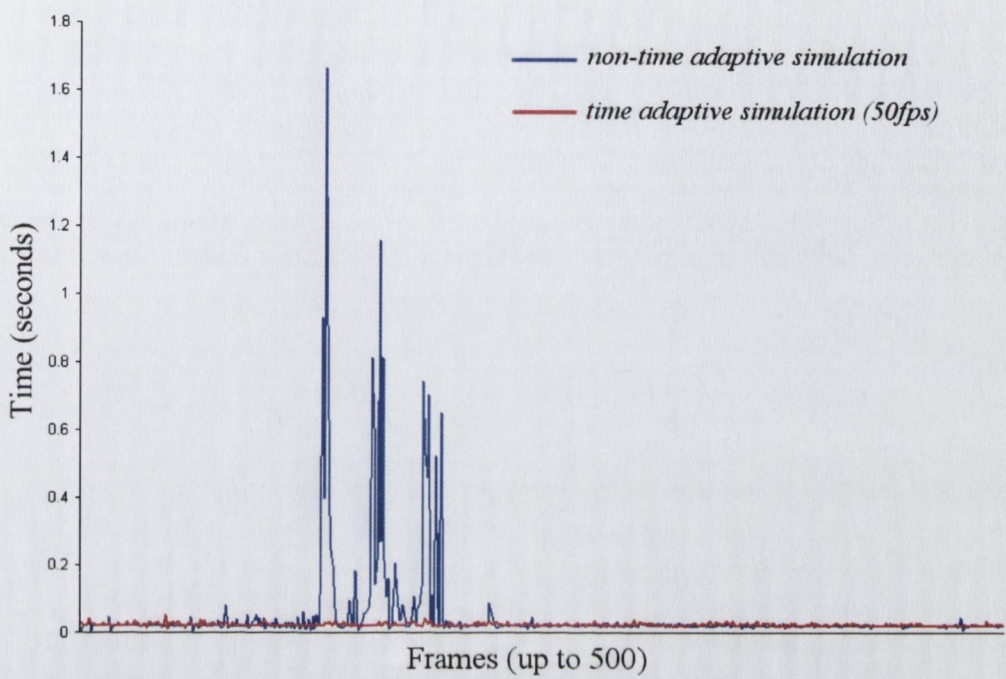
the number of sphere contacts is one of the major issues in hindering the overall performance time during collision resolution, which in itself must be accounted for and adapted as appropriate if temporal constraints are not to be broken.

The area of adaptability of collision resolution has unfortunately only been preliminarily considered by a small band of researchers in previous work in time-critical collision handling. Such an area is of great importance if a fully time adaptive system is to be made possible. Encouraged by previous work on this subject matter, an investigation into the adaptability of collision resolution processing and the proposal of novel solutions (by way of the point reduction schema and response timing table) was further considered in this chapter.

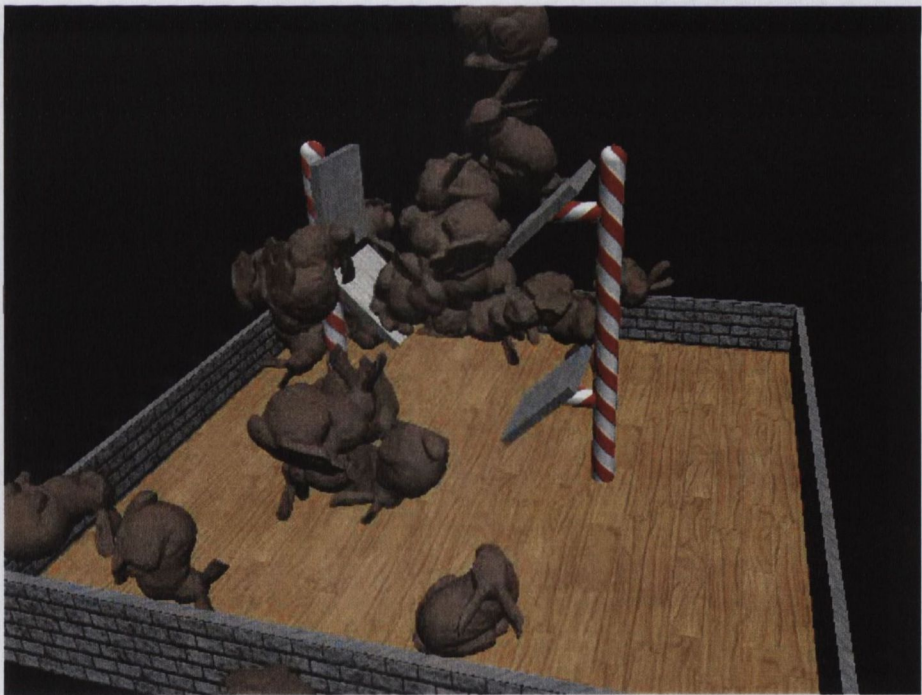
Problems with tunnelling and the issues that can arise within a time adaptive system when trying to resolve such issues were also discussed. It was shown that basic backtracking schemes are not viable approaches when progressive refinement of collision data is being used in a simulation. Such schemes can only serve to cause further problems, as over-approximations in contact queries are no guarantee of actual object collision, even if interpenetration has been detected. Instead, we proposed an elementary temporal lookahead scheme which is by no means perfect. The proposed scheme cannot really be considered a *Conservative Advancement* scheme but, however, does attempt to predict interpenetration before it happens so that such visual artifacts do not occur.

Throughout the discussions within this chapter, the achievement of physical plausibility over physical correctness has been the main criterion for many of the solutions proposed. As such, this allowed us to make many assumptions that would otherwise have been dismissed if physical accuracy was the main priority.

Although many topics on collision handling have been addressed within the chapter, we have only very briefly mentioned collision resolution issues. We have not discussed how we might approach approximating appropriate contacting data for collision resolution, given premature interruption during collision detection, or how we may do so in a refinable manner. Given the spherical nature of sphere-trees, certain problems can arise during contact data resolution, which have not been previously considered by researchers in time-critical collision handling. In the next chapter, we shall present the idea of *Closest Feature Maps* (CFMs), which aim to resolve the problems associated with current approaches to approximating refinable contact data during time-critical collision detection. The proposed CFM scheme aims to preserve the refinability of collision data during time-critical collision detection but also aims to provide better approximations of contact data that otherwise would affect the overall plausibility of responses for certain physical events.



(a) Frame performance timings for the system tests



(b) The test scene

Figure 5.14: A scene and timing results from the time adaptive framework performance test

Chapter 6

Closest Feature Maps

Up until now we have neglected to mention how one might go about obtaining useful information for collision resolution after we have detected a collision. The traditional approach in a collision detection scheme that utilises BVHs such as sphere-trees, is to have each leaf of the tree contain the information about which polygon(s) it bounds. Upon traversal down to the leaf level signifying a positive collision, more expensive and exact schemes such as polygon-polygon, face-edge or similar intersection tests are further conducted. From these tests, useful collision information such as point(s) of contact and normal(s) of contact are obtained for collision resolution. In such collision detection approaches, the BVHs are utilised as a detection acceleration technique only.

For time-critical collision detection, such an approach may not be possible due to the uncertainty that we may ever reach the leaf level of our BVH. As an adaptation to the possibility of interruption within time-critical schemas, it is possible to not only have the leaf nodes of a tree contain information about which polygons they bound but also nodes within all other levels of the tree may contain such information if so desired. This may seem to facilitate the idea of interruptible collision detection but however is quite expensive and not at all feasible as a working solution for real-time applications. Take for example the worse case scenario whereby we have no time to do collision detection and must take the top level sphere (assuming the BVH is a sphere-tree) for collision processing. In such a case, the information within the root sphere tells us that it bounds the entire body and hence exact intersection tests must be done for all polygons of the body. This is a very expensive process and defeats the entire purpose of interruptible collision detection, which aims to trade off accuracy for speed in the collision detection stage. For time-critical collision detection, a BVH must be treated not only as a detection acceleration technique, but also as inherently

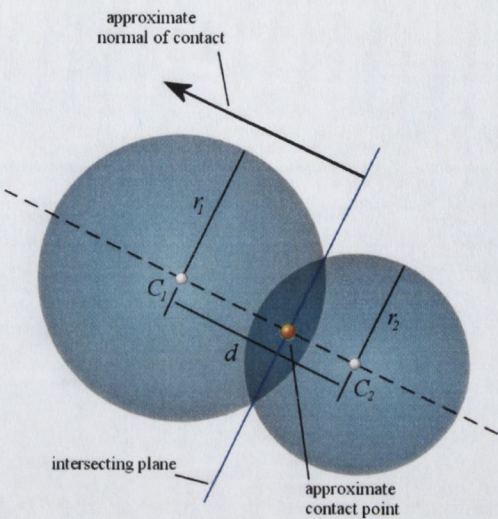


Figure 6.1: Approximating collision normal and point of contact between a colliding sphere pair during interruptible collision detection

part of the entire collision detection process, whereby a quick definitive answer must be obtainable through the use of the BVH.

6.1 Approximating Contact information

The problem of what to do during interruption in time-critical collision detection remains an open question. Hubbard, who first proposed the idea of time-critical collision detection, only used a very naïve scheme during collision response, choosing to simply reverse the direction of the incoming velocity during detection of a collision. Dingliana and O’Sullivan, on the other hand, have adopted the idea of time-critical collision detection with sphere-trees from Hubbard’s work and also considered what to do after collisions have been interrupted. They have proposed an approach that quickly determines approximate points of contact and normals of contact using proximities of the colliding spheres of the sphere-trees for calculation of desired contact information.

The solution proposed by Dingliana and O’Sullivan is an elementary yet elegant one that accounts for interruption within all levels of the sphere-tree without the need for excessive calculations. For each pair of colliding spheres detected, an approximate contacting normal, n , for that collision is calculated by a simple calculation of the vector that runs through the center of both colliding spheres:

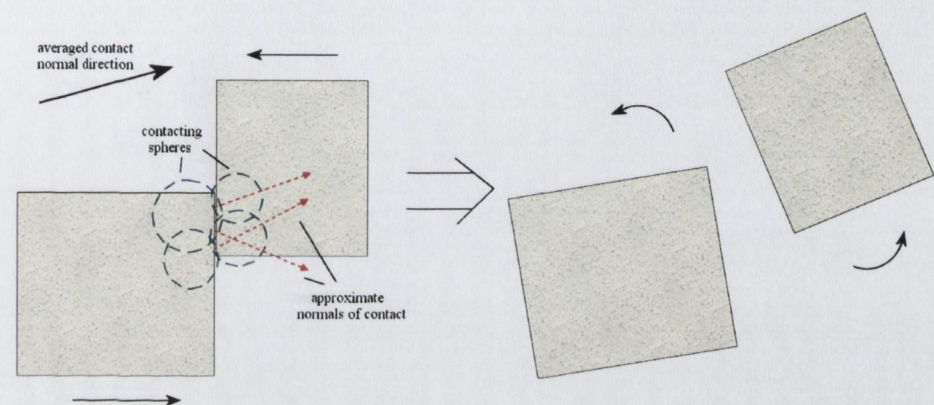
$$n = \frac{C_1 - C_2}{|C_1 - C_2|} \quad (6.1)$$

An approximate collision point for this sphere pair collision is simply obtained by considering the intersection of the line through the centres C of each sphere and the intersecting plane of both spheres. By projecting along the distance between the two centres d , taking into account the ratio of the difference in sphere sizes, the point of intersection and thus the point of contact is easily approximated:

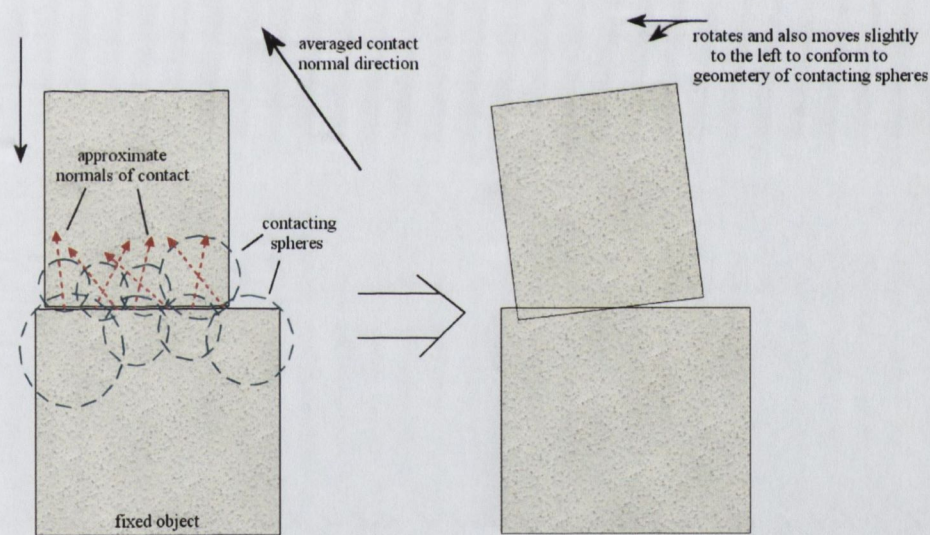
$$p_{\text{contact}} = C_2 + n * \left(d * \frac{r_2}{r_1 + r_2} \right) \quad (6.2)$$

In the above equation, $r_{1,2}$ simply denotes the radius of the spheres and n , the unit normal, which is obtained from utilising Equation 6.1. The corresponding sphere centre that we choose to project out from is determined by how we calculated the normal. Since Equation 6.1 gives us a normal pointing away from the centre of sphere 2 (i.e. C_2), we likewise use this normal, n , to project away from C_2 towards the centre of the first sphere C_1 in Equation 6.2. The ratio of the difference in sizes of each contacting sphere (i.e. judged through the spheres' radii) must be accounted for as this affects the distance to the point of intersection of the intersecting plane. For two spheres of the same size and hence the same radius we are only required to travel half the distance from the corresponding sphere's centre to obtain our desired point of contact (i.e. bringing us to the mid point of the line running through the centres of both spheres). However as sphere sizes become dissimilar it is no longer a case of the intersection plane intersecting at the mid point of the line between the sphere centres. As one of the sphere's get progressively smaller, the distance that we need to travel from its centre to get to the point of intersection of the intersecting plane and hence our approximate point of contact becomes less, and vice versa for when the sphere gets progressively larger. We can determine the fraction of the distance to travel from the line between the two sphere centre's if we account for the ratio of the corresponding sphere radius to the combination of both sphere's radii (i.e. the last part of equation 6.2), in affect accounting for sphere size differences. Figure 6.1 illustrates this process of approximating collision point and contact normal.

While this solution is an elegant one and works quite well within situations of refinable collision handling, at times the produced information may be too approximate. For objects in constant motion, the effects of overly approximate contact information may not be noticeable due to the expected chaotic nature of dynamics. However, for situations where



(a) Consequences of normal approximation does not effect overall plausibility of object response



(b) Inappropriate approximation of contacting normals causing unwanted rotation and shifting during response

Figure 6.2: Cases where we can and can't get away with over approximation of contact data

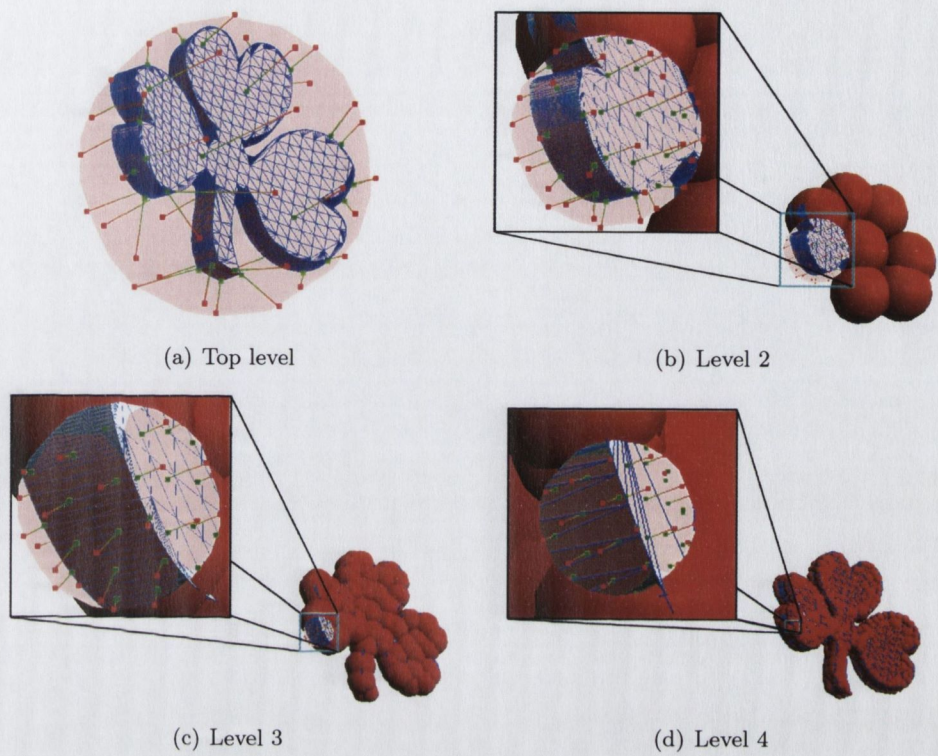


Figure 6.3: Closest Feature Map of four sphere-tree levels for Shamrock model

contact information is required for more than a small period of time, such as cases of continuous contact, over-approximation of contact information may cause unwanted anomalous dynamic behaviour. Consider two cases of a pair of rectangular objects, in 2D, colliding with each other as illustrated in Figure 6.2. In the first case, where both objects are freely floating through space, during a collision, over-approximations of contact normals do not adversely effect the plausibility of the simulation. Now, consider the second case where one object, which is in motion, falls directly on top of the other object, which is stationary and fixed. Assuming that the second object is falling under gravity, the scenario will eventually bring the two bodies to a state of rest causing them to be in a final stacked position. In such a case, it is necessary to have as close a contact normal approximation as possible to the underlying polygonal representation of the model. Using the previously described method to approximate contacting normals may result in unwanted rotations and shifting during collisions, due to bad normal approximations, thus detracting from the overall plausibility of the simulation.

6.1.1 Improved Approximations Using Closest Feature Maps

To address the problems mentioned above and to provide better contact approximations, we propose the novel concept of *Closest Feature Maps* (CFMs) for refinable collision handling. Essentially, a CFM is a mapping of contact information based on the closest features from a set of sample points around a sphere that bounds the model or part of the model. It not only has the advantage that, the further down the sphere-tree we manage to traverse, the more refined the contact information will be, but also, all contact information can essentially be obtained from the polygon level rather than through rougher approximations made on the two contacting spheres, no matter where in the sphere-tree we are approximating from. We illustrate a CFM for four sphere-tree levels in Figure 6.3. The CFM approach presented here sacrifices memory consumption for more accuracy during collision handling.

