Perceptually-Adaptive Collision Detection for Real-time Computer Animation

Carol O'Sullivan

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
University of Dublin, Trinity College
Department of Computer Science
June 14th, 1999

Declaration

I declare that the work described in this thesis has not been submitted for a degree at any other university, and that the work is entirely my own.

Signature

_____

Carol O'Sullivan,
June 14th, 1999

Permission to lend and/or copy

I agree that the library in Trinity College may lend or copy this thesis upon request.

Signature

_____

Carol O'Sullivan,
June 14th, 1999

Acknowledgements

Thanks to my supervisor Dr. Steven Collins for his advice and guidance, and for his prompt reading of the final manuscript. Many thanks also to Dr. Ralph Radach from the Institute of Psychology in the Technical University of Aachen for all his help with respect to the psychophysical aspects of this work.

I am also grateful for the support of Professor John Byrne, Head of the Computer Science department in Trinity College Dublin and of Dr. Michael Sherwood-Smith in the Department of Computer Science in University College Dublin. Thanks also to Dr. Ronan Reilly from University College Dublin for helping to formulate some of the ideas of this thesis, to Dr. Shane O'Mara from the department of psychology in Trinity College Dublin, and to my post-graduate student John Dingliana for his co-operation. This work has been funded in part by a basic research grant from Enterprise Ireland.

Finally, thanks to my family for their support: Tomás and Nora, Ingrid and Jürgen, my husband Michael, and my children Denis, Rachel, and Oscar.

Abstract

Perceptually-Adaptive Collision Detection for Real-time Computer Animation
Carol O'Sullivan
Supervisor: Dr. Steven Collins

The aim of interactive animation systems is to create an exciting and real experience for viewers, to give them a feeling of immersion, of “being there". The tendency in the past has been to attempt to achieve this by matching as closely as possible the physics of the real world, with varying degrees of success. However, it is the human visual system that receives and interprets the visual cues from the surrounding environment, and it ultimately determines what we perceive. Therefore, we must look beyond the laws of physics to find the secret of reproducing visual reality.

In interactive animation applications such as VR or games, it cannot be predicted in advance how a user or the entities in a virtual world will behave, so the animation must be created in real-time. There are many bottlenecks in such systems, collision detection being a major one. A trade-off between detection accuracy and speed is necessary to achieve a high and constant frame-rate. However, it is possible to reduce perceived inaccuracy by taking perceptual factors into account, and also by estimating where on the screen a viewer is looking, possibly using an eye-tracking device, or by attaching more importance to certain objects in a scene, or to regions of the screen.

In this thesis we present the first perceptually-adaptive collision detection algorithm. New collision scheduling strategies are also presented and evaluated, along with a new interruptible algorithm to test for the intersection between two sphere trees. A model of human visual perception of collisions is developed, based on two-dimensional measures of **eccentricity** and **separation.** The model is validated by performing psychophysical experiments, in the first study of its kind. We demonstrate the feasibility of using this model as the basis for perceptual scheduling of interruptible collision detection in a real-time animation of large numbers of homogeneous objects. The user's point of fixation may be either tracked or estimated. By using a priority queue scheduling algorithm, perceived collision inaccuracy was significantly improved. The ideas presented here are applicable to other tasks where the processing of fine detail leads to a computational bottleneck.

Contents

List of Figures

1 Introduction

1.1 Introduction

The demand for highly interactive computer systems is increasing rapidly, as people wish to communicate with computers in a more natural fashion. This need for interaction adds to the pressure on developers of systems to produce realistic, real-time animations. However, this high degree of interaction with the user may also be exploited, in order to adaptively improve the realism of real-time animations. In order to maintain a user's immersion in the animation, a high and constant frame rate is necessary. Realistic rendering, motion synthesis, and collision handling all place an impossibly high computational load on graphics workstations. In a scene with many moving objects, maintaining the target frame rate, while rendering each frame to the highest level of image realism and controlling motion and interaction of objects to full accuracy, is virtually impossible with single-processor work-stations.

Figure 1.1: The wildebeest stampede from "The Lion King". ã Disney

Many interactive animation systems, such as games or simulations, require large numbers of virtual entities which are moving and interacting with each other, and/or with one or more users. In these applications, we cannot predict in advance how the user or the entities will behave, so we must create the animation as we watch it, i.e. in real-time. This means that the image must be re-drawn at least 10 times per second, although for true real-time performance the generation of up to 60 frames per second (f.p.s) may be required. Hence, there will be 100 milliseconds available, at the most, to update all entities in the simulation, and then render the new scene. There are many bottlenecks in such systems. Depending on the level of realism required, rendering and motion synthesis algorithms require a large amount of processing power. In multi-user systems, network lag is a major issue.

Some solutions to these problems could be to increase the computational power, add hardware accelerators, and develop parallel algorithms that may be implemented on multiple processors. However, using such approaches, the problem is postponed rather than eliminated, and such computational power will not be available to a wide range of users. An additional challenge is maintaining a constant frame rate. The time taken to render a given scene is dependent on the current level of complexity. Some frames may require only one object to be rendered, whereas a sudden change of view may cause many more interacting objects to be visible in subsequent frames. Obviously, the latter set of frames will take longer to process than the former, regardless of the computational power available. Hence, increasing computational power alone is not the solution. It may sometimes be necessary to trade realism and accuracy for speed, and aim to achieve optimal realism in the time available, thus maintaining a high and constant frame-rate.

1.1.1 Collision detection

In many interactive real-time animation systems, such as those described above, the entities often need to be viewed not as geometric shapes devoid of physical properties, but as real entities having properties such as mass, moment of inertia, elasticity, and friction. Their motions are constrained not only by their own physical properties, but also by collisions with other objects. If two solid objects collide in the real world, they bounce off each other, or break into pieces, and deform if their surfaces are non-rigid. In a computer world, geometrically modelled objects would simply float through each other.

A **Collision Handling** system is necessary to enforce solidness, and ensure that entities behave as expected when they come into contact, i.e. they should not interpenetrate, and their behaviour subsequent to collision should be compatible with their physical properties. This involves two very distinct phases: **Collision Detection**, and **Collision Response**. Detection is a problem of kinematics, while response is a problem of dynamics.

A possible application of real-time collision handling could be a game where a user must navigate his/her way through a rockfall without being hit. As multiple rocks are falling, they hit off each other and the edges of a ravine, either bouncing or breaking into smaller rocks. In order to achieve this effect, collisions between the rocks and with the ravine edges must be detected. Many other application areas exist: Large-scale Virtual Reality (VR) systems with thousands of moving entities; Crowd simulations; Educational simulations with chemical molecules or blood cells. An example could be the wildebeest stampede scene from the Disney film "The Lion King" (see Figure 1.1), where behavioural rules were used to simulate the behaviour of many objects, similar to the approach taken for flocking of birds (or Boids) in [Reynolds 1987]. The stampede scene was generated off-line using many hours of computer time. To achieve anything approaching such an effect in real-time, efficient collision handling algorithms are necessary.

In addition to increasing the speed and efficiency of collision handling algorithms, it is also essential to maintain a constant frame rate. Consider a room with a hundred very bouncy balls in it. As long as the balls are evenly distributed around the room, the number of collisions will be reasonable, with several small groups of two or slightly more balls coming into contact with each other every few milliseconds. However, eventually the balls all drift to one corner of the room. Now each ball in the room will be in contact with many other balls, each of which will also be colliding with many others. There are now very many possible combinations of entity-entity collisions to detect every few milliseconds, so the time to process collisions, and hence the frame rate, will increase dramatically. This could result in several frames that take 200, 300 or even more milliseconds to process until the balls again separate to a more even distribution. The resulting animation will be very jerky and unrealistic.

In some applications, such as scientific simulation, fully accurate physically-based collision handling is necessary. In these cases, solutions must be considered such as degrading the image realism to maintain physically-based response, or increasing the number of processors and developing parallel algorithms for tasks such as rendering and collision detection. However, in a growing number of applications, what is important is that viewers perceive events, such as collision response, to be accurate. Testing all potentially colliding objects at full resolution is not always necessary to achieve this goal. In fact, in some cases this testing may be fully redundant, for example, if the viewer is not actually looking at the objects when they collide.

1.1.2 Visual Perception

What effects do properties such as size, velocity, colour, and semantics have on our perception of objects and their interactions? Could a combination of weights be given to these effects, in order to prioritise and schedule collision processing? What if it was known where exactly in the visible scene the viewer was looking? We shall see later in this thesis that it is a well-established fact that visual and spatial acuity falls off rapidly with increasing eccentricity of stimuli from the point of the eye's fixation. Is this also true for collisions? Would a viewer be less likely to notice a repulsion or interpenetration if it happened on a part of the screen at which they were not directly looking, or how close to the point of fixation, and how significant must the anomaly be for it to be noticed? These considerations begin to reveal the complexity of the problem, and lead to the following two conclusions:

- A study of human visual perception could yield important information about how viewers perceive collisions, and hence enable a prioritisation of potential collisions to process within a given frame of an animation.

- Estimating the position on-screen where a viewer is looking could be very useful for real-time collision handling. One promising method of achieving this it by using an eye-tracking device, or alternatively to predict the most likely position using scene information.

Traditionally, the use of eye-trackers has been confined to medical and scientific research. These types of trackers are very accurate, but also very invasive, involving the use of head restraints, bite bars, and even scleral coils which are inserted into the eye in a contact lens. This technology is too intrusive and immobile for use in an interactive graphical system (see Figure 1.2). In order to convince the computer graphics community that this form of technology is feasible, the eye-tracking technology must be much more usable. There are, however, some mobile, non-intrusive, and low-cost trackers being developed, and we have experimented with one such model from Vision Control Systems™ (see Figure 1.3). Of course, the drawback at present with such systems is a loss of accuracy, both spatial and temporal, and we found that this eye-tracker was infeasible for use in this research for these reasons. However, once the use of such trackers in areas other than medical and scientific becomes established, the technology will improve dramatically.

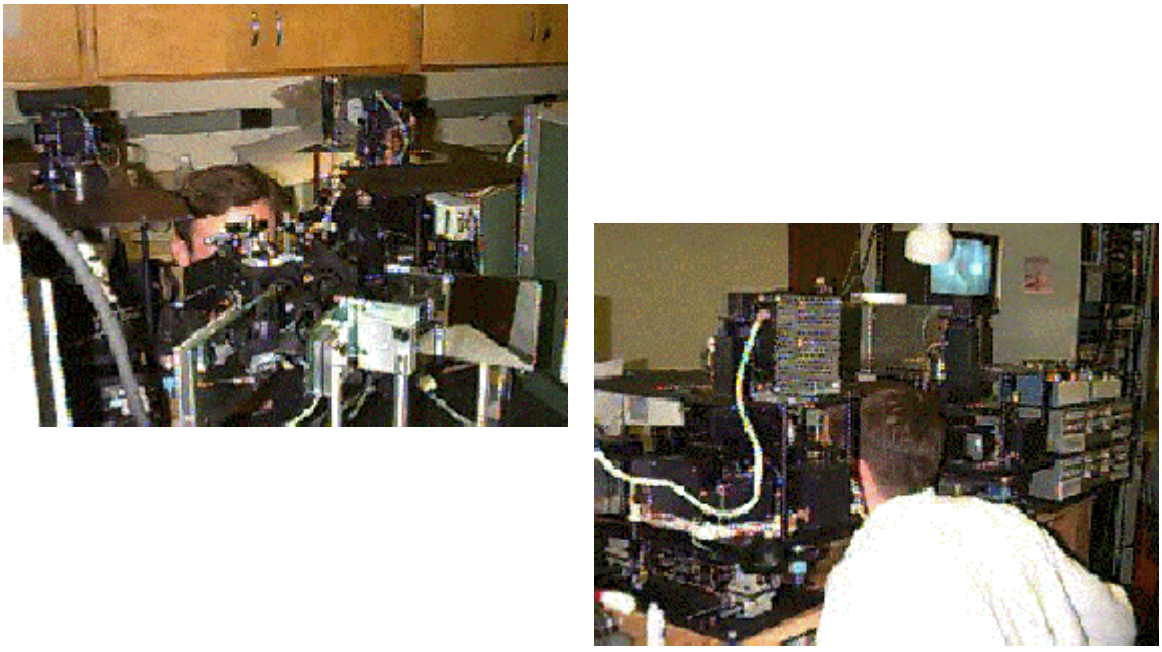


Figure 1.2: A highly accurate but intrusive eye-tracker [1]

[1] Images downloaded from http://hering.berkeley.edu/zbackups/facphotos/SRI.html. The eye-tracker used is the SRI Dual-Purkinje Eye-Tracker™, which measures eye position according to the reflections of infrared light off the eye.

Figure 1.3: The low-cost, non-intrusive eye-tracker from Vision Control Systems

1.2 Objectives and Scope

In this thesis, we address the problem of real-time collision detection between large numbers of visually homogeneous (i.e. similar) objects. The following options are available at present:

- Perform fully accurate collision detection: This ensures the realism of collision handling if enough time is available, but causes frame-rate problems for increasing numbers of objects

- Degrade of detection accuracy for speed: This option may deliver a high and constant frame rate, but the degradation of accuracy can have un-predictable results, and the animation may look very unrealistic.

We propose a new algorithm for real-time collision detection, which degrades accuracy for speed, but reduces the perceived inaccuracy of the animation through the use of a perceptual model. We use the results of existing psychological research and also perform our own psychophysical experiments to develop and validate this model.

1.3 Organisation of the thesis

Chapter 2, Previous Work: This chapter reviews the current state of the art in collision detection, and justifies further research in this area.

Chapter 3, Visual Perception: This chapter starts by reviewing the precedents for perceptually-driven techniques in the wider field of computer graphics. To provide an understanding for later discussion, the basics of visual perception are explained from various perspectives. Finally, a detailed discussion of the factors which affect the perception of collisions is provided.

Chapter 4, The Application: A real-time animation system was developed to test the ideas presented in this thesis. The design of this system is explained in this chapter, as are the collision detection algorithms developed. Our new scheduling strategies are also presented, along with new metrics which can be used to measure total and perceived collision inaccuracy. Models of human visual perception of collisions are developed.

Chapter 5, Analysis and Performance: A comprehensive suite of tests were carried out to test the performance of our collision detection algorithms, both in terms of efficiency and perception, under a range of conditions. The perceptual models described in the previous chapter were used to schedule collisions, and the metrics were used to measure the inaccuracy introduced by our new algorithms. The results of these tests are presented here.

Chapter 6, Psychophysical Experiments: In this chapter we describe the psychophysical experiments that were conducted on human subjects to determine the validity of the model of collision perception developed in Chapter 4, and used to schedule collisions in Chapter 5.

Chapter 7, Conclusions and Future Work: Finally, the ideas of this thesis are summarised in, along with ongoing extensions and suggestions for future research directions. This work is not just relevant to the problem of collision detection. Many, if not most, of the ideas may be applied to other bottlenecks in the wider field of computer graphics. Some of these other applications are also discussed.

1.4 Contributions of the Thesis

- We present ***the first perceptually-adaptive collision detection algorithm***.

- Our approach has been validated through ***psychophysical experiments***, which investigate the sensitivity of humans to degradations in collision accuracy. To our knowledge, no such psychophysical experiments have ever been conducted before.

- ***New collision scheduling strategies*** are also presented and evaluated. Again, we are unaware of any other research which implements and compares different methods of ordering collision processing.

- A ***new interruptible algorithm*** to test for the intersection between two sphere trees has been developed, which integrates the best properties of two existing algorithms

- We have also contributed to the fields of Human Computer Interaction, and eye-movement research, by proposing a potential ***new application of eye-tracking technology and eye-movement analysis*** for interactive computer systems.

2 Previous Work

In this thesis, we are proposing the hypothesis that adaptive techniques, coupled with a model of visual perception of collisions, may be used to enhance the realism of collision handling in real-time applications of many homogeneous objects. We also maintain that this approach may be extended to other areas where the processing of fine detail leads to a computational bottleneck. Before we present and validate our solutions, we must first examine the current state of the art in collision detection. This chapter first discusses the applications of collision detection in Section 2.1. In Section 2.2 the methods currently used for the various phases of collision detection algorithms are presented. In Section 2.3, the application of adaptive refinement to this problem is examined, and alternative solutions are considered in Section 2.4.

2.1 Applications of Collision Detection: Past, Present and Future.

The original motivation for Collision Detection algorithms arose in areas such as CAD and Robotics, where the desire was to do more work on the computer, and to off-load work from the human designer or planner. Problems such as handling many different CAD shape descriptions, VLSI layout, robot path planning, bin packing, and assembly planning gave rise to off-line algorithms, where objects positions and motions were fully predictable over time, and “what if” analysis was done to generate an optimum solution. Real-time performance was not an issue in such systems, and fully accurate mathematical intersection tests could be utilized.

As time progressed, the desire for realtime and interactive systems increased. For example, CAD designers want to see the results of their new layout immediately, not several minutes later. As they reposition an object, they want to try it out in several locations, and the collision tests with other objects must be performed while they are doing this, in real-time. In such a case, it is not possible to predict in advance what will happen. This is totally at the control of the human designer, (who is by nature non-deterministic).

Robotics applications in the past were typically characterized by static scenes, where one or more robotic devices performed pre-specified tasks. Therefore, pre-processing could be used to predict where collisions would happen. However, with the development of more autonomous robots, who can operate in unknown environments, and whose behaviour is impossible to predict in advance, this situation has also changed. The goal in robotics is no longer just efficient path planning, but to support a real-time safety system that warns of imminent collisions, such as proposed by [Shaffer and Herb 1992]. Most of the robotics research has also been into the more complex *Collision Avoidance* problem. However, as most collision avoidance systems incorporate collision detection by necessity, this research is also very relevant in the field of computer graphics, and much cross-over research has occurred between the fields. For example, the collision detection algorithm proposed in [Lin and Canny 1991] and [Lin 1993] has been used extensively in both robotics and animation.

The range of applications which require collision detection is extensive. Vehicle simulators are one case in point, where the users manipulate a steering device, and attempt to avoid obstacles in their way. In molecular modeling, simulations allow interactive testing of new drugs to examine how molecules interact and collide with each other. Training and education systems that realistically model the movement of objects within the geometric constraints of their layout, allow designers to experiment interactively with different strategies, e.g. to assemble or disassemble equipment, to perform a virtual surgery, or to test different paths that a robot could take. Such simulations are a safe and cheap way to teach.

Human character animation is one of the most challenging topics in computer animation, and collision detection is an important issue here also. As a figure moves, collisions must be detected between the virtual person and its environment, its clothes and hair, and self-collisions between limbs and digits. Haptic interfaces are devices that allow humans to interact manually with virtual environments, and to actually feel a sense of touch via these devices as if they were really touching the virtual objects. It is necessary to detect collisions between the skin of the real person and virtual objects, as in [Singh et al. 1995]. Virtual actors may be added to real scenes, as in [Thalmann and Thalmann 1995]. Here, collisions must be detected between a virtual actor and a real scene.

Virtual Reality (VR) applications allow users to enter a computer-generated virtual world and interact with graphical objects and virtual agents with a sense of reality. Such systems may be either immersive, or desktop based. One thing they have in common is a requirement for extremely high and constant frame-rates. Physical interactions such as touching, hitting and throwing are usually triggered by collision. The more objects in the environment, and the more complex these objects are, the higher the burden on the engine which powers the animation, and hence the greater the need for extremely rapid collision detection. Increasing performance of V.R. is often driven by hardware developments, as in [Rohlf and Helman 1994], but significant opportunities exist to increase performance via algorithmic improvement.

We have seen that there is a requirement for collision detection in almost all systems that animate graphical objects, or which need to determine potential collisions with real objects in advance. This poses a significant challenge in the area of real-time systems, and as the demand for more realism and interaction in such systems is constantly increasing, it is also likely to remain a challenge for quite some time into the future.

2.2 Hybrid Collision Detection

When animating more than two objects, the most obvious problem which arises is the $O(N^2)$ problem of detecting collisions between all N objects. This is known as the all-pairs problem. It is obvious that this is an undesirable property of any collision detection algorithm, and several techniques have been proposed to deal with it. Hybrid collision detection, a term coined by [Kitamura et al. 1994], refers to any collision detection method which first performs one or more iterations of approximate tests to identify interfering objects in the entire workspace and then performs a more accurate test to identify the object parts causing interference. [Hubbard 1995] and [Cohen 1995] also propose hybrid algorithms for collision detection. The former refers to the two levels of the algorithm as the ***broad phase***, where approximate intersections are detected, and the ***narrow phase***, where exact collision detection is performed. Such an approach is essential for acceptable collision detection performance. The narrow phase itself may also consist of several levels of intersection testing between two objects at increasing levels of accuracy, the last of which may be fully accurate. We shall refer to these as the ***exact level***, and the ***progressive refinement levels***.

2.2.1 Narrow Phase: The Exact Level

Any collision detection algorithm depends on the technique used to model the objects, and the data structure used to represent that model. The narrow phase, where exact collision detection is performed, depends greatly on the object representation scheme used.

Polygonal representations of surfaces are widely used to represent surfaces in 3D graphics. Surfaces, which are intrinsically planar, such as those on building exteriors and cabinets, can be easily and naturally represented in this way. However, virtually any surface can be represented by polygons if the number of polygons is sufficiently high. This leads, however, to an approximate representation only. In order to reduce the faceted effect of these surfaces, the number of polygons has to be increased to such an extent that affects space requirements and computation time of algorithms processing the surface. Many different special cases must be handled, and cases may occur where polygon edges “tunnel” through each other, or smaller surfaces pass through an entire polygon. A major advantage to using polygonal representations of objects is the fact that many manufacturers provide specialized acceleration hardware to implement common operations on polygons, such as clipping and shading.

Polyhedral methods are not well suited to surfaces that deform in time, and which roll or slide against each other. In such cases *High-level surface representations* are more desirable. One such representation is a collection of *Parametric Patches*, which are regions on a curved surface bounded by parametric curves. The number of parametric patches needed to approximate a curved surface to a reasonable degree of accuracy is many fewer than the number of polygonal patches that would approximate it to the same level. *Implicit Surfaces* are defined using implicit functions, and also have some desirable properties, such as being closed manifolds. This means that they define a complete solid model, not just the surface as in the case of parametric surfaces and polygonal models. Collision detection algorithms have been developed to process collisions between objects modeled via these techniques [Von-Herzen et al. 1990] [Snyder et al. 1993][Shene and Johnstone 1991]. For a more extensive survey of collision detection techniques between a variety of geometric models, see [Lin and Gottschalk 1998].

The drawback of these higher-level solutions is that the algorithms for processing them are usually more complex, and hence slower. This is another important reason that polygonal representations are prevalent. They are most useful for modelling objects which are deformable, such as cloth garments and blobby, jelly-like substances which decompose and split over time. We do not address the problem of collisions between such objects in this thesis. We are concerned with collisions between rigid objects, which retain their shape over time. Such objects are usually modelled using the polygonal representations mentioned above, and hence there has been much activity in the development of efficient collision detection algorithms between these models.

Polygonal models can be either convex, or non-convex. If a polygon is convex, that means that the line between any two points inside the polygon must also lie completely inside (see figure 2.1). The concept extends to three dimensions, where a surface composed of polygons is called a polytope. If the polytope is convex, then any line between two points inside this area must lie completely within the surface defined by such a polytope. Most of the work on collision detection techniques has concentrated on detecting collisions between convex polytopes. Such approaches fall into two broad categories: *Feature-based methods*, and *Simplex-based methods*.

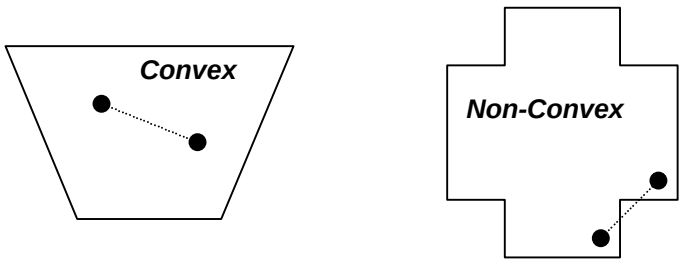


Figure 2.1: A convex and a non-convex polygon

Feature-based methods concentrate on the inter-relations between the vertices, edges and faces of two polytopes, i.e. their features. The main goal of such algorithms is to detect whether two polytopes are touching or not. The most significant of these algorithms is the *Lin-Canny* algorithm mentioned in Section 2.1 [Lin and Canny 1991] [Lin 1993] which has been regarded until recently as the fastest solution to such problems.

The Lin-Canny algorithm determines whether two objects are disjoint or not, by computing the distance between their closest features. It tracks these features, and caches them between subsequent calls to the algorithm. In this way it exploits coherence, because the closest features will not change significantly between successive frames, and “feature-stepping” is used to keep the closest features up to date, i.e. if the closest features have changed, they are going to be adjacent to the those cached, and hence finding them is quite efficient. The algorithm to track these features runs in expected constant time if the collision detection time-step (i.e. the steps which the animation takes before each iteration of the collision detection algorithm) is small relative to the speed at which the objects are moving. The algorithm has been built into a general collision detection package I-Collide, described in [Cohen et al. 1995] and [Ponamgi et al. 1997] which is freely available on the World Wide Web. This has contributed to the popularity of their algorithm.

The Lin-Canny algorithm does not handle penetrating polytopes, however, and if such a condition arises the algorithm enters an infinite loop. A quick-fix to this problem is to force termination after a maximum iteration, and return a simple result stating that the objects have collided. However, this solution is quite slow, and no measure of inter-penetration is provided. This is a significant weakness, because the condition of inter-penetrating objects will occur very frequently unless the objects are moving quite slowly, and/or if the detection time-step is quite small. This is unrepresentative of most situations in which real-time collision detection is necessary. If inter-penetration occurs, and more information is needed about the exact time of contact, back-tracking is necessary to pin-point the exact instant in time when collision occurred. This is a slow and cumbersome process. The code is also very complex, with many special cases being handled separately (e.g. parallel features), and it is difficult to configure, with several numerical tolerances which need to be adjusted to achieve the desired performance.

[Mirtich 1998] presents the V-Clip (Voronoi-Clip) feature-based algorithm which has been inspired by the Lin-Canny algorithm, but claims to overcome the chief limitations of that algorithm. It handles the penetration case, needs no tolerances to be adjusted, exhibits no cycling behaviour, and is simpler to implement due to fewer special-case considerations.

Simplex-based algorithms have provided improvements on the Lin-Canny algorithm. A simplex is the generalisation of a triangle to arbitrary dimensions. The approach in these cases is to treat a polytope as the convex hull of a point set. Operations are then performed on simplices defined by subsets of these points. The first of such algorithms was presented in [Gilbert et al. 1988] and is commonly referred to as ***GJK***. The main strength of this algorithm is that, in addition to detecting whether two objects have collided or not, it can also return a measure of interpenetration. [Rabbitz 1994] improved upon GJK by exploiting coherence, and [Cameron 1997] developed it further to produce the algorithm which is known as ***Enhanced GJK***. This algorithm achieves the same almost-constant time complexity as Lin-Canny, while eliminating most of its main weaknesses. Mirtich claims that the V-Clip algorithm requires fewer floating-point operations than Enhanced GJK, and is hence more efficient, but it is also admitted that the GJK algorithms return the best measures of penetration.

The proponents of the above algorithms usually claim that extending their work to cater for non-convex polytopes is easily accomplished. Non-convex polytopes can be represented by hierarchies of convex components. A “pass the parcel” approach is recommended, with collision checking being performed between the convex hulls of successive subsets of convex components, which are then unwrapped when a collision is detected. Results are rarely presented for such operations, and the focus of the validation performed is mainly on the efficiency of the intersection tests between two convex objects. Mirtich admits that although this technique works well for slightly non-convex objects, it becomes very inefficient as the level of non-convexity increases. Therefore, these techniques are very useful for situations where a small number of convex, or slightly non-convex objects are interacting in real-time, but in other situations techniques based on hierarchical representations are much more suitable.

2.2.2 Narrow Phase: Progressive Refinement Levels

The progressive refinement levels of the narrow phase of a collision detection algorithm are often based on using bounding volumes and spatial decomposition techniques in a hierarchical manner. Hierarchical methods have the advantage that as a result of simple tests at a given point in the object hierarchies, branches below a particular node can be identified as irrelevant to the current search and so pruned from the search.

Trees of bounding volumes are used, each level approximating the object. This is a form of Level Of Detail (LOD) representation of the object. This differs from the polygonal levels of detail used in multiresolution methods for faster rendering of complex objects, or surfaces such as mountainous terrain [Hoppe 1996, 1997, 1998][Rossignac and Borrel 1993]. In such techniques, the aim is to render an approximation which is as visually similar to the original model as possible. LODs for collision detection are always conservative approximations to the object, and the choice of volume is usually based on the speed of their intersection tests. More recently emphasis has been placed on their ability to approximate the geometry of the bounded object. The following hierarchies have been used:

- ***AABB-trees*** [Von Den Bergen 1997]. Axis Aligned Bounding Boxes are used, the advantage of these being their ease of computation and overlap testing.
- ***Octrees*** [Sammet and Webber 1988][Kitamura et al. 1994] Octrees are built by recursively sub-dividing the volume containing an object into eight octants, and retaining only those octants which contain some part of the original object as nodes in the tree. Such a data structure is simple to produce automatically, and lends itself to efficient and elegant recursive algorithms. The disadvantage of this approach is that each level of the hierarchy does not fit the underlying object very tightly.
- ***Sphere Trees*** [Hubbard 1996][Palmer and Grimsdale 1995][Quinlan 1994]. The main advantages of using spheres are that they are rotationally invariant, making them very fast to update, and it is very simple to test for distances between them, and test for overlaps. The disadvantage is that spheres do not approximate certain types of objects very efficiently. Hubbard attempts to improve upon this by building first a medial axis surface, which is like a skeleton representation of an object, and then placing the spheres upon this to provide a tighter-fitting approximation to the object. [O'Rourke and Badler 1979] also developed a method of tightly fitting spheres to an object.
- ***C-trees*** consist of a mixture of convex polyhedra and spheres [Youn and Wohn 1993]. This has the advantage of choosing primitives which best approximate the enclosed object, but a major drawback is that the hierarchy must be created by hand, and cannot be produced automatically. A similar approach is taken in [Rohlf and Helman 1994].

- ***OBB-trees*** [Gottshalk et al. 1996]. These hierarchies consist of tightly-fitting Oriented Bounding Boxes. It is claimed that using an algorithm based on a separating axis, it can accurately detect all the contacts between large complex geometries at interactive rates. However, it is admitted that other methods are very good at performing fast rejection tests, and a disadvantage of OBB-trees over Sphere trees is that they are slower to update. A similar approach is taken in [Klosowski et al. 1997], who use ***hierarchies of k-DOPs***, or discrete orientation polytopes, which are convex polytopes whose facets are determined by half spaces whose outward normals come from a small fixed set of k orientations. Again, they implement it with a small number of highly complex objects, for the purposes of haptic force-feedback. If there are a large number of objects between which fast rejection or acceptance is needed, the update time needed for these approximations is likely to add an unacceptable additional burden.
- ***ShellTrees*** [Krishnan et al 1998, a, b]. These trees consist of oriented bounding boxes and spherical shells, which enclose curved surfaces such as Bezier patches and NURBS. They are particularly suited to collision detection between the higher-order surface representations discussed in the previous section.

2.2.3 Broad Phase Collision Detection

[Hubbard 1995] highlights three potential weaknesses of collision detection algorithms. The most serious of these is the ***all-pairs weakness*** discussed above, where every object in the scene must be compared with every other one at every collision timestep of the animation. Most research has concentrated on alleviating this problem. A second problem is what he calls the ***fixed-timestep weakness***. Allowing the objects to move larger distances before checking whether they intersect leads to a more efficient algorithm, but it is possible that some collisions will be missed, and objects will tunnel through each other. Decreasing the size of the time-step would reduce the chances of this happening, but would cause a lot of extra unnecessary intersection tests. The third weakness refers to the narrow phase, and is the ***pair-processing weakness***. This refers to the non-robust properties of algorithms such as Lin-Canny discussed in Section 2.2.1, which must handle many special cases and can exhibit strange behaviour such as cycling.

Hubbard recommends the use of an adaptive timestep, which becomes small when collisions are likely, and large when they are not. For the broad phase of his algorithm, 4-dimensional structures called space-time bounds are used, which provide a conservative estimate of where an object may be in the future. The fourth dimension represents time. Overlaps of these bounds trigger the narrow phase, which is based on hierarchies of spheres. Collision detection between sphere trees is robust, thus solving the pair-processing problem. Using the space-time bounds, attention is focused on the objects which are likely to collide, and those far away can be ignored, thus alleviating both the all-pairs weakness and the fixed-timestep weakness. [Cameron 1990] also addresses the fixed-timestep weakness through the use of four-dimensional bounding structures.

In [Cohen et al. 1995] multiple object pairs are “pruned” using bounding boxes. Overlapping bounding boxes then trigger the narrow phase of the algorithm. Their “Sweep and Prune” algorithm orthogonally projects axis-aligned bounding boxes of all objects onto the x, y and z-axes. This results in intervals, of which overlaps in all three dimensions indicate overlaps of the corresponding bounding boxes. Because of coherence, the relative positions of objects will not change significantly between frames, so insertion sort is used to keep the interval lists sorted, which runs in almost linear time for almost-sorted lists.

This algorithm is extremely desirable, because it handles the all-pairs weakness, and has small computational overheads. It does not tackle the fixed-timestep weakness, but runs in almost constant time for a given number of objects. This makes it preferential in situations where a constant frame rate is required in the presence of large numbers of interacting objects. The use of an adaptive timestep could be undesirable in these circumstances because processing would slow down when many objects were close to each other, due to the smaller size of the time-steps, and then speed up as they became more evenly distributed. This would give a non-constant frame-rate, and hence lead to a jerky animation. However, it may be feasible to use an adaptive time-step in conjunction with adaptive techniques for the narrow phase testing (and ideally for other operations also such as rendering). Efficient load-balancing mechanisms would be important in this case.

2.3 Adaptive Refinement

In Virtual Reality applications, in order to create an illusion of real-time exploration of a virtual world, an interactive frame rate of more than 10 frames per second (f.p.s) must be achieved, and this frame rate must also be constant. If the frame rate is too slow, or too jerky, the interactive feel of the system is destroyed. In a very complex environment, the objects may be modeled using thousands or even millions of polygons. To render these polygons completely accurately, with full hidden surface removal, shading, and collision detection is far beyond the capabilities of currently available workstations. A highly variable frame rate would be the result. One solution to this problem is to adjust image quality adaptively in order to maintain a uniform frame rate.

We have seen in the previous section that the use of an adaptive timestep is recommended in [Hubbard 1995]. However, he goes further than this and recommends adaptive refinement at the narrow-phase level also. The idea of *interruptible collision detection* is presented, which allows the accuracy of intersection tests to be progressively refined until a target time has elapsed. Collisions are detected between the sphere-trees of objects in round-robin order at increasing levels of detail, descending one level of all sphere trees at each iteration of the algorithm, until interruption occurs. This enables a fast, albeit approximate, answer when required.

As far as we can determine, this algorithm and the algorithms proposed in this thesis and its companion papers [O'Sullivan and Reilly 1997][O'Sullivan and Dingliana 1999][O'Sullivan et al. 1999][O'Sullivan and Radach 1999] are **the only existing algorithms for interruptible collision detection** presented to date. What is lacking in Hubbard's approach is that no attempt is made to prioritise collisions in order to reduce the visual impact of inaccuracies resulting from interruption, a weakness which this research remediates.

2.4 Other possibilities

We have shown that one solution to the bottleneck of collision detection is to keep the frame rates high by sacrificing simulation accuracy to achieve visual realism. However, perceived inaccuracy is not the only important form of accuracy in some applications.

If an application involves some pure forward simulation, then any geometric inaccuracy in any collision causes all subsequent parts of that simulation to be wrong. For example, a simulation may need to simulate some safety-critical situations, and perform a what-if analysis on various configurations of objects and their environment. In such situations it would be inappropriate to degrade accuracy. Thus, interruptible collision detection makes sense in many but not all applications. One solution in such situations is to degrade image quality while maintaining the integrity of the physical simulation. Another solution is to increase the computational power available. Modern graphics workstations have greatly improved performance, but even they cannot provide the high levels of realism and real-time performance required by some applications. More computational power is sometimes needed than can be provided by any one processor, so an obvious solution is to harness the power of several processors, and have them perform the task in parallel.

[Sillitoe et al. 1994] use a transputer-based architecture for the efficient parallel evaluation of clash-detection tests between modeled solids. [Swift et al. 1993] presents a parallel algorithm for generating octrees. [Borgolte et al. 1993] found that computational effort for on-line collision avoidance between a group of robots can be kept constant by using parallel processing hardware. [Hariyama et al. 1994] use a mutiprocessor constructed by several identical VLSI processors, in conjunction with a high-speed, large-capacity Content-Addressable Memory (CAM), to perform collision detection between a vehicle and obstacles at high speed. Obstacles are represented as a union of rectangular solids, and a vehicle is represented by a set of discrete points covering its surface. They found that if enough processors and CAM are available, any desired performance to any level of precision is possible.

[Tseng and Wu 1995] apply orthogonal neural networks to detect collisions between multiple robot manipulators that work in an overlapped space. They find that the property of parallel processing enables the network to detect collisions in real-time, and also that the computing time for collision detection is almost the same, regardless whether the collision is between two, three or more manipulators. [Chande et al. 1993] states the collision problem as that of points on one robot coming in close vicinity to the points on another robot, and proposes a neural network based methodology to avoid collisions in real-time. The network can also be implemented in VLSI.

A problem with the above approaches is that they only give a yes-no answer as to whether a collision has, or is about to happen. They do not produce accurate information about the points of contact, which is unimportant in robotics, but very necessary in graphical applications where collision response must be computed. However, [Yuan 1995] presents a feasibility study of using a neural network to detect a collision between two convex polyhedra and measure the degree of intersection if a collision occurs. It can also provide additional information on the best direction to separate two colliding polyhedra.

3 Visual Perception

3.1 Introduction

The aim of interactive animation systems is to create an exciting and real experience for viewers, to give them a feeling of immersion, of “being there”. A visual experience must be created for them that mimics the events that happen in their real environment, or that matches their expectations of what might happen in an environment that they have never experienced (e.g. in space, or a high-speed racing car, or a virtual world). The tendency in the past has been to attempt to achieve this by matching as closely as possible the physics of the real world, with varying degrees of success. Of course, what a person perceives is strongly affected by the physical behaviour of the world around them, but it is the human visual system that receives and interprets the visual cues from the surrounding environment, and it ultimately determines what we perceive.

Seeing is believing, but there are many situations in which the eye can play tricks. People can be convinced that they have seen or experienced something that is contrary to reality. For example, they can be convinced that their train is moving, only to perceive that it was the train beside them moving in the opposite direction while theirs remained motionless in the station; There are roads upon which cars seem to roll uphill; After a waterfall is observed for some time, the rocks beside it appear to move upwards; Of the two lines in Figure 3.1, most people would perceive line b as being the longer, but they are actually identical in length. Phenomena such as these convince us of the necessity of looking beyond the laws of physics to find the secret of reproducing visual reality.

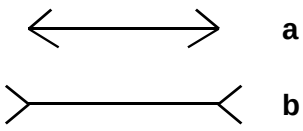


Figure 3.1: The Müller-Lyer illusion

Amazing though it may seem, many of the functions of the human brain, including the visual system, still remain a mystery. The science of the brain appears to be still in its infancy, and new theories and facts about the brain are constantly emerging. Therefore, there is no complete and rigorous theory that can be studied and applied. All that there is to know about the physical architecture of the brain is not yet clear, and new techniques are still being developed to examine it. It is in this climate that Professor Semi Zeki, one of the world's foremost researchers into the visual functions of the brain, has written: "*I hope that no one will be deterred from asking new questions and suggesting new experiments simply because they are not specialists in brain studies. Leaving it to the specialist is about the greatest disservice that one can render to brain science in its present state. The question that the most humble person can ask about the brain is often surprisingly sophisticated and one to which the most accomplished specialist has no answer."* [Zeki 1993]

In Section 3.2, we review some of the perception-based techniques used currently in computer systems, and discuss the feasibility of tracking a viewers point of fixation. Some facts about visual perception are indisputable, such as the physiology and functions of the eye and the parts of the brain that handle visual perception, i.e. the visual cortex. In Section 3.3 we provide an overview of the present knowledge of the physiology of the eye and the visual pathways. Explaining how basic visual information is acquired and encoded is complex, but not impossible. Explaining how this information is subsequently processed and interpreted is another matter entirely. Visual perception is so central to our existence, that it has been the subject of intense research for centuries. However, the variety of approaches has given rise to many contradictory findings, which have been defended passionately by their proponents. This situation has arisen mainly due to the vastly different starting points of the researchers. Some of these different approaches are presented in Section 3.4.

Finally, in Section 3.5 we extract from this vast, sometimes confusing, and often contradictory, body of research some ideas that can help to achieve our ultimate aim, i.e. to understand human visual perception of collisions, and subsequently exploit this knowledge. We adopt a strategy of picking and mixing the techniques and most convincing results from the most well established areas of research.

3.2 Visual Perception in Computer Systems

In recent years the realisation has been growing within the computer graphics community of the advantages to be gained by using knowledge of human perception. The results of research in visual perception have been used to improve the visual realism of synthetic images in [Ferwerda et al. 1996][Ferwerda et al. 1997][Greenberg et al. 1997][McNamara et al. 1998] [Myskowski et al. 1999]. A heuristical approach to real-time shadow generation is recommended in [Meaney and O'Sullivan 1999]. Perceptual factors such as size and speed of objects have been used to choose the levels of detail (LOD) at which to render objects in a scene in [Funkhouser and Sequin 1993]. They use a hierarchical representation of objects, which maintains representations of objects at multiple levels of detail. Image quality is adjusted adaptively in order to maintain the target frame rate. A cost/benefit heuristic is used to choose the level of detail at which to render the object, thus trading accuracy for speed. A similar approach is taken in [Luebke and Erikson 1997] and in [Gossweiler 1994]. [Reddy 1998] discusses selection criteria for specification and evaluation of LOD representations.

The advantages of simulating plausible motion, as opposed to physically accurate motion, have been investigated in [Barzel et al. 1996]. Adaptive refinement for dynamic simulations is used in [Carlson and Hodgins 1997], which presents the idea of Levels of Detail for simulation of legged creatures. They explore techniques for reducing the computational cost of simulating groups of such creatures by using less accurate simulations for individuals when they are less important to the viewer or to the action in the virtual world. Two important criteria for the use of LODs was that the outcome of the game should not be affected by their use, and that the movements should appear visually unchanged to the viewer. A preliminary investigation was made into the effects on the game of the use of such LODs. In [Hodgins et al. 1998] psychological experiments were conducted to determine whether a viewer's perception of human figure motion is affected by the geometric model used for rendering. They found that subjects were better able to observe changes with a polygonal model than with a stick figure model. In [Chenney and Forsyth 1997] a physical simulation process is coarsely approximated for objects outside the view frustum, thus maintaining some level of realistic behavious for objects out of sight, but saving some time by reducing the accuracy where it cannot be noticed.

The idea of concentrating effort on the portion of a Virtual World which is visible, i.e. inside the viewing frustum, is a useful one. This idea can be taken further, however, by determining the current regions of interest in the visible scene. It has long been established that many visual processing tasks deteriorate at increasing eccentricities from the fixation point [Aubert and Foerster 1857] [Weymouth 1858]. Therefore, knowing the location on the screen where a viewer is looking could very useful in many real-time computer systems. The most obvious method to achieve this is to use an eye-tracking device. In the past, the most common use of eye-trackers has been in medical and scientific research. These types of trackers are very accurate, but also very invasive, involving the use of head restraints, bite bars, or scleral coils which are inserted directly into the eye. More recently, more mobile, non-intrusive, and low-cost trackers have been developed, and their use has been gaining increasing support, in particular in field of Human Computer Interaction (HCI) [Ware and Mikaelian 1987][Jacob 1993][Jacob 1994][Jacob 1995][Skerjanc and Pastoor 1997]. A comprehensive review of eye-tracking technology and applications is provided in [Glenstrup and Engell-Nielsen 1995].

In the field of computer graphics such technology is also being advocated. In [Deering 1992] it was stated that: *"(eye-tracking) is a promising long term solution, since gaze direction can be exploited for other purposes such as identifying the region of screen space – corresponding to the foveal portion of the retina - that deserves to be rendered with high spatial detail.*" Gaze-contingent degradation of video resolution was implemented in [Wampers and van Diepen 1999] and [Duchowski 1998]. Gaze-directed adaptive rendering was used in [Ohshima et al 1996], and LOD management for rendering through peripheral degradation has been evaluated in [Watson et al. 1997], using a visual search task to determine the effects while wearing a Head Mounted Display (HMD). In [Janott and O'Sullivan 1999] eye-tracking data is used to order vertex-collapses of multiresolution meshes, thus retaining perceptually important regions at fuller accuracy for longer. However, the use of raw eye gaze information is potentially problematic, and in [Stiefelhagen et al 1997] it is acknowledged that: "*the problem of deriving the focus of attention from the user's 'low level' eye gaze patterns has not yet been adressed. In fact, even if we could have a perfect gaze tracking system, we still have the problem to find a user's focus of attention using only the gaze information. A high level user model is needed to deal with involuntary eye-movements*."

Alternatively, task-specific semantics may be used to determine the fixation point, by attaching the point of fixation to an important object in the animation, e.g. the ball in a football game, an important character in a virtual world. [Funkhouser and Sequin 1993] do not track the user's eye position, so they simply assume that objects appearing near the middle of the screen are more important than ones near the side, and reduce the benefit of each object by an amount proportianal to its distance from the middle of the screen.

Salience maps [Mahoney and Ullmann 1988] are used to prioritise regions of a scene with respect to the probability that they will be the target of a viewer's fixation, and could also be a potential means of either predicting the current regions of interest, or to smooth the raw eye-tracking data. A Kalman filter is a mathematical tool that was used in [Welsh 1996] to smooth incomplete motion-tracking data, and this could also be implemented to counteract the spatial and temporal inaccuracies of the eye-tracker, and the physiological 'noise' described above. A Kalman filter model of the human cortex is presented in [Rao and Ballard 1997], and is used to explain the fixation behaviour of monkeys freely viewing a natural scene.

3.3 Physiology and Neurophysiology

We now provide an overview of the anatomy and physiology of the eye and the neurophysiology of the visual cortex. More detailed explanations may be found in [De Valois and De Valois 1988][Wandell 1995] and [Zeki 1993]. In summary, light from a source arrives at the eye, and is focused onto photoreceptors, causing the formation of an image. Section 3.2.1 examines the physiology and functions of the eye that allow this to occur. This image is then transformed into electrophysiological signals, which communicate several different representations of the image to the brain via the visual pathways, discussed in Section 3.2.2. Finally, in Section 3.2.3 we explain how these neural representations are then transformed into many cortical representations in the visual cortex itself.