The CFM is pre-calculated and loaded in with the sphere-tree information during system startup. A set of sample points is defined around each sphere within the sphere-tree. During the pre-calculation stage it is possible for us to determine for each sample point on the corresponding sphere :

1. the closest feature on the model to each sample point
2. the closest point on the surface of the model to the sample point in question on the bounding sphere
3. the outward pointing normal from the closest feature to the sample point

If a sphere within the sphere-tree is involved in a collision we simply approximate the point of contact using Equation 6.2 in the same manner as that proposed by Dingliana and O'Sullivan. We then determine the closest sampling point on the sphere from this approximate contact point and from that sample point we may obtain the desired closest feature information for collision resolution. The closest feature information can be anything from the type of closest feature (i.e. edge, vertex or face) from the sampling point, to the outward pointing normal of the closest feature. If so desired we can further approximate the contact point by taking the point found to be the closest point on the surface of the model to the sampling point in question. Of course, it is not necessary to store all the suggested closest feature information within each sphere of the sphere-tree, only those that we deem necessary. In practice, to reduce the memory requirements we only retain the normals of the closest features to each sample point on the sphere¹. We use these normals

¹Perhaps a more appropriate name for the Closest Feature Map if utilised in this manner would then be a *Normal Map*

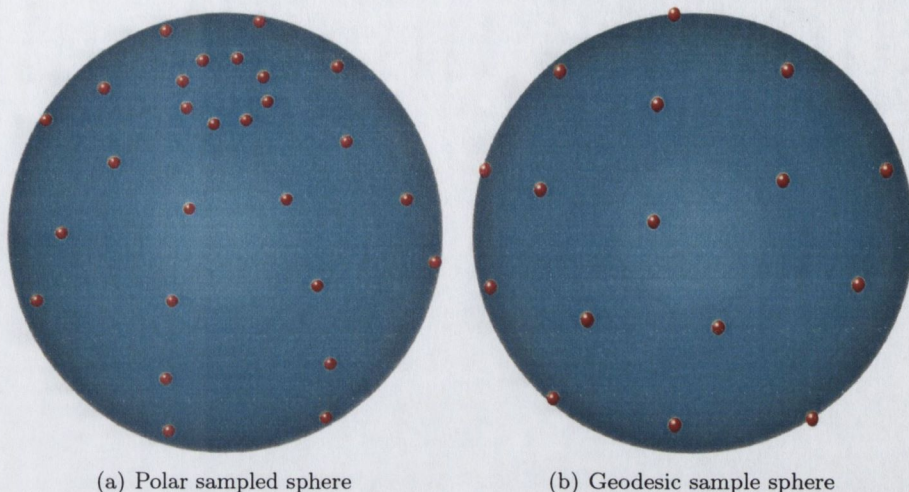


Figure 6.4: Difference in point distribution for polar and geodesic sampling

as the contact normals during collision detection rather than approximating normals in the manner as described by Equation 6.1. Such normals are closer approximations to the real normals needed for contact resolution.

6.2 CFM Construction

Before we can utilise the Closest Feature Maps scheme within the collision handling phase in our simulations, we must first construct them before simulation starts. Construction of CFMs involves several steps. The first is to determine how we wish to distribute the sampling points for each sphere within the sphere-tree. This is important as the distribution and density of sampling points around each sphere strongly affects our approximations of closest feature information during collision handling.

For sample point construction we essentially have two choices. We may choose to use a *polar* sampling or *geodesic* sampling scheme. The difference between these two distribution methods is illustrated in Figure 6.4. Sample points constructed using polar sampling are not evenly spaced but have the advantage that during collision handling, closest sample point searching can be done much more efficiently and intuitively. On the other hand, geodesic sampling ensures even and fair distribution of all sampling points on the sphere but nearest sample point searching during collision handling cannot be handled as efficiently.

6.2.1 Polar Sampling

For both polar and geodesic sample point construction, the algorithms can be greatly simplified if we consider a unit sphere during processing. Later we may adjust the scale of the sphere and hence the sample points distribution, with respect to the specific sphere in the sphere-tree that we are considering.

Polar sampling assigns sample points on the sphere's surface using a projection from the sphere's centre in the direction defined by the angles ϕ and θ . The corresponding cartesian coordinates are trivially calculated using the spherical polar equations:

$$x = \sin(\phi) * \cos(\theta) \quad (6.3)$$

$$y = \sin(\phi) * \sin(\theta) \quad (6.4)$$

$$z = \cos(\phi) \quad (6.5)$$

Unfortunately, while this is an elementary method to use for generating spherical sample points, the points that are thus generated are not evenly distributed across the surface of the sphere. As can be seen from Figure 6.4(a), the closer to the poles of the sphere we get, the more clustered the sample points are to each other. This results in very poor sampling of closest features as sample points in close proximity to each other will usually determine the same closest feature information. This is wasteful in terms of memory usage when storing such information.

To alleviate unnecessary closest feature information duplication during bad point clustering, it may be advantageous to use an adaptive value of θ during sample point creation depending on proximity to the spherical poles. Since we are assuming a unit sphere during sample point creation, this is easily done if we sample θ with an interval proportional to $\sin(\phi)$. This is exactly what we do during sample point creation.

6.2.2 Alternative Geodesic Sampling Approach

An alternative, and perhaps better, solution is to utilise the vertices of a platonic solid as sample points on the sphere. If we were to fold out any platonic solid, we would see that it consists of evenly distributed vertices that belong to a series of equivalent faces, which are essentially convex regular polygons. Each of these solids can be further refined by subdivision to give us more and closer spherical sample points if required.

From the five known platonic solids, perhaps the most relevant and beneficial solid for

constructing sample points around a sphere is the icosahedron. This is because, perhaps apart from the dodecahedron, the icosahedron gives the best initial fit to a sphere. For our purposes, it is not necessary to have too fine a spherical approximation with the icosahedron, as the finer the approximation gets, the closer the sample points on the sphere surface will be to each other. This fine clustering of points (although they may be equidistant from each of their neighbours) can lead to a lot of duplication of closest feature information, which is wasteful of memory. However, too coarse an approximation may also not be desirable, as we wish to catch as much polygonal closest feature information as possible without duplication.

6.3 Determining Closest Feature Information from Sample Points

We have proposed two choices for determining distribution of sample points around a sphere but have not yet mentioned how one might proceed to obtain closest feature information from the underlying bounded polygonal model. We shall discuss such a method here.

A naïve approach to determining the closest feature to a point from an underlying polygonal mesh is to step through every polygon of the mesh and test for closest distance of the point to the polygon surface. However, this is an extremely expensive process and can be highly optimised. A volumisation approach may be adopted to cull out unnecessary space occupied by the mesh that may be deemed too far away to the point being tested. This is in fact the approach that Hubbard [76] and Bradshaw [21] adopt for speeding up closest point inside-outside testing of medial axis vertices during sphere-tree construction and also for their sphere occupancy error tests. We adopt the same closest point algorithm as used by Hubbard and Bradshaw within their sphere-tree generation approaches to aid us in obtaining the closest feature information for each sample point on the spheres of our sphere-tree. Unlike Hubbard and Bradshaw, we are not concerned about whether a point lies inside or outside the polygonal mesh. We already know that our point resides on a sphere that bounds the polygonal mesh in some way and we wish to obtain the closest polygonal surface feature to this point regardless. Here we shall present a brief overview of the closest point algorithm along with a further search speed-up approach as suggested by Bradshaw.

The closest point algorithm iterates through each polygonal face of a triangulated polyhedron, maintaining the closest distance from the point p on the sphere's surface to a perpendicularly projected point p' that resides on the surface of the polygonal mesh. In

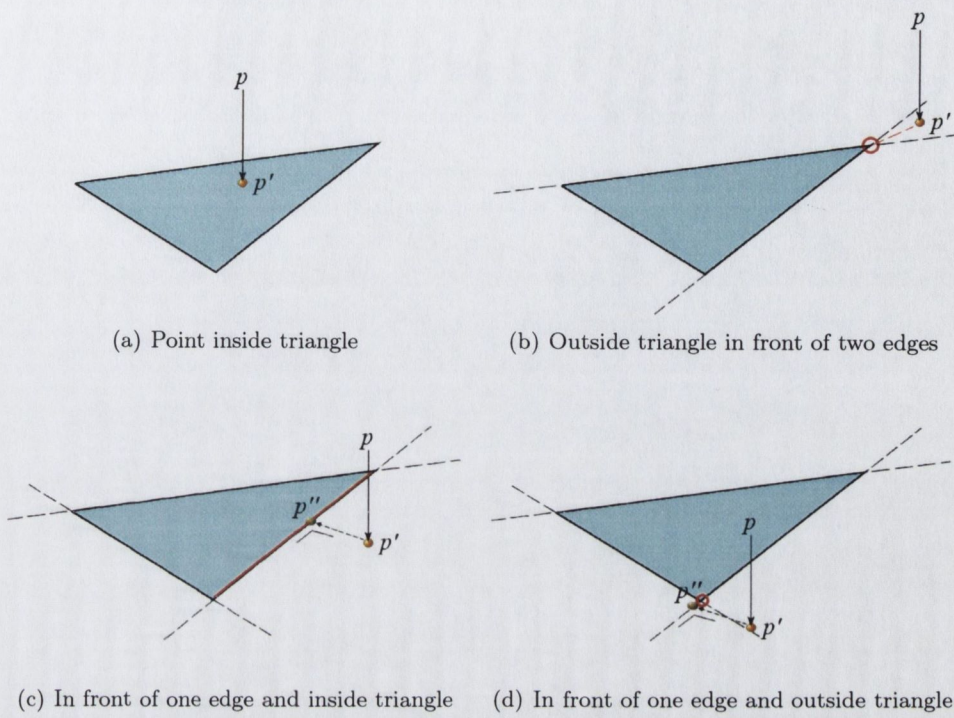


Figure 6.5: Different cases that can be encountered during closest point searching

checking for the closest point to a triangle on the mesh surface, three individual cases are considered:

Face

- p' lies behind every edge of the triangle (Figure 6.5(a)).

Vertex

- p' lies outside of the triangle but is in front of two edges (Figure 6.5(b)).
- p' lies in front of one edge and the perpendicular projection of p' (i.e. p'') lies outside of the triangle (Figure 6.5(d)).

Edge

- p' lies in front of one edge and the perpendicular projection of p' (i.e. p'') lies inside of the triangle (i.e. behind two edges, Figure 6.5(c)).

In all cases, once the closest feature has been determined, the calculation of the corresponding normal to the discovered closest feature is a trivial matter.

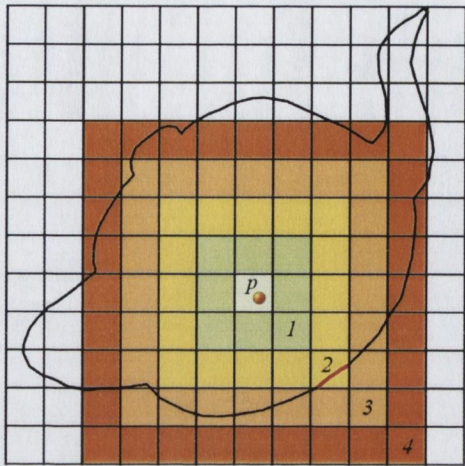


Figure 6.6: 2D example of voxel traversal for speed up of closest point testing on a polygonal mesh

In order to speed up the search for the closest polygon to our sample point, we adopt the use of a SEADS (Spatially Enumerated Auxiliary Data Structure) grid. For each voxel in the SEADS grid a list of intersected polygons is maintained. A quick distance check may be conducted from the sample point to the voxel before doing a more expensive closest point test to quickly eliminate those voxels and hence those polygons that are deemed too far away. Hubbard maintains a sorted list of voxels in accordance with their distance from the point p in order to further speed up processing. Bradshaw, however, discovered that this sorting overhead accounted for a large amount of work when conducting inside-outside tests and hence suggested an alternative approach. Instead of maintaining a sorted list of voxels based on distance from the point p , he suggested that we traverse the voxels by first starting with the voxel that contained point p and work our way outwards in ever-increasing concentric rings around p . We adopt this suggestion when traversing the voxels in the SEADS grid to further speed up closest polygon searching during our closest feature searches.

Figure 6.6 illustrates a 2-dimensional example of Bradshaw’s suggested voxel traversal algorithm. To transform this process into three-dimensions, the three dimensional equivalent of the concentric rings would be a set of voxels that make up concentric spheres around the point p and the underlying algorithm remains the same.

Upon determination of the closest feature for each sample point on the spheres in our sphere-tree, the type of information we wish to retain is dependent on how we wish to use it later on. It is possible to retain the closest feature (i.e. face, edge or vertex) to the sample

point, the nearest point on the mesh surface and the normal from the closest feature all at once for each sample point. However, we must bear in mind that the more information we retain, the more memory we will utilise when loading our Closest Feature Map into our dynamics system. It is unlikely that we need all that information, so a better strategy is to only save that which is of interest to us, e.g., the normals to the sample point so that we may use them to approximate contact normals later on. (Of course, we can retain all the previous information without loading all of it into the system. However, our CFM file size may be considerably larger than we wish in that case.)

6.4 Efficient Nearest Sample Point Determination

During simulation, it is not enough to determine whether a collision has occurred or not. We must also be able to determine useful information that will allow us to resolve the collision in a plausible manner. The information from the Closest Feature Map from each sphere in the sphere-tree is hence used for this purpose.

The next main issue is how to determine the nearest sample point (so as in order to determine the closest feature) to the estimated contact point in a quick and efficient manner without compromising the overall processing time allotted. Obviously, an exhaustive nearest neighbour search is out of the question as this is far too computationally expensive, especially if there is a large number of sample points to consider. If such an approach were to be utilised, the CFM scheme would become as useful as doing an underlying all-pairs polygon intersection test for the entire polygonal mesh in question. Therefore, we need to have quick and efficient search schemes in place for determining the nearest sample points to the estimated collision point. Fortunately, the polar sampling schema provides us with an elementary and constant time retrieval solution, providing that we set up the data structure as appropriate.

6.4.1 Polar Coordinate Searching

A simple and efficient search algorithm based on the polar coordinates of the contact point can be used to quickly determine the closest sample point and hence the relevant CFM information. For the purposes of subsequent retrieval, during Closest Feature Map creation we retain the sample points (and hence the CFM information) in a sorted order to be thus loaded into the dynamics system. The order of sorting is twofold. For each sphere within the sphere-tree, we first sort in ascending order of ϕ and for each block in the sorted

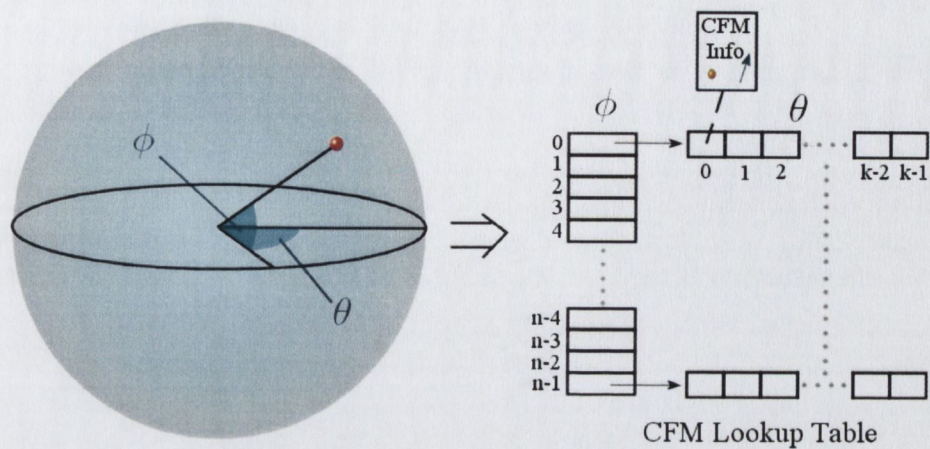


Figure 6.7: Determining closest feature information during collisions from polar sampling

ϕ list we associate a sorted list of θ which corresponds to the *ring* of sample points at that ϕ angle. We illustrate this data structure in Figure 6.7. The combination of the ϕ and θ blocks represent the corresponding nearest sample point on the sphere created for those coordinates. The appropriate closest feature information can be quickly retrieved for a corresponding sample point on the sphere if we know the index position of each point within the sorted list. Since this is essentially a lookup process, the retrieval strategy is conducted in a constant time manner, regardless of number of sample points per sphere.

It should be noted here that the sorted list is treated as residing within the sphere’s body space coordinates as the sample points were created as such. The CFM information that is contained in the list and the spheres in the sphere-tree, however, is viewed as residing in the body space coordinates of the model they relate to. If we did not take this view, it would become a very computationally expensive process to constantly have to update the list and the sphere-tree to correspond to the current world space orientation and position of the body that is bounded as it moves through space.

To correspond to the conventions of the data structure of the ϕ and θ list, we must ensure that we first translate the approximate contact point p into the corresponding sphere body space coordinates before beginning closest feature data retrieval. To account for non-constant polar sampling intervals of θ for each ring at angle ϕ , the equations for data retrieval can be expressed as:

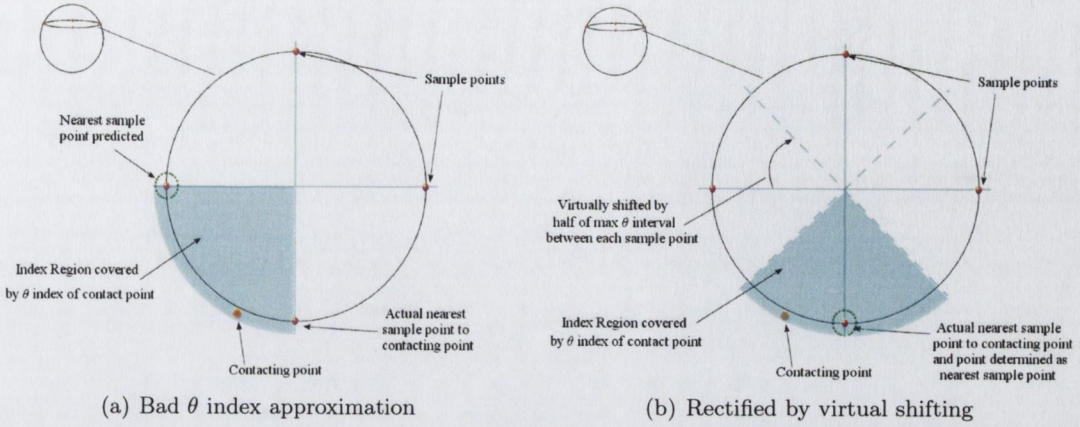


Figure 6.8: Bad θ indexing strategy during polar searching and rectified situation

$$Index_{\phi} = \frac{\phi * (ListSize_{\phi})}{180} \quad (6.6)$$

$$Index_{\theta} = \frac{\theta * (ListSize_{\theta})}{360} \quad (6.7)$$

whereby $ListSize_{\phi/\theta}$ are the respective sizes of the ϕ and θ lists.

The above approach is very straightforward, although there is a slight problem with the indexing strategy for θ as described above. We desire that we retrieve the closest sample point to the approximated contacting point during indexing, however Equation 6.7 will sometimes retrieve the neighbour of the closest sample point instead. This error arises because Equation 6.7 effectively dictates that, for any point falling within the region swept in an anti-clockwise direction from a specific sample point to the next sample point, that sample point is to be considered the closest sample point to any point falling within the region. However, the correct view that should be taken is that only the regions specified by a half way sweep in both a clockwise and anti-clockwise direction to each of the sample point's neighbours on the ring should be considered valid nearest point regions for that sample point. This problem can be rectified by simply shifting the contact point by half the θ interval between each sample point in an anti-clockwise direction before using Equation 6.7 to obtain the appropriate index of the closest sample point. In this way, what we are effectively doing by shifting the contact point, is virtually shifting every sample point's closest point region back by half a sample interval so that the contact point now resides within the correct region bounds during index determination. Figure 6.8(b) illustrates the correct new region

as a result of this change. As can be seen, the contact point now resides in the correct closest point region of the correct nearest sample point.

Fortunately, the ϕ index search does not suffer from this problem. This is because we only consider a half ring during the ϕ traversal of sampling and also we do not attempt to create rings at the poles of the sphere during sample point creation. As a result, Equation 6.6 naturally assumes the correct nearest point regions (to the contacting point) for each ring, even to the extent of expanding the nearest point regions of the appropriate sample points to account for the *null* regions near the poles of the sphere.