3.3.1 The Eye

Without eyes, there is no sight, so this is where any study of vision must begin. The eye is the sensory organ of sight. It detects information from the environment in the form of light, and transmits this information by a series of electro-chemical changes to the brain. The human eye is a roughly spherical, hollow, sensory organ. It is held within a protective bony cavity in the front of the skull, called the *orbit*. It consists of an outer wall and a central cavity. The outer wall or globe consists of three layers (figure 3.2), which are:

- The Fibrous layer (outer coat)
- The Vascular layer (middle coat)
- The Nerve layer (the inner layer)

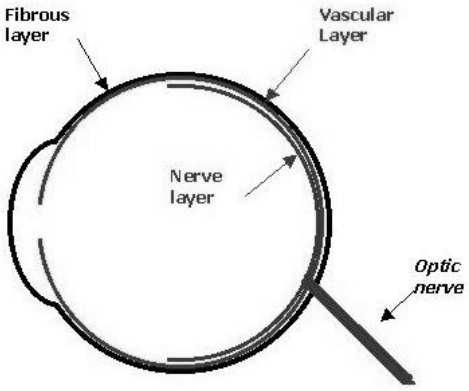


Figure 3.2: The layers of the eye

The fibrous layer (figure 3.3) consists of the *Sclera* at the back, which is opaque, and the *Cornea* at the front, which is transparent. The junction where they meet is called the *Limbus*. The *Vascular Layer*, (figure 3.4) also called the *Uveal Tract*, is composed mostly of blood vessels and capillary nets. Its main function is to supply nutrition to the other layers of the eye. It consists of three parts: the *Choroid*, the *Ciliary Body*, and the *Iris*.

The crystalline lens is transparent, elastic, biconvex in cross section, and roughly circular from the front. The functions of the lens are refraction of light and *Accommodation***,** which is the focus adjustment of the eye. The Zonules of Zinn are numerous radially arranged fibres attached between the ciliary body and the lens around its circumference. Tensions in these fibres are adjusted by the muscles of the ciliary body, changing the shape of the lens and thus its powers of accommodation.

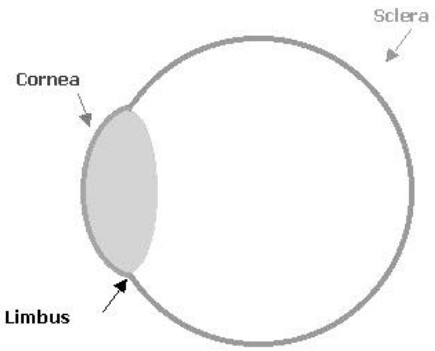


Figure 3.3: the fibrous layer of the eye

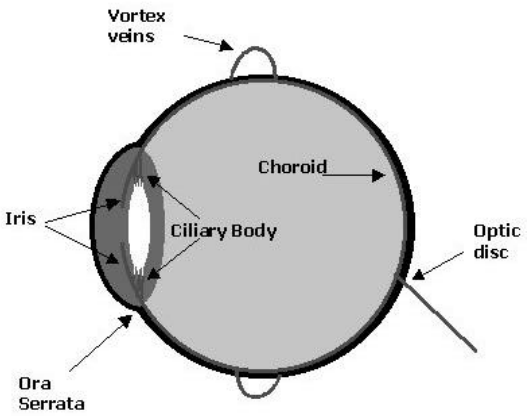


Figure 3.4: the vascular layer of the eye

The nerve layer is more commonly known as the *Retina* (figure 3.5), and it is a multi-layered tissue, consisting of 10 layers. The nerve cells, or neurons, which exist in the retina are the same type as those in the brain. Therefore, the retina is part of the central nervous system. The retina contains light sensitive cells called *Visual Receptors*, is attached to the brain via the *Optic Nerve*, and can be subdivided into three regions: The *Peripheral Retina*, the *Macula Lutea*, and the *Fovea Centralis*.

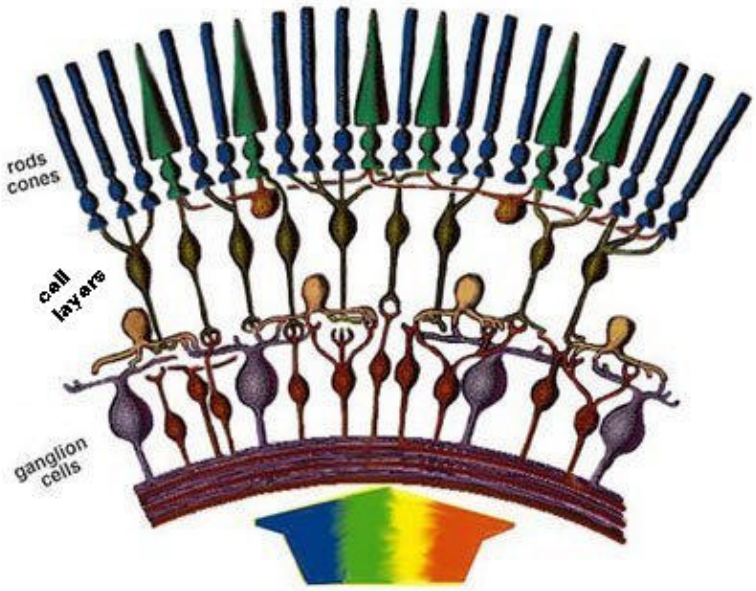


Figure 3.5: A diagram of the Retina [1]

The visual receptors in the retina are called *Rods* and *Cones*. Rods are cells that function best at low light intensity, i.e. in *scotopic illumination*, and produce black and white vision. They are primarily used for contrast determination. Cones operate at high or medium levels of light intensity, i.e. in *photopic illumination*. They produce colour vision, and are needed for fine discrimination of detail, or visual acuity. The rods do perceive colour, but not as strongly as the cones, whose contrast sensitivity is lower.

[1] Adapted from an image downloaded from http://insight.med.utah.edu/Webvision/imageswv/schem.jpeg

The retina contains about six million cones and 100 million rods. Information from the rods and cones is conveyed by the nerve fibres in the inner layers of the retina to leave the eye via the optic nerve. The optic nerve leaves the eye via the *Optic Disk*, where there are no visual receptors, and hence gives rise to a physiological *blind spot*. The peripheral retina contains mainly rods and a few scattered cones. Because cones are necessary for discrimination of detail, visual acuity in the periphery is poor. The *macula lutea*, so called because it looks like a yellow spot, is at the optical centre of the retina. It contains an abundant amount of cones, and very few rods. It therefore produces a high level of visual acuity. The *fovea centralis* is a central depression at the centre of the macula. The cones here are tightly packed, and there are no rods. It is responsible for the highest levels of visual acuity. Each retina is divided into four segments, with the fovea at exact centre. The *nasal retina* is that nearest the nose, and it looks at the *temporal field of view*, and the *temporal retina*, at the side of the eye, looks at the *nasal field of view*. Hence, the nasal retina of the left eye and the temporal retina of the right eye look at the left half of the field of view, i.e. the *left hemi-field*, while the temporal retina of the left eye and the nasal retina of the right eye look at the *right hemi-field*. Each hemi-field is in turn subdivided into an *upper quadrant* and a *lower quadrant*.

The definition of the precise position of a projection of the light from a point in the world onto a retina is determined by the direction of gaze, the fixation point, and the distance of the eye from that point. The fixation point is serviced by the fovea, and the angle between the vector from the fovea to the fixation point, and the vector from the fovea to any other point in the field of view, determines the region of the retina which services that point (see figure 3.6). The larger this angle is, the more peripheral the region of the retina is that services it. Thus can the precise position of a point on the retina be referenced. We can therefore speak of a point in the periphery as being situated at an eccentricity of 15° in the lower right quadrant, or of a point in central vision being situated at an eccentricity of 3° in the upper left quadrant. A definition of visual angle can be found in [Olzak and Thomas 1986]. The visual angle θ is related to the distance between two stimulus elements h, and the viewing distance d, as follows:

$$\tan\left(\frac{q}{2}\right) = \frac{h}{2d} \qquad \textbf{\textit{Equation 1}}$$

q is measured in degrees, while *h* and d can be measured in any units, as long as the same units are used for both. When q is small, a good approximation to the above is:

$$q = 57.3 \times \frac{h}{d} \qquad \textbf{Equation 2}$$

Equation 2 overestimates q by about 1% when its true value is 10°. The peripheral parts of the retina deal with vision beyond 5°. The macula lutea is concerned with the central 5° of vision, and the fovea handles the central 1° of vision.

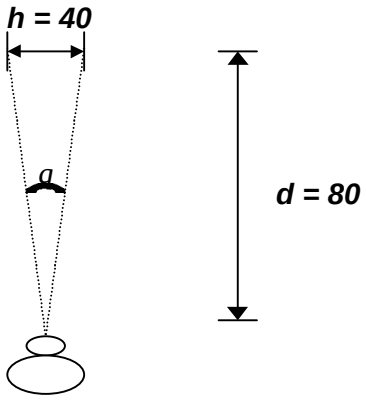


Figure 3.6: Calculation of Visual angle.

To summarise, visual perception results from a series of optical and neural transformations. The first visual transformation is the formation of the retinal image by the eye, which occurs as follows:

- Light from a source arrives at the cornea
- It is focused by the cornea and lens onto photoreceptors on the retina
- The photoreceptors transform light into neural signals, which are communicated through the several layers of retinal neurons to the neurons whose output fibres constitute the optic nerve
- The neural signals are carried out of the eye by the optic nerve through the optic disk, and are carried to the brain for further processing.

3.3.2 The Visual Pathways

The visual pathways are the paths taken by nerve impulses between the eye an the brain when the retina is stimulated by light. They consist of seven structures (see figure 3.7):

- Retina
- Optic Nerve
- Optic Chiasm
- Optic Tract
- Lateral Geniculate Nucleus (LGN)
- Optic Radiations
- Visual Cortex

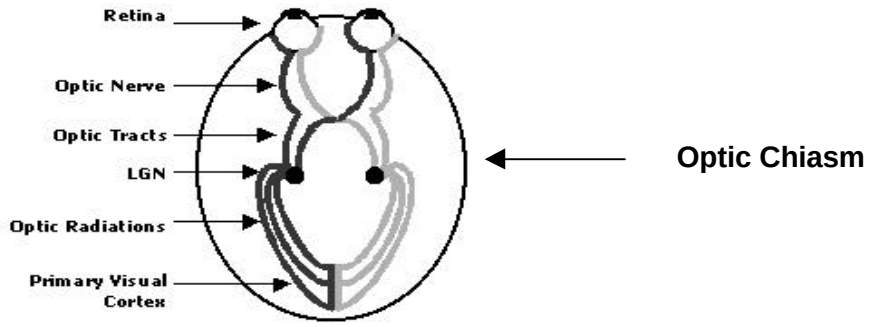


Figure 3.7: The Visual Pathways

Light stimulates the rods and cones of the retina. A first stage of image processing takes place in the retina itself, where there is a large number of interconnecting ganglion cells that perform basic processing such as edge-enhancement. Electrochemical messages are then passed to nerve fibres in the retina and then via the optic nerve to the optic chiasm. Here, information from the temporal (outer) half of each retina continues to the same side of the brain. Information from the nasal (inner) half of each retina crosses to the other side of the brain within the optic chiasm. The rearranged nerve fibres then pass through the optic tract to the lateral geniculate body, and on through the optic radiations to reach the visual cortex in the occipital lobe of the brain (i.e. at the back).

The cells of the retina are part of the central nervous system, and hence are called *neurons*. Retinal neurons have different shapes and sizes, but a typical structure is shown in figure 3.8. The dendrites receive input from other neurons, and the axon carries the neuron's output onto its ultimate destination. The cell body performs whatever processing function is necessary for that type of cell.

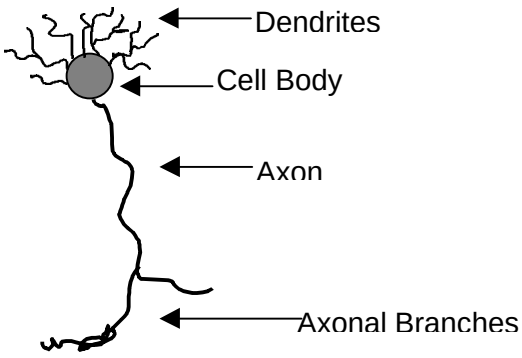


Figure 3.8: Typical Retinal Neuron

The retina contains 3 distinct layers of cells, and two connective layers. The outer nuclear layer contains the cell bodies of the photoreceptors, i.e. the rods and cones. The axons of the photoreceptors make contact with the dendrites of the next layer of cells in the outer plexiform layer. The cell bodies of the bipolar and horizontal cells are located in the inner nuclear layer. The bipolar cells make connections onto the dendrites of the ganglion cells within the inner plexiform layer. The ganglion cell layer contains the ganglion cells, the axons of which comprise the optic nerve. These provide the only output to the brain, and therefore are the most interesting.

The ganglion cell layer has two types of cells, M cells, so-called because they project onto the M-cells of the magnocellular layer in the LGN, and P cells which project onto the P-cells in the parvocellular layer of the LGN. The M cells of the retina are sensitive to low contrasts, respond transiently, and have axons which conduct very rapidly. Hence, it is considered that these are important for motion perception. They are not wavelength-sensitive. The P cells of the retina respond to high contrasts, have a sustained reponse, and are wavelength selective. Hence, they are considered to be important for colour perception.

The fibres of the optic nerves carry information from the retina of each eye, and these fibres meet and cross over at the optic chiasm. The fibres from the nasal part of the retina cross over, and those from the temporal retina do not cross over, but continue on to the same hemisphere of the brain. Therefore:

- The fibres from the temporal retina of the left eye, and the fibres from the nasal retina of the right eye continue on to the left hemisphere of the brain. This means that this half of the brain looks at the right half of the field of view.
- The fibres from the nasal retina of the left eye and the fibres from the temporal retina of the right eye connect to the right hemisphere, which therefore looks at the left half of the field of view.

The visual pathway now becomes the optic tract, whose fibres terminate in the cells of the LGN. The LGN has 6 layers. The upper 4 layers, the parvocellular or P layers, have small cells, and the lower two, the magnocellular or M layers, have large cells. The two sets of cells have different destinations in the visual cortex. Most cells in the P layers are wavelength sensitive, whereas those in the M layers are not. Again, this indicates that the M layers transfer information about motion to the visual cortex, whereas the P layers transfer information about colour. Finally, the axons of the LGN cells travel in the optic radiation on to the visual cortex.

The differences in the behaviour of cells in the retina and the LGN indicates the existence of at least two parallel visual pathways, one for motion, and one for colour. In fact, there are four pathways:

1. A motion pathway
2. A dynamic form pathway
3. A colour pathway
4. A form pathway linked to colour

However, before we can explain these pathways fully, we must first look at the architecture and functional segregation of the Visual Cortex, and the nature of the projection from the retina, via the visual pathways, onto this part of the brain.

3.3.3 The Visual Cortex

The visual cortex occupies the occipital lobe of the brain. It consists of several areas, the largest of which is the *Striate Cortex*. The striate cortex is a large sheet of cells at the back of the brain, so-called because the architecture of its cells, i.e. its *cytoarchitecture*, is very distinctive and consists of stripes. This part of the brain is also known as the primary *visual cortex*, or *area V1*, and for a long time was considered to be the only area of the brain responsible for visual perception. However, it was discovered that other areas of the brain, collectively called the *pre-striate cortex*, were also responsible for visual processing. These areas are referred to as areas, V2, V3 etc…Hence, we shall refer to the striate cortex as area V1 of the brain from this point onwards.

The area V1 on each side of the brain receives information about the opposite side of the field of view. The cells in the retina project in an orderly, systematic way onto the cells of the LGN, so that adjacent retinal points are mapped onto adjacent cells in the LGN. The cells in the LGN are mapped in a similarly systematic way onto the cells of area V1. However, magnification has occurred along the visual pathways, so that one retinal cell will project onto hundreds of cortical cells, which process the output of each incoming fibre. This mapping from the retina onto area V1 is known as a *topographical map*, because each part of the retina is represented in a given part of area V1.

Area V1 is a layered structure, and contains cells which behave in different ways, depending on the properties of the visual stimulus about which the information has been transferred. The receptive field of a cell is that part of the field of view which is projected onto this cell, (via the retina and visual pathways). Most of what we know about the receptive field of cortical cells follows from the work of [Hubel and Wiesel 1968]. They recorded the activity of cortical neurons while displaying oriented lines, bars and spots. Cells with oriented receptive fields were discovered, which responded to stimuli in some orientations better than others. This property is known as *orientation selectivity*. Cells have also been discovered which respond well when a stimulus moves in one direction, but not in another [Zeki 1974]. This property is known as *direction selectivity*. Some cells have no orientation or direction preference, but are wavelength-specific, and some are both orientation *and* direction selective.

The main function of area V1 is as a functional segretator, i.e. it selects the important information for certain sub-tasks, such as colour, motion, or form perception, and redirects the important information to the appropriate areas in the pre-striate cortex, i.e. areas V2, V3, V4, V5 and V6. Area V2 also contains the functional groupings of cells outlined above, and projects to the same specialized areas as above, namely V3, V4 and V5. Therefore, it also will send signals related to different properties of the field of view to those parts of the brain which are responsible for processing those attributes. It is also, therefore, a segregator. Area V3 lies next to B2, and receives a direct, point-to-point mapping from area V1. This means that the retina is also topographically mapped onto it. Recordings from this area of the brain have shown that the vast majority of cells are responsive to lines of a specific orientation, but show no difference in response when the colour of a stimulus is changed.

Next to V3 lies area V4. This area receives a direct input from the region of foveal representation in V1, but its predominant input is from area V2. The mapping is not so topographical as was the case from V1 to V3, but the receptive fields of cells in an area of V4 are usually in roughly the same area of the field of view. The cells in this area are heavily wavelength-specific, and only the central 40° of the retina is mapped in this area. Hence it is inferred that this is the part of the brain responsible for colour vision, and form from colour. Lying directly behind area V4 is area V5 (also called area MT). It receives a direct input from area V1. The cells in this area are all responsive to different types of motion, but the colour of the stimuli is unimportant. 90% of the cells are directionally selective. Hence, area V5 is considered to be responsible for motion perception. Finally, area V6, which is located in a different area of the brain, also receives input from area V2. The mapping from the retina is very complicated, and recently it has be suggested that one of its functions is that of space representation in the brain. All of these areas output many signals to other parts of the brain, or back into other parts of the same area.

We may now explain the four visual pathways in the visual cortex introduced above in more detail. The motion pathway (M-pathway) starts in the M ganglion cells of the retina, which are relayed to the M layers of the LGN and from there to a layer of direction-selective cells in area V1, and then onto area V5, both directly and via area V2.

Another pathway, the M dynamic form pathway, is derived also from the M ganglion cells of the retina. It is also relayed to the same layer of V1, and from there to area V3, both directly and via area V2. The P pathway originates in the P cells of the retina, which are relayed to the P layers of the LGN, and onto those layers of area V1 which are wavelength specific. These layers consist of “blobs” of cells, which are responsible for colour, and the areas between the blobs are called the “interblobs”, which are responsible for form. From there they divide to create a colour pathway, and a form pathway linked to colour. These two paths relay to area V4 both directly, and through area V2.

3.4 Theories of Visual Perception

As has been stated above, a wide range of approaches have been applied to explain human visual perception. An overview of the most important of these theories may be found in [Gordon 1996]. We consider two of the most significant theories, which are quite complementary to each other.

Psychophysics is a set of techniques used to study mappings between events in the environment and levels of sensory responses, by conducting non-invasive experiments on humans. To be more specific, psychophysicists are interested in exploring thresholds, i.e. what is the minimum distance and/or size required to recognise a letter? At what point in the periphery may a gap between two objects be perceptible?

Neurophysiology is the search for explanations of perceptual phenomena by examining actual neural mechanisms, most commonly of animals with cortical architectures similar to humans, or by examination of human brains posthumously. There are many good reasons for taking this approach. As more advanced techniques are being developed to examine the architecture of the cerebral areas, more facts are emerging about the functioning of the brain. For example, it has long been known that visual acuity falls off when stimuli are located in the periphery of the eye. It is now established why this occurs, both at the retinal and cerebral levels. This makes knowledge attained by other means much more secure.

From the perspective of some researchers, it may be more satisfying to deal with actual physical structures such as neural mechanisms, than with abstract (and often un-substantiated) psychological concepts. However, although advances in neurophysiology have been immense in recent years, this reductionist approach is often not sufficient to explain how the brain processes all but the most simple of visual events. Nevertheless, both approaches complement each other: psychophysical results are more secure when they can be linked to known physical structures, and the work of neurophysiologists is often motivated by the need to explain phenomena discovered in psychophysical research.

Our approach in this thesis is to discover facts about visual perception of collisions through ***psychophysical experiments***, but to use the results of ***neurophysiological investigations*** both to guide the design of the experiments, and to explain our findings.

3.5 Collision Perception

To determine the factors which influence a viewer's perception of a collision, and the extent of their contribution, we must first try isolating these factors, and testing them individually. We need to carefully consider each potential factor, and decide on the best way to test its effects. This will lead to a set of psychophysical experiments, the results of which will be used to schedule collision testing in our application.

The overall hypothesis to be tested is that there are certain factors that affect a human's ability to notice whether two objects have collided realistically or not. To consider using the factor in the prioritisation algorithm, the effects observed should be significant, and occur in most, if not all, of the subjects tested. In addition, they must be *robust*. There is no point in using a factor that only occurs under some highly specialised conditions; e.g. sitting a specific distance from the screen, with the head tilted at a particular angle. The vision literature is filled with descriptions of experimental results that can only be reproduced if the conditions of the original experiment are exactly replicated. We are looking for factors that can be generalised over a wide range of conditions, because we wish to apply them in a real-world scenario.

The list of potential factors includes:

1. Type of collision event
2. Eccentricity
3. Separation
4. Location
5. Orientation
6. Motion: velocity and direction
7. Presence/Number of distractors

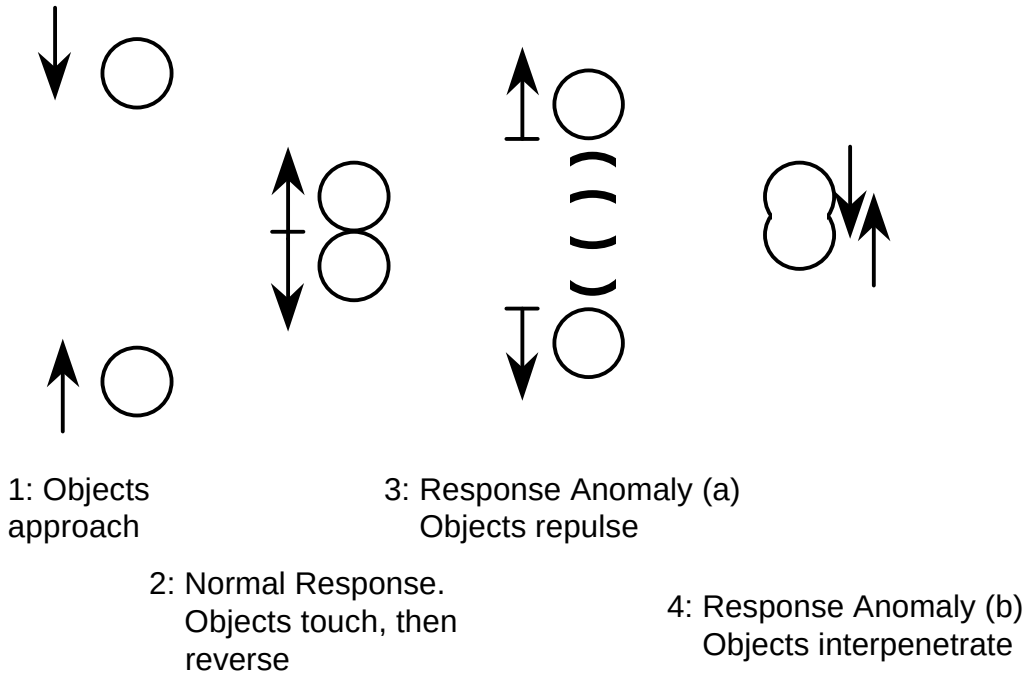


Figure 3.9: Different types of collision events

3.5.1 Type of collision event

In the applications being considered, three types of collisions may occur (see figure 3.9):

- "True" collisions, where entities touch, the collision is detected, and fully accurate collision response occurs. We may consider this as being the control situation for experimental purposes.
- Interpenetrations, where the entities also touch, but the collision is not detected or is ignored by the application. The entities are therefore allowed to continue on their previous path, even though it causes them to merge into each other to a greater or lesser degree.
- Repulsions, where the entities are close to each other but have not actually touched. In this case the application decides to take the chance that they are actually touching, and accepts this situation as a true collision. This may then cause the entities to change their paths, causing a repulsion effect.

The application can control which type of collision anomaly will occur most often, if not always. It could be argued that a detailed study should be conducted into which type of anomaly is most disturbing to the viewer. However, there are certain points to be made in favour of allowing only repulsions to occur:

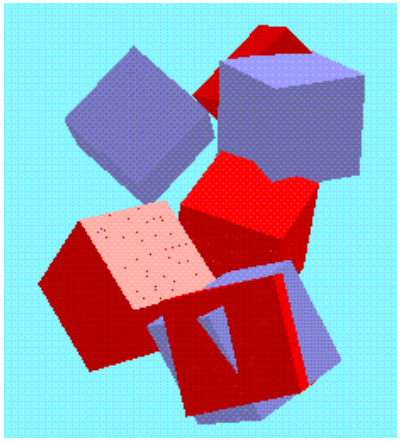


Figure 3.10: Interpenetrating entities of different colours

- The effect of one entity piercing through another is very noticeable. Observations suggest that this effect is more disturbing than the effect of repulsion, especially if the entities are of different colours (see figure 3.10).

- Another problem with interpenetration is that the anomaly lasts longer than repulsion. With repulsion, the anomaly occurs for only one frame, and then the entities move apart as if they had actually collided. If the viewer has not noticed the anomaly at that frame, they will be none the wiser if they subsequently observe the two entities. However, if the collision between two interpenetrating entities is ignored for several frames, they will continue to interpenetrate further and further. This would happen if the collision is happening somewhere that the viewer cannot see very well, e.g. outside the viewport or at the opposite end of the screen, and in this case would probably not be noticed. But what if the view is suddenly swerved, or a saccade to that exact point on the screen occurs? The application will immediately prioritise the collision as much more important, and attempt to separate the objects, but it will be too late. The viewer will be presented with the offending interpenetration and is highly likely to find the effect disturbing.

In addition, the visual perception of repulsion has well documented parallels in the study of spatial vision, hyperacuity, and brain physiology. The visual perception of interpenetration involves more complex, and less well researched, neural mechanisms. It may be the case that some level of interpenetration must be accepted as a trade-off for speed in certain applications, so a possible extension to this work would be to study the perceptual response of the human visual system to this anomaly also. However, for the reasons outline above, we consider the study of interpentration to be outside the scope of this thesis. We have therefore decided to study the effects of repulsion vs. true collision only.

3.5.2 Eccentricity

The fact has long been established in vision literature that many visual processing tasks deteriorate at increasing eccentricities from the fixation point [Aubert and Foerster 1857] [Weymouth 1858]. The eccentricity effect can be exploited in a real-time application by tracking the user's fixation position.

When the viewer is looking directly at a collision, it should be given a higher priority than a collision occurring at a slight eccentricity, which itself should be given a higher priority than other collisions presented more peripherally. A summary of the physiological reasons for decreased spatial resolution in the periphery appears in [DeValois and DeValois 1988]:

- Information projected onto the central part of the retina, i.e. the fovea, receives more processing, because there are more cones concentrated there.
- There is an almost one-to-one correspondence between the photo-receptors in the fovea, and the ganglion cells there. In the periphery, hundreds of photoreceptors can converge onto just one ganglion cell.
- The receptive field sizes of cells in the fovea are smaller than in the periphery (i.e. the area of the visible scene which a foveal cell must process is smaller than those cells with greater eccentricities). In addition, there is a higher density of ganglion cells in the fovea. In fact, with eccentricity, receptive field size increases and density decreases in a log function. This means that detection of fine details is facilitated in the fovea, and deteriorates with eccentricity.
- The representation of the fovea in the LGN is magnified, and more magnification occurs when the LGN projects onto the primary visual cortex, V1. This means that in the LGN and V1, there is more representation for the fovea than for the periphery, allowing for more acute visual processing (cortical magnification).

Two additional factors may also be mentioned, the effect of similar elements in the field of view may degrade perception in the periphery, as could the distribution of attention, i.e. if a person is concentrating on a complex task at the point of fixation, their perception of stimuli in their peripheral field of view is likely to be much poorer. The cortical magnification theory claims that by increasing the size of stimuli according to the amount of cortical area onto which that part of the retina projects, the effect of eccentricity can be neutralised. This can either be done directly, where subjects are tested at different retinal eccentricities and the results compared with cortical mapping functions [Weymouth 1958], or indirectly, where the size of stimuli is scaled in inverse proportion to the cortical magnification factor (M) at each eccentricity. This technique is known as M-scaling, and is explained in [Rovamo and Virsu 1979] [Carrasco and Frieder 1997].

As early as 1857, M-scaling was shown to equalize performance of two-point separation in the near periphery [Aubert and Foerster 1857]. However, in the same study, the theory failed completely for two-point separation in the far periphery. It also fails in many other cases, such as in [Strasburger et al 1994]], where both size and contrast had to be increased to neutralise the eccentricity effect. There is controversy about whether it works in the case of Vernier Acuity (the ability to detect slight offsets between lines) [Beard et al 1997]. From this discussion it is likely that the ability of humans to detect collision anomalies deteriorates almost linearly with eccentricity of presentation. Cortical magnification is definitely one explanation for this, but from the preceding discussions it becomes obvious this is not the only reason, because other factors, such as direction of motion, presentation orientation, size of stimuli, will all have an influence.

3.5.3 Separation

Separation, or gap-size between two colliding objects is also likely to affect the ability of humans to detect an anomaly. This is because there is a topology-preserving mapping from the cells in the retina to the cells in the primary visual cortex, called a retinotopic mapping, and it is quite precise, enabling spatial location information to be efficiently processed [Tootell et al. 1982]. There is also the question of how much its effect is affected by the size of the entities, their velocity, and all the other variables being discussed.

3.5.4 Location

As mentioned in Section 5.4, the effect of eccentricity is not symmetric. We would like to determine the useful field of view (UFV) for the perception of collision anomalies, i.e. to test what effect the location (e.g. above, below, above left etc...) of a collision has. In [Nies et al. 1998] they found that a 75 per cent confidence region for a visual search task was quite elliptic, with performance being better in the horizontal regions rather than in the vertical. Other effects were also evident, such as better detection to the right than to the left. They did find that there were big differences between subjects in the size and shape of the useful field of view. Hence, we will need to test collisions occurring in different regions, determined by the 9 cardinal directions, i.e. left, right, up, down, up-left, up-right, down-left, and down-right.

3.5.5 Orientation

The visual cortex contains many cells that are orientationally selective, i.e. they perform best when the stimuli in their receptive fields are oriented at a particular angle [Hubel and Wiesel 1968]. Is there a pattern that can be exploited by our application, i.e. are the orientationally-selective cells grouped together in such a way that certain regions and/or eccentricities exist where collision detection is worse or better? In [Fahle 1986], it was found that curvature detection in peripheral vision was best for stimuli which were oriented towards the fovea, i.e. there was an anisotropy of orientation discrimination, with detection being better for lines that were oriented radially (away from the point of fixation), than those presented with an isoeccentric orientation (all points equidistant from fixation point. However, in [Yap et al. 1987] the opposite effect was found for a three-dot bisection task, where subjects had to indicate how close to the centre between two fixed dots a third flashed dot was). They distiguished between the orientation of the stimulus and the direction of stimulus displacement and found that the direction of offset was more important than the orientation of the stimulus, and found that isoeccentric bisection was better than radial bisection.

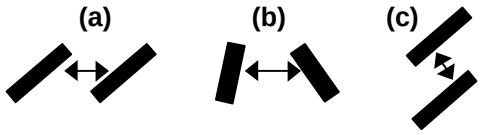


Figure 3.11: Different stimulus orientations (a)(b), and displacement direction (c)

3.5.6 Motion: velocity and direction

Image motion has been observed to have a degrading effect on various types of visual task. In [Chung et al. 1996] they tested the effect of stimulus velocity on Vernier acuity (the ability to detect offsets between two lines), and discovered that vernier thresholds worsen as the velocity increases. They explain this by a shift of sensitivity to mechanisms of lower spatial frequency. I.e, the faster a stimulus moves, the less fine detail the retina and hence the visual cortex can determine. Hence, we suggest that the faster two entities are moving, the less likely it is that an erroneous gap left between them when they collide will be noticed.

There are also cells which are sensitive to direction of motion in the area of the cortex responsible for motion detection, i.e. the middle temporal visual area, MT (also called area V5) [Zeki 1974]. [Albright 1989] describes a "centrifugal directional bias" in area V5 of the brain of the macaque monkey, where more cells were responsive to directions away from the fovea. A likely reason for this anisotropy is that as an animal moves, the visible scene is constantly expanding away from the fovea. Cells responsive to certain types of spiral motion have also been found [Graziano et al. 1994].

3.5.7 Presence/Number of Distractors

The presence of distractors is also likely to affect the ability to accurately detect collision or non-collision, as is the type of distractor. These issues arise in the area of Visual Search [Saarinen 1994][Treisman 1982]. If the distractors are in a clearly distinguishable perceptual grouping from the target to be searched for, the identification of this grouping occurs automatically, without any attention or search being necessary. Such a grouping may be of similar colour, orientation or common movement, which differs from the target. This is referred to as a *preattentive pop-out task*. If such an obvious grouping is not immediately apparent, it is necessary to focus attention on each item in turn, and this is called a *serial search task*. In such a task, performance is significantly worse than in the pop-out tasks. In the tasks that we are considering, e.g. simulations of large numbers of homogeneous interacting entities, such as crowd scenes or rockfalls, there will not be obvious perceptual groupings of objects. Therefore, the ability of viewers to detect a collision anomaly, i.e. a repulsion, in such a scenario is of major interest to us.

3.5.8 Other Factors

Other important factors may also affect the perception of collisions, such as colour [Dobkins and Albright 1993], texture [Saarinen et al. 1987], shape, depth and visibility. Incorporation of all these factors into our model may make it overly complex, and impact on the performance of the system. There is an infinity of variable combinations and possible experiments, so we will pick a set of conditions that are useful to investigate the principal questions. Our strategy is to follow one line in the space of possibilities, which can later be extended and refined. The design of the psychophysical experiments to investigate these questions are presented in Chapter 6 of this thesis.

4 The Application

To apply and test the concepts and algorithms proposed in this thesis, we need a real-time animation system. A simple application was developed which allows implementation and testing of various different collision scheduling and testing strategies. The ultimate aim is to incorporate the resulting algorithms into a "real" application, such as the rock falling one described in Chapter 1.

4.1 Overall design of application

The application may be considered as consisting of an object, of type **Animation**. This **Animation** object represents a "world" in which entities exist, move around and interact depending on their physical properties, and are rendered and displayed on a 2-Dimensional display. The **Animation** object consists of:

- The entities in the world, represented as an array of objects of type **Entity**.
- A volume/box within which the entities move and interact
- Viewing parameters, i.e. a description of the current state of the "synthetic camera" used to gain a view of the world.
- Lists and tables to keep track of interactions and collisions between objects:
  - An "all-pairs" table that sets a flag for every pair of entities if the projection of their bounding boxes onto each of the x, y and z axes overlap. Overlaps in all three dimensions means that the bounding boxes themselves overlap, indicating a potential collision of the entities themselves.
  - Two or more lists of collisions, represented as linked lists of objects of type **Collision**:
    - The active collision lists: One or more lists of potential collisions created from the all-pairs table, each consisting of all pairs of entities suspected of colliding due to an overlap of their bounding boxes, but which need further processing by our intersection-testing algorithm to determine whether they are really colliding or not.

– The real collision list, which contains all pairs of entities that have been detected as really colliding by our intersection-testing algorithm.

The main methods/operations that an **Animation** object can perform are:

- Construction/Initialisation, i.e. a new animation can be instantiated, specifying exactly how the "world" should be, i.e. number of objects, dimensions of the box, initial viewing parameters. The objects will be given their initial positions in the world, and the collision lists and tables will be set to reflect the interactions of the objects.
- Execution, i.e. once the world has been created, with all the objects in it, the animation may then begin. In its simplest form, execution consists of the following loop:

  DO
  - Update the position of all objects
  - Update the all-pairs table by testing object bounding boxes for overlaps in all three dimensions
  - Generate the active collision list from the all-pairs table
  - Process the active collision list, removing collisions as they are resolved, placing detected collisions on the real collision list, and discarding those where the objects are found not to be touching.
  - Process the real collision list, computing the appropriate collision response for each colliding pair of objects.
  - Draw all objects

  UNTIL animation is terminated.

The **Entity** objects contain all information about the entity, such as position, size, translational and rotational velocity, colour, material, along with a pointer to an object of type **Sphere**, which will be the root of the sphere tree that approximates it. We have used a very simple volumetric representation for the entities in our application. Each entity is defined by a 3-dimensional array of 1's and 0's, a 1 indicating the presence of a cube, a 0 representing the absence. This simplifies the tasks of rendering, updating and generating sphere-trees greatly, and it is for this reason that we chose this scheme.

Figure 4.1 shows some of the objects we used in our application. Although this modelling scheme is fairly simplistic, the collision detection routines are designed to work with any sphere-trees generated from any type of model. Therefore, our work may easily be adapted to work in more general cases.

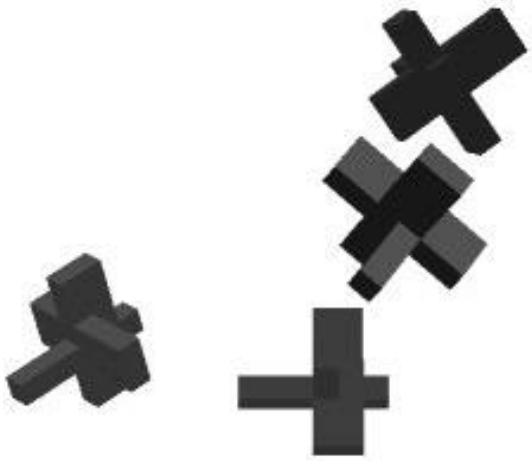


Figure. 4.1: Sample Entities

The main methods/operations that an object of type Entity may perform are:

- Initialisation: Set initial properties, i.e. colour, position, etc... and generate the sphere tree, centred at origin.
- Update: generate the new position of the entity from its translational and rotational information
- Draw: All rendering is achieved with the use of the OpenGL™ graphics library.

The **Sphere** objects are the building blocks of the sphere tree that approximate an object. A sphere contains the information about its radius, its centre relative to the origin, and its centre relative to the entity in its updated state. The sphere-tree is defined to approximate the entity centred at the origin. Only when this entity is involved in a broad-phase collision will the sphere tree be moved to the update position of the object. This is a very simple operation, as spheres are rotationally invariant. The sphere also contains pointers to other spheres: i.e. to its child-list, its parent, and its next sibling. These pointers allow the sphere tree to be built up.

The **Collision** objects are nodes which may be linked together to make a dynamic list. These objects consist of:

- Two pointers to objects of type Entity, i.e. to the entities involved in the collision
- A pointer to the previous collision in the list (if any).
- A pointer to the next collision in the list (if any).
- A sphere hit list (explained below)
- The centre of collision (see below)
- The distance of the centre of collision from the (estimated) fixation location
- The priority of the collision.
- The status of the collision (i.e. colliding, not colliding, or further processing required).

The main methods/operations that an object of type **Collision** may perform are:

- Initialisation
- Maintain the links to the previous and next collisions in the list
- Intersection test:
  - Tests one level of one sphere tree against one sphere on the other tree. Section 4.2 discusses the sphere-tree intersection algorithm in more detail.
- Set collision priority:
  - This is where we will use the heuristic/model developed as the result of our psychophysical and physiological studies.

4.2 Collision Detection

In Chapter 2, we learned that many efficient collision detection algorithms involve two or more phases of detection at varying degrees of accuracy. We also adopt this strategy, and implement both a broad phase algorithm to isolate potential collisions, and an interruptible narrow phase algorithm which allows more accurate collision detection, depending on the amount of processing time available.

4.2.1 Broad Phase

We use the ***Sweep and Prune*** algorithm proposed by [Cohen et al. 1995] to create a list of potential collisions. This algorithm uses the fact that:

> *For two 3-dimensional objects to overlap in 3-dimensional space, their 2-dimensional projections onto each of the xy, xz, and yz planes must overlap in* ***all three*** *cases.*

If axis-aligned bounding boxes are used, they can be projected onto the x, y and z-axes, resulting in three sets of one-dimensional ***intervals***. Intersection of a pair of 3-dimensional bounding boxes would result in overlaps of their corresponding intervals in all three dimensions. This leads to a very quick and simple one-dimensional algorithm.

During the first frame of the animation, the bounding boxes are generated for all objects, and projected onto the x, y and z-axes. A list is constructed for each dimension, containing the endpoints of all intervals corresponding to that dimension. These lists are then sorted using an efficient sorting algorithm for previously unsorted lists, such as ***Quick Sort***. Any intervals that overlap are then detected, and if overlaps occur in all three dimensions for a pair of bounding boxes, the Narrow Phase is triggered.

At each subsequent iteration of the application, bounding boxes are updated, and appropriate changes made to the interval lists. Animations typically exhibit ***Inter-frame coherence***, i.e. because each frame represents a snap-shot of all objects at a particular point in time, and the time-step between frames is very small (e.g. a minimum of one-tenth of a second), the positions of the objects relative to each other will not change dramatically. Hence, because the lists were previously sorted, ***Insertion Sort*** is used to keep them sorted. This sorting algorithm runs in close to ***linear time*** on almost sorted lists. Again, overlaps in all three dimensions will trigger the Narrow Phase of the algorithm.

One issue which arises is whether to use ***Fixed-Size*** bounding cubes, which are large enough to hold the convex object at any orientation, or ***Dynamically-Sized*** bounding boxes, which will be recomputed at every frame to be the smallest axis-aligned box that contains the object at its current orientation. Although dynamically sized boxes are more accurate, they add a computational load at each frame. Fixed-sized cubes are simpler to update, but may give rise to many unnecessary Narrow Phase collision tests.

The best choice of bounding volume may depend on the shape of the objects. If they are almost spherical, the fixed cube fits it well, and if they're long and thin, dynamically sized rectangles fit better, and give rise to fewer overlaps. However, the number of objects moving in the scene is a more important factor. [Cohen et al. 95] found that if many objects are moving, the computational burden of updating bounding volumes at each frame could significantly degrade performance. However, if only a few objects are moving, the reduction in Narrow Phase collision tests achieved by using tighter bounding boxes, outweighs the computational cost of computing the boxes. We choose to use fixed-size bounding boxes for our application, as we will be animating large numbers of entities.

4.2.2 Narrow Phase

For the narrow phase of our algorithm, we have developed an interruptible algorithm based on sphere trees. Spheres are frequently used in computer graphics as approximations to objects. One reason is that it is very simple to test for intersections between them. Another more important reason is the fact that they are rotationally invariant. Because of this property, it is possible to build a hierarchy of spheres to approximate any non-convex object once in a pre-processing phase, centred at the origin. Whenever we wish to test for a collision between two entities, we translate and rotate the centres of the spheres on each approximating tree as we need them, and test for intersections between them. We have adapted a "staircase" algorithm from [Palmer and Grimsdale 1995], and have made it interruptible, as in [Hubbard 1995]. The sphere trees are generated during a pre-processing phase, each tree consisting of 4 levels of spheres, each level representing a closer approximation to the surface of the object (see Figure 4.2).

[Hubbard 1996] lists three requirements for generating hierarchies of spheres:

1. The pre-processing phase must be automatic, with no user-intervention necessary
2. The hierarchy must be structured in such a way as to make searching it efficient, with each level eliminating the need to search a significant subset of the next level
3. Each hierarchy should fit the entity as tightly as possible.

Because of criterion 2, (i.e. efficient searching and elimination at each level), a tree is the obvious data structure to use. Another possible structure is a Directed Acyclic Graph (DAG) in which parents can share children. The structure of the sphere-tree, which we use, is shown Figure 4.3. Each sphere in the tree contains a pointer to the first sphere in its child list. If this sphere is a leaf, this pointer will be NULL. In turn, each sphere contains a pointer to its parent (NULL for the root), and to its next sibling in the sibling list, if any.

Sphere trees may be built in such a way that children must fully cover all parts of the object that their parent does, or simply a subset of those parts. We have chosen to build our sphere-trees to represent a conservative over-approximation of the object's exposed volume. Collisions involving any uncovered areas of the surface will remain undetected by the detection system so we require that our hierarchy of spheres at each level encompass the object completely. On the other hand, we need to ensure that there are no redundant spheres in our sphere-tree such as those that are occluded by other spheres and thus play no part in the actual collision detection. The accuracy of collision detection is dependent on the sphere-tree representations of the objects in our system, so we would desire that the sphere trees fit the object as tightly as possible. At the same time, limiting the number of nodes at each level of the sphere tree directly implies less computation for the detection mechanism so a simple model would also be desirable. What we require then is an automatic method of generating sphere-tree representations, which will be as tight as possible and yet simple enough so that it would be feasible to use it in our real-time application. The method implemented in our system is based on octree subdivision of the object volume.

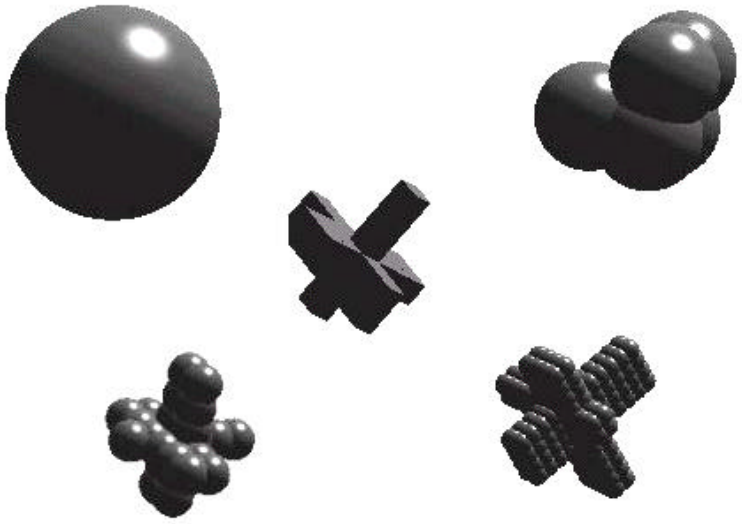


Figure 4.2: An entity and 4 levels of its sphere-tree

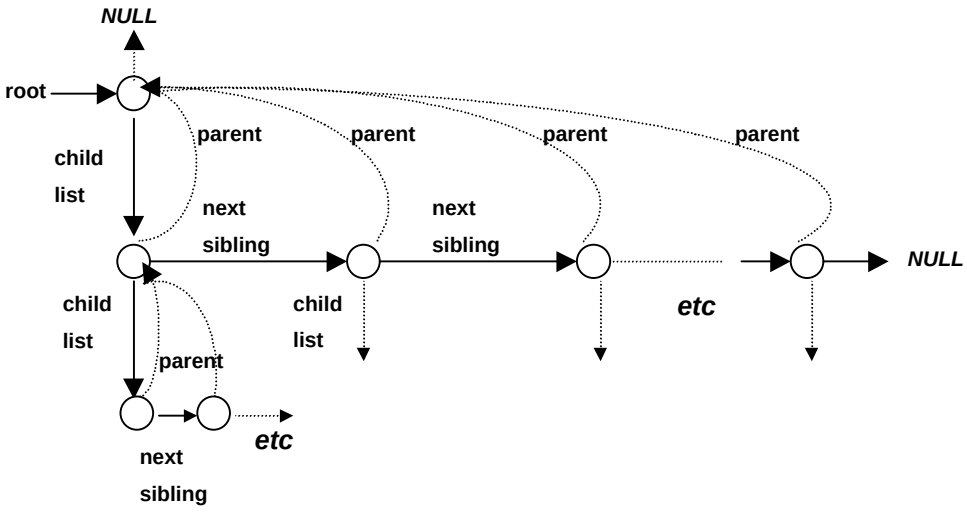


Figure 4.3: Design of a sphere-tree

An octree representation of the object is generated by recursive subdivision [Sammet & Webber 88]. The smallest bounding cube needs to be determined, which will completely encompass the object. This is the coarsest level of the octree and represents the bounding cube for our object. This cube is then subdivided into eight equal partitions or octants. If any of these partitions contains any part of the object then it is enabled as a node on the octree. Each octant is then recursively subdivided in a similar way up to the level of decomposition required. Determining whether or not a 3D sub-partition contains any part of the object can be done in several ways depending on the method initially used to describe the object's volume.