6.4.2 Geodesic Coordinate Searching Alternative

Although a geodesic sampling scheme gives an even distribution of sample points on the spherical surface, there is unfortunately no obvious solution that allows us to relate the sample points so that an efficient indexing approach can be utilised during closest point determination. A nearest neighbour search algorithm that exploits attributes of the sample point distribution in other ways than the previously described polar search scheme must be considered in this case.

If one wished to adopt a geodesic sampling approach or similar distribution scheme, it may be possible to utilise some sort of divide and conquer strategy in order to further speed up searching or perhaps even a voxel based approach such as that described in Section 6.3. Unfortunately, all retrieval strategies essentially boil down to a nearest neighbour search strategy rather than an indexing strategy such as the previously described polar sampling schema. So, although geodesic sampling guarantees a fair distribution of the sample points around the sphere, we lack the constant time retrieval advantage of the closest feature data that the polar sampling scheme provides us with. Perhaps the best we can hope for is a near constant time retrieval of closest feature data based on the chosen search strategy.

6.5 Discussion and Results

During simulation, it is important that we retrieve contact data as quickly as possible once a collision has been detected. For a time-critical dynamics system, the approximation of the required contacting data for collision resolution should not further hinder the imposed timing constraints of the entire collision handling process.

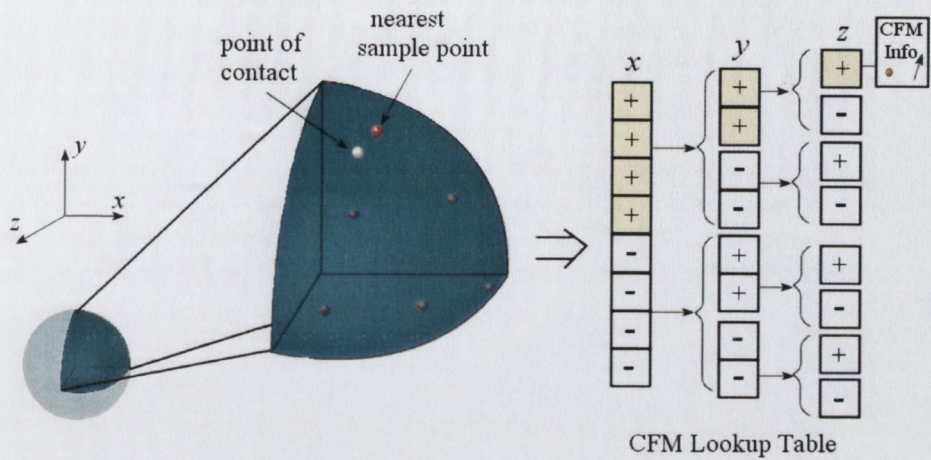


Figure 6.9: Alternatively adopted geodesic search scheme

Retrieval Timings

Timing tests were conducted for both the polar and geodesic sampling schemes for comparison. For the geodesic sampling scheme we adopted an elementary divide and conquer approach to closest feature data retrieval. The quadrant of the hemisphere in which the nearest neighbouring sample point resided was determined and then an exhaustive nearest neighbour search of the remaining points was conducted in order to obtain the appropriate closest feature data (Figure 6.9). This scheme proved to be four times faster than a linear search scheme. The results of the test are illustrated in Figure 6.10 and the scene which was used for the test are illustrated in Figure 6.11. For comparison, the results of a naïve linear search scheme are also given. The timing tests were all run on a 1Ghz AMD Athlon with 500megs of RAM running a GForce 3 graphics card.

For the polar sampling approach, a ϕ interval of ten was specified during sample point creation, giving us a distribution of nine rings of sample points around the sphere. An adaptive polar point creation algorithm was used to limit the clustering of points during sphere sample point creation. This strategy created a distribution of sixty five sample points per sphere in total. For the geodesic sampling approach, we allowed subdivision of the icosahedron approximation to two levels during sample point creation. This in turn gave us forty two equidistant sample points around a sphere. The linear search scheme was performed on a distribution of forty two sample points per sphere.

A constant time retrieval of contact data is obviously most favourable. As such, the polar sampling scheme is preferred due to its elementary but yet guaranteed constant time

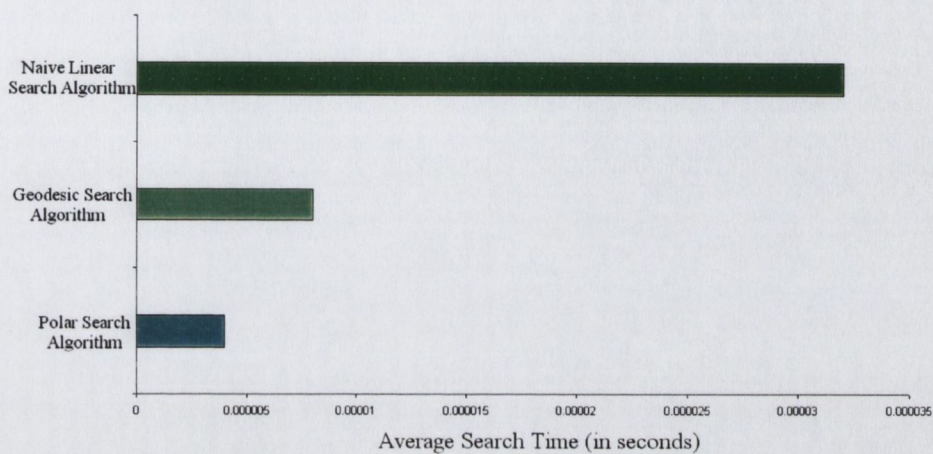


Figure 6.10: Average timings per frame per sphere for search algorithms

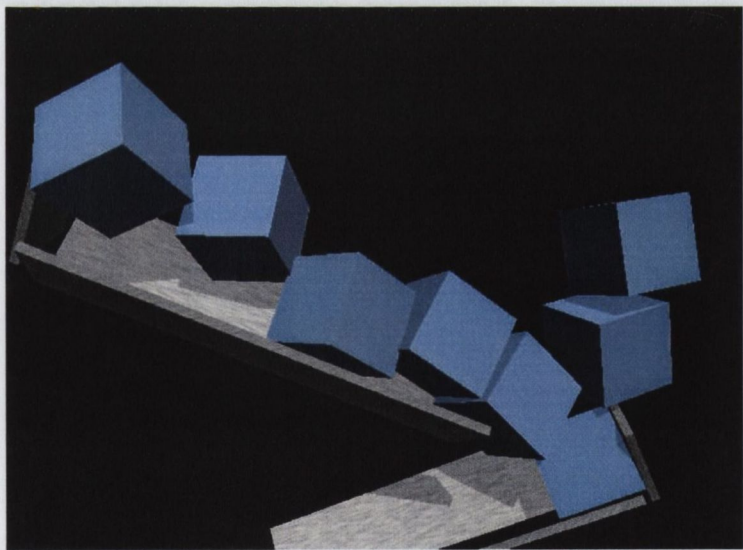


Figure 6.11: Snapshot of the simulation scene used for the search timing tests

No. of Stacks	Full Polar	Adaptive Polar
5	0.371169	0.17718
10	0.144332	0.119856
20	0.0717484	0.0661784

Table 6.1: Standard deviation of distances between neighbouring sample points in a polar sampling scheme

retrieval solution. However, if the goedesic sampling scheme is used or any other similar point distribution scheme for bounding volumes other than spheres, then a voxel based search scheme such as the once described in Section 6.3 is recommended. Other temporal coherence schemes may be similarly adopted to further speed up closest feature retrieval.

Point Distribution

In order to better gauge the accuracy of the polar sampling scheme, the standard deviation of distances between neighbouring sample points was taken. The ideal standard deviation in which we would prefer is obviously zero as this suggests equidistant points (as is the case for the geodesic sampling scheme), which in turn reduces the possibility of unnecessary duplicate closest feature information. Standard deviation of distances were taken and compared for both the full and adaptive polar sampling scheme for points created with 5, 10 and 20 stacks. The results of this are shown in Table 6.1. As the results show, adaptive polar sampling greatly improves the *quality* of point distribution as the number of contacting points are reduced. As point distribution increases, neighbouring points start to become fairly close to equidistant from each other, so much so that the adative scheme only provides minor improvements. However, it must be noted that as the number of sampling points increases so too does the possibility of producing too many unnecessary duplicate closest feature data. The ideal situation is to have as much unique closest feature information as possible utilising the least number of sample points per sphere.

Other Considerations

Is very possible that, depending on the adopted sphere generation algorithm, at lower levels of the sphere-tree some bounding spheres may only bound a single polygon. Such a case should be checked for during the closest feature determination and sample point creation process. It may be that, in such a case, the sphere simply has the same closest feature information for all sample points distributed over the sphere. In such an instance it is not

necessary to store or even maintain excessive sample points for the sphere in question but merely the associated bounded closest feature. Thus, when such a sphere is involved in a collision, the appropriate closest feature information is trivially obtained. Such a check during the pre-processing stage invariably leads to further speedup of closest feature data retrieval for the geodesic scheme.

The question arises why we didn't simply use the an averaging of the normals of the closest features bounded by the corresponding sphere or even just use information from a single mean face. While this in itself may seem like a sensible idea and does potentially give us a very quick data retrieval strategy, there are cases whereby this averaging scheme once again may provide us with undesirable over-approximation of contact data for collision resolution. Such an instance is best illustrated if one takes the case of a sphere bounding the corner of a box object. In such a case the sphere bounds at least two faces of the box that are perpendicular to each other. If we take the case of Figure 6.2(b) where we have one box falling directly on top of each other, the averaging of normal data in such an instance will produce normal information that causes unwanted rotation during initial contact which will detract from the overall plausibility of the resultant animation.

The question has also arisen on why we did not choose to use barycentric interpolation of the closest features of the triangle of the three closest sample points rather than just using the geometric feature information at the closest single sample point. It has been argued that utilising barycentric interpolation in this manner would avoid discontinuities in collision response during some cases in continuous contact. This assertion only holds true if we take contact point information at the polygonal level through the closest feature map during continuous contact situations as contact points will then tend to jump from one closest feature to another due to the nature of the algorithm. However, since we advocate using the colliding spheres to quickly approximate contact points as described by Equation 6.2, this is a sufficient approximation of required contacting points and does not exhibit discontinuities during cases of continuous contact. The determination and use of closest feature contact normal information does not suffer any such problems. If we did use barycentric interpolation of closest feature normals, it may be that the information given to us may be overly smooth (due to the attributes of the method) which can be undesirable with certain models, especially ones that exhibit numerous sharp edges. This barycentric approach may also suffer from ill conditioned information for situations such as that of the previously described example. Furthermore, our method has the advantage in that it provides a faster retrieval strategy since less calculations are needed to obtain the required

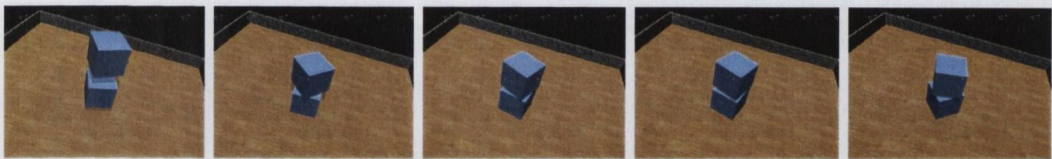
contact data (i.e. only required to determine one instead of three nearest sample points in order to determine our data). Also, if needed, it is possible with our approach to sample at a higher polygonal level of detail treating it as such during collision resolution while displaying the model at a lower polygonal level of detail during simulation.

We may further optimise our CFM schema by enhancing how we deal with the point distribution around the bounding spheres of the sphere-tree if we adopt a progressive refinement strategy for the point distribution scheme as well. In other words, the further down the tree hierarchy that we progress, the more sparse the point sampling on each sphere need be, since the possibility of catching duplicate closest feature information to associated sample points on each sphere increases as sphere size decreases. The sampling rate at each level of the hierarchy really is dependant on how geometrically detailed the model that we are bounding is. For models that are geometrically quite low in detail (i.e. have larger and less polygons) we may not need as many sample points at the higher nodes of the hierarchy for say a model with a higher level of detail (i.e. exhibiting smaller and more polygons).

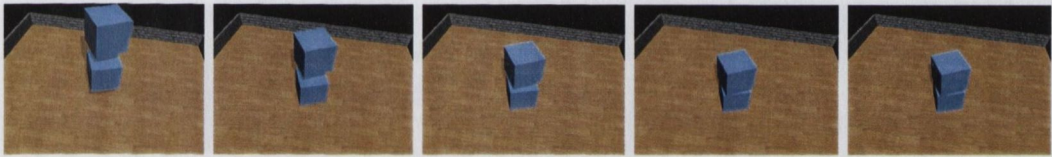
Continuous Contact Model

For our simulations, an impulse based approach similar to that of Mirtich [108] is used for cases of continuous contact. This impulse based approach was decided upon mainly for its simplicity and its advantage in handling both cases of resting contact and collisions under the one umbrella. A constraint force approach such as the one described by Baraff [7] could have quite easily have been used instead but we decided against this method as it was a little more involved than Mirtich's method. Regardless of the method used for resting contact it is still important to have correct contact data (i.e. normal of contact and point of contact). Incorrect or too much of an over-approximation of the contact data will cause the object to behave implausibly (e.g. slowly rotate into each other or objects hang too far over the edge of each other before eventually toppling).

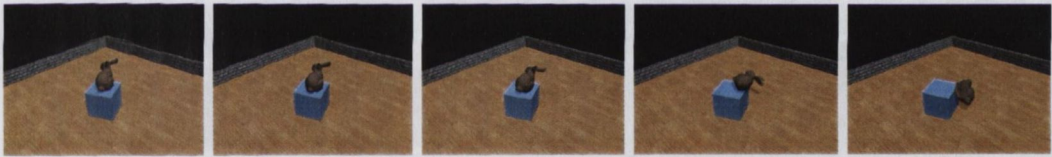
To illustrate the merits of the Closest Feature Maps approach, we present frames from two animations in Figure 6.12. It can be seen that the approach that utilises the approximation scheme proposed by Dingliana and O'Sullivan causes unwanted rotations and anomalous physical behaviour for such scenarios. The Closest Feature Map scheme, however, rectified this situation by allowing the top object to correctly rest on top of the box without causing unwanted rotational behaviour.



(a) Non-CFM approach - Box landing on top of another box



(b) CFM approach - Box landing on top of another box



(c) Non-CFM approach - Bunny landing on top of another box



(d) CFM approach - Bunny landing on top of another box

Figure 6.12: Two example scenes illustrating Closest Feature Map approach rectifying anomalous contact response

6.6 Conclusions

The novel idea of *Closest Feature Maps* to better approximate contact data in a refinable manner during interruption of collision detection was presented in this chapter. The major issues and stages involved in constructing and using CFMs for more accurate refinable contact approximation were presented and two alternative sampling strategies were discussed. CFMs were proposed to resolve the over approximation issues arising from Dingliana and O'Sullivan's approximation scheme for obtaining appropriate contact data for time-critical dynamics. Although we sacrifice some memory in using the proposed CFM approach, we believe that in order to ensure that physical plausibility is not hindered from over approximations of contact data, in certain circumstances this is a fair trade.

Although methods for determining various potentially important contact data were presented, it is however not necessary to utilise or store all the determined information. It suffices to use the approach suggested by Dingliana and O'Sullivan to approximate contact point data. However, if more accurate surface point data or feature information is needed, CFMs can be utilised to quickly determine such data. In practice, in order to reduce the memory footprint of CFMs, it is only necessary to store closest surface normal information for collision resolution as poor normal approximation invariably produces implausible collision response.

For sphere-tree BVHs, we advocate the use of the polar sampling scheme over the alternatively proposed geodesic sampling scheme. Our results have shown that the standard deviation of the distances between neighbouring sample points in the adaptive polar scheme is relatively low, suggesting that the sample point distribution is comparable to a scheme that produces a distribution of equidistant points.

It could be argued that perhaps it is a worthy sacrifice to have more sample points around a sphere to better approximate closest point data using the polar sampling scheme because the retrieval of such data is always guaranteed to be very fast regardless. Whereas, even though the geodesic sampling approach guarantees a fair distribution of sample points and hence needs less sample points to better approximate closest feature data, its timing performance in searching for such data may offset this advantage.

It should be noted that CFM approximations are only as accurate as the sphere sample point distribution will allow. Of course, too sparse point sampling may produce too approximate contact data. However, it is also possible to have an overabundance of unnecessary point samples, whereby closest feature determination becomes overly pedantic, to such a degree that memory is needlessly wasted.

Chapter 7

The Perception of Physical Plausibility

For the achievement of desired real-time frame rates within an interactive dynamics system, it is unfortunately sometimes a necessity to sacrifice certain details within the simulation so that this goal may be maintained. The obvious choice is to degrade geometric detail based on certain perceptual heuristics. There are many techniques available [102], whereby the criteria for the level of detail settings is based on heuristics or user defined limits. However, the work in this thesis concerns itself with the choice of physical detail degradation in order to conform to desired real-time performance constraints.

In doing so, the problem of simulation shifts from that of trying to achieve accurate physical behaviour to that of trying to maintain physically plausible behaviour. This in itself is by no means a trivial task, due mainly to the subjective nature of the subject matter. Studies have shown in the past that people in general have strong preconceptions of what to expect during a dynamic event [37]. As such, within a dynamic event, what may seem plausible for one person may be construed as exhibiting anomalous behaviour by another person. This makes the task of trying to quantify plausibility for physical events quite difficult.

It is therefore not overly surprising to discover that, to date, there has been no generic means of evaluating or predicting the fidelity of dynamic events occurring in an animation. Given the importance of having a good perceptual basis for a globally acceptable measurement of plausibility in dynamic simulations, we discuss in this chapter a proposed perceptual metric for evaluating the visual fidelity of physically based animations from our recent collaborative work with O'Sullivan et al [121]. The first steps towards applying this

metric to the case of a physically based animation system that is influenced by dynamic constraints is also presented here. We also describe some preliminary experiments exploring the conceptual plausibility of time-critical animations and also a preliminary study of a person's preconceptions of physics in a simulated physical event.

7.1 Previous Work

We first present, previous work by graphics researchers who have exploited the concept of physical plausibility to produce viable physical animations. We also examine some of the studies conducted in the perceptual domain pertaining to physical distortions.

7.1.1 Physical Plausibility Simulations

Barzel et al [14] first proposed striving for simulation plausibility over accurate simulation. They provided several interesting thought experiments to further reinforce the notion of plausible motion over realistic motion. Preliminary tests on a bouncing ball and the motion of pool balls were conducted to verify the *conjectures* presented within the thought experiments. Barzel et al concluded that simulations need not be *"predictive modelers of the physical world"* but instead *"have the quite distinct task of being plausible mimickers of the physical world, while at the same time providing flexibility and controllability for animators"*

Following on from the work of Barzel et al, Chenney and Forsyth [35] developed a system that exploited user uncertainty, whereby several animations based on constraints that satisfied a plausibility distribution were created and presented to the user to choose from. A Markov Chain Monte Carlo method was used to sample the space of possible solutions and a Gaussian distribution function was used as a measure of the plausibility distribution. A perturbation of the collision normals was used to guide simulations towards desired constraints and to provide the various solutions within the plausibility distribution curve.

Popovic et al [125] exploits physical uncertainty to develop an interactive system whereby animators are allowed to interactively adjust and control the outcome of physical simulations. Objects are allowed to be dragged and positioned into a desired state while the system adjusts state parameters such as initial position, orientation and velocity, along with required surface contact properties such as contact normals as needed in order to achieve the final desired outcome. The final outcome of the simulations play to the perceptual tolerance of the animator.

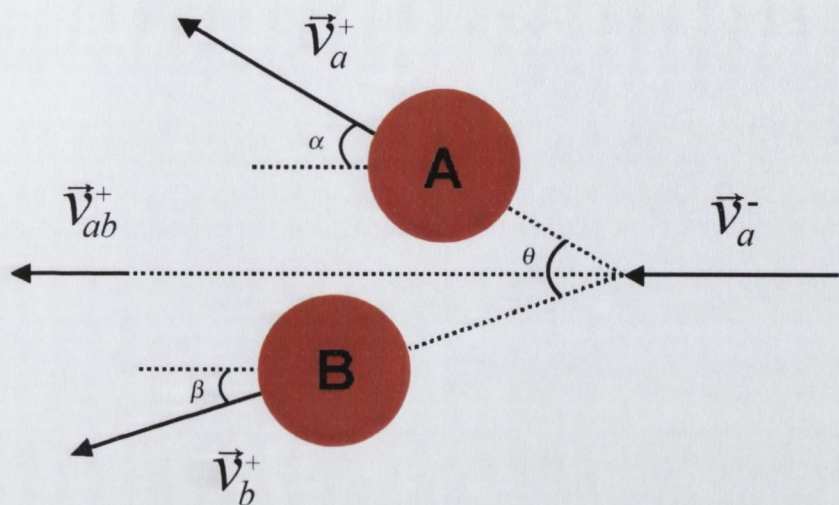
To account for aesthetic demands, Barzel [12] takes advantage of a person's preconceptions of physical behaviour and fakes the dynamics of ropes and springs using kinematic means in order to give the animator control over the animation. This is important within a production environment, where aesthetic requirements of a director may overrule the actual requirement for true physics. The approach uses a default rest shape and animators are then allowed to perform *gross* modification and wave deformations on this rest shape. The result is very convincing dynamic behaviour of ropes and springs. The method was employed during production of Pixar's Toy Story, most noticeably in the animation of the springs of the *Slinky Dog* character.

As we have previously mentioned, Hubbard [76] introduced the idea of interruptible collision detection, whereby detection processing could be stopped once it ran over an allotted amount of time, thus providing us with a level of detail collision detection schema. However, Hubbard only used a simple, non-physically based response. Dingliana and OSullivan [48] later described how information useful for contact modelling and response could be harvested during the collision detection process and then used to determine an approximate physically based response. This work is further extended with the work described in this thesis.

7.1.2 Studies In Perception

The effect of model type used for representing human figures on the perception of human motion was explored by Hodgins et al [74]. Their studies showed that people were better able to observe changes with polygonal models than they were with stick figure models.

O'Sullivan and Dingliana [120] conducted a series of psychophysical experiments to determine the perceptual consequences of the degradation of collision response accuracy and to measure people's ability to notice anomalous collisions. They examined the role of response accuracy, eccentricity, causality, separation and distracters in their experiments. Their results showed that anomalous collisions within the viewer's peripheral range and in the presence of many distracters were less likely to be detected by the viewer. However, O'Sullivan and Dingliana's results also concluded that delays within contact response affected the plausibility of the simulation as the observer's sense of causality was broken. These findings of people's strong sense of causality concurred with those of Michotte's [104]. These results further support our notion of the importance of having good constant frame rates within a dynamics simulation rather than better physical accuracy of collision events, which could hinder frame rate consistency.



v_a^- is the initial velocity of object A (object B is initially stationary); v_a^+ and v_b^+ are the velocities after impact; v_{ab}^+ is the post-collision resultant velocity vector; α and β are the post-collision vector angles and $\theta = \alpha + \beta$.

Figure 7.1: Variables manipulated to study the perception of dynamic anomalies.

While O’Sullivan and Dingliana’s results suggest that a random spin can often hide a bad collision and is sometimes even more plausible than the physically accurate response, this factor was not investigated in any further detail. Nor were the effects of the distorted post-collision trajectories and velocities on the ability to detect an anomaly investigated. These and other types of distortions can occur and must be taken into account in any metric that attempts to measure visual plausibility.

Clement [37] demonstrated that students intuitions concerning simple physics problems (i.e., their naïve physics) were highly inaccurate. Proffitt and Gilden [127] showed that people are reasonably good at detecting dynamic anomalies when there is one dimension of kinetic information present i.e., what they call a particle motion, but are less competent with so-called extended body systems, where there are more dimensions of visual information e.g., both linear and angular velocity. In another study, Gilden and Proffitt [58] showed that people used heuristics based on kinematic data, such as velocity after impact and the way that objects spun, to judge the relative mass of two objects involved in a collision. These studies indicate that people have certain weaknesses in detecting dynamic anomalies, that worsen as the dimensionality of the problems increase.

Kaiser and Proffitt [85] presented a series of experiments that investigated the ability of

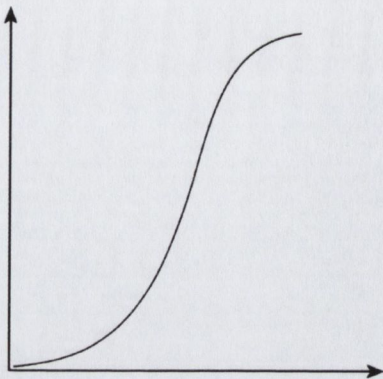


Figure 7.2: An ogive function

people to discriminate between accurate and dynamically anomalous collisions. The variables manipulated within Kaiser and Proffitt’s study are illustrated in Figure 7.1. The methodology used in this study was a randomly interleaved staircase design [39, 99]. The staircase (or up-down) method is an effective psychophysical technique for identifying thresholds, since it ensures that most of the trials are presented near the threshold for that particular observer.

A perceptual threshold, of course, is not an absolute point on the sensory continuum, but rather is stochastic. That is, a stimulus level is associated with a probability of detection. The function commonly used in psychophysics to fit these response curves is a cumulative normal distribution, which is the integral of a normal (i.e. Gaussian) distribution. The general name for this family of S-shaped functions is an *ogive* (Figure 7.2); where the 50% “yes” response rate intersects the ogive, the abscissa represents the absolute threshold stimulus intensity. This view of a threshold as a random variable fits very nicely with Cheney and Forsyths [35] concept of plausibility as a probability function. As was previously mentioned, they used a Gaussian function to estimate the probability that a deviation of the normal from the correct direction is acceptable. Later, we will describe a derivation of a similar set of probability functions as a first step toward estimating the probability that viewers of an animation will perceive certain anomalies. The 50% intersection rate was set using the results of the experiments described in our collaborative work with O’Sullivan et al [121]. We will now briefly present these studies.

7.2 Recent Studies in Dynamic Distortion Thresholds

O'Sullivan et al [121] ran a series of experiments to investigate the thresholds for the perceptibility of dynamic distortions. Kaiser and Proffitt's investigations of angular, momentum and temporal distortions¹ were followed in these studies, along with a new study of anomalous angular velocity discriminations.

Differing from Kaiser and Proffitt's studies, these new sets of studies were conducted in a three dimensional environment using three dimensional stimuli (i.e. a diffuse shaded sphere, block, t-shape, l-shape and a cross), whereby Kaiser and Proffitt only used flat two dimensional disks. Figure 7.3 shows the scenes from the different studies conducted on angular distortion from these tests. In addition to this, a larger display was also used and two different velocities were considered for some of the experiments and different viewpoints.

As in Kaiser and Proffitt's studies, a random interleaved staircase method was used in all the experiments. Four different experiments were run to find the acceptance thresholds for angular distortion. Each of these experiments shifted the angle α and/or β and also shifted, contracted and expanded the θ angle as well. In the momentum experiments the post-collision velocities of the colliding balls were manipulated. In the temporal distortion experiments, two differently induced temporal distortions were examined (i.e. the struck ball was either delayed after or moved off before the collision with the second ball). For the angular velocity experiments, four objects of increasing complexity were hit by a striking ball whereby angular velocity was then proportionally added or subtracted. The results from these experiments are shown in Figure 7.4.

It was concluded from the results of these studies that people were almost 100% more accepting of angular distortions than in Kaiser and Proffitt's study. This was probably due to the fact that the objects in these studies were in three dimensions. It was also concluded that the bias of the θ distortions in the angular distortions were mainly due to the struck ball's post-collision trajectory. The results from the O'Sullivan et al's studies further showed that people preferred a speed-up of linear and angular velocity as opposed to a slow-down after a collision event. This may be attributed to the fact that each person was told before these experiments that the simulations were taking place in a frictionless environment. It is possible that this fact may have somewhat affected a person's preconceptions of physical behaviour. Positive acceptance thresholds for some gaps were found in the temporal distortion studies, conforming to previous studies conducted by O'Sullivan

¹O'Sullivan et al referred to these distortions as spatio-temporal distortions as visibility of the physical separation between colliding balls as well as temporal factors are taken into account.

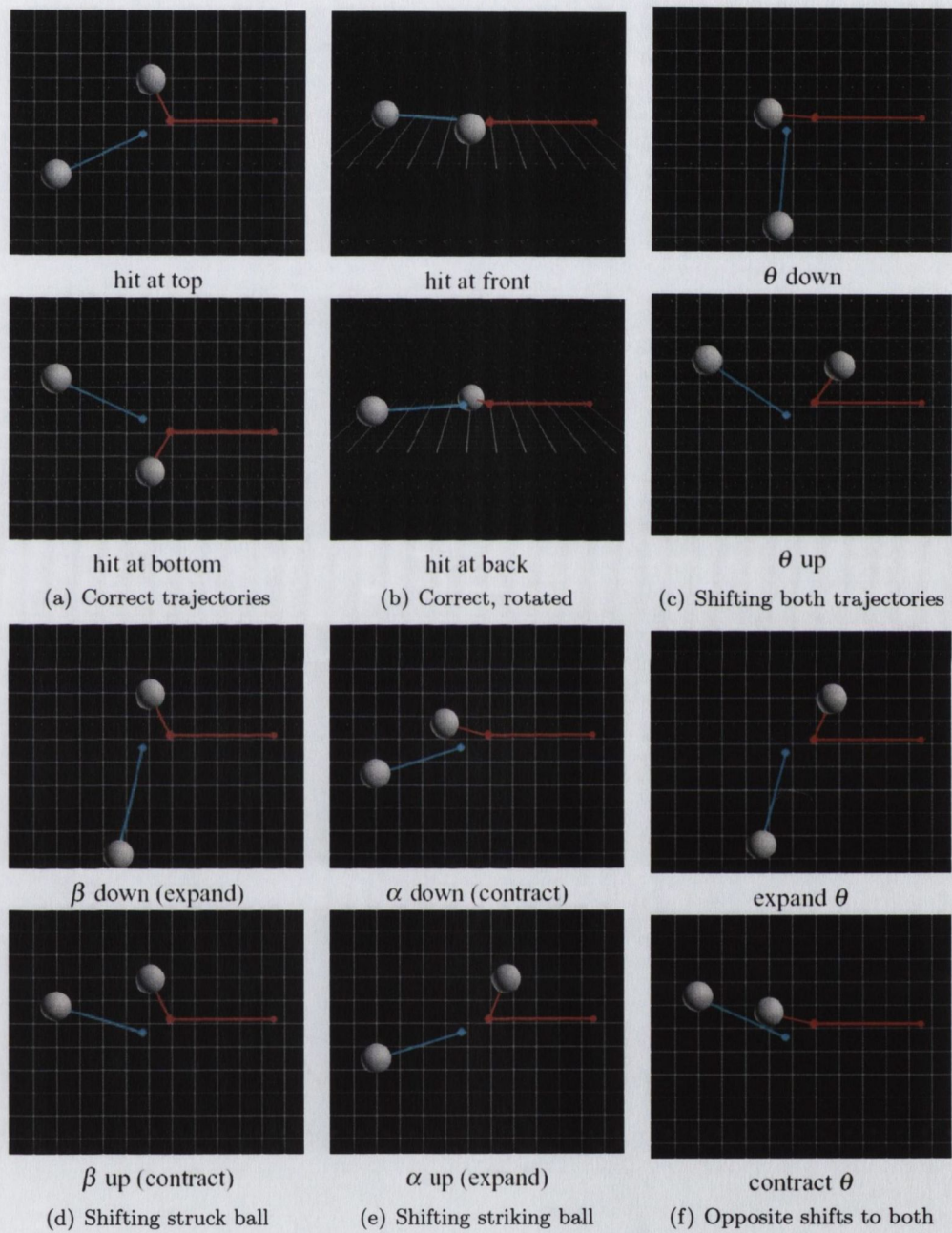


Figure 7.3: Angular Distortions. The red and blue lines indicate the trajectories of the centres of the striking and struck balls respectively

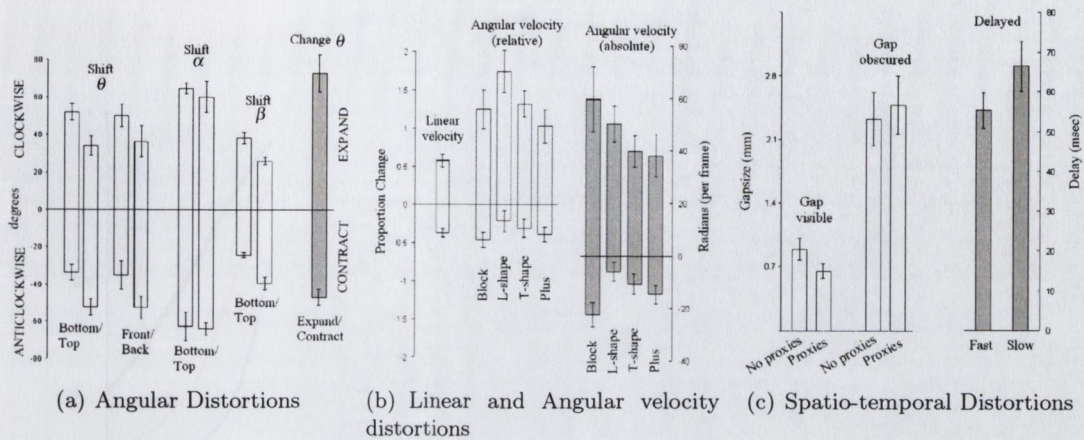


Figure 7.4: Results from psychophysical experiments

and Dingliana [120].

7.3 A Proposed Perceptual Metric

A means of estimating the quality of a simulation that displays dynamic events needs to be extrapolated from the results of Figure 7.4. As in Chenney and Forsyth [35], a similar concept of the “probability” of a simulation is used. A high probability value means that the animation is visually plausible, a value of 1 being fully plausible with respect to a given referent for reality. The X_{50} estimates for each type of anomaly (i.e. the magnitude of distortion at which a person will see that anomaly 50% of the time) is known. The probabilities of all other magnitudes can be estimated by fitting a Gaussian probability function to this data. This is achieved as follows: if θ is the known threshold value, the standard deviation δ for the probability function can be determined as follows:

$$p(\theta) = \frac{1}{\sqrt{e^{\left(\frac{\theta}{\delta}\right)^2}}} = \frac{1}{2} \Rightarrow \delta = \sqrt{\frac{\theta^2}{\ln(4)}} \tag{7.1}$$

These distributions are not necessarily symmetric, so two different standard deviations for each pair of opposite anomalies (e.g. expansion vs. contraction) are used. The probability functions for all anomalies tested are shown in Figure 7.5. Based on the data that are derived from our applications, it needs to be decided which of these functions can be used and also how they should be combined to derive some form of quality measure. The probability, or visual fidelity measurement, of an animation is viewed to consist of three

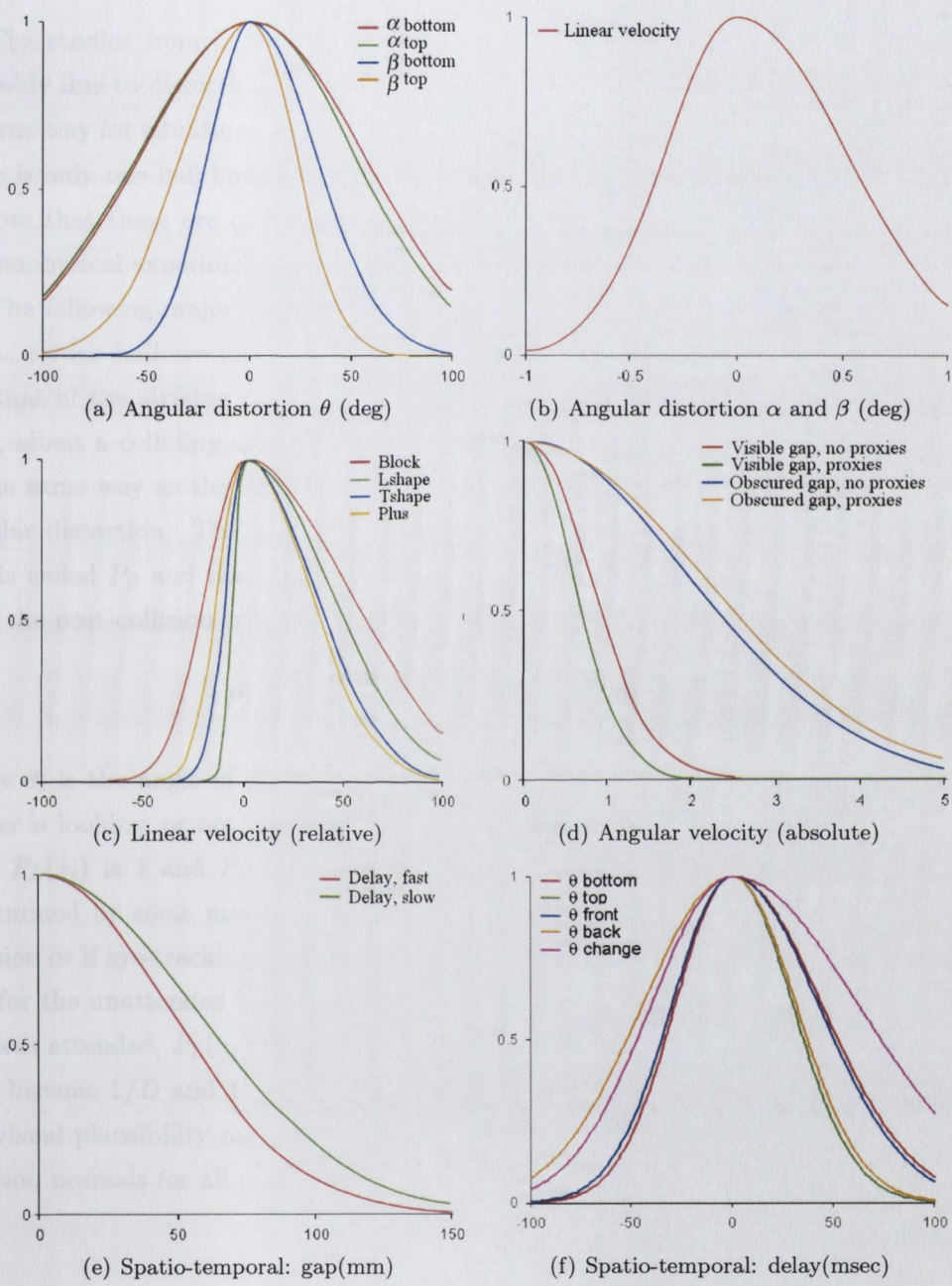


Figure 7.5: Gaussian probability functions for effects observed

components:

$$P(A) = f(P_{angular}(A), P_{momentum}(A), P_{spatiotemporal}(A)) \quad (7.2)$$

The studies from O'Sullivan et al showed that the biases for angular distortions are probably due to distortions in the struck ball's trajectory, but this needs to be generalised in some way for situations where both colliding objects are in motion pre-collision, or when there is only one ball bouncing off a static object, such as a floor or walls. It is important to note that these are only assumptions, to guide the development of a metric. Further psychophysical experiments would be advisable in order to validate their reliability.