In our present system we require that each object has a voxel representation of its volume so determining whether or not to attach a child node simply involves checking if the corresponding octant contains an enabled voxel. Once the required octree has been generated, it is a simple matter to find the smallest radius of sphere required to completely encompass any particular node of the octree. Finally we remove all nodes from the octree which are occluded on all sides by other nodes and the finished sphere-tree is obtained by generating all the spheres corresponding to the remaining nodes of the octree. It may be stressed here, that although our entities presently have a volumetric representation, this can be easily extended to cater for more general geometric objects. In addition, there are more tightly-fitting sphere hierarchies, such as those described in [Hubbard 1996] and [O'Rourke and Badler 1979], allowing further optimisation if required.

To perform adaptive collision detection, we must approximate each entity with a sphere tree during a pre-processing phase. We test for intersections between two sphere trees as follows: Take two sphere-trees, tree 1 and tree 2, which approximate two objects whose bounding boxes overlap in all three dimensions. This has caused an object of type **Collision** to be created, with pointers to the roots of both trees. Each collision object contains a pointer to a list of **sphere hits**. A sphere hit contains a record of the current state of the collision object; i.e. what spheres on one tree must be tested against what spheres on the other tree. This consists of a target sphere on one tree, and a test list of spheres on the other tree.

At the start of the algorithm, the list consists of one sphere hit, i.e. the root of tree 1 is the target, and the root of tree 2 is the test list. If these do not intersect, then there is no collision between the objects. If they do, a new sphere hit is added to the collision's sphere hit list, the root of tree 2 becomes the target sphere, and the children of the root of tree 1 now become the test list. Continuing on in this fashion, every time an intersection is found between a target sphere and a member of the test list, a new sphere hit is created. The intersecting member of the test list becomes the new target sphere, and the children of the old target becoming the new test list. In this way, the algorithm is fully interruptible, allowing the detection to descend one level of one tree at a time, reducing the complexity of the algorithm, and enabling a fast, albeit approximate, response when necessary.

4.3 Adding Interruption

The application performs the broad-phase testing, and creates a list of collision objects called the active collision list. This is a linked list of all collisions that have not yet been resolved. As collisions are resolved, they are placed on the real collision list if a real collision was detected, or are destroyed, if it has been determined that the entities are definitely not colliding. If all collisions are fully resolved, the frame-rate will be highly variable. Therefore, it is sometimes desirable to interrupt processing when a target time has elapsed. The control of the processing order of collisions on the active collision list will therefore necessitate some form of scheduling mechanism, discussed in the next section.

Regardless of the scheduling algorithm used, at any point during collision processing, there will be unresolved collisions on the active list, and resolved collisions on the real list. At some point the application will deem that collision processing should stop. In our case the criterion for stopping is when a pre-defined target time has been exceeded. However, other criteria could also be used, e.g. a request from a client to a server, or user intervention. When the request for an interruption is generated, collision processing must stop, leaving a list of real collisions and a list of unresolved collisions on the active collision list. These unresolved collisions are added to the end of the real collision list. We could also choose to reject these collisions, thus allowing interpenetration, but for reasons explained in Chapter 3, we will accept all unresolved collisions as being real.

There are four possible results at each iteration of the test for a collision object:

1. An intersection is detected between two leaves of the trees. In this case the objects are deemed to be colliding, and the collision is resolved. The collision will be removed from the active collision list, and added to the real collision list.

2. No intersections are detected between any of the targets of each sphere hit and the members of the test lists. In this case, the objects are definitely not colliding, and the collision is resolved. The collision is removed from the active collision list and destroyed.

3. An intersection is detected between the target sphere and a test sphere of at least one sphere hit, but at most one of the spheres is a leaf, so the collision test is non-conclusive. In this case, new sphere hits are created, and the collision remains on the active collision list for further processing if needed.

4. During the iteration, the application indicates that it wishes to interrupt collision processing. In this case, if the sphere hit list is non-empty, the entities are deemed to be colliding, and the collision is removed from the active collision list and added to the real collision list.

Although our application creates 4 levels of sphere trees for every object, the intersection algorithm has been designed to handle intersection tests between two sphere trees of unequal height. This is handled by stopping the crossover in the algorithm. In the standard case, we add a new sphere hit between the children of the old target, and the intersecting member of the test list. However, if the old target has no child, the new sphere hit is created with the old target remaining as target, and the test list becomes the child list of the intersecting member of the old test list. The target remains the same until an intersection is detected between it and a leaf of the other sphere tree, or until an interruption occurs. This means that, in the future, more complex objects can be approximated by more levels of more spheres, and more simple objects by only a few levels (or even just one, in the case of spherical objects).

4.4 Measuring Inaccuracy and Prioritising Collisions

This application provides a framework within which different collision detection, prioritisation and scheduling algorithms may be implemented and evaluated for computational and perceptual performance. We now need metrics to determine collision inaccuracy in each frame in order to evaluate the effectiveness of our techniques. In order to justify a thorough investigation of the factors which affect human perception of collisions, we first need to show that our strategy of using a perceptual model to schedule collision processing is feasible and useful. From our discussion of visual processing in Chapter 3, we can develop a ***plausible*** perceptual model of collision detection. This model may then be used both to prioritise collisions, and also to estimate perceived inaccuracy. In this way, we can test the feasibility of our approach, and also focus on the type of psychophysical data we wish to gather.

When considering the inaccuracy present in a frame of an animation, we must distinguish between **geometrical inaccuracy** Δ, and **perceived inaccuracy P**. The geometrical inaccuracy in a scene is an estimate of the overall three-dimensional error that has been incurred by accepting non-collisions as real, causing entities to repulse without touching. However, not all collision inaccuracies contribute equally to the inaccuracy perceived by the viewer in a single frame of an animation. Hence, the perceived inaccuracy **P** present in two frames of an animation with identical geometrical inaccuracy Δ**,** may be quite different depending on how the frame is viewed.

4.4.1 Overall inaccuracy: Δ

Calculating the overall collision detection inaccuracy present in a frame is a much simpler task than calculating the perceived inaccuracy, as it is purely a function of the positions of the entities relative to each other in three-dimensional space.

To illustrate this concept, let us consider an example in 2 dimensions. In the scene in Figure 4.4, there are 6 pairs of objects deemed to be colliding, even though they are not actually touching. Typically, these objects would not be spheres, but more complex objects. The spheres are simply there to illustrate the concept. The actual separation, or gap size, between the entities in each collision is also shown.

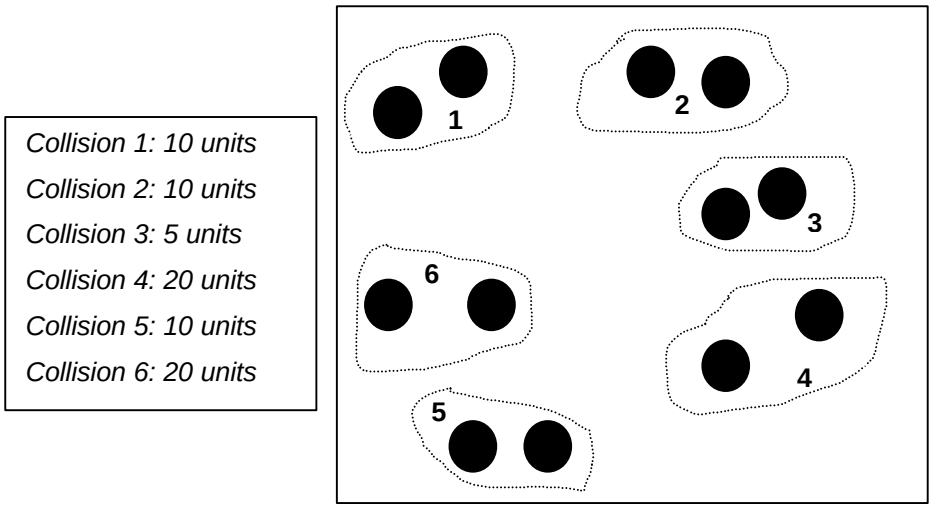


Figure 4.4: A 2D depiction of a scene with erroneous collisions

We have defined geometrical inaccuracy Δ for a frame as being an estimate of the overall three-dimensional error that has been incurred by accepting non-collisions as real, causing entities to repulse without touching. We can estimate this error by expressing it as the sum of the sizes of all potential gaps left during such "non-collisions":

$$\Delta = \sum_{i=1}^{N} g_i$$

where N is the number of collisions, and g_i is the gap-size between the entities in collision i. The value of Δ for the scene in Figure 4.4 is 75.

So, how do we calculate the separation between two non-colliding entities? If we could do this completely accurately, we wouldn't need interruptible collision detection, because we would then know whether all objects are colliding or not. In our applications, we cannot calculate the exact size of the gap between two entities, as this would take an excessive amount of time and completely defeat the purpose of approximate collision testing. Instead, we can use the information available to us to estimate an upper bound on the maximum separation that might occur between two entities that have been erroneously detected as colliding.

Consider the potentially colliding pairs in Figure 4.5. The highest level spheres of both objects in all three cases are overlapping. Therefore, if collision processing were interrupted at this level, all three collisions would be deemed to be real collisions. In cases (a) and (b), there is the same level of intersection of the spheres, but the separation between the objects in (a) is much larger that that in (b). However, ***we have no way of detecting this difference***, because we have interrupted collision handling at a point where this information is unavailable. However, we do know that the spheres are intersecting, and we have enough information to know to what extent they are intersecting (i.e. radii and distance between the centres). We therefore have several choices available to us to estimate the potential error in this collision.

The minimum separation possible would be 0, because the objects may actually be touching (case (b)). However, it is also possible that the objects are so oriented as to maximise the possible gap between them (case (a)). No matter what the shape of the object, the largest separation possible between two objects whose bounding spheres intersect, is the distance between the centres of those spheres. This distance is therefore an upper bound on the maximum erroneous separation between the two objects, i.e. a worst-case estimate of the error. We can see that in case (c), the spheres are closer together, hence the estimated error will be smaller. An alternative method of calculation the error could be to pre-compute the Hausdorff distance[1] for all spheres in each tree, and use the sum of these distances as our estimate.

Now consider the situation when collision processing is allowed to progress to a higher level of accuracy, for the same objects at the same orientations. From Figure 4.6. it can be seen that in this situation the collision in case (a) will be fully resolved at this stage, as it has been definitely determined that the objects cannot be colliding. The collisions in cases (b) and (c) will still be accepted as real, but what should we use as the error estimate in this case?

---

[1] The Hausdorff distance [Preperata and Shamos 1985] is the maximum over all points on the sphere's surface, of the minimum distance from that point to the enclosed object. It thus gives a more accurate upper-bound of the potential gap left between two objects whose bounding spheres have touched, especially if the objects are not very well fitted by the sphere. [Hubbard 1996] however, had difficulty developing an algorithm to compute this measure correctly for non-convex polyhedra, so an approximation was used.

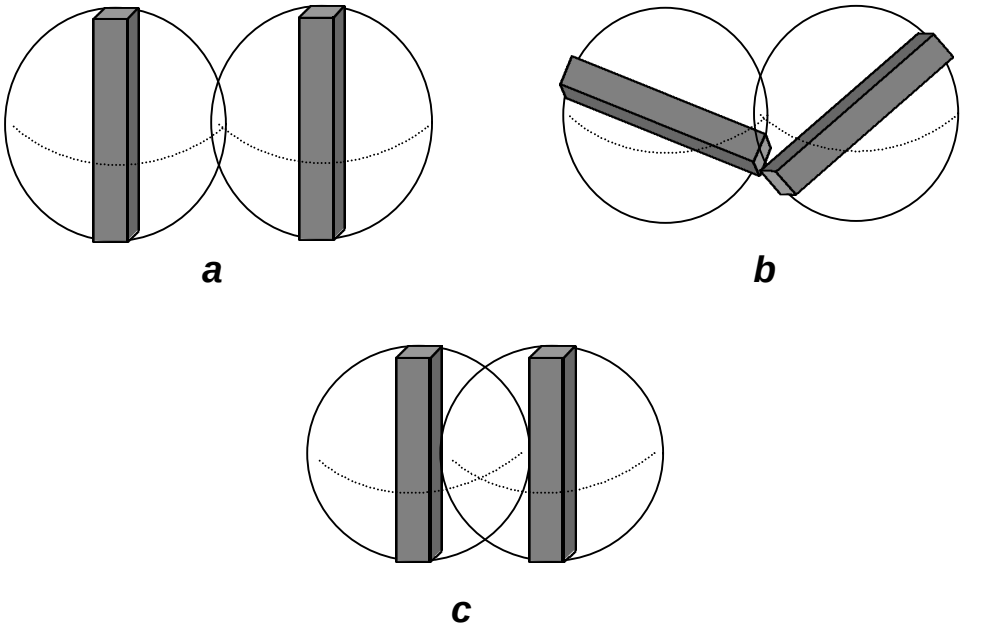


Figure 4.5: Three possibilities for collisions detected at lowest level of accuracy

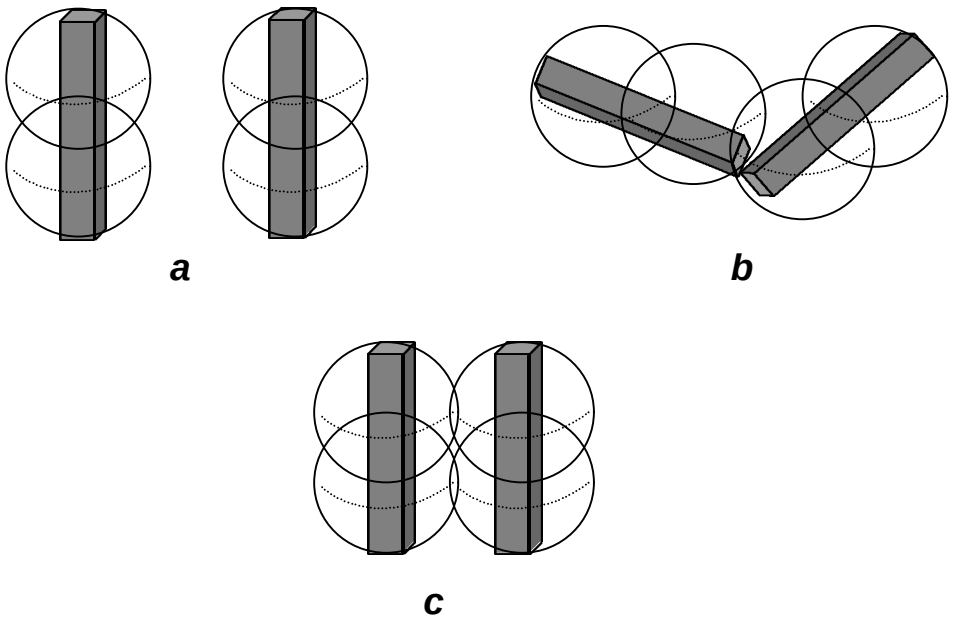


Figure 4.6: Three possibilities for potential collisions allowed to proceed to a higher level of accuracy

Further information is now available, i.e. the distance between spheres lower down in the sphere tree hierarchies which have been determined as intersecting. The simplest estimate of the error in this case is to take the distance between the centres of the last two spheres detected as intersecting. Using this estimate, the error in case (b) will be estimated to be smaller than when processing was interrupted at the bounding sphere level. In case (c), the error will remain the same, but this is because of the parallel orientation of the spheres relative to each other. In the majority of cases, the error estimated at this level will be significantly smaller. This method of estimation is extended to incorporate all collisions interrupted at all levels of accuracy.

To summarise, we use the information available to us to estimate an upper bound on the maximum erroneous separation or gap size between two entities. We find the three-dimensional distance between the centres of the last two spheres found to be intersecting from the sphere-trees of each colliding pair, or the distance between the centres of the two entities if collision testing is interrupted before any spheres have been tested. The geometrical inaccuracy ∇ in a frame is the sum of these distances. As the collision detection algorithm descends deeper into the sphere trees of the colliding objects, the estimated error decreases, as is appropriate.

4.4.2 Perceptually-weighted Inaccuracy: $\mathbf{P}$

In a given frame of an animation of multiple objects, some of which are colliding erroneously, not all collision inaccuracies contribute equally to the inaccuracy perceived by the user. From our pilot experiments, our study of vision literature, and our observation of the application running in its base form, the most obvious factors which make a “bad” collision noticeable are eccentricity and separation. Eccentricity is the on-screen distance of the collision from the viewer's fixation point. In this case, we use separation to mean the erroneous gap between the on-screen projection of two objects falsely detected as colliding. It should be obvious, therefore, that the estimate of perceived inaccuracy will therefore be based on ***two-dimensional*** information. The exact relationship between eccentricity and gap-size remains to be determined by psychophysical means, and from our previous discussions, it is highly probable that these factors alone are not sufficient to estimate perceived collision inaccuracy.

However, the interactions between the many potential factors are likely to be so complex, that we could spend years testing them before coming up with a definitive model. Instead, we can hypothesize what this relationship might be for the purposes of our feasibility tests, and then use the results of these tests to direct our future psychophysical investigations. There is no point in performing exhaustive psychophysical tests to create a model, only to discover that the overall strategy is of no use. Instead, we define a plausible model, which can subsequently be changed and improved if it is shown to be a useful tool.

In order to calculate the eccentricity of a collision, we need to find the two-dimensional distance in screen units between the collision and the user's point of fixation. If we track the user's gaze, we will know the fixation point F**,** expressed in screen co-ordinates, for each frame. Alternatively, for testing purposes we can fix this point in the centre of the screen (which is actually suggested in [Funkhouser and Sequin 1993] as being the most likely point of fixation in any case), or attach it to some very noticeable object in the animation. The question is now: to what part of the collision do we measure the distance? Should we measure the distance to the nearest point, or to the furthest point, or to the mid-point? The latter option would seem to be the most suitable, but what if the intersecting bounding spheres are of varying size, as in Figure 4.7.

A better approach would be to determine the 3D line which joins the two centres, c1 and c2, of the intersecting bounding spheres, and then to find the segment of this line which lies inside the intersecting region. Then find the mid-point of this line segment, which we will call the ***Centre of Collision:*** **C.** We calculate this point as follows:

- Let dx, dy and dz be the differences between the x, y and z co-ordinates of c1 and c2. Then the 3-dimensional distance between the two objects centres is:

$$d_{3d} = \sqrt{dx^2 + dy^2 + dz^2}$$

- Now we want to find the points of intersection of the line between c_1 and c_2, and the spheres themselves. Let t_1 be the distance along the line from c_1 to the point of intersection with sphere 1, and t_2 the distance from c1 to the point of intersection with sphere 2. We can see from Figure 4.8 that:

$$t_1 = \text{radius}(\ \textit{sphere 1}\ )$$

$$t_2 = d_{3d} - \text{radius}(\ \textit{sphere 2}\ )$$

- Substituting into the parametric equation for the line, where $c_1 = (cx_1, cy_1, cz_1)$ and $c_2 = (cx_2, cy_2, cz_2)$ and the co-ordinates of d, i_1 and i_2 are similarly defined:

$$i_1 = c_1 + d \cdot \frac{t_1}{d_{3d}}$$

$$i_2 = c_2 + d \cdot \frac{t_2}{d_{3d}}$$

- The centre of collision C = (c_x, c_y, c_z) is the midpoint between points i_1 and i_2:

$$C = i_1 + \frac{i_2 - i_1}{2}$$

- We find the projection of the point in two-dimensional screen co-ordinates, v = (v_x, v_y) using the gluProject function[2], and then finally find the eccentricity **e** by finding the distance between v and the fixation location **F** = (f_x, f_y):

$$e = \sqrt{(v_x - f_x)^2 + (v_y - f_y)^2}$$

[2] As mentioned above, all rendering is achieved through the use of the OpenGL™ library, which provides the function GluProject. This function transforms the specified object coordinates into screen coordinates, using the information about current viewing parameters such as the type of projection (parallel or perspective), and position of the virtual camera.

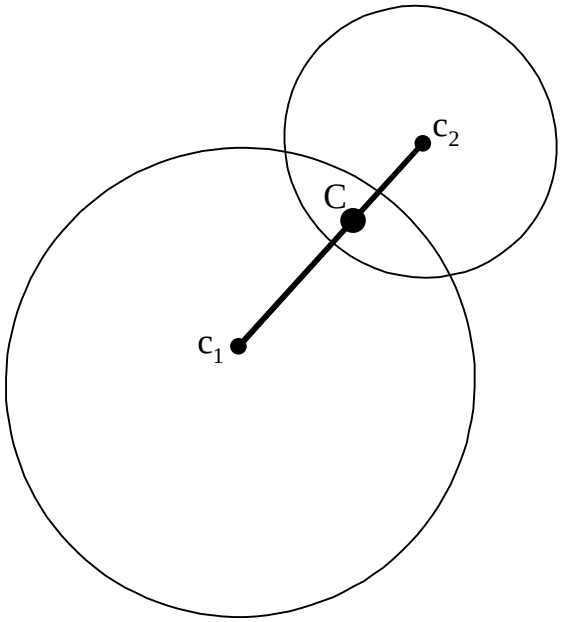


Figure 4.7: The location of the Centre of Collision, C

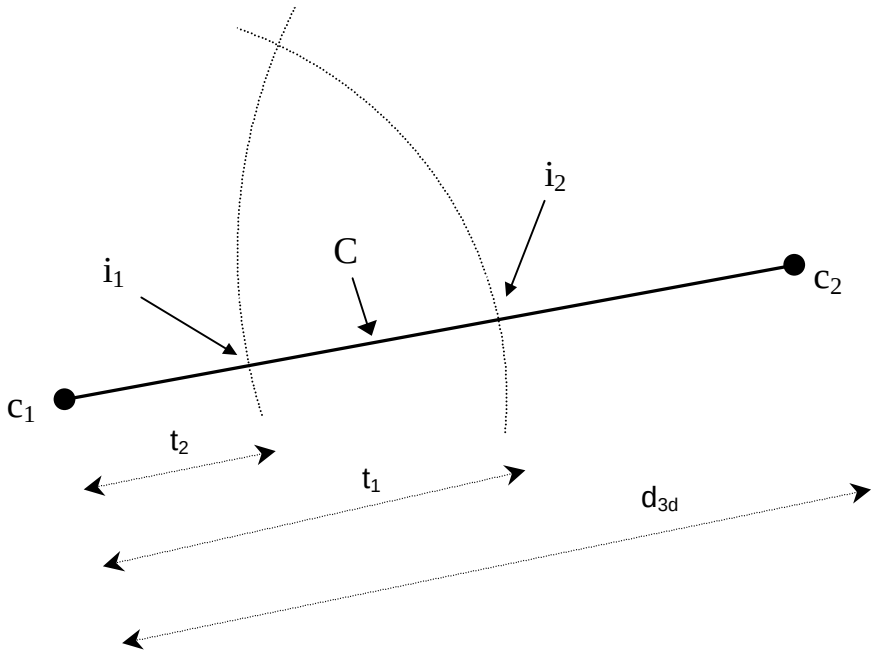


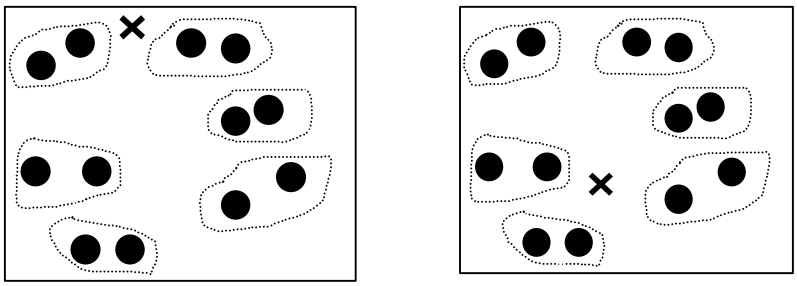
Figure 4.8: Calculation of the Centre of Collision, C

We will use this measure of eccentricity to weight the contribution of each erroneous collision to the inaccuracy perceived by the user: Collisions closer to the fixation point contribute more to the perceived inaccuracy of a frame than those further away and hence should receive higher weighting. Collisions further away should receive lower weighting. Similarly, the on-screen separation may also be used to weight each collision in some way, with larger gaps contributing more to inaccuracy than smaller ones. In a similar way to the way we measured the maximum three-dimensional error present in an incompletely-processed collision, we can calculate an upper bound on the 2-dimensional separation as follows: We take the centres of the last two spheres found to be intersecting as before, but this time we find the x, y locations in screen co-ordinates of their projections (again using gluProject). We then simply find the 2-dimensional distance between them. We will refer to this two-dimensional gap as g' in order to distinguish it from the three-dimensional error estimate g.