The following major assumption is made that allows us to extrapolate: the distortions in the struck ball are more noticeable because the viewer is attending its trajectory and not that of the striking ball. Hence, the striking ball simply becomes another distracting item, albeit a colliding one. All other visible but unattended colliding objects are treated in the same way as the striking ball i.e., they exhibit no bias in terms of the direction of angular distortion. The probability function to be applied to the single struck/attended ball is called P_β and the striking/unattended balls, P_α . Hence, the probability associated with the post-collision trajectory of any object x_i involved in a collision is:

$$P_{angular}(x_i) = P_L(x_i)P_\beta(\theta_i) + P_{NL}(x_i)P_\alpha(\theta_i) \quad (7.3)$$

where θ_i is the angle of distortion and $P_L(x_i)$ and $P_{NL}(x_i)$ are the probabilities that the viewer is looking, or not looking, at object x_i . If there is only one object bouncing around, then $P_L(x_i)$ is 1 and $P_{NL}(x_i)$ is 0. If there are two objects colliding, then if it can be determined by some means which ball is being attended e.g., if it was stationary before collision or if eye-tracking is used, then the probability for the attended ball becomes $P_\beta(\theta_i)$ and for the unattended ball is $P_\alpha(\theta_i)$. If both balls were moving and we don't know which one was attended, $P_L(x_i)$ and $P_{NL}(x_i)$ are each 0.5, while in the presence of D distractors, they become $1/D$ and $1 - 1/D$ respectively. A formulation for the angular component of the visual plausibility metric of an animation A can be suggested to be a function of the collision normals for all objects involved in collisions:

$$P_{angular}(A) = f_A(P_{angular}(x_i))\forall i \quad (7.4)$$

The momentum distortion is described similarly:

$$P_{momentum}(A) = f_M(P_{LV}(x_i), P_{AV}(x_i))\forall i \quad (7.5)$$

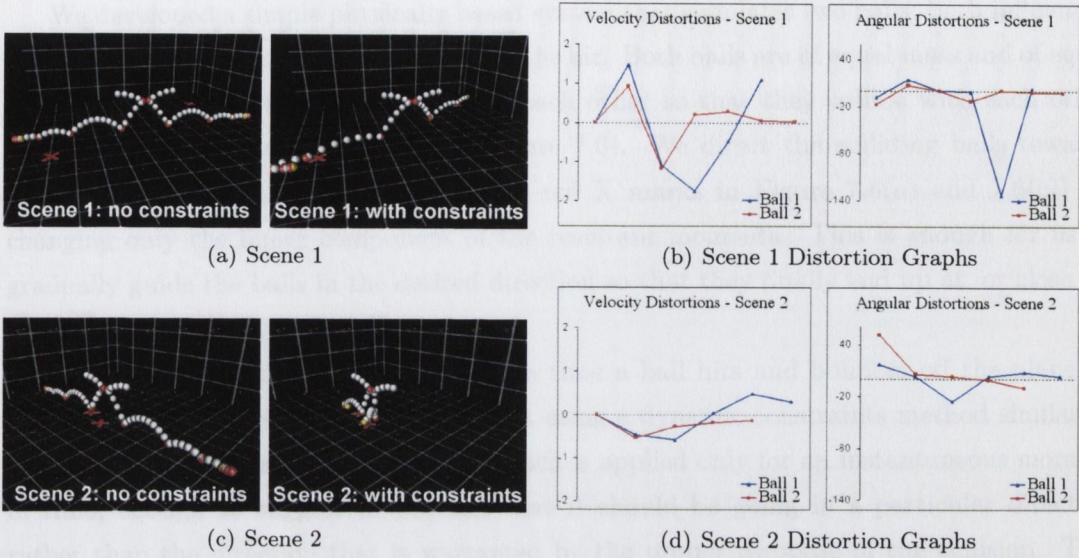


Figure 7.6: Two Scenes from the constrained dynamics framework case study with graphs showing the deviation from normal response to collisions for each scene.

where the linear and angular velocity contributions P_{LV} and P_{AV} are shown in Figure 7.5 (P_{AV} is selected based on the number of potential edge transitions). Finally, the gap factor relates to each collision pair c_j , while the delay factor affects all collisions equally, so the spatio-temporal component can be described to be:

$$P_{spatiotemporal}(c_j) = f_{ST}(P_{gap}(c_j), P_{delay}) \forall j \quad (7.6)$$

The simple experiments described by O’Sullivan et al do not yet allow us to claim with authority how these perceptual factors interact and thus to derive relative weights.

7.4 A Constrained Dynamics Case Study

In order to demonstrate the applicability of the previously proposed metric, we present here a case study using a constrained dynamics example. This section describes our contributions to the SIGGRAPH paper [121] where the work previously discussed in this chapter was first published. This specific case study was so chosen as it provided a good means of measuring deviations from actual physical behaviour (in this case of a bouncing ball). Also, animations resulting from such a case study was not too far different to the controlled bouncing ball animations in which was created by Chenney and Forsyth [35].

We developed a simple physically based system that simulates two balls, both influenced by gravity, starting at an initial position in the air. Both balls are of equal mass and of equal dimensions. They are then sent towards each other so that they collide with each other before hitting the plane below them (Figure 7.6). We direct the colliding balls towards desired spots on the plane, (indicated by red X marks in Figure 7.6(a) and 7.6(c)) by changing only the linear component of the resultant momenta. This is enough for us to gradually guide the balls in the desired direction so that they finally end up at, or close to, the desired positions.

After the initial two ball collision, each time a ball hits and bounces off the plane, it is further directed towards the desired spot using a dynamic constraints method similar to that described by Barzel and Barr [13], which is applied only for an instantaneous moment of time, enough to suggest to the ball that it should be going in a particular direction rather than the direction that is warranted by the proper response of the collision. This approach differs from Cheney and Forsyth's [35], who augmented the contact normal at each collision to eventually guide the ball to its desired spot.

The dynamic constraint applied to the ball for that instant of time also affects the velocity magnitude of the ball, causing a speed-up or slow-down of the ball after it strikes the plane. We do not apply this dynamic constraint at the first time of collision (i.e. before the balls hit the plane) but instead we just simply re-orientate the response velocity vector towards the desired spot, leaving the post collision velocity magnitude alone (as can be seen in Figure 7.6(b) and 7.6(d)). We do this as the pull from the dynamic constraint is not strong enough for it to initially set each ball along the desired path, as rather, its job is to act a subtle discriminator to the natural path of the ball at each collision. If left to run for a while, the dynamic constraints will eventually influence the balls to the desired spots. However, we wish to "help" it along its way at the start by initially pointing it in the right direction.

It is possible to increase the "influence" that the dynamic constraint has on the ball at each collision so that it is no longer more than just a "suggestive influence". However, the influence of the dynamic constraint on the ball is actually a moderately weak constraint force acting for that split second of time, pulling the ball towards the desired spot. If we increase the influence, we also increase this force, thus further increasing the response velocity at a collision. If the constraint force is increased too much, this results in the ball "flying" rapidly towards the desired point in almost a linear path, just as if it was being pulled rapidly towards the point (as it actually is). We do not desire this to happen as we

	Scene 1		Scene 2	
	$P_{angular}$	$P_{momentum}$	$P_{angular}$	$P_{momentum}$
Product:	0.01	0.06	0.17	0.14
Average:	0.89	0.84	0.88	0.85
Minimum:	0.02	0.37	0.35	0.52

Table 7.1: Probabilities for the Constrained Dynamics case study.

wish to maintain the believability of the bounce during a collision. For that reason, we only have the constraint force as a suggestive force at each collision. Also, due to the manner in which the dynamic constraint approach imposes a constraint force for a split second of time only at collisions, the balls will often not land exactly on the spot but close to it as they can overshoot the mark, because no constraints are imposed upon them during free flight. We stop the simulation when either ball reaches a state classified as rest to avoid them rolling forwards and backwards due to the imposed constraint causing the ball to try to reach the end-point.

For each collision, we record the original angles between the incoming and outgoing linear momentum, the new augmented angles due to the applied changes of the new outgoing momentum, and the original and newly augmented velocity magnitudes of each ball. To evaluate the angular distortion probability $P_{angular}$, we applied Equation 7.3. The probability of looking P_L or not looking P_{NL} at each ball is 0.5. The objects did not have any angular velocity distortions, so the momentum probability $P_{momentum}$ was calculated using the linear velocity probability P_{LV} alone and there were no spatio-temporal distortions. The results are shown in Table 7.1 in the form of probability averages, products (as used in Cheney and Forsyth [35]) and minimums. The averages are similar, whereas the minimums and products would suggest that the animation in Scene 2 has a higher visual fidelity. The interpretation of this information is up to the developer or designer. It could be used, for example, to improve an algorithm to generate only animations with a lowest error bound, or to choose an aesthetically distorted simulation for a game or movie.

7.5 The Plausibility Of CFMs and Point Reduction

In order to better validate our proposed Closest Feature Map and point reduction schemes we present here the results of tests conducted in order to determine the overall physical plausibility of animations created with such schemes. Later we shall present the results of visual fidelity probability evaluations on our CFM scheme utilising the proposed new metric discussed within this chapter.

To validate the physical plausibility of our CFM and point reduction schemes we set up several force choice experiments whereby observers were asked to choose between pairs of animations created with and without the use of CFMs and point reduction. From these results, it was hoped that the overall merits of our scheme in terms of viewer preferences to simulated physical behaviour could be determined.

Three different pairs of animations were setup in order to test the physical plausibility of CFMs along with another two pairs of animations to test the plausibility of the point reduction scheme. Ten observers were asked to choose, in terms of personal preference, from each pair of animations shown and the results then recorded. None of the observers were told which method each animation was using or what the experiment was about, but rather to just simply look at the animation pairs and to choose the animation they preferred most from each pair. To reduce potential biasing from distracters within our animations, we chose to only have two animated objects within each animation so that an observer's attention was not taken away from the overall physical behaviour of such objects.

CFM Plausibility

For each CFM animation shown to observers, we compared these animations to that created using Dingliana and O'Sullivan's [48] approximation scheme. We first showed an animation of one box falling on top of another box such that they come to an eventual stacked resting state. This was so chosen as the objects were relatively simple which reduced expectations of anomalous physical behaviour and also such an animation produces subtle physical anomalies for Dingliana and O'Sullivan's scheme (i.e. shifting of the top box to conform to the spherical bounding volume geometry) which the CFM scheme corrects. We were curious as to how alert observers were to such subtle anomalous behaviours.

For the second pair of animations, we chose to drop the two box objects onto a set of ramps whereby any subtle anomalous behaviour was harder to detect. For the third animation pair, we repeated this setup with the ramps but this time with two more complex shaped objects (i.e. a dragon model) so that judgement of physical behaviour became a little more challenging still. Figure 7.7(a) shows these scenes.

From the results of these experiments, shown in Figure 7.7(b), it can be seen that, for the ramps scenes, there was very little difference in terms of preference for animations (in fact almost equal preference). We cannot conclusively say with the difference of one person that the non-cfm scheme was preferred however, within the continuous contact case (i.e. scene 1), there was an overwhelming preference for the animation conducted with our CFM

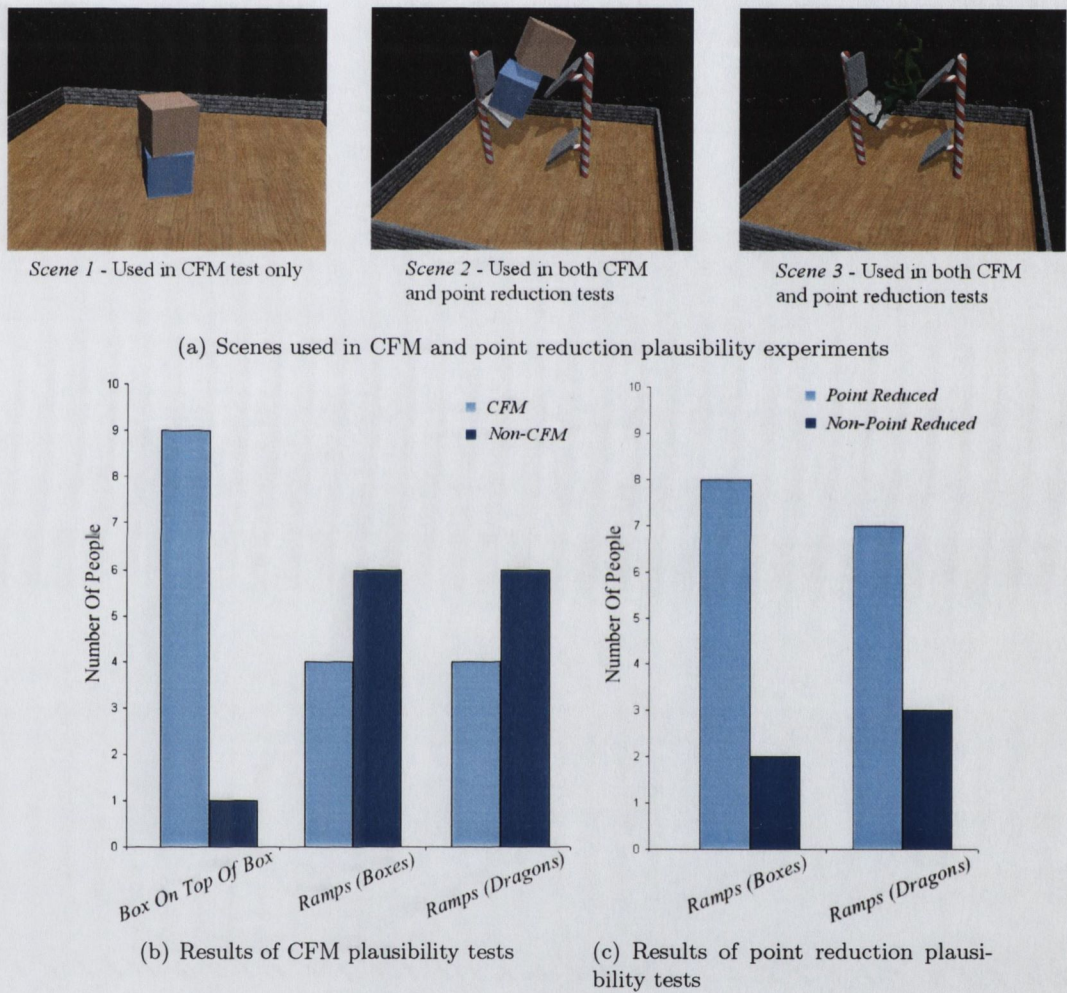


Figure 7.7: Scenes and experimental results of CFM and point reduction plausibility tests

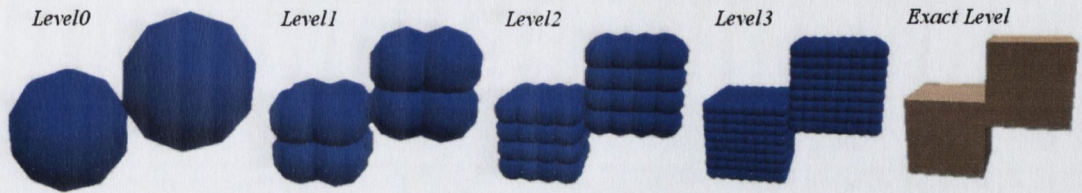


Figure 7.8: Varying levels of bounding sphere proxies for box objects used for visual fidelity probability test of CFMs

scheme. From these results, it may be concluded that our CFM scheme is much better at maintaining physical plausibility within animations of cases of continuous contact where some sort of stacking is involved over that of Dingliana and O’Sullivan’s approximation scheme. However, in cases of non-continuous contact whereby stacking of objects is not involved or else not visually so, our CFM scheme is comparable to using Dingliana and O’Sullivan’s approximation scheme.

Point Reduction Plausibility

In order to validate the physical plausibility of the point reduction scheme we conducted the experiments with similar ramp scenes as those of the CFM experiments. The box on box scene was not considered for the point reduction tests as such a setup produced the same physical behaviour (as was expected) for both animations. Again, for each pair of animations, one animation was produced with the point reduction scheme in place and the other without (thus resulting in large amounts of contacting points to be considered during collision resolution).

Surprisingly, as can be seen from Figure 7.7(c), the majority of people it was discovered, preferred the animations produced with our point reduction scheme in place. We can only conclude from this result that it is actually better to use points of contact around the outer edges and vertices of an object during collision resolution (as is the case in most collision handling schemes) than to try to simulate for the entire contacting area of objects. This is an encouraging result as it not only validates our point reduction schema but also lends merit to it. Such a result further suggests that standard practice of using approximations through contacting vertices, points on a contacting edge and other likewise approximations for collision resolution is more than perfectly acceptable.

7.5.1 Evaluation Of Visual Fidelity Probabilities

To further determine the overall plausibility of animations created with our closest feature map schema, we took the new plausibility metric proposed within this chapter and applied it to animations produced for four different bounding sphere levels for a simple box object. The animations consisted of two box objects in a zero gravity environment with one box object moving towards another initially stationary box object. The two boxes eventually collide providing us with a physical response whereby the appropriate response parameters were then recorded.

We did this for animations using our Closest Feature Map scheme and then for animations using Dingliana and O'Sullivan's approximation scheme so that we could determine the difference in visual fidelity for each schema along with the merits of our scheme.

In order to validate our data with the proposed plausibility metric we needed some sort of reference in order to compare this data with. In this case, it was necessary to also simulate and determine the physical response from an exact polygonal level detected collision. This data was then used as our reference data during fidelity probability calculations. A simple bounding box overlapping axis and then polygon-polygon intersection test was used to detect collisions and for determining appropriate collision data for this exact level simulation.

For each animation in both the CFM and non-CFM cases we measured the magnitude of the linear and angular velocity responses, the angle of the outgoing linear momentum and the nearest distance between the surface of each colliding box. From these results, we evaluated the angular distortion probability $P_{angular}$ for each object, x and y in the animation, along with the linear and angular velocity distortion probabilities, P_{LV} and P_{AV} respectively. Also, the probabilities for the gap between each object, P_{gap} , for each proxy level was determined. For the case of evaluating the angular distortion probability $P_{angular}$, the probability of looking P_L for the moving box (i.e. object x) was determined to be 1 and the probability of not looking P_{NL} for this moving box was 0. This was the opposite way around for object y , the initial stationary box (i.e. $P_L = 0$ and $P_{NL} = 1$). We did not consider any delay distortion probabilities P_{delay} as there was no delay for collision response within these animations.

We present the evaluated probabilities for both the CFM and non-CFM schemas in Figure 7.9. As can be seen from the results presented by Figure 7.9(a), our CFM method showed to exhibit an extremely high visual fidelity for all bounding sphere levels for all evaluated probabilities barring the P_{gap} probability. This result can be attributed to the fact that with the CFM method, data for collision response is approximated at the polygonal

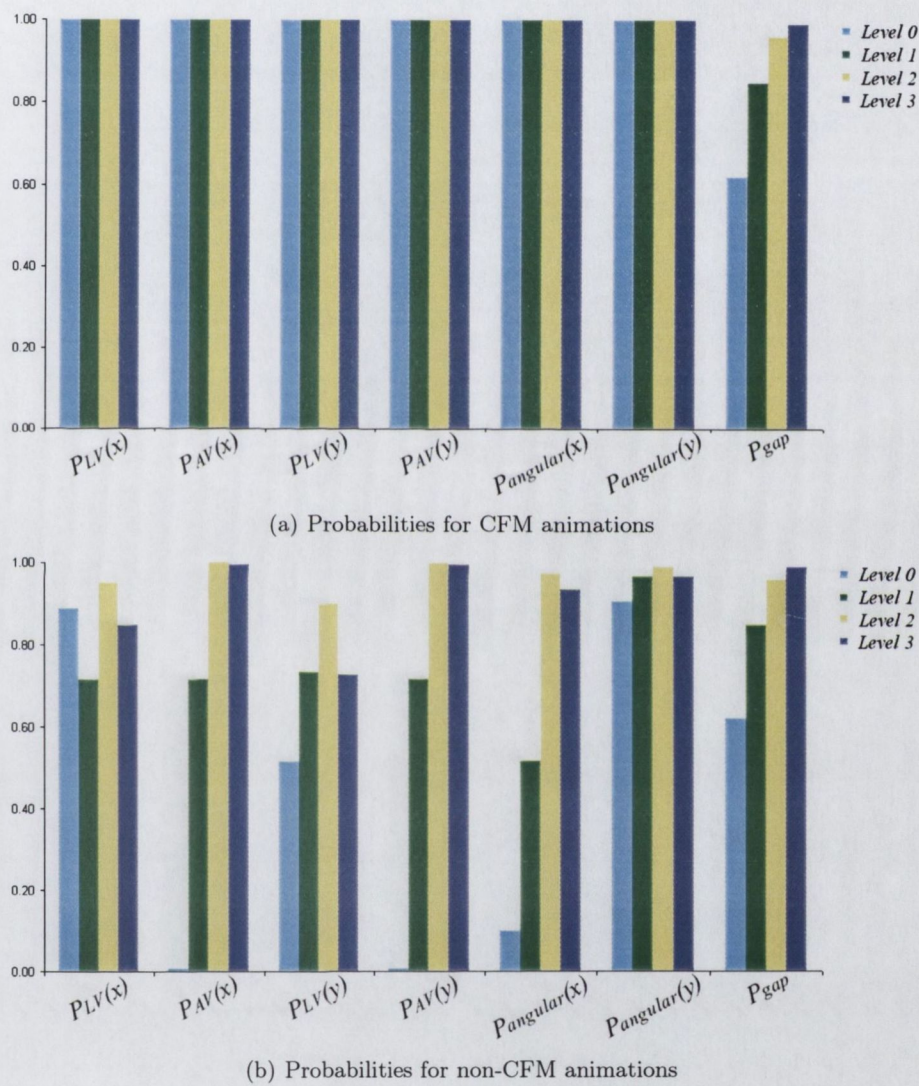


Figure 7.9: Results of the visual fidelity probabilities of CFMs in an animation

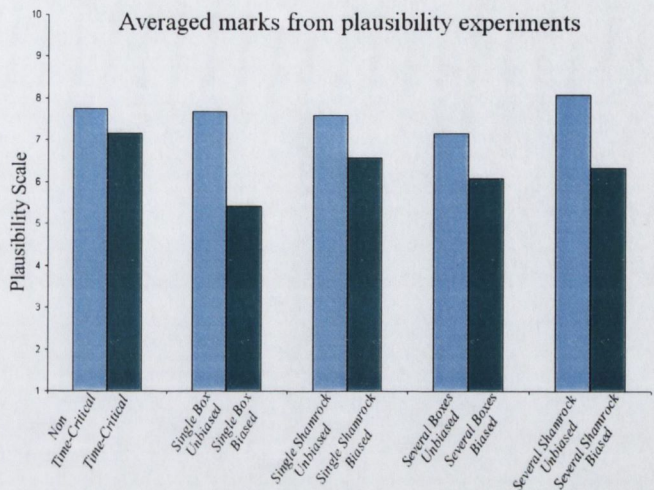


Figure 7.10: Results of preliminary tests on plausibility of time-critical approach and physical preconceptions

level rather than through a rougher approximation through the bounding sphere geometry as is the case with Dingliana and O’Sullivan’s non-CFM method. From Figure 7.9(b), we can see that Dingliana and O’Sullivan’s approximation method does provide a progressive refinement in plausibility as we traverse down the sphere-tree hierarchy, however the angular velocity probabilities, P_{AV} , for the root node sphere of the tree exhibits an overall very poor visual fidelity. This can be seen in the animations as little to almost no spin imparted on the box objects during the collision event. This however is not the case with animations created with our CFM method. The P_{gap} probability is the same for both methods as the distance between the colliding objects are the same due to the fact that we use the same sphere-tree BVH for both animations.

These encouraging results further lend weight to validating the advantages of our CFM schema for use within time critical dynamics. For animations of simple geometric objects such as a cube, these results suggests that the CFM method preserves the overall physical behaviour for all bounding sphere levels.

7.6 Pilot Studies In Time-Critical Plausibility

We ran two preliminary experiments to determine the physical validity of the animations produced by a time critical approach and secondly to examine how much people are influenced by their preconceptions of correct physical behaviour.

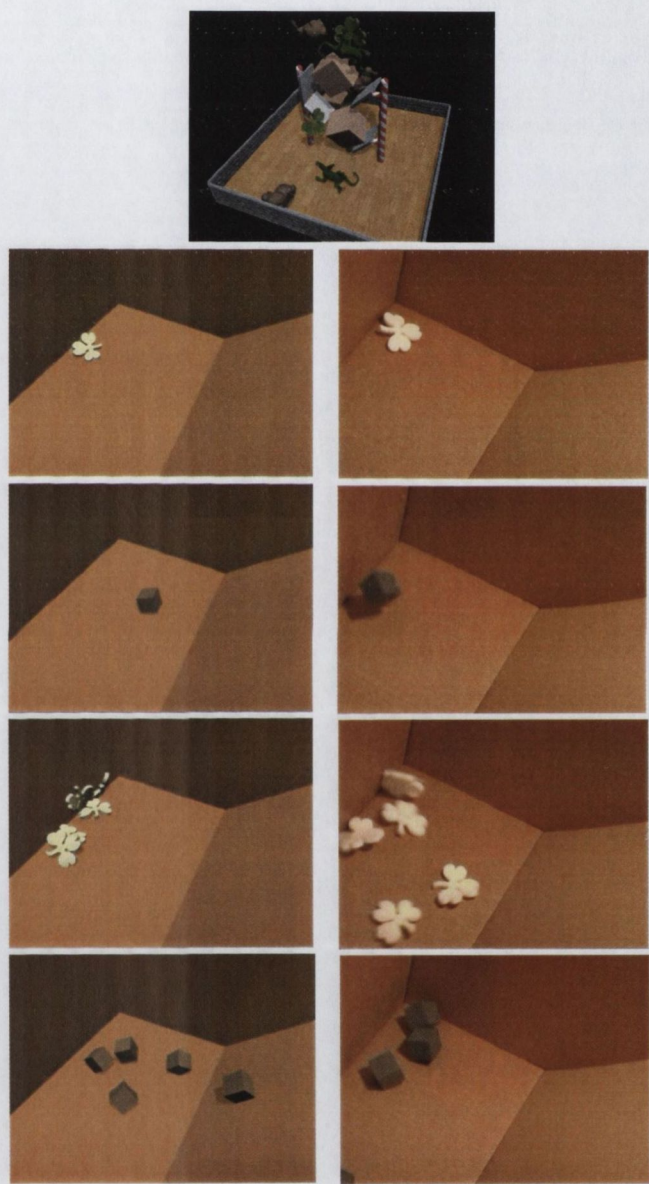


Figure 7.11: Scenes used in the plausibility experiments

Plausibility of Time-Critical Animations

We wished to determine the level of plausibility of time-critical approaches compared to non-time-critical approaches. To this end, we conducted a preliminary experiment to gauge how much more plausible or implausible an animation conducted in an interruptible manner with refinable approximate collision data would be. We took 13 observers and asked them to rate from 1 to 10, with 10 being the most plausible and 1 being not plausible at all, the plausibility of two given animations. Each animation featured the same scene with the same initial setup of 20 rigid bodies all lined up in a straight line one on top of another, ready to fall on a set of ramps once the simulation started, as illustrated in Figure 7.11. One of the animations was done using a time-critical approach set to run at a constant 20 frames per second, just enough to give us an almost fine level of refinement, while the other animation was non-time-critical and resolved all collisions down to the lowest sphere level of the sphere-tree bounding volume hierarchy. We did not tell the volunteers about the difference in method for producing each animation. Instead we simply asked them to rate the animations that they saw with no further additional information.

Although the time-critical approach ran at a constant frame rate of 20 frames per second and the non time-critical approach ran at variable frame rates up to the achievable maximum, each simulation was invariably captured to avi and ran at 40 frames per seconds to reduce biasing caused by different frame rate speeds. The actual variation in frame rates for the non-time-critical simulation ranged from a maximum of 23 frames per second to a minimum of 4 frames per second. We randomised the order in which we showed the simulations to each observer to reduce biasing of comparison between animations. Each observer was allowed to view an animation as many times as they wanted and were told to judge by their impression of what they were seeing and to try and not make comparisons to what they had previously seen. We show the results from these tests in Figure 7.10.

The results showed that almost half the people had trouble distinguishing between the time-critical and non time-critical animation. Out of the 13 observers, 2 found the time-critical simulation to be more plausible, 4 couldn't see any difference between the two animations and the remainder only slightly rated the non-time-critical approach as being moderately better (in most cases by only a single mark). Those that found the non time-critical simulation to be more plausible gave reasons of too much spin or not enough spin during collisions which acted to impact upon their judgement. Interestingly, these were also the same reasons given by those that thought that the time-critical approach was better. This suggests that angular distortions play more of a role than linear distortions in affecting

the plausibility outcome of a physically based simulation.

Given that a 10 mark meant completely plausible and a 5 mark referred to not sure, we considered any score close to or around the 7 mark (i.e. halfway between completely plausible and not sure) to be a good measure of overall physical plausibility. It was hypothesised that any reasonably physically sound simulation should at the very least fall around that halfway mark. From this, the results from these preliminary experiments do go some way to suggesting that time-critical animations can maintain a reasonably good level of plausibility. It must be stressed that these results are very preliminary and much more stringent testing need to be conducted. However, these pilot studies have shown that a time-critical approach to physically based simulations does not necessarily lead to an overly degraded perception of animations.

Variable Preconceptions of Physical Behaviour

As part of the same set of experiments, we also conducted a study into how much people are potentially influenced by their preconceptions of physical behaviour when making judgements about physical simulations, given no immediate means of comparison. For this set of experiments, we initially showed 4 animations followed by a real world example and then the same set of animations again. These are illustrated in Figure 7.11. The testing procedure was the same as the last experiment. We again asked the observers to rate the plausibility of the animations between 1 and 10 for each of the four shown simulations. However, this time, we showed the observers the same simulations for a second time, after we showed them a real world comparison, and then asked them to reevaluate the simulations given their new perception of physical behaviour from the real world examples.

The animations were of a single box, a single shamrock model, multiple boxes and multiple shamrock models falling down an inclined plane into a corner (Figure 7.11). The box model was chosen to gauge preconceptions of physical behaviour on a simple geometric shape while the shamrock was to gauge preconceptions on a slightly more complex shape. Multiple instances of the objects were shown along with single instances of the objects in order to gauge the difference in preconceptions when there were more than one object to mentally evaluate. Again, observers were told to judge the animations based on just that animation alone and to try and not judge what they were seeing with what was previously seen. Observers were not told beforehand that they would be later shown real world examples of the simulated scene so as not to bias them initially. The real world scene was constructed using a cardboard box and different bits of cardboard pieces.

We expected the last four set of results from these experiments to be lower than the previous four sets but we were curious as to how much lower. We were curious as to how much people trusted their preconceptions of physical behaviour and how easily they could be influenced. Figure 7.10 illustrates these results.

For the last set of simulations, observers were told to take into consideration dynamic chaos (e.g. objects are always thrown down differently in the real world) and also to account for the fact that we could not exactly replicate the material properties in the simulated scene to that shown in the real world scene. The one comment that a few observers made when shown the real world scenes was that objects moved a lot faster and stopped a lot quicker than they expected in the real world. This would suggest that some people do indeed have much looser preconceptions of physical behaviour than others and hence can be more accepting of what could be classified as above normal anomalous physical behaviour in dynamic simulations.

The results for the first four simulations shown to observers more or less conformed to those of the results gathered from the animations in the first experiment previously described. As was expected, the results for the last four simulations shown to observers were evaluated lower. However, on average, the last set of simulations were only marked about a mark or 2 marks lower. In some cases, the simulation was marked higher the second time around than before the observer had been shown the real world examples. We may hypothesize from the results of this set of experiments that people do have very strong preconceptions of physical behaviour that do indeed play a large part when asked to make judgements on physical accuracy when observing simulated dynamic behaviour. From the results gathered, physical plausibility is indeed, it would seem, a very subjective concept. However, the results of this preliminary study also indicate that there exists a fine cloud of globally acceptable answers that, if our simulations fall within this range, can potentially satisfy the majority of observers.

7.7 Conclusions

In this chapter, we have looked at the concept and issues of plausible dynamic simulation and presented a perceptual metric for the evaluation of visual fidelity within physically based simulations. We also discussed a case study of a constrained physically based system and how we may apply the proposed perceptual metric to such a case. The results of our force choice experiments conducted to evaluate the physical plausibility of our CFM and

point reduction schemas was presented. The visual fidelity metric discussed in this chapter was then used to further validate the merits of our CFM scheme by applying it to various animations created with the scheme. It was suggested by this metric that the visual fidelity of animations created using our CFM scheme was indeed quite high.

We further presented some preliminary experiments conducted by us to determine the extent of physical plausibility within time-critical simulations and also to attempt to determine the extent to which preconceptions actually influence our judgements of physical simulated events. It is evident from the discussions in this chapter that physical plausibility is not an elementary concept to evaluate.

The studies discussed and presented in this chapter are very preliminary. The perceptual metric presented is not absolute but rather it is a first step taken towards the development of such a metric. The studies discussed in Section 7.6 warrant more work under more controlled conditions. The ratings used in these pilot studies are very subjective measures. More stringent techniques, such as forced-choice or the staircase method should be used instead in future experiments. However, these pilot studies may serve as a basis for further study in that general direction.

Chapter 8

Conclusions And Future Work

In this thesis we have presented the concept of time adaptive dynamics and the issues involved in implementing a fully comprehensive system to achieve this. The idea of plausible rather than realistic physical behaviour as a general goal was adhered to. Based on this principle, many simplifications could be adopted that allowed us to propose viable algorithms for conforming to required timing constraints.

The refinable nature of the collision response approach in the time adaptive framework does not lead to repeated outcomes for animations starting with the same initial states, whereas non-time-critical physical simulations will always give us the same outcomes from repeated simulations. This in itself is an unexpected advantage of the time adaptive scheme, as it provides a model of dynamic chaos similar to that which exists in the real world. This advantage can be used in such situations as interactive entertainment whereby it is desirable to have the extra detail of dynamic chaos within the virtual world to enhance overall player immersion.

The results presented in this thesis have shown that it is possible to have a viable time adaptive system if physical plausibility is adhered to. There is much scope for the expansion and further incorporation of other approaches into this work. The ideas and methods presented are not definitive and many other viable alternative strategies may exist to either further improve or complement such approaches.

Like Barzel et al [14], we also advocate the advantages of physical plausibility over physical accuracy for cases where accuracy is not a necessity or desirable. It is hoped that the work and ideas presented in this thesis have demonstrated the advantages of adopting such a strategy.

8.1 Summary of Contributions

This thesis investigated the concept of viable solutions for the domain of time-critical physically based simulations and a number of novel contributions to this end were presented:

1. The main goal of this thesis was to investigate and resolve the multitude of issues involved in the implementation of a fully time adaptive dynamics system. To this end, sphere-tree bounding volume hierarchies were utilised in aiding us with interruptible collision detection and refinable response data. However, despite the advantages that such a data structure has for interruptible collision detection and refinable response data determination, it also brings with it some problems. These problems were addressed with our proposal of using the novel concept of ***Closest Feature Maps for refinable contact data approximation*** (pages 141-150). The disadvantage of previous approaches to finding good approximations of contacting data for collision resolution was that the sphere geometry was used for this purpose. This could lead to bad contact approximation, especially in cases of continuous contact, which could potentially lead to anomalous physical behaviour. Closest Feature Maps addressed this problem by approximating contact data at the polygonal level rather than the bounding sphere level. We presented two approaches for constructing and sampling our Closest Feature Maps. These being, a polar sampling approach and a geodesic sampling approach. Each one had its advantages and disadvantages. The polar sampling allowed us very quick guaranteed constant time retrieval of desired closest feature information but, however, the sphere sampling was not uniformly distributed. The geodesic sampling approach provided uniform sphere sampling but lacked the speed advantage in retrieving desired closest feature information of the polar sampling scheme.
2. We also investigated the issue of overabundance of contact data during collision detection. Such an issue had not been considered in the past, within the realms of efficient refinable response processing. A ***point reduction schema*** (pages 121-125) was presented and this was incorporated to allow us to efficiently ***adapt our collision response processing to required timing constraints*** (pages 199-200).
3. In this thesis we also investigated the challenge of having a fully comprehensive ***scheduling strategy for time adaptive dynamics*** (pages 105-109). A scheduling approach based on retroactive schemes for maintaining constant frame rates in

level of detail rendering was presented. This scheduling scheme allowed the adaptation of not only collision detection to frame timing constraints but also of the collision resolution processing.

4. In order to allow us more time to conduct collision handling for each frame in the simulation, we considered an alternative integration scheme to that of the more common Runge-Kutta schemes, known as *Verlet* integrators. In simple spring tests conducted, Verlet integrators were shown to have almost as good stability and accuracy as that of the fourth order Runge-Kutta integrators but with far fewer calculations. These schemes, originally designed to handle variable updates in the linear domain only, had to be adapted for rigid bodies which required state variable updates in both the linear and rotational domain. A **hybrid Verlet Runge-Kutta integration scheme, called *RKVerlet* (pages 85-90)**, was presented as a novel solution.
5. We also discussed, in this thesis, a metric to evaluate the visual fidelity of physically based animations and presented the **first steps towards applying this metric in a meaningful way to a physically based system (pages 168-171, 174-176)**. Pilot studies into the plausibility of simulations produced by time-adaptive approaches were also presented (pages 176-180).
6. A fully viable time adaptive dynamics framework that takes advantage of an observer's uncertainty about physical behaviour was the culmination of all the work presented in this thesis.

8.2 Future Work

The area of time-critical physically based simulation is a relatively new area of research and as such there is much work that can still be done. For the methods presented in this thesis, there is a broad scope for further improvements and alternative considerations. We present in this section, suggestions for possible future work that could be pursued.

Further Enhancement Of Collision Detection

Even though the sphere-tree hierarchy allows for fast culling of non-colliding areas during collision detection, there is still much scope to further enhance the overall speed of the collision detection scheme.

One possible consideration could be in perhaps utilising the point reduction scheme described in this thesis to quickly cull out further unnecessary spheres during our traversal. Since we know that we only desire a predefined number of contacting spheres then perhaps it would be more advantageous to not bother trying to determine collisions between the parent spheres whose children we are going to ultimately discard. So, instead of doing point reduction to contacting points at the end of the collision detection process, this proposed scheme would conduct point reduction at each level as we traverse down the sphere-tree. In this manner, it is proposed that we can efficiently disregard spheres at a higher level so that we don't waste our time doing additional collision detection on those spheres and their children.

Additionally, it may also be advantageous to consider incorporating some form of caching or hill climbing schema that will take advantage of temporal coherence between frames. Temporal coherence speedup schema are regularly utilised in many other collision detection approaches to further enhance detection processing efficiency.