We now have two metrics to measure the separation and eccentricity of a given collision. We now wish to find an appropriate way of modelling the interaction of these factors within a given frame of an animation of many colliding objects. We can express this perceptually weighted measure of inaccuracy as follows:

$$\mathrm{P} = \sum_{i=1}^{N} f\left(g_i^{'}, e_i\right)$$

where N is the number of collisions in a scene, $g_i^{'}$ is the estimated 2-dimensional error gap size between the entities in collision i, and e_i is the eccentricity of the centre of collision i from the fixation point **F**.



a **b**

Figure 4.9 two frames, identical except for the fixation point.

One possible estimate of the above function ***f*** is:

$$f_1\left(g_i^{'}, e_i\right) = \begin{cases} \dfrac{g_i^{'}}{e_i} \text{ if } e_i \geq 1 \\ \\ g_i^{'} \text{ if } e_i = 0_1 \end{cases} \quad : Z \times Z \rightarrow R$$

This provides a plausible model of how the visual system might work. At equal eccentricities larger gap sizes contribute more to the overall inaccuracy and at equal gap sizes larger eccentricities contribute more. For example, look at both frames in Figure 4.9. The cross indicates the location of the fixation point **F**. Table 4.1 shows the calculated values for frame **a** in Figure. 4.9. Table 4.2 shows the values for the frame **b**. The positioning of the entities in both cases is identical, and hence so is the value of Δ. However, the perceptual value $\mathbf{P}$ of the frame in **a** is better than in **b** because the fixation point **F** in **b** is closer to the larger collision gaps, which will therefore be more noticeable.

i	$\mathbf{g_i}$	$\mathbf{e_i}$	$\mathbf{g_i/e_i}$
1	10	30	0.333
2	10	30	0.333
3	5	50	0.100
4	20	60	0.333
5	10	80	0.125
6	20	90	0.222

Table 4.1, $\boldsymbol{P}$ *= 1.446,* D *= 75*

i	$\mathbf{g_i}$	$\mathbf{d_i}$	$\mathbf{g_i/d_i}$
1	10	60	0.166
2	10	60	0.166
3	5	50	0.100
4	20	50	0.400
5	10	30	0.333
6	20	40	0.500

Table 4.2, $\boldsymbol{P}$ *= 1.665,* D *= 75*

For testing purposes, both gap size and eccentricity are measured in pixels on a monitor of size 37.5x30 centimetres at a resolution of 1024x768 pixels. Figure 4.10 shows the above function, $\boldsymbol{f_1}$, for a range of eccentricities and gap sizes. However, we can see that this function decreases very rapidly up to value 20, which corresponds to less than a centimetre on the screen. This assumes a very dramatic fall-off in human ability to detect gaps at increasing eccentricity from the fixation point. Pilot studies have indicated that for larger gap sizes, this fall-off may be more gradual. In addition, with the use of a highly accurate and intrusive eye-tracking device, we can guarantee that the estimated fixation location is within a millimetre or two of the actual fixation. However, if we use a tracker with lower spatial resolution, and do not employ the use of bite-bars and other such restraints, the best accuracy we can achieve may only be within a circle of several millimetres diameter. Therefore, the above model may be too refined for our purposes. Nevertheless, if the visual and mental task at the viewer's point of fixation is very complex, such a "tunnel vision" effect as described by the perceptual function $\boldsymbol{f_1}$ may be evident. Another possible estimate of the perceptual function could be:

$$f_2\left(g_i^{'}, e_i\right) = \frac{g_i^{'}}{\exp\left(\frac{e_i}{C}\right)}$$ where C is a constant.

This function allows a more gradual fall-off, and different values of the constant C allow us to control the shape of the function. Figure 4.11 plots this function for different values of **e**, **g** and **C**. It may well be the case that neither of these models are a suitable approximation, or it may be that both may be appropriate, depending on the situation. We may also find that it is not possible to model the perceptual function with one function alone, but with several. For example, we may discover from our psychophysical experiments that a function such as the following (shown in Figure. 4.12) is more appropriate:

$$f_3\left(g_i^{'}, e_i\right) = \frac{g_i^{'}}{\exp\left(\frac{e_i}{C^{'}}\right)} \qquad C^{'} = \begin{cases} C_1 \text{ if } e_i < 100 \\ C_2 \text{ if } e_i < 200 \\ C_3 \text{ if } e_i \geq 200 \end{cases}$$

We are not yet in a position to state which, if any, of the above models is the most appropriate. In fact, the final model will almost definitely be more complex than the ones we have just proposed. However, it is now possible to test our application with respect to a plausible model of perception, allowing us to manipulate frame times and scheduling mechanisms, and test their effect on the viewer, as measured by our hypothetical perceptual metric. At a later stage, a more accurate model of perception of collision inaccuracy, based on psychophysical data, can be used. This initial process may be viewed a study of the feasibility of using perceptual metrics to improve the perception of an animation which uses interruptible collision detection.

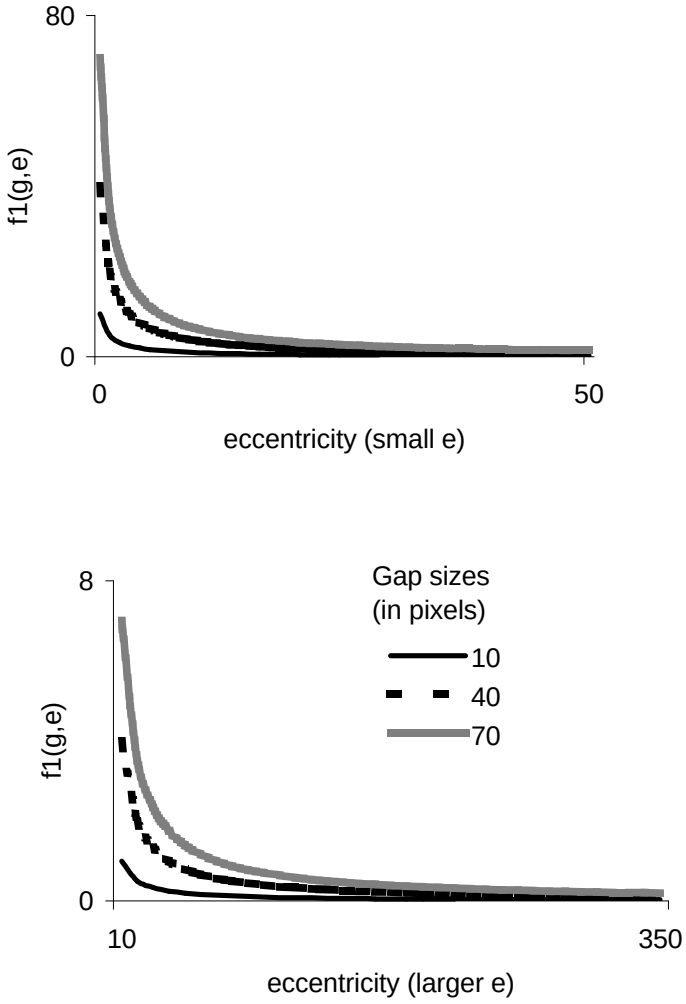


Figure 4.10: Perceptual function f_1.

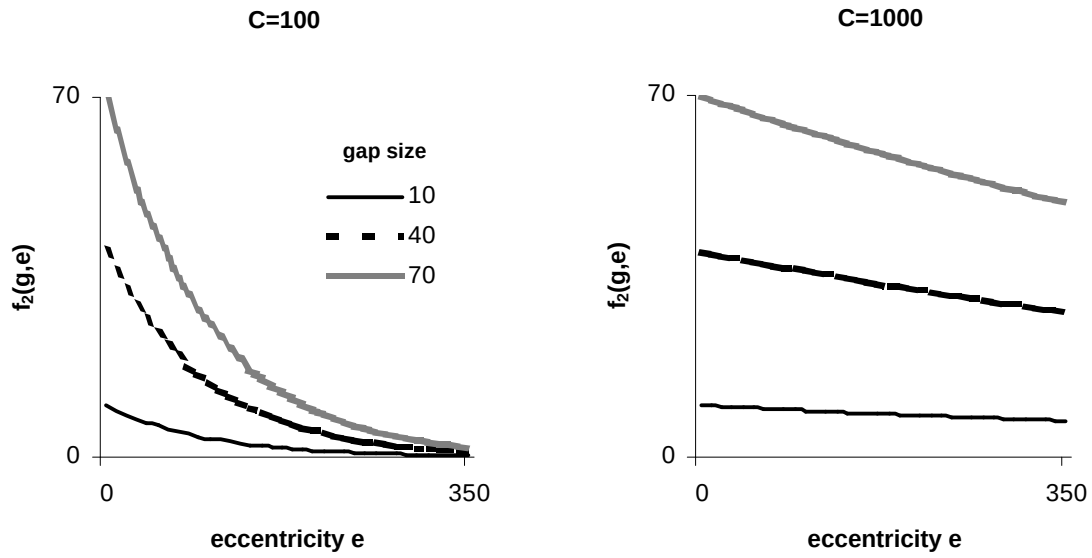


Figure 4.11: Perceptual function f_2 for different values of C.

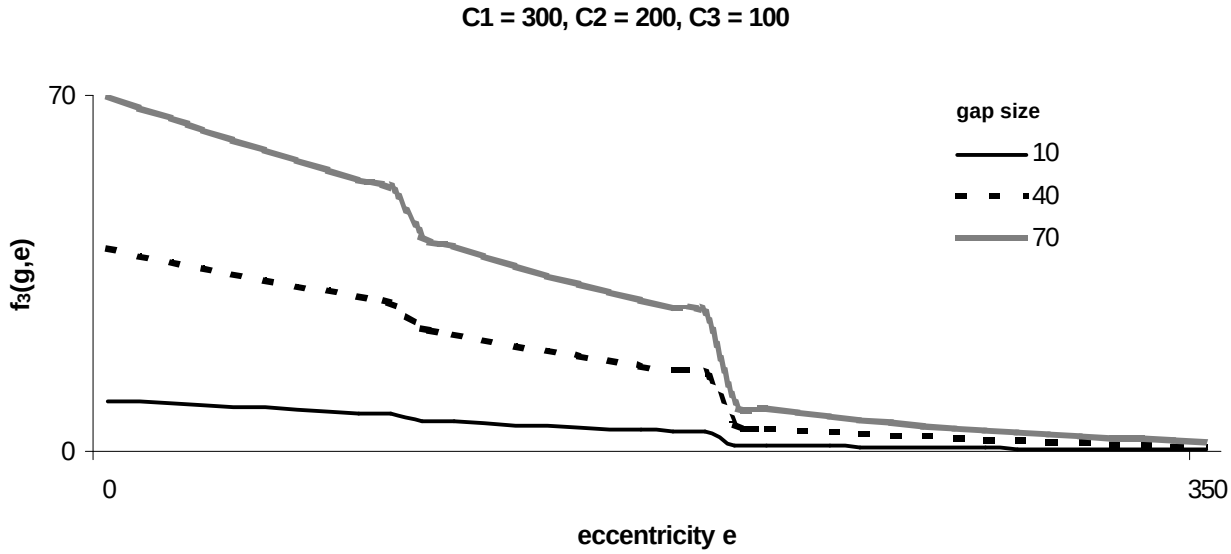


Figure 4.12: Perceptual function f_3

4.4.3 Collision Priority

As an initial pass at estimating the perceptual importance of a collision, we use the metric **P** described above. It is possible to choose any of the three estimates of ***f***, to determine if and how they impact upon performance. In addition to eccentricity and separation, this metric takes account of three additional factors that could affect collision perception:

- Distance from the viewpoint: P is a 2-dimensional metric, calculated from the perspective projection of 3-d intervals onto a 2-dimensional plane. Thus, the further away a collision is, the smaller the projection of an erroneous gap will be.
- Size of objects. The larger the objects are, the larger the estimated erroneous gap between them will be.
- Visibility of collision points / angle between the colliding entities. If two entities are oriented towards each other along a line orthogonal to the viewer, it will not be possible to tell whether they are actually touching or not, because one of the objects will occlude the points of collision. In this case, the error gap between them when projected onto the view-plane will be zero. If however, the objects are oriented towards each other on a plane parallel with the view-plane, any inappropriate separation between them will be most noticeable. In this case, the projection of the erroneousgap will be at a maximum. As the angle between objects changes from orthogonal to parallel, the projected gap will also continuously increase from 0 to the maximum.

This perceptual metric is by no means conclusive, but it provides a good starting point to test our application, and to establish the principle that using perceptual information can help to increase the realism of collision handling.

4.5 Scheduling

The key to controlling the collision inaccuracy perceived by a viewer in a given frame of an animation lies in the scheduling method adopted. We have seen in previous sections that upon completion of the broad phase of collision testing, an active list of collisions exists, one collision object for each pair of entities whose bounding boxes overlap.

[Hubbard 1995] recommends resolving such collisions in round-robin order, descending one level in the hierarchy of every sphere tree at each iteration of the algorithm, until interruption occurs. However, no account is taken of the perceptual importance of each collision. We will refer to this strategy as ***round-robin scheduling*** from now on. Another strategy, ***sequential scheduling,*** is to start at the first collision and fully resolve each collision in turn, before moving onto the next collision, until completion or interruption. Again, perception is ignored with this strategy.

In ***perceptually sorted sequential scheduling***, a perceptual importance is attached to each collision, using the metric **P** described above, and the active collision list is sorted based on this priority. We use a version of quick-sort adapted for linked lists to achieve this. The list is subsequently processed sequentially, as in sequential scheduling, but collisions which are most important perceptually will be resolved first, leaving the more unimportant collisions to be resolved only if there is time left.

Finally, we can generate not one active collision list, but a set of priority queues, and round robin within them. A higher priority queue would be resolved first, and only when all collisions on that queue have been resolved would the next highest queue be processed. This is called ***priority queue scheduling***. This strategy would be good for handling collisions of equal perceptual importance. Adaptations of this mechanism are possible, where one or more of the priority lists are also sorted, perhaps allowing those close to the fixation point to be sorted, while those outside a certain limit all have equal priority. We will call this adaptation ***priority sorted queue scheduling***.

Our application has been designed to support any of these scheduling strategies. As we learn more about human visual perception of collisions, we will discover which scheduling strategy is most appropriate in a given situation, or we may indeed come up with other scheduling strategies.

5 Analysis and Performance

This chapter describes some experiments which were carried out to test the collision detection algorithms described in the previous chapter. The aim of these experiments was to establish the feasibility and usefulness of our approach. It was essential to evaluate the techniques proposed, in order to justify and guide the later psychophysical studies.

5.1 Introduction

A suite of experiments were designed to examine the following issues:

- Non-interruptible collision detection vs. Interruptible collision detection. Is interruptible collision detection feasible?
- Perceptual interruptible collision detection vs. Non-perceptual interruptible collision detection. Does a perceptual model help to improve perceived inaccuracy?

In the former case, the criteria for "good" performance were frame-rate, variation of rate, and inaccuracy caused by interruption. In the latter case, frame-rate is kept constant, and perceived inaccuracy is what is being measured. Some questions that need to be answered when running these experiments are as follows:

– What kind of frame-rate can we expect if we don't interrupt our collision detection algorithm, and is there any room for improvement?
– Can the frame-rate be improved upon by interrupting the algorithm?
– If we do interrupt, what kind of inaccuracy are we introducing?
– Does the use of perceptual information help to reduce this accuracy?
– How do we set the parameters of interruption to optimise frame-rate vs. inaccuracy? E.g. how should we set the target time for collision detection, that maintains an acceptable frame-rate and an acceptable level of accuracy?
– What scheduling strategy is most suitable under different circumstances?
– What effect does the choice of perceptual model have?

Among other options, the application was designed to run with or without graphics, and with or without interruption. To test its performance, we ran the application with 10, 30, 100, 300, and 500 objects respectively. To isolate the effects of collision detection, each test was run with no graphics, and only the time taken to perform collision handling was measured. This included broad-phase testing (i.e. sweep and prune), generation of the active collision list, detection of collisions, and collision response determination.

We recorded results from 5000 frames for each number of objects. For each frame we recorded:

- **T** : The time spent on collision handling
- **B** : The number of broad-phase collisions
- **R** : The number of real collisions
- **L1, L2, L3, L4** : The number of narrow phase tests that were resolved at each of levels 1, 2, 3 and 4 of one or other sphere tree.
- Δ : The overall inaccuracy
- **P** : The perceived inaccuracy

The experiments ran on a Pentium PC, 233 MHz, running Microsoft Windows '95. As this is a pre-emptive multi-tasking environment, each test was started with the process priority: REALTIME_PRIORITY_CLASS, and all other non-necessary processes were ended. This was done to minimise the disruption caused by other processes taking up CPU time while the experiment was running, and allow for more accurate assessment of the collision detection performance. We used the Win32 system call: GetCurrentTime() to measure the time elapsed in milliseconds. The object shapes and sizes were identical in each test, and in order to enable fair comparisons between tests with different numbers of objects, we had to ensure that the density of objects inside each volume was the same. We wanted to create a worst-case scenario, where there were many collisions at each frame. Therefore, the objects had to be tightly packed into the volume. Packing 10 objects of size 0.3 x 0.3 x 0.3 into a volume of size 1x1x1 (virtual units) ensured a high number of collisions at each frame. Therefore, we increased the size of the volumes for larger numbers of objects proportionally, i.e. 30 objects were packed into a volume of size 1.44, i.e. the cubed root of 3; 100 were packed into a volume of size 2.15, i.e. the cubed root of 10, and so on.

Volumes were centred on-screen, and the front was always the same distance from the viewpoint, i.e. at position 10 along the z axis. The size of the screen viewport was identical in each case, i.e. full screen size (see Figure 5.1). See Table 5.1 for the exact configurations.

Objects	Size	xmin	xmax	ymin	ymax	zmin	zmax
10	1.00	-0.50	0.50	-0.50	0.50	10.00	11.00
30	1.44	-0.72	0.72	-0.72	0.72	10.00	11.44
100	2.15	-1.08	1.08	-1.08	1.08	10.00	12.15
300	3.11	-1.55	1.55	-1.55	1.55	10.00	13.11
500	3.68	-1.84	1.84	-1.84	1.84	10.00	13.68
1000	4.64	-2.32	2.32	-2.32	2.32	10.00	14.64

Table 5.1 Volume configurations

5.2 Interruptible vs. Non-interruptible collision detection

The aim of this set of experiments was to determine whether interruptible collision detection is feasible. In order to do this, we first ran all experiments with no interruption. Then we repeated all tests, this time interrupting at 0, i.e. stopping processing after broad-phase testing only. Based on these results, we chose a target time X for each number of objects, and interrupted collision handling at this time.

5.2.1 No Interruption

This set of experiments tested collision processing performance at the highest possible level of accuracy, i.e. processing every collision to the deepest level of each sphere tree necessary for rejection or acceptance of the collision as real. The mean results for all measured parameters are shown in Table 5.2. Figure 5.2 provides a graphical overview of these figures. In order to reach an overall target frame rate of at least 10 frames per second, the collision handing should take up only a part of the 100 milliseconds available per frame. Therefore, in the absence of any load balancing schemes, we can estimate that at most 50 milliseconds should be spent on collision handling per frame.

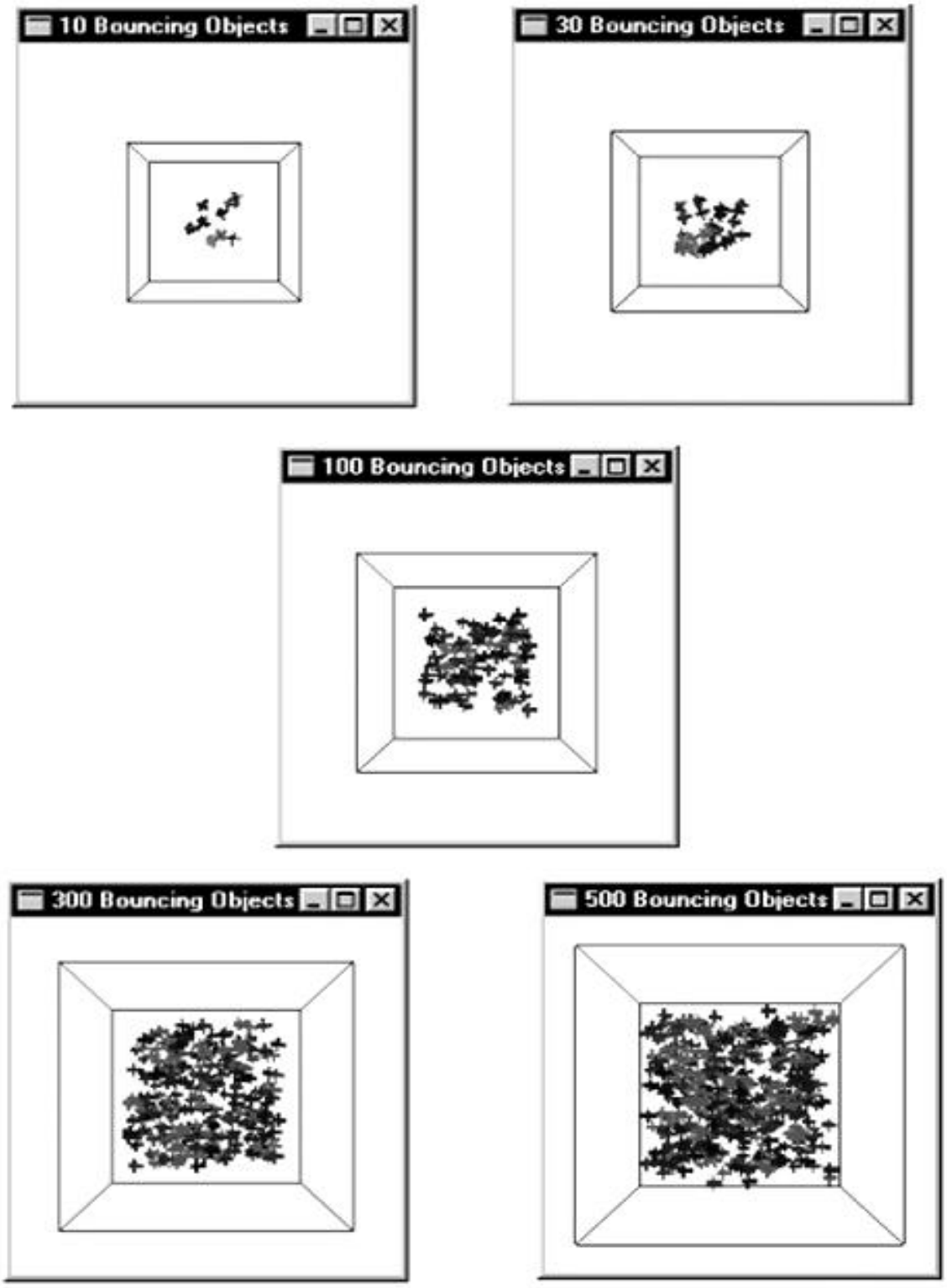


Figure 5.1: On-screen projections of experimental volumes and objects.