Collision Handling With Alternative Bounding Volume Hierarchies

Sphere-trees are not the only existing bounding volume hierarchy that could be exploited for interruptible collision detection. There are many alternative bounding volume hierarchies that can be considered for collision detection and that can be quite easily adapted for time critical collision handling.

Other hierarchical data structures based on simple primitives such as axis-aligned or oriented bounding boxes could also be considered as viable alternatives for the collision detection stage instead of sphere-trees. However, with each different bounding volume hierarchy considered, there may potentially be a new set of problems that will arise from utilising such a structure. A thorough study is warranted on the advantages and disadvantages, in terms of both efficiency and visual fidelity, of adopting a different bounding volume hierarchy for time critical collision handling.

More Efficient Geodesic Sampling Search Schemes

Although the geodesic sampling strategy in the Closest Feature Map schema provides a balanced sample point distribution, the search for the closest feature information can still be further improved. At the moment, the search scheme is four times as efficient as the basic linear search scheme. However, it is believed that this can be vastly improved through the use of clever data organisation coupled with an optimum data structure designed to

enhance regional analysis.

One possible approach that may be considered is to utilise a similar voxel space scheme that was described for the closest feature determination during the Closest Feature Map construction pre processing stage. A local voxel space scheme incorporating the same *concentric ring* optimisation strategy could be utilised.

Another alternative consideration would be to utilise temporal coherence in some form in our search strategy. For example, if we find that a contact point resides in the same spherical quadrant as during the previous frame or its distance from the last contacting point is within some minimum threshold, we can assume that the nearest spherical sample point is either the same one as before or one of its neighbours.

Further Enhancement To Closest Feature Map Scheme

The Closest Feature Map approach as described could possibly suffer from an overabundance of data. As such, it may be advantageous to consider appropriate data compression schema that will limit the amount of data that needs to be stored.

Additionally, the geodesic and polar sampling schemes may no longer apply to other bounding volumes of different geometry. If other geometric bounding volumes are utilised instead of spheres, Closest Feature Maps may have to be adapted to conform to the new bounding volume geometry. Specifically, the sample point distribution schema must be adapted to the geometric shape to ensure that a fair point distribution is adhered to.

Further Studies Into Plausibility And Time-Critical Approaches

We have presented pilot studies in this thesis that try to determine and evaluate the level of perceptual degradation of physical plausibility within dynamic simulated scenes generated by systems that utilise a progressively refinable time critical approach. The results generated from these studies, as they stand, cannot be viewed as conclusive, but do provide, however, an initial bases for further perceptual studies in this general direction of research. The testing methods and rating scheme presented are quite subjective and more controlled and stringent testing approaches based on, for example, the *interleaved staircase* approach will be considered.

Incorporation Of Interruptible Rendering For A Complete System

A fully comprehensive system based on both geometric and simulation level of detail is the next logical step. The framework described in this thesis culminates in the first complete time adaptive dynamics system. However, the system can be further extended to be a fully comprehensive overall *time adaptive animation system* by incorporating time critical processing into the other processing stages of the simulation pipeline that we have purposely neglected to explore in this thesis. Specifically, the rendering stage can also be incorporated into a time-critical heuristic by perhaps using a schema such as that described by Wooley et al [145]. The full incorporation of time adaptive processing for all stages in the simulation pipeline would then dictate that scheduling heuristics based on fairness and/or critical prioritisation be considered and developed.

A.1 Basic collision impulse assumptions

During a dynamic event, or collision event t_c , there may be two or more bodies colliding with one another. If this is the case, then all bodies that are in a state of collision are to be treated simultaneously. Each point of contact on the colliding bodies and the influences through those points (i.e. impulses) are viewed as affecting the overall outcome of the collision between that body and the body or bodies which it is colliding with. So for each contact point on a colliding body, that point must be considered to have an influence on

Appendix A

Resolving Multiple Contacting Collisions

Within this thesis, we have mentioned the notion of collision response resolution but provided little detail on how we might go about resolving collisions once the appropriate contact data has been obtained. It is important that we have appropriate collision response formulations in place that allow us to resolve a collision between two or more objects in a physically plausible manner. It is too naïve an assumption to simply reverse the velocities whenever a collision is detected. Instead we must incorporate some sort of physical law into collision resolution in order to maintain a high degree of plausibility. It was mentioned in Chapter 5 that we chose to resolve collisions in a simultaneous manner for multiple contacting points through the formulation of a *Linear Complementarity Problem*. For completeness, we present here the derivation for obtaining appropriate impulses given the problem of multiple contacting points during a collision. The formulations presented below are based on Baraff's [3] suggestions for dealing simultaneously with multiple points of contact.

A.1 Basic collision impulse assumptions

During a dynamic event, at collision time t_c , there may be two or more bodies colliding with one another. If this is the case, then all bodies that are in a state of collision are to be treated simultaneously. Each point of contact on the colliding bodies and the influences through those points (i.e. *impulses*) are viewed as affecting the overall outcome of the collision between that body and the body or bodies which it is colliding with. So for each contact point on a colliding body, that point must be considered to have an influence on

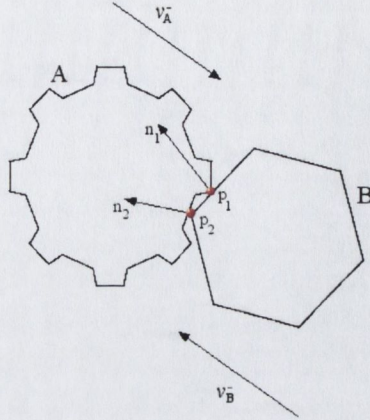


Figure A.1: Contacting points and normal directions during collision between two objects. Collision at p_1 affects how p_2 reacts and vice versa. Both points p_1 and p_2 are the same for both objects A and B at collision.

how all other contacting points on that body will effect the behaviour of the body and vice versa. Figure A.1 illustrates the relation between body A and B during a collision.

For any pair of bodies, A and B, during a collision at time t_c , the i^{th} point of contact on body A should be such that it is equal to the i^{th} point of contact on body B (see Figure A.1). Let p_i denote the i^{th} point of contact at time t_c on a pair of bodies A and B so that

$$p_{A_i}(t_c) = p_{B_i}(t_c) = p_i(t_c) \quad (\text{A.1})$$

whereby $p_{A_i}(t_c)$ and $p_{B_i}(t_c)$ denote the i^{th} point of contact at collision time t_c of bodies A and B respectively.

Now let us consider the velocity of each body. At any time t , the velocity of any point on an unrestricted moving body ξ may be expressed as

$$\dot{p}_\xi(t) = v_\xi(t) + \omega_\xi(t) \times (p_\xi(t) - x_\xi(t)) \quad (\text{A.2})$$

where $v_\xi(t)$ is the linear velocity of body ξ at time t , $\omega_\xi(t)$ is the angular velocity at time t and $x_\xi(t)$ is the center of mass of the body at time t . For notational convenience let us denote $(p_\xi(t) - x_\xi(t))$ as the variable $r_\xi(t)$ from now on. Also, let us represent pre- and post-collision events using the superscripts $-$ and $+$ respectively. So the pre-collision velocity of point i at time t_c is denoted as $\dot{p}_i^-(t_c)$ and post-collision velocity is $\dot{p}_i^+(t_c)$.

With this in mind, let us consider what happens during a collision event. According to Newton's law of restitution for frictionless collisions, at collision time t_c :

$$v_{rel}^+(t_c) = -\epsilon v_{rel}^-(t_c) \quad (\text{A.3})$$

v_{rel} being the *relative velocity in the normal direction* of the colliding objects which can be expanded to be

$$v_{rel} = n \cdot (\dot{p}_A - \dot{p}_B) \quad (\text{A.4})$$

and ϵ being what is known as the *coefficient of restitution*. This variable, measured between 1 and 0, determines how *elastic* a resultant collision is and can be viewed as how bouncy the materials from both colliding bodies are. For a perfectly elastic collision, $\epsilon = 1$, and for a perfectly inelastic collision $\epsilon = 0$. For a detailed proof please see [46].

Before we continue any further, it may be advantageous to clarify the definition of an impulse to prevent any confusion. Informally speaking, the incoming velocity of a body during a collision event is perturbed by an *impulse* Ψ . This differs from constraint forces that enforce non-penetration. We may simply look at Ψ as a very large force acting over a very small interval of time, applied to two newly colliding objects, in such a manner as to instantaneously change the approach velocities of both objects so that they each push the other apart. Impulses relate to momentum so in fact a more formal definition of an impulse would be *the instantaneous change in momentum over a very small time period*. The direction in which Ψ acts relates to the direction of the normal of contact. All this may be expressed as

$$\Psi = F\Delta t = M\Delta v = fn \quad (\text{A.5})$$

In the above equation, $F\Delta t$ denotes a large force acting over a very small time period, M is simply the mass of the colliding body, Δv being the change in velocity of the colliding body and f the magnitude of the resultant impulse due to the collision. Though the resultant impulse variable Ψ is a vector term, the variable f (the impulse magnitude) is an unknown scalar and is such that $f \geq 0$. This is what our resolver wishes to solve for. So, according to Newton's third law of motion, the final calculated impulse(s) should act positively on one body and equally but oppositely on the other. Let us say that Ψ acts positively on body A and equally but oppositely on body B, thus conforming to Newton's third law of motion. Thus for body A, $\Psi_A = fn$ and for body B, $\Psi_B = -fn$.

Now, let us deduce the change in velocity due to an applied impulse during a collision

at time t_c . It is possible to use the information thus far to derive the change in velocity as an expression in terms of the resultant impulse Ψ . Thus the change in velocity, Δv , at any point on a body ξ due to an imparted impulse Ψ at time t_c is

$$\Delta v_\xi(t_c) = \frac{\Psi_\xi(t_c)}{M_\xi} + I_\xi^{-1}(t_c)(r_\xi(t_c) \times \Psi_\xi(t_c)) \times r_\xi(t_c) \quad (\text{A.6})$$

Where $\frac{\Psi_\xi(t_c)}{M_\xi}$ is simply the linear component of the change and the second part, $I_\xi^{-1}(t_c)(r_\xi(t_c) \times \Psi_\xi(t_c)) \times r_\xi(t_c)$ the angular component. The variable I^{-1} is the inverse inertia tensor of the body and all other variables are as described previously. Knowing this, the equation for the *post collision* velocity of point i on a body ξ at time t_c can be expressed as

$$\dot{p}_{\xi i}^+(t_c) = \dot{p}_{\xi i}^-(t_c) + \Delta v_{\xi i}(t_c) \quad (\text{A.7})$$

We now have v_{rel}^+ as a linear function of f .

A.2 Simultaneous collision points

Recall that during a collision event, $v_{rel}^+(t_c) = -\epsilon v_{rel}^-(t_c)$. This assumption needs to be extended to fit the criteria of multiple points of collision. Let us first make the assumption that the coefficient of restitution, ϵ , relates to each collision point individually on the colliding body rather than the body as a whole. Since each colliding point on the colliding body influences the other, so too must ϵ by virtue of relation. So, in short, for each colliding point on the body during a collision event, that point may possess a different elasticity to any other point on that body even though it is part of the same body. Furthermore, the colliding body may be pushed away by a third (or more) body or bodies in a simultaneous, more powerful, collision thus cancelling the effect of the previous collision and breaking the rule in Equation (A.3). To account for this, let us change the constraint imposed by Equation (A.3) to one that reflects this situation better:

$$v_{rel}^+(t_c) \geq -\epsilon v_{rel}^-(t_c) \quad (\text{A.8})$$

Of course this implies that if $v_{rel}^+(t_c)$ exceeds $-\epsilon v_{rel}^-(t_c)$, then our impulse magnitude f for that point must be zero as the initial collision contact assumption no longer holds. We can write these constraints down as

$$v_{rel}^+(t_c) + \epsilon v_{rel}^-(t_c) \geq 0 \quad (\text{A.9})$$

$$f(t_c) \geq 0 \quad (\text{A.10})$$

$$f(t_c)(v_{rel}^+(t_c) + \epsilon v_{rel}^-(t_c)) = 0 \quad (\text{A.11})$$

From this point onwards, for the sake of clarity, let us drop the t_c variable from our equations as they are going to get quite messy. Instead, it will be assumed that all equations from here onwards refer to time t_c .

Back to the problem at hand. The above three constraints, (A.9), (A.10), (A.11) formulate what is referred to as a *Linear Complementarity Problem*. More formally, an LCP can be stated as:

Given a known vector $\mathbf{b} \in R^n$ and a known matrix $\mathbf{A} \in R^{n \times n}$ the problem is to find a vector $\mathbf{f} \in R^n$ such that

$$\mathbf{w} = \mathbf{A}\mathbf{f} + \mathbf{b} \geq 0 \quad (\text{A.12})$$

$$\mathbf{f} \geq 0 \quad (\text{A.13})$$

$$\mathbf{f}^T(\mathbf{w}) = 0 \quad (\text{A.14})$$

or to show that no such vector $\mathbf{f}$ exists.

Cottle et al. [40] and Murty [114] both give excellent detailed explanations of LCPs and various solution methods.¹

Within the collision resolver, the constraints (A.9), (A.10), (A.11) are considered and the appropriate A matrix and b vector formulated. The vector f , the list of required impulse magnitudes, are then solved for with an appropriate LCP solver. Let us now extend the *post collision velocity* equation (A.7) so that it accounts for simultaneous multiple colliding points. Recall that we said each impulse acting on each contacting point on a colliding body will influence all other contact points on that same body. This must be taken into account in the updated equation formulation. So updating (A.7) will give us

$$\dot{p}_{\xi_i}^+ = \dot{p}_{\xi_i}^- + \sum_{j=1}^k \frac{f_{\xi_j} n_j}{M_{\xi}} + I_{\xi}^{-1} \sum_{j=1}^k (r_{\xi_j} \times f_{\xi_j} n_j) \times r_{\xi_i} \quad (\text{A.15})$$

If we substitute equation (A.15) into equation (A.4), the resultant formulation describes

¹The latter book is unfortunately out of print but is however, freely available in electronic form on the web

the relative velocity at point i in the normal direction after the collision, taking account of all other influences acting through all other contacting points on the colliding objects.

and

$$v_{rel_i}^+ = v_{rel_i}^- + n_i \cdot \left(\sum_{j=1}^k \frac{f_j n_j}{M_{A_i}} + \lambda_{A_i} \sum_{j=1}^k (r_{A_j}^* f_j n_j) - \sum_{j=1}^k \frac{-f_j n_j}{M_{B_i}} - \lambda_{B_i} \sum_{j=1}^k (r_{B_j}^* (-f_j n_j)) \right) \quad (A.16)$$

where $\lambda_{A_i} = r_{A_i}^{*T} I_{A_i}^{-1}$, $\lambda_{B_i} = r_{B_i}^{*T} I_{B_i}^{-1}$, $f_{A_j} = f_{B_j} = f_j$ and $v_{rel_i}^-$ is as in Equation (A.4). Note that we negate the impulse magnitude f acting on object B in the above formulation so as to conform to Newton's third law of motion. To neglect to do so would give us an improper model of the resultant impulses acting on each object.

The above equation (A.16) has been rearranged for notational convenience and clarity. The reader should have no problem in seeing that it was obtained by simple substitution. Perhaps the only confusing thing is the variables with the $*$ superscript. Barzel et al [13] name this *transformation* as the *dual of a vector* and hence we follow suit here. This formulation is just a convenient way for us to express a cross product of two vectors as a matrix vector multiplication. Equation (A.17) below might perhaps give a better picture of what is happening. The appendix B in [13], as well as giving a brief description of vector dual properties, also gives a very good overview of point behavior on a rigid body which is highly applicable to the equation formulations presented here. The reader may want to refer to chapter 4 of Goldstein [61] for the proof of why this vector cross product duality relation holds.

input

main

direction

value

$$a \times b = a^* b = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \quad (A.17)$$

Taking the above equation (A.16) and substituting for the left hand side of equation (A.3) brings us one step closer to obtaining our required b vector and A matrix in our LCP problem. The last thing for us to do is to bring over the right hand side of equation (A.3) and further rearrange to give:

value

sign

$$b_i + \sum_{j=1}^k f_j A_{ij} \geq 0 \quad (A.18)$$

where

value

$$b_i = v_{rel_i}^-(1 + \epsilon_i) \quad (\text{A.19})$$

and

$$A_{ij} = n_i \cdot \left(\left(\frac{n_j}{M_{A_i}} + \lambda_{A_i}(r_{A_j}^* n_j) \right) - \left(\frac{-n_j}{M_{B_i}} + \lambda_{B_i}(r_{B_j}^* (-n_j)) \right) \right) \quad (\text{A.20})$$

Each element ij in the A matrix can be got from the above by straightforward substitution.

The above Equation (A.20) may still seem a bit overwhelming so let us further simplify the equation by further variable substitution. Let

$$\begin{aligned} A_{A_{ij}} &= \frac{n_j}{M_{A_i}} + \lambda_{A_i} r_{A_j}^* n_j \\ A_{B_{ij}} &= \frac{-n_j}{M_{B_i}} + \lambda_{B_i} r_{B_j}^* (-n_j) \end{aligned}$$

so that equation (A.18) can now be expressed as:

$$b_i + \sum_{j=1}^k f_j n_i \cdot (A_{A_{ij}} - A_{B_{ij}}) \geq 0 = w_i \quad (\text{A.21})$$

We now have all the information we need to be able to solve for the impulse magnitude f . The collision resolver module, given the appropriate list of collision nodes, will use the above formulae to determine the appropriate constraint formulation to give to the LCP solver. The LCP solver itself in turn will then solve for vector f , the list of appropriate impulse magnitudes, which it can either return back to the resolver to pass on to the main update module in the simulator or return the information to the main update module directly. If the resolver was given only one collision point, then the response problem simply reduces down to one linear equation in one unknown which can be solved very efficiently and mirrors the more traditional single point impulse resolution methods. The impulse magnitudes calculated through the resolver will then be applied as appropriate to the states of the rigid bodies in the simulation so as to change their momentum to reflect a collision (recall impulses are related to momentum).

Within the time adaptive system described in this thesis, we actually use an implementation of *Lemke's* algorithm for solving LCPs. This is a very popular simplex based pivoting algorithm not unlike the simplex algorithms used for solving *Linear Programming* problems. We shall not discuss this algorithm here but refer the interested reader to Cottle [40] and Murty [114] for a thorough treatment on the algorithm.

One interesting note about LCPs and *Quadratic Programming* (QP) problems should also be mentioned here. If the matrix $\mathbf{A}$ in a LCP is positive semidefinite (i.e. for any vector $\mathbf{x}$, $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$), then the LCP problem is also equivalent to a QP problem. In the case of frictionless collisions between multiple bodies our matrix $\mathbf{A}$ is always positive semidefinite. As such it is possible to formulate the LCP problem into an equivalent QP problem and solve it in that form instead.

Bibliography

- [1] H.C. Andersen. Rattle: a velocity version of the shake algorithm for molecular dynamics calculations. *Journal of Computational Physics*, 52:24–24, 1983.
- [2] A. Aubel, R. Boulic, and D. Thalmann. Real-time display of virtual humans: Levels of detail and impostors. *IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on 3D Video Technology*, 10(2):207–217, 2000.
- [3] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *ACM SIGGRAPH*, 23(3):223–232, 1989.
- [4] D. Baraff. Curved surface and coherence for non-penetrating rigid body simulation. *ACM SIGGRAPH*, 24(4):19–28, 1990.
- [5] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *ACM SIGGRAPH*, 25(4):31–40, 1991.
- [6] D. Baraff. *Dynamic Simulation of Non-Penetrating Rigid Bodies*. PhD thesis, Cornell University, 1992.
- [7] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. *ACM SIGGRAPH*, pages 23–34, 1994.
- [8] D. Baraff. Linear-time dynamics using lagrange multipliers. *ACM SIGGRAPH*, pages 137–146, 1996.
- [9] D. Baraff and A. Witkin. Partitioned dynamics. Technical Report CMU-RI-TR-97-33, Robotics Institute, Carnegie Mellon University, 1997.
- [10] D. Baraff and A. Witkin. Large steps in cloth simulation. *ACM SIGGRAPH*, 32:43–54, 1998.

- [11] G. Barequet, B. Chazelle, L. J. Guibas, J. S. B. Mitchell, and A. Tal. BOXTREE: A hierarchical representation for surfaces in 3D. *Computer Graphics Forum*, 15(3):387–396, 1996.
- [12] R. Barzel. Faking dynamics of ropes and springs. *IEEE Computer Graphics and Applications*, 17(3):31–39, 1997.
- [13] R. Barzel and A. H. Barr. A modelling system based on dynamic constraints. *ACM SIGGRAPH*, 22(4):179–188, 1988.
- [14] R. Barzel, J. F. Hughes, and D. N. Wood. Plausible motion simulation for computer graphics animation. *Computer Animation and Simulation '96, Proceedings of Eurographics CAS workshop*, pages 183–197, 1996.
- [15] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanic and Engineering* 1, pages 1–16, 1972.
- [16] B. Berenbrug. *Designing a Class Library for Interactive Simulation of Rigid Body Dynamics*. PhD thesis, Eindhoven University of Technology, 2000.
- [17] G. Van Den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools: JGT*, 2(4):1–14, 1997.
- [18] G. Van Den Bergen. *Collision Detection in Interactive 3D Computer Animation*. PhD thesis, Eindhoven University of Technology, 1999.
- [19] F. Bertails, T-Y. Kim, M-P. Cani, and U. Neumann. Adaptive wisp tree - a multiresolution control structure for simulating dynamic clustering in hair motion. *Eurographics/SIGGRAPH Symposium on Computer Animation '03*, pages 207–213, 2003.
- [20] J. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1):3–9, 1979.
- [21] G. Bradshaw. *Bounding Volume Hierarchies for Level-of-Detail Collision Handling*. PhD thesis, Trinity College Dublin, 2002.
- [22] G. Bradshaw and C. O'Sullivan. Sphere-tree construction using dynamic medial-axis approximation. *ACM SIGGRAPH Symposium on Computer Animation*, pages 33–40, 2002.

- [23] G. Bradshaw and C. O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics*, 23(1):1–26, 2004.
- [24] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM SIGGRAPH*, 21:594–603, 2002.
- [25] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. *Symposium on Computer Animation, ACM SIGGRAPH/Eurographics Symposium Proceedings*, pages 28–36, 2003.
- [26] S. Cameron. A study of the clash detection problem in robotics. *Int. Conf. Robotics and Automation*, pages 488–493, 1985.
- [27] S. Cameron. Efficient intersection tests for objects defined constructively. *International Journal of Robotics Research*, 8(1):3–25, 1989.
- [28] S. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automation*, 6(3):291–302, 1990.
- [29] S. Cameron. Enhancing GJK: computing minimum penetration distances between convex polyhedra. *In Proceedings of the International Conference on Robotics and Automation*, pages 3112–3117, 1997.
- [30] D. A. Carlson and J. K. Hodgins. Simulation levels of detail for real-time animation. *Graphics Interface '97*, pages 1–8, 1997.
- [31] J.E. Chambers and M.A. Murison. Pseudo-high-order symplectic integrators. *Astronomical Journal*, 119:425–433, 2000.
- [32] S. Chenney. *Controllable and Scalable Simulation for Animation*. PhD thesis, University of California, Berkeley, 2000.
- [33] S. Chenney, Okan Arıkan, and D. Forsyth. Proxy simulations for efficient dynamics. *Eurographics '01 Short Presentations*, 2001.
- [34] S. Chenney and D. Forsyth. View-dependent culling of dynamic systems in virtual environments. *ACM Symposium on Interactive 3D Graphics*, pages 55–58, 1997.
- [35] S. Chenney and D. Forsyth. Sampling plausible solutions to multi-body constraint problems. *ACM SIGGRAPH*, pages 219–228, 2000.

- [36] S. Chenney, J. Ichnowski, and D. Forsyth. Dynamics modelling and culling. *IEEE Computer Graphics and Applications*, 19(2):79–87, 1999.
- [37] J. Clement. Students' preconceptions in introductory mechanics. *American Journal of Physics*, 50(1):66–71, 1982.
- [38] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-collide: An interactive and exact collision detection system for large scale environments. In *Proceedings of ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.
- [39] T. Cornsweet. The staircase method in psychophysics. *American Journal of Psychology*, 75:485–491, 1963.
- [40] R. W. Cottle, J-S. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, Inc., 1992.
- [41] G. Debunne, M. Desbrun, A. Barr, and M-P. Cani. Interactive multiresolution animation of deformable models. *10th Eurographics Workshop on Computer Animation and Simulation (CAS99)*, 1999.
- [42] G. Debunne, M. Desbrun, M-P. Cani, and A. Barr. Adaptive simulation of soft bodies in real-time. *Computer Animation '00*, pages 133–144, 2000.
- [43] G. Debunne, M. Desbrun, M-P. Cani, and A. Barr. Dynamic real-time deformations using space and time adaptive sampling. *ACM SIGGRAPH*, pages 31–36, 2001.
- [44] M. Desbrun and M-P. Cani-Gascuel. Smoothed particles: A new approach for animating highly deformable bodies. *7th Eurographics Workshop on Animation and Simulation*, pages 61–76, 1996.
- [45] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects. *Graphics Interface*, pages 1–8, 1999.
- [46] E. A. Desloge. *Classical Mechanics*, volume 1. Wiley-Interscience, 1982.
- [47] J. Dingliana. *Adaptive Levels of Detail for Interactive Collision Handling*. PhD thesis, Trinity College Dublin, 2003.
- [48] J. Dingliana and C. O'Sullivan. Graceful degradation of collision handling for physically based animation. *Computer Graphics Forum*, 19(3):239–247, 2000.

- [49] P. Dworkin and D. Zeltzer. A new model for efficient dynamic simulation. *Computer Animation And Simulation, EGCAS '93*, pages 135–148, 1993.
- [50] D. H. Eberly. *Game Physics*. Morgan Kaufmann, 2004.
- [51] D. Fincham. Leapfrog rotational algorithms. *Molecular Simulation*, 8:165–178, 1992.
- [52] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *ACM SIGGRAPH*, 27:247–254, 1993.
- [53] J-D. Gascuel, M-P. Cani, M. Desbrun, E. Leroi, and C. Mirgon. Simulating landslides for natural disaster prevention. *8th Eurographics Workshop on Animation and Simulation*, 1998.
- [54] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [55] T. Giang, G. Bradshaw, and C. O'Sullivan. Complementarity based multiple point collision resolution. In *Proceedings of Eurographics Ireland '03*, pages 1–8, 2003.
- [56] T. Giang, R. Mooney, C. Peters, and C. O'Sullivan. ALOHA: adaptive level of detail for human animation: Towards a new framework. *Eurographics '00 Short Presentations*, pages 71–77, 2000.
- [57] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, 1988.
- [58] D. Gilden and D. Proffitt. Understanding collision dynamics. *Journal of Experimental Psychology: Human Perception and Performance*, 15(2):372–383, 1989.
- [59] A. Glassner. *An Introduction To Ray Tracing*. Morgan Kaufmann, 1989.
- [60] M. L. Gleicher. *A Differential Approach to Graphical Interaction*. PhD thesis, Carnegie Mellon University, 1994.
- [61] H. Goldstein. *Classical Mechanics*. Addison-Wesley, 1971.
- [62] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *ACM SIGGRAPH*, pages 171–180, 1996.

- [63] F. S. Grassia. Practical parameterization of rotations using the exponential map. *The Journal of Graphics Tools*, 3(3):29–48, 1998.
- [64] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor. H-COLLIDE: A framework for fast and accurate collision detection for haptic interaction. *In the Proceedings of IEEE Virtual Reality Conference*, pages 38–45, 1999.
- [65] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: fast neural network emulation and control of physics-based models. *ACM SIGGRAPH*, pages 9–20, 1998.
- [66] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM SIGGRAPH*, 22(3):871–878, 2003.
- [67] W.F. Van Gunsteren and H.J.C. Berendsen. Algorithms for macromolecular dynamics and constraint dynamics. *Molecular Physics*, 34:1311–1327, 1977.
- [68] J. K. Hahn. Realistic animation of rigid bodies. *ACM SIGGRAPH*, 22:299–308, 1988.
- [69] M. Hauth, O. Etzmuss, B. Eberhardt, R. Klein, R. Sarlette, M. Sattler, K. Daubert, and J. Kautz. Cloth animation and rendering. *Eurographics Tutorial T3*, 2002.
- [70] T. He. Fast collision detection using QuOSPO trees. *In Proceedings Symposium on Interactive 3D graphics*, pages 55–62, 1999.
- [71] T. He and A. Kaufman. Collision detection for volumetric objects. *In Proceedings 8th IEEE Visualization '97 Conference*, 1:27–35, 1997.
- [72] C. Hecker. Physics, part 4: The third dimension. *Game Developer Magazine*, pages 15–27, June 1997.
- [73] L. J. Hettinger and G. E. Riccio. Visually induced motion sickness in virtual environments. *Presence*, 1(3):306–310, 1992.
- [74] J. Hodgins, J. O'Brien, and J. Tumblin. Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):307–316, 1998.
- [75] R. Holloway. Viper: A quasi real-time virtualenvironment application. Technical Report TR92-004, Department of Computer Science, The University of North Carolina, 1992.

- [76] P. Hubbard. *Collision Detection for Interactive Graphics Applications*. PhD thesis, Brown University, 1995.
- [77] P. Hubbard. Real-time collision detection and time-critical computing. In *Proceedings of the 1st Workshop on Simulation and Interaction in Virtual Environments*, pages 92–96, 1995.
- [78] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.
- [79] D. Hutchinson, M. Preston, and T. Hewitt. Adaptive refinement for mass/spring simulation. *7th Eurographics Workshop on Animation and Simulation*, pages 31–45, 1996.
- [80] T. Jakobsen. Advanced character physics. In *Game Developer's Conference*, 2001.
- [81] D. R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, (3):404–425, 1985.
- [82] P. Jiménez, F. Thomas, and C. Torras. 3D collision detection: A survey. *Computers and Graphics*, 25(2):269–285, 2001.
- [83] M. P. Johnson. *Exploiting Quaternions To Support Expressive Interactive Character Motion*. PhD thesis, Massachusetts Institute of Technology, 2003.
- [84] Z. Kacic-Alesic, M. Nordenstam, and D. Bullock. A practical dynamics system. *Symposium on Computer Graphics*, pages 7–16, 2003.
- [85] M. Kaiser and D. R. Proffitt. Observers' sensitivity to dynamic anomalies in collisions. *Perception and Psychophysics*, 42(3):275–280, 1987.
- [86] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. *ACM SIGGRAPH*, 20:269–278, 1986.
- [87] Y. Kitamura, H. Takemura, N. Ahuja, and F. Kishino. Efficient collision detection among objects in arbitrary motion using multiple shape representations. In *Proceedings 12th IAPR Int. Conf. On Pattern Recognition*, 1:390–396, 1994.
- [88] J. Klein and G. Zachmann. ADB-trees: Controlling the error of time-critical collision detection. *8th International Fall Workshop Vision, Modeling, and Visualization (VMV)*, pages 37–45, 2003.

- [89] J. Klein and G. Zachmann. Time-critical collision detection using an average-case approach. *Proc. ACM Symp. on Virtual Reality Software and Technology (VRST)*, pages 22–31, 2003.
- [90] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.
- [91] A. Kol, B. Laird, and B. Leimkuhler. A symplectic method for rigid body molecular simulation. Technical Report 1997/NA5, DAMTP, Silver St., Cambridge CB3 9EW, UK, March 1997.
- [92] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using ShellTrees. *In Proceedings of Eurographics '98*, 17(3):315–326, 1998.
- [93] S. Krishnan, A. Pattekar, M. Lin, and D. Manocha. Spherical Shells: a higher order bounding volume for fast proximity queries. *In Proceedings Workshop on the Algorithmic Foundations of Robotics*, pages 122–136, 1998.
- [94] J.D. Lambert. *Numerical Methods for Ordinary Differential Systems : The Initial Value Problem*. Wiley, 1991.
- [95] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Dept. of Computer Science, University of North Carolina, 1999.
- [96] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast distance queries using rectangular swept sphere volumes. *In Proceedings IEEE International Conference on Robotics and Automation*, 2000.
- [97] B. Leimkuhler, S. Reich, and R.D. Skeel. Integration methods for molecular dynamics. *IMA Volumes in Mathematics and its Applications*, 82:161–186, 1996.
- [98] A. Leslie and S. Keeble. Do six-month-old infants perceive causality. *Cognition*, 25:265–288, 1987.
- [99] H. Levitt. Transformed up-down methods in psychoacoustics. *Journal of Acoustical Society of America*, 49:467–477, 1971.

- [100] M. C. Lin. *Efficient Collision Detection for Animation and Robotics*. PhD thesis, University of California at Berkeley, 1993.
- [101] M. C. Lin and S. Gottschalk. Collision detection between geometric models: A survey. *In Proceedings IMA Conference on Mathematics of Surfaces*, pages 33–52, 1998.
- [102] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2003.
- [103] J. Mezger, S. Kimmerle, and O. Eitzmuß. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG*, 11(2):322–329, 2003.
- [104] A. Michotte. *The Perception of Causality*. Basic Books, New York, 1963.
- [105] V. J. Milenkovic and H. Schmidl. Optimization-based animation. *ACM SIGGRAPH*, pages 37–46, 2001.
- [106] B. Mirtich. Hybrid simulation: combining constraints and impulses. *Proceedings of First Workshop on Simulation and Interaction in Virtual Environments*, 1995.
- [107] B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.
- [108] B. Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California at Berkeley, 1996.
- [109] B. Mirtich. Rigid body contact: Collision detection to force computation. Technical Report TR98-01, Mitsubishi Electronics Research Lab, 1998.
- [110] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, 1998.
- [111] B. Mirtich. Timewarp rigid body simulation. *ACM SIGGRAPH*, pages 193–200, 2000.
- [112] B. Mirtich and J. Canny. Impulse-based simulation of rigid bodies. *Symposium on Interactive 3D Graphics*, pages 181–188, 1995.
- [113] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *ACM SIGGRAPH*, 22(4):289–298, 1988.
- [114] K. G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Helderman-Verlag, 1988.

- [115] D. O'Brien, S. Fisher, and M. Lin. Automatic simplification of particle system dynamics. In *Proceedings of Computer Animation '01*, pages 210–218, 2001.
- [116] I.P. Omelyan. Numerical integration of the equations of motion for rigid polyatomics: The matrix method. *Comput. Phys. Commun.*, 109:171–183, 1998.
- [117] I.P. Omelyan. A new approach to the integration of rotational motion in systems with interacting rigid bodies. *Journal of Computational Physics*, 1999.
- [118] C. O'Sullivan. *Perceptually-Adaptive Collision Detection for Real-time Computer Animation*. PhD thesis, Trinity College Dublin, 1999.
- [119] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees. In *Proceedings Spring Conference in Computer Graphics*, pages 83–92, 1999.
- [120] C. O'Sullivan and J. Dingliana. Collisions and perception. *ACM Transactions on Graphics*, 20(3):151–168, 2001.
- [121] C. O'Sullivan, J. Dingliana, T. Giang, and M.K. Kaiser. Evaluating the visual fidelity of physically based animations. *ACM SIGGRAPH*, 22:527–536, 2003.
- [122] C. O'Sullivan and R. Reilly. React: Real-time adaptive collision testing, an interactive vision approach. in *Proceedings of the Eighth International Eurographics Workshop on Computer Animation and Simulation, EGCAS'97, September 2-3*, pages 163–175, 1997.
- [123] I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, 1995.
- [124] M. K. Ponamgi, D. Manocha, and M. C. Lin. Incremental algorithms for collision detection between polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):51–64, 1997.
- [125] J. Popovic, S. M. Seitz, M. Erdmann, Z. Popovic, and Witkin A. Interactive manipulation of rigid body simulations. *ACM SIGGRAPH*, pages 209–217, 2000.
- [126] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

- [127] D. Proffitt and D. Gilden. Understanding natural dynamics. *Journal of Experimental Psychology: Human Perception and Performance*, 15(2):384–393, 1989.
- [128] X. Provot. *Animation Réaliste de Vêtements*. PhD thesis, Synthim-INRIA, 1997.
- [129] S. Quinlan. Efficient distance computation between non-convex objects. In *Proceedings International Conference on Robotics and Automation*, pages 3324–3329, 1994.
- [130] R. Rabbitz. Fast collision detection of moving convex polyhedra. *Graphics Gems IV*, P. S. Heckbert Ed., Academic Press, Cambridge, MA, pages 83–109, 1994.
- [131] M. Reddy. *Perceptually Modulated Level of Detail for Virtual Environments*. PhD thesis, University of Edinburgh, 1997.
- [132] S. Redon, A. Kheddar, and S. Coquillart. CONTACT: arbitrary in-between motions for continuous collision detection. In *Proceedings of IEEE Workshop on Robot-Human Interaction, ROMAN'01*, Sept. 2001.
- [133] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. In *Eurographics*, 2002.
- [134] S. Redon, A. Kheddar, and S. Coquillart. Hierarchical back-face culling for collision detection. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.
- [135] H. Sammet and R. Webber. Hierarchical data structures and algorithms for computer graphics. *IEEE Computer Graphics and Applications*, 8(3):48–68, 1988.
- [136] A. Smith, Y. Kitamura, H. Takemura, and F. Kishino. A simple and efficient method for accurate collision among deformable polyhedral objects in arbitrary motion. In *Proceedings 12th IAPR Int. Conf. On Pattern Recognition*, 1:136–145, 1995.
- [137] A. Spiegl. Physical modelling of non-rigid objects and interactive display using bsp trees. Master's thesis, University of Texas at Austin, 1993.
- [138] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *ACM SIGGRAPH*, 21(4):205–214, 1987.
- [139] R. Turner and E. Gobbetti. Interactive construction and animation of layered elastically deformable characters. *Computer Graphics Forum*, 17(2):135–152, 1998.

- [140] K. Uliano, R. Kennedy, and E. Lambert. Asynchronous visual delays and the development of simulator sickness. In *Proceedings of Human Factors Society, 30th Annual Meeting (1986)*, pages 422–426, 1986.
- [141] L. Verlet. Computer experiments on classical fluids, thermodynamical properties of lennard-jones molecules. *Physical Review*, 159:98–103, 1967.
- [142] P. Volino, M. Courchesne, and N. Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *ACM SIGGRAPH*, 29:137–144, 1995.
- [143] A. Wilson, E. Larsen, D. Manocha, and M. C. Lin. Impact: A system for interactive proximity queries on massive models. Technical Report TR98-031, Dept. of Computer Science, University of North Carolina, 1998.
- [144] A. Witkin, D. Baraff, and M. Kass. Physically based modelling. *ACM SIGGRAPH Course Notes*, 2001.
- [145] C. Woolley, D. Luebke, B. Watson, and A. Dayal. Interruptible rendering. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 143–151, 2003.
- [146] G. Zachmann. The Bboxtree: Exact and fast collision detection of arbitrary polyhedra. *SIVE95 (First Workshop on Simulation and Interaction in Virtual Environments)*, 1995.