Figure 5.2(a) plots the mean time taken for collision handling in each frame of the animation for each number of objects. We can see that collision handling times are unacceptably high for 100, 300 and 500 objects. Table 5.3 shows the mean min, max, variances and standard deviations. We can see from this data that with large numbers of objects, our frame times will be too slow and variable for real-time performance. Hence, interruption is definitely needed to achieve a high and constant frame rate. However, we need to measure what effect this will have on the accuracy of the collision detection, and the resulting collision response. This will need to be measured both in terms of total or geometrical inaccuracy, and of perceived inaccuracy.

Figure 5.2(b) shows the number of broad-phase collisions detected per frame, and the number of real collisions that were detected after full collision detection. It can be seen that a large number of potential collisions eliminated before the end of each frame. Figure 5.2(c) shows exactly where in the collision testing phase each of these collisions were eliminated or accepted. Looking at the difference between the number resolved at a level and the number of real collisions shows us how many have been eliminated at that level. We can see that the largest number of collisions are eliminated at level 1 of the sphere trees, i.e. at the bounding sphere level, an equal number are eliminated at levels 2 and 3, and that almost all collisions that descend to level 4 of either sphere tree are actually real collisions, with only a small number being eliminated at that point.

	10	**30**	**100**	**300**	**500**
T	6.86	27.44	117.77	419.00	744.29
B	13.28	67.33	293.54	1057.86	1883.92
R	0.95	3.54	14.98	51.45	90.19
L1	5.05	28.23	130.22	483.29	871.13
L2	3.32	16.76	70.86	252.73	447.40
L3	3.78	18.00	74.30	259.45	456.19
L4	1.13	4.34	18.16	62.39	109.20
Δ	0.04	0.14	0.58	1.98	3.47
P	0.11	0.27	0.77	1.80	2.67

Table 5.2 Results from no interruption

Times	**10**	**30**	**100**	**300**	**500**
Mean	6.86	27.44	117.77	419.00	744.29
Min	0.00	5.00	64.00	295.00	579.00
Max	45.00	85.00	220.00	560.00	975.00
StdDev	5.94	10.01	21.16	39.42	54.56

Table 5.3 Statistics for collision handling times with no interruption

	10	**30**	**100**	**300**	**500**
T	0.23	1.06	4.49	20.53	42.35
B	12.72	65.93	278.20	951.13	1652.71
R	12.21	65.37	277.62	950.58	1652.16
L1	1	1	1	1	1
L2	0	0	0	0	0
L3	0	0	0	0	0
L4	0	0	0	0	0
Δ	5.58	30.88	132.92	457.07	796.91
P	18.37	69.42	193.36	434.87	612.36

Table 5.4: Results from interrupting at 0

Times:	**10**	**30**	**100**	**300**	**500**
Mean	0.23	1.06	4.49	20.53	42.35
Mode	0.00	0.00	5.00	20.00	40.00
Median	0.00	0.00	5.00	20.00	40.00
Min	0.00	0.00	0.00	14.00	34.00
Max	6.00	7.00	10.00	30.00	51.00
StdDev	1.04	2.06	1.57	2.02	2.72
Variance	1.09	4.23	2.46	4.09	7.39

Table 5.5 Statistics for collision handling times when interrupting at 0

	10	30	100	300	500
Time needed for housekeeping:	0	0	5	20	40
Time set for interruption:	5	20	44	30	10

Table 5.6 Times chosen for interruption at X

	at 5	*at 20*	*at 44*	*at 30*	*at 10*
Objects	**10**	**30**	**100**	**300**	**500**
T	3.31	19.35	46.97	51.65	51.59
B	11.71	60.46	245.52	837.06	1695.87
R	2.13	6.46	45.66	397.36	1545.74
L1	4.58	24.89	112.47	417.51	150.68
L2	2.86	15.42	65.34	22.81	0.00
L3	2.46	13.96	22.77	0.00	0.00
L4	0.21	0.73	0.00	0.00	0.00
Δ	0.61	1.22	12.67	154.01	717.97
P	1.53	1.71	11.59	138.21	576.01

Table 5.7 Results for Interruption at X

Times:	**10**	**30**	**100**	**300**	**500**
Mean	3.31	19.35	46.97	51.60	51.59
Mode	5.00	20.00	45.00	50.00	50.00
Median	5.00	20.00	45.00	50.00	50.00
Min	0.00	5.00	44.00	42.00	39.00
Max	10.00	51.00	150.00	125.00	100.00
StdDev	2.40	2.93	3.77	5.08	4.09
Variance	5.75	8.59	14.25	25.80	16.76

Table 5.8 Statistics for Interruption at X

5.2.2 Interrupting at 0

To answer the question of when to interrupt, the same tests as above were run, but this time interrupting collision handling at 0. What this means is that after broad phase testing occurred, and the active collision list was generated, narrow-phase testing was interrupted after the very first intersection test (i.e. between the root spheres of the first two objects), and all active collisions were transferred to the real collision list. In other words, this test is a measure of the time needed for broad-phase testing, list maintenance, and collision response determination (with all potential collisions treated as real). The results are shown in Table 5.4, and plotted in Figure 5.3.

We can see from Figure 5.3(a) that the time for collision handling falls below our target time of 50 milliseconds in all cases. Table 5.5 shows the statistics for collision handling times. The maximum mean time spent on collision handling was 42.35 milliseconds for 500 objects, for which the maximum time for one frame was 51 milliseconds. This compares to a maximum mean time of 744.29 milliseconds in the non-interruptible case, and a longest collision handling time for one frame of 975 milliseconds. Hence, we have shown that it is possible to provide some level of collision handling, for up to 500 objects in our case, within a time that can guarantee a reasonable frame rate. But at what price?

Figure 5.3 (b) shows the plots of the number of broad-phase collisions detected, vs. the number of collisions treated as real. We can see that these two numbers are identical in all cases. This point is also illustrated in table 5.4 where we can see that no broad-phase collisions are eliminated at any level of the sphere-trees (apart from the very first sphere test before interruption occurs), and that all potential collisions are accepted as real collisions. What does this mean in practice? If all broad phase collisions are accepted, this means that objects that are significantly separated from each other will be treated as having touched, and will move apart in the next frame. The effect of this is shown in Figure 5.3(c), with the mean total inaccuracy rising dramatically with increasing numbers of objects to a maximum of 796.91, as opposed to a maximum of 3.5 in the non-interruptible case, shown in Figure 5.2(d), and a corresponding rise in the mean perceived inaccuracy also, to a maximum of 612.36, as opposed to a maximum of 2.67 when we were not interrupting. It is highly probable that this level of inaccuracy would lead to very unrealistic behaviour of colliding entities.

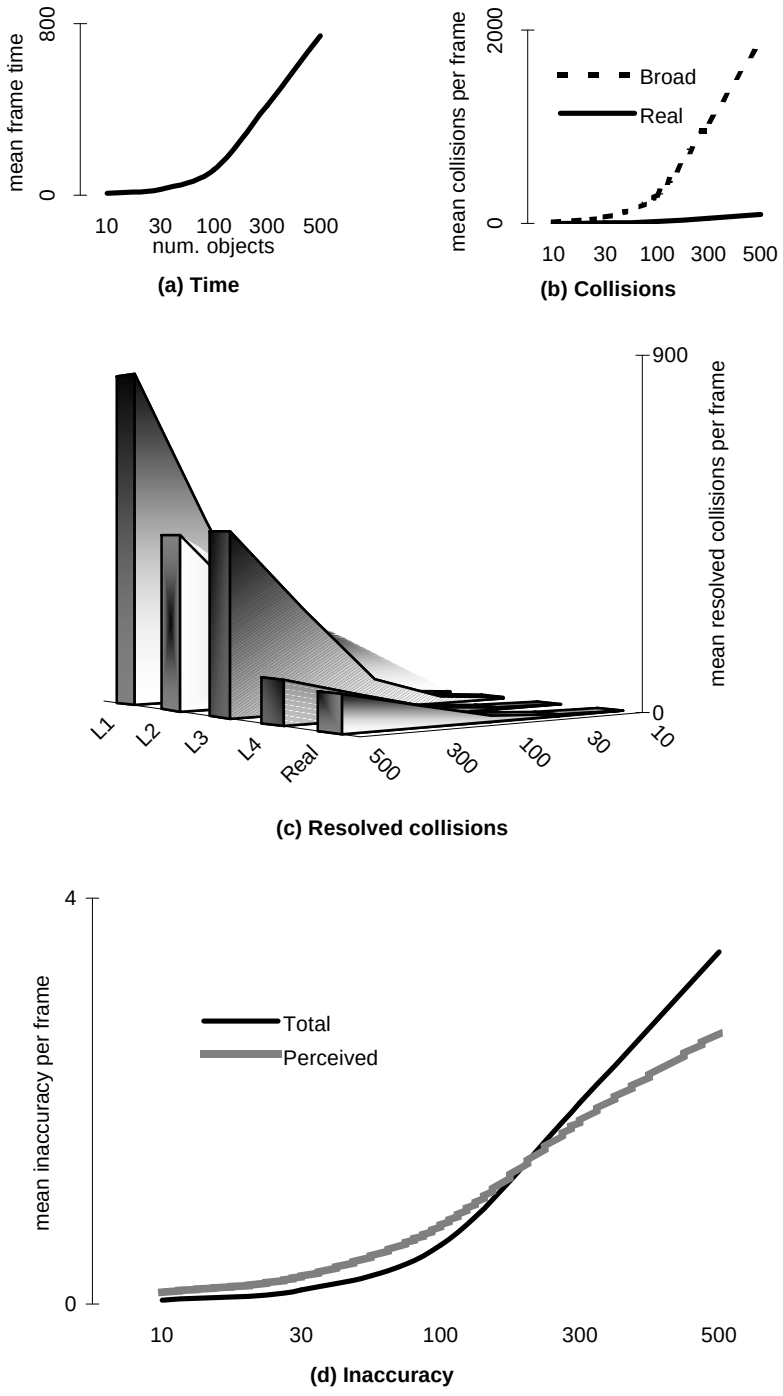


Figure 5.2: Results for no interruption

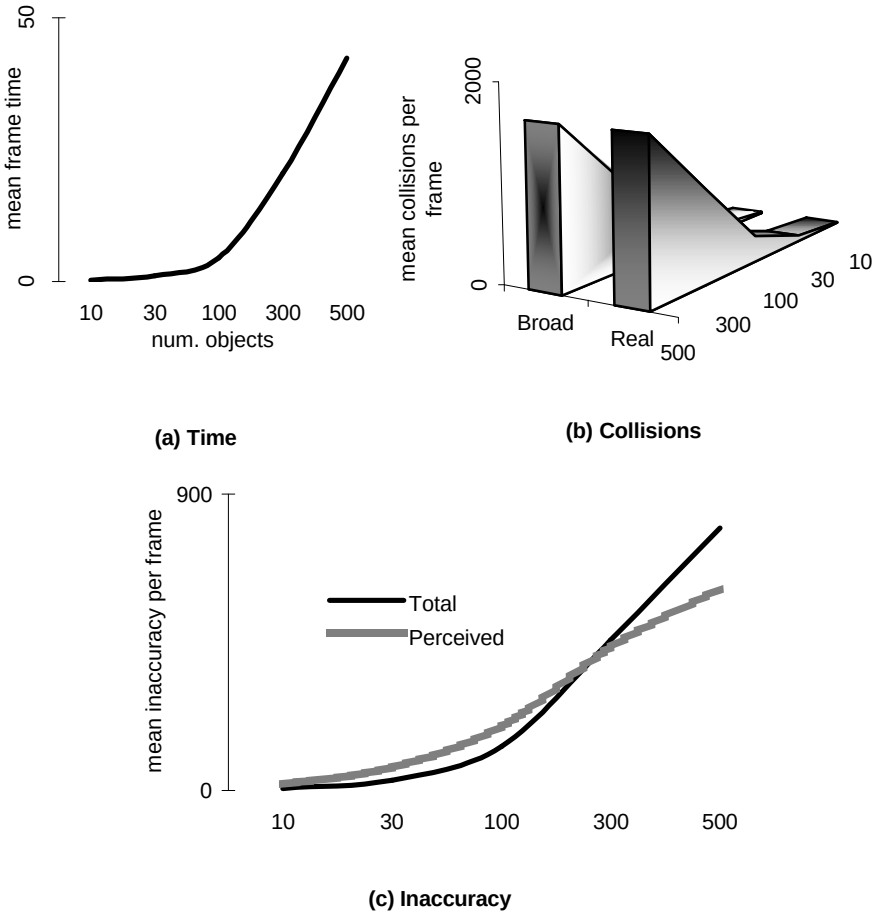


Figure 5.3: Results when collision handling is interrupted at 0

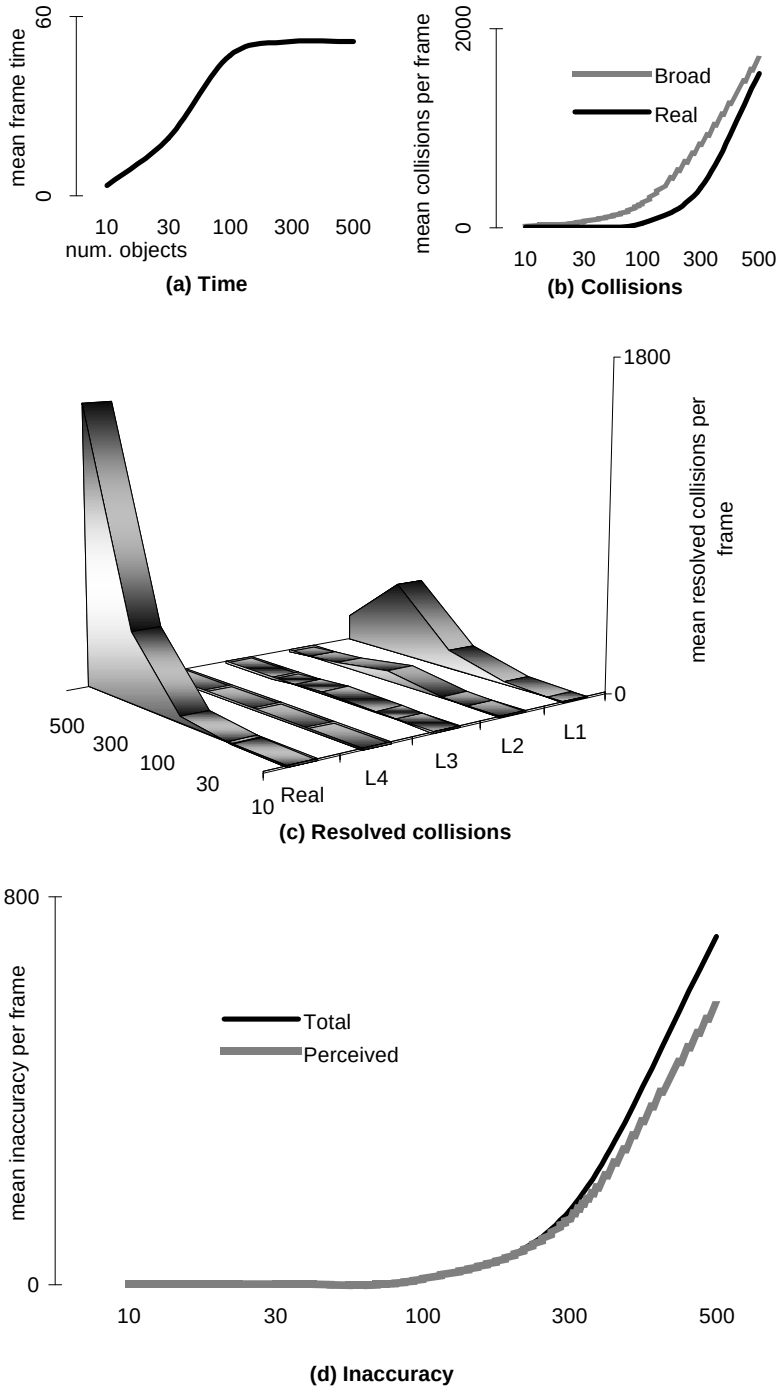


Figure 5.4 Results when collision is interrupted at target time X

It is worth noting at this point, that the units of measurement of total collision and perceived collision are completely different. Total collision is measured in the virtual units of our 3-dimensional world, and is therefore a 3-dimensional metric. Perceived collision is measured in physical units, i.e., the number of screen pixels separating the projected center of two objects or spheres, divided by the number of screen pixels separating the projected centre of a collision from the fixation location of the viewer, (for these experiments, this was held constant at the centre of the screen viewport.) Nevertheless, it can be seen that the two measures are strongly correlated. Our aim will be to reduce this correlation as much as possible, i.e. If we cannot reduce the overall inaccuracy, then we will try to reduce the perception of that inaccuracy.

Obviously, interrupting at 0 is not a feasible method of collision detection, as it is equivalent to accepting all bounding box tests as collisions. The application builds an active collision list after the broad-phase, and interruption occurs immediately, thus causing the creation of the real collision list directly from the active one. This introduces a totally redundant set of overheads. However, it was never our intention to consider interrupting at 0. This test was intended to estimate how long broad-phase testing, collision response, list manipulation and other house-keeping would take. We needed to know this in order to set target times for interruption in the subsequent tests.

5.2.3 Interrupting at X

From table 5.5 we can find the the mean, mode and median times for the tests where interruption was at 0. We use these values to choose the interruption time in each case, to allow as much time as possible for collision handling, while still reaching a target frame time of 10 f.p.s. As previously discussed, an estimate of the maximum time that could be allowed for collision handling is half the frame time, i.e. 50 milliseconds. In the case of 10 objects and 30 objects, this target time was reached with no interruption. However, the variance of frame times was very high in both these cases. Our aim in using interrupting for up to 30 objects is, therefore, to reduce this variance, and hence achieve a more constant frame rate, with the minimum impact on perceived inaccuracy. However, with 100, 300 and 500 objects, we cannot achieve our target frame time without interruption, and hence our aims in these cases are to achieve a target time, reduce variation, and mimimize perceptual inaccuracy.

Table 5.6 shows the time chosen in each case for interruption. The results for interruption at X are shown in Table 5.7, and plotted in Figure 5.4. We can see from table 5.8 that we have, in all cases, achieved a mean time that does not significantly exceed 50 milliseconds (see Figure 5.4(a) vs, Figure 5.2(a) for the no interruption case). We have also managed to reduce the overall number of collisions automatically accepted as real in the at-0 case (see Figure 5.4(b) vs. Figure 5.3(b)) and to eliminate more broad phase collisions at each level of the sphere tree (see Figure 5.4(c) vs. 5.3(c)). Obviously, we have also reduced both the total and perceived inaccuracy from the at-0 case (see Figure 5.4(d) vs. Figure 5.3(d)).

5.2.4 Discussion

The main objective of running this set of experiments was to test the feasibility of using interruptible collision detection to maintain a constant collision handling time. The results are summarized in Figures 5.5 and 5.6. Figure 5.5 shows histograms of the numbers of frames that took a certain amount of time to perform collision handling. For example, looking at the first histogram for 10 objects, no interruption, the first bar indicates the number of frames out of 5000 that spent 0 milliseconds on collision testing. The second bar represents the number that took greater than 0 and less than or equal to 2 msecs to perform collision testing, the third bar represents >2 and <=4, and so on. We can see from the first column of histograms that there was a large range and variation of collision handling times when no interruption was used.

The range and variation becomes much smaller when collision handling is interrupted at 0 (second column). The third column of histograms represent the collision handling times when we interrupted at our target time. For 10 and 30 objects, we have not significantly reduced the time of the majority of frames, but we have eliminated those times which were increasing the variation, i.e. the frames that were taking significantly longer than the mode. For 100 objects, we have completely achieved our goal of making almost all frames spend 50 milliseconds or under on collision handling. For 300 and 500 objects, we have almost reached this goal, with some of the times seeping into the next bar. By further fine-tuning of the target time, we could reduce this time even further. Figure 5.6 allows a further comparison of the collision handling performance for all parameters.

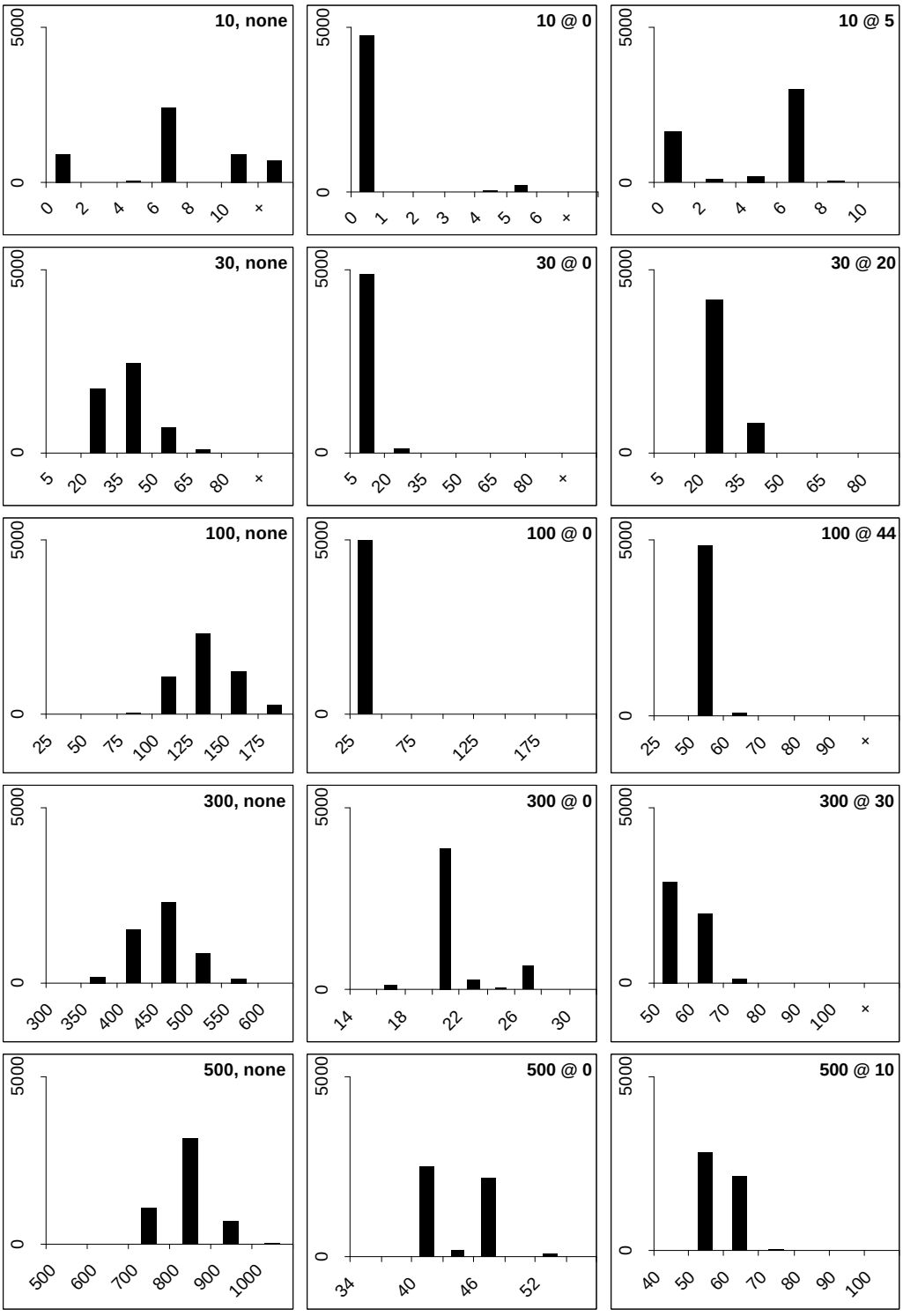


Figure 5.5 Histograms of collision handling times

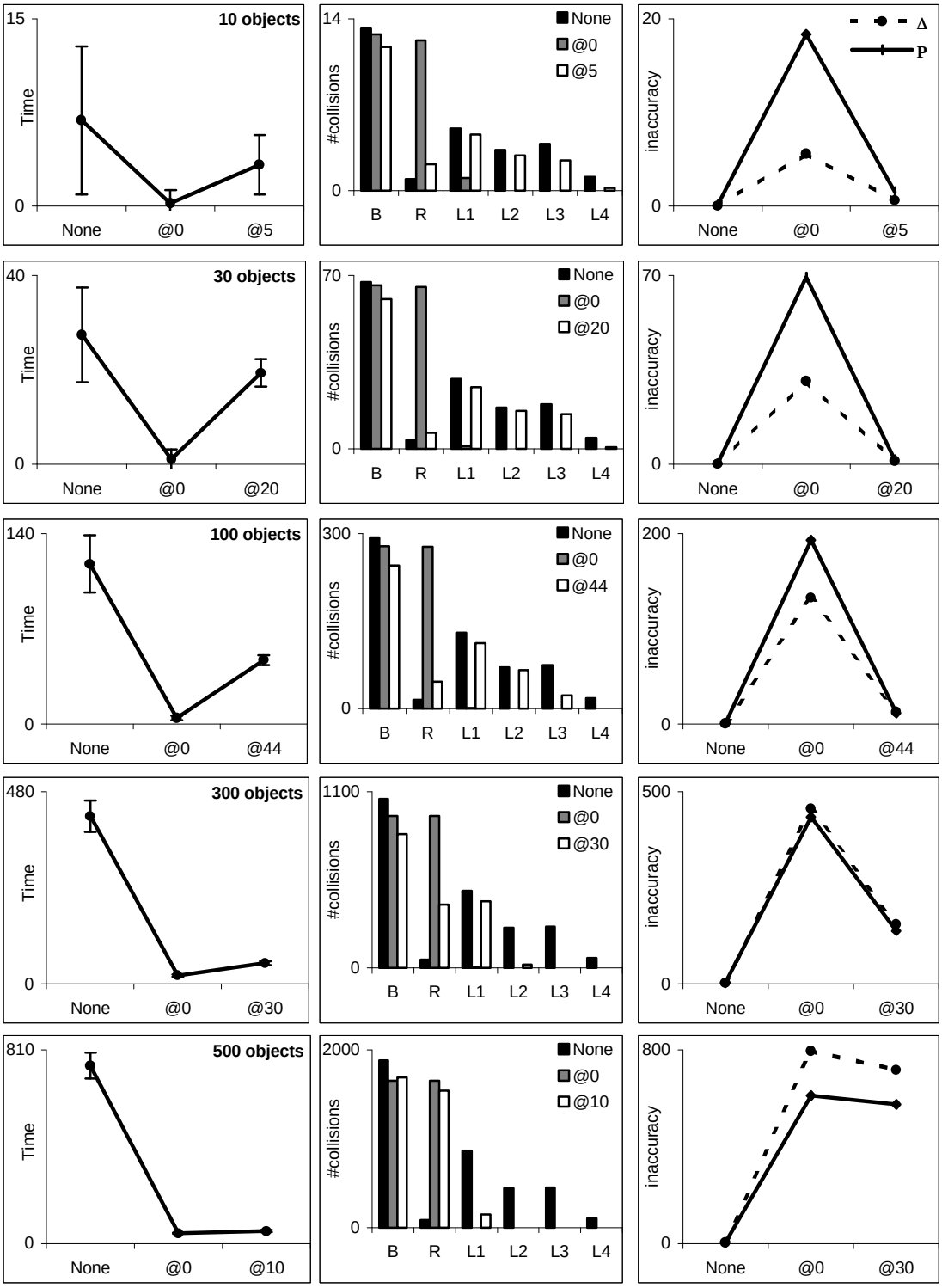


Figure 5.6: Comparison of collision handling values (error bars show standard deviations from mean).

We can see in all cases the mean time T decreases significantly along with the time of interruption. The variance of time T decreases even more dramatically, and the decrease gets larger as the number of objects increases. The number of broad phase tests remains fairly constant, although in every case the number of broad phase tests has been reduced slightly in the interruption at x cases. This is probably because more collisions are being resolved before they actually touch, and are moving away from each other when interruption occurs, whereas when no interruption occurs, their bounding boxes may continue to interpenetrate for several frames before an actual collision is detected or rejected. The reason that it rises again slightly in the interruption at 0 case, is because we never get lower than the bounding boxes, so that in a closely-packed world such as ours, the entities get themselves into a kind of vibrating state, where they can never move more than a step or two without colliding with something else.

The number of collisions accepted as real, however, increases significantly as the time for interruption decreases. This is because more non-collisions are being treated as real. For all numbers of objects, the increase for interrupting at 0 is very significant, because all broad-phase tests have been accepted as real. For 10 and 30 objects, the increase in the number of real collisions when interrupting at x, is not so significant. This is because we have chosen our interruption times to aim to achieve the mode collision handling time when full collision handling was performed. Hence, collision detection accuracy is deteriorated in only the minority of frames which would take longer than the mode if not interrupted. However, the higher the number of objects involved, the greater is the increase in the number of collisions accepted as real from the fully accurate case to the interruption at x case. This increase in real collisions is reflected in the increases in overall inaccuracy and perceived inaccuracy.

We can see that the mean number of collisions that descend to each level of the sphere trees is also affected by the interruption time. When no interruption occurs, the highest number of collisions are rejected at the L1 level, an equal number are rejected at the L2 and L3 levels, and a small number are resolve as colliding or not at the L4 level. The number of real collisions in these cases is usually slightly less than the number resolved at the L4 level, as some have been accepted, and some rejected. In the interrupting at x cases, for 10 and 30 objects, the pattern is not so different.

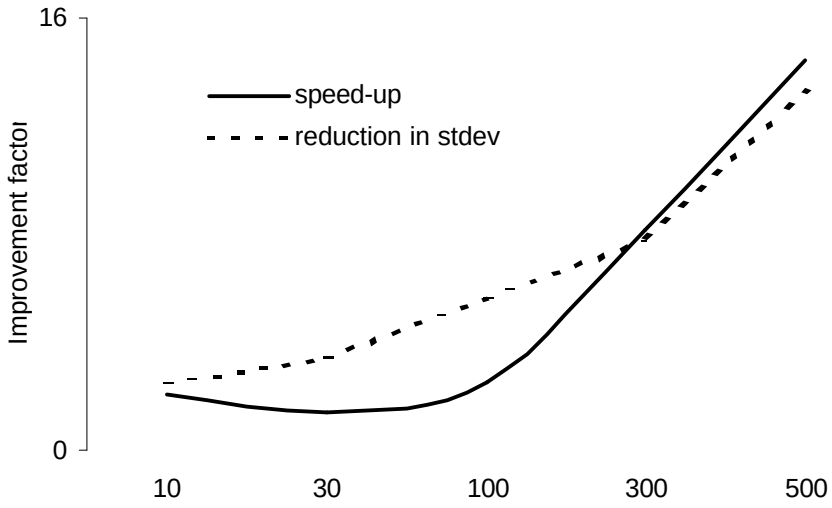


Figure 5.7 Improvements in average collision handling times and time deviation when using interruptible collision detection.

We do have some collisions which have managed to descend all 4 levels of the tree, but not all of them have made it. Therefore, the number of real collisions is slightly more than the number of collisions resolved at level 4. For 100 objects, no collisions made it to level L4, so the number of real collisions rises. For 300 objects, only a few have made it to level L2, so the number of real collisions rises even further. The worst case is with 500 objects, where the time for collision was set so short, that not all of the collisions managed even to descend to the first level of the tree.

We can therefore conclude that it is feasible to use interruptible collision detection for real-time animations, with the advantage being reduced variation of collision handling times in all cases. When used with other time-critical algorithms for rendering and motion synthesis, this will contribute to a high and constant frame rate. For up to 100 objects, the reduction in accuracy is not too damaging, but as the number of objects increases, both the perceived and total accuracy that is achievable in the target time decreases significantly. This is the motivation for our next set of experiments. Some level of inaccuracy must be accepted as a trade-off for shorter and less variable collision-handling times. However, we can attempt to decrease the perceived inaccuracy by using the perceptual metrics described in Chapter 4 to schedule the further processing of collisions detected in the broad phase.

5.3 Perceptual Interruptible Collision Detection vs. Non-Perceptual Interruptible Collision Detection

We have seen from the results presented in the previous section, that as we decrease the amount of time available for collision detection, both the perceived and total inaccuracy increases. As the number of objects increases, the reduction in accuracy becomes more severe. Totally accurate collision detection between large numbers of non-convex objects in real-time will always give highly variable frame rates. With more powerful computers and optimised algorithms, it may take longer for the worst effects of this to become apparent. I.e. The variation in collision handling times is evident from our analysis for 10 objects, but becomes really disturbing to the viewer after 30 objects. With more CPU power, and more fine tuning of the application (e.g. less pointer arithmetic, less dynamic memory handling), the number of objects handled without very noticeable deterioration could be increased, but this would just be postponing the inevitable. The following set of experiments start from the premiss, validated by the results of the previous section, that interruptible collision testing is both necessary and feasible. Keeping collision time constant, we examine the feasibility of using perceptual metrics to counteract the negative effects of interruptible collision testing, i.e. increased inaccuracy.

5.3.1 Perceptually Sorted Sequential Scheduling

In this set of experiments, the interruption time was set to be the same as in Section 5.2.3 and collision handling was always interrupted at that time. Hence, we once again try to allow as much time as possible for collision handling, while keeping that time as constant as possible. In the previous set of experiments, we used ***round-robin scheduling***, (as in [Hubbard 1995], whose approach we are trying to improve upon). This time we used ***perceptually sorted sequential scheduling***. As an initial pass at testing the feasibility of perceptual sorting, we used the perceptual metric $\mathbf{P}$ with function f_1 both to measure inaccuracy, and to schedule the processing of collisions. As before, broad phase collision testing produces an active list of potential collisions. Now however, this list is sorted, using a version of ***quick sort*** that works on linked lists. Each collision is thereafter resolved sequentially, until interruption occurs. The results of these tests are presented in table 5.9. Comparisons with the results of the previous section are plotted in Figure 5.8.

Interrupting:	*at 5*	*at 20*	*at 44*	*at 30*	*at 10*
Objects:	**10**	**30**	**100**	**300**	**500**
T	3.56	18.87	45.64	50.32	55.74
B	11.77	55.45	237.34	820.93	1577.38
R	2.62	5.91	42.61	622.30	1512.74
L1	4.10	22.57	105.42	132.20	44.41
L2	2.59	13.08	48.28	39.59	12.57
L3	2.79	13.76	40.50	26.65	7.98
L4	0.43	2.75	5.10	2.96	0.64
Δ	0.87	1.35	14.77	289.52	728.87
P	1.59	1.33	9.86	180.82	453.75

Table 5.9 Perceptually sorted using function f_1, interrupting at x

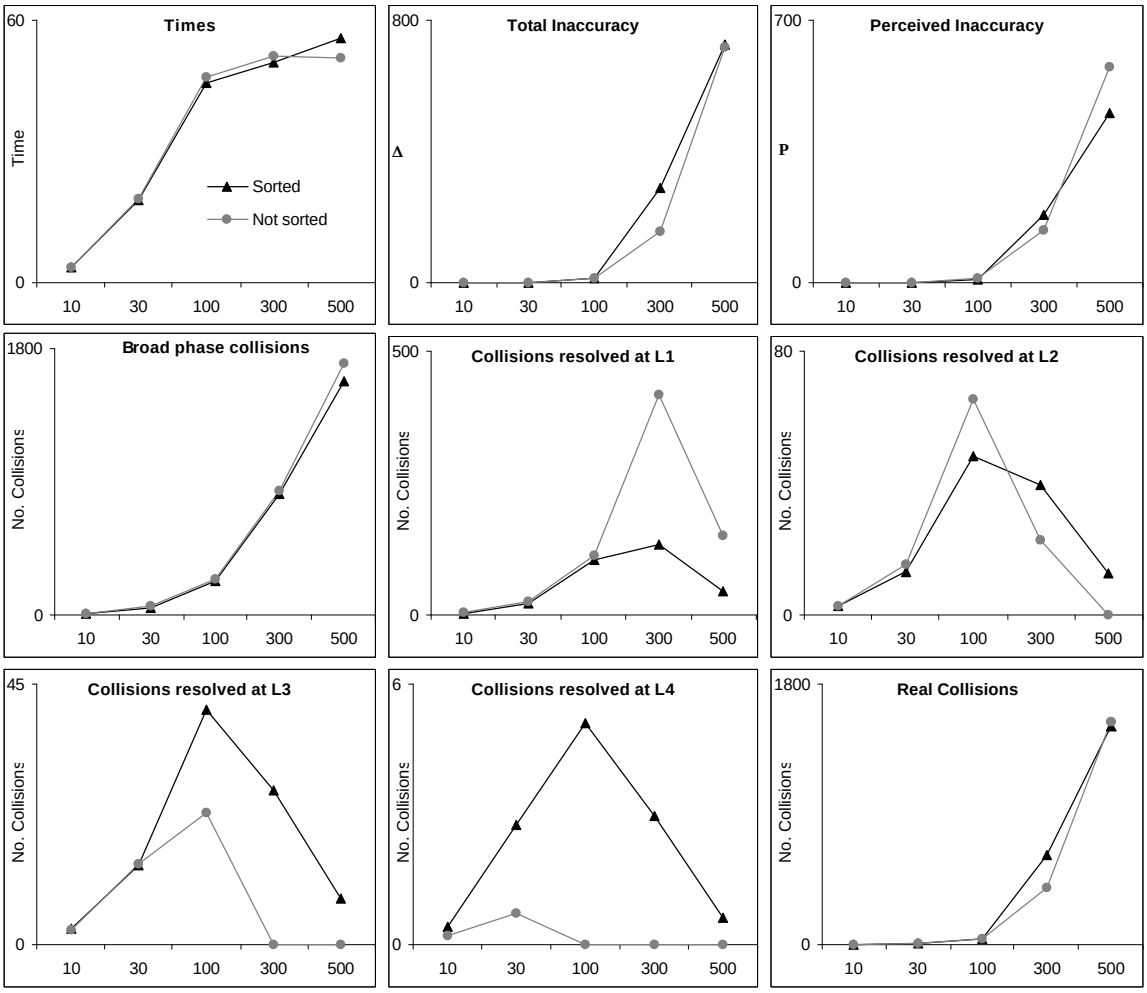


Figure 5.8: Comparison of non-sorted vs. sorted interruptible collision testing

We can see from Figure 5.8 that the added overhead of performing a quick-sort at each frame has reduced the time available for collision detection, and hence in the 300 object case, has caused an increase in Δ, so that the absolute value of $\mathbf{P}$ is higher than in the round-robin case. This can be seen more clearly in Figure 5.9(a), which shows the absolute values of **P** for both scheduling mechanisms. In the 500 object case, there is so little time available for collision detection when interrupting after 10 milliseconds, that the already very high geometrical inaccuracy is not much higher, whereas the perceived inaccuracy is slightly improved.

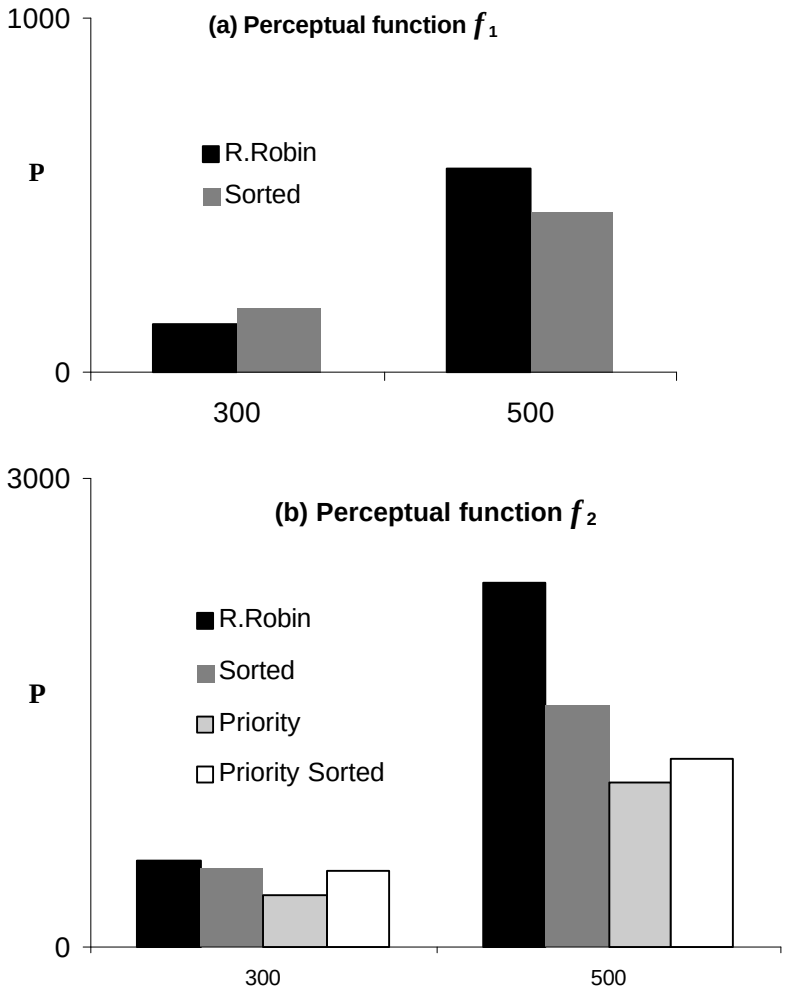


Figure 5.9: Comparison of perceived inaccuracy with various scheduling mechanisms

We speculated in Section 2 as to whether function f_1 was an appropriate model of the human visual system. It became evident from some pilot psychological experiments, described in Chapter 6, that f_2 is more likely to be appropriate. To this end, we repeated the above tests for both 300 and 500 objects, but this time using function f_2 both to measure inaccuracy and to sort collisions. We can see from Figure 5.9(b) that based on this model, there is an improvement in perceived inaccuracy when sorted scheduling is used, and this is most noticeable in the 500-object case. (Note: f_2 has a different scale, and cannot be compared directly with the f_1 values).

5.3.2 Priority Queue Scheduling

Next, we ran the tests using priority queue scheduling. We achieved this by simply setting an eccentricity from the fixation point, and creating one active list of collisions inside this region, and another of those outside the region. We tried two approaches: priority only, and priority sorted. In the former case, the collisions on the first active list are resolved in simple round-robin fashion, and only when they are all resolved are the collisions on the second list processed. In the second case, the first priority list is sorted and processed sequentially, and the second list is processed in round-robin order, thus reducing the sorting overhead. The results are shown in Figure 5.9(b). It is clear from these results that the simple priority queue scheduling produced the best results for both 300 and 500 objects, approximately halving inaccuracy in both cases.

Most sorting algorithms display an average-case O(N log N) performance. We cannot assume overall coherency in our system, because users will often move their eyes, creating a new sorting order with respect to eccentricity. However, if they do fixate for a few seconds in an area, concentrating on some complex task, the ordering may not change very much. In this case QuickSort exhibits a worst-case performance of $O(N^2)$, because the list is almost-sorted. Insertion sort would produce faster sorting times in these cases, but by mixing algorithms we are introducing another element of variability. Perhaps some other sorting algorithm would provide better, more consistent performance, and this may be worth investigating. However, the evidence is quite strong that the simple priority-list strategy provides good improvements with the minimum of overhead. Even by reducing the number of collisions to be sorted, by using a priority sorted queue strategy, the simpler mechanism provided superior improvements.

6 Psychophysical Experiments

6.1 Introduction

In Chapter 4 we introduced some plausible models of collision perception, based on two perceptual factors: eccentricity and separation. In Chapter 5, we found that using a simple priority queue scheduling algorithm with one of these models, function f_2(e, g), produced promising results. Some pilot experiments, described in Section 6.1.2, reinforced our opinion that this model is more appropriate than function f_1 which exhibits a much too rapid fall-off in acuity. We present the results of these experiments first, as they guided the design of the more rigorous experiments, which are described in subsequent sections. In Section 6.2, we describe the psychophysical experiments which were carried out on human subjects to determine their response to collision anomalies under a range of conditions. In Section 6.3 and we present the results of these experiments, which will be used to validate our model. In Section 6.4, we show that within the scope of the applications being considered in this thesis, the function f_2(e, g) provides a good approximation to observed human performance. We also discuss the situations in which our model is not appropriate, and look to ways in which it could be made more representative and general.

6.1.1 Motivation

We are seeking answers to the following questions:

(a) Is the model valid? We have found that using perceptual function f_2 (e, g) to prioritise collisions provides the best reduction in perceived inaccuracy when used with a priority queue scheduling algorithm. The reduction has been measured, however, using f_2 as the metric. If the model is not representative of true human perception, this renders the results of Chapter 5 meaningless. If, however, we find that humans exhibit similar eccentricity and gap effects to those modelled by this function, then we know that our model is a valid one, and that our techniques are feasible and useful.

(b) Under what circumstances is the model invalid, or incomplete? In Chapter 3 we reviewed relevant literature in the area of Visual Perception. From this, we deduced that eccentricity and separation were the most likely candidates to affect human perception of collision anomalies. However, we also found evidence that other factors strongly influence the perception of similar events. To what extent do these and their interactions influence collision perception also? How can our model be adapted to be more representative of human behaviour?

6.1.2 Pilot Experiments

A set of pilot experiments on 20 subjects was conducted to gain experience with psychophysical techniques, and to establish some common methods and conditions. These were then followed by the more extensive studies described in Section 6.2. The duration of each experiment was kept short, about 15 minutes, in order to encourage volunteers, and normal room lighting conditions were maintained. Collisions were presented at 5 eccentricities (at a viewing distance of approx 800 mm) of 20, 40, 60, 80 and 100mm, (i.e. at 1.4, 2.9, 4.3, 5.7 and 7.2 degrees of visual arc), and 8 locations (up, down, right, left, up-right, up-left, down-right, and down-left). Green spheres with 3-dimensional shading effects on a light-blue background served as stimuli. A red sphere appeared in the centre of the screen for fixation purposes. (See Figure 6.1). The spheres appeared and moved towards each other for 1 second, and then moved apart for 1 second after either touching, leaving a small gap of 4 mm, or a larger gap of 8 mm. The direction of motion and direction of offset were equal, and were randomised to be horizontal, vertical, oblique-left, or oblique-right (see Figure 6.2). The initial distances in each case were 40mm, 44mm and 48 mm respectively.

In order to keep the experimental time short, there were no replications of each combination of variables. Subjects were required to indicate after each collision whether the objects had touched or not. They were instructed to fixate on the red sphere at all times, and not to look directly at the green stimuli. However, it must be stressed at this point that there was no way of validating during the execution of the pilot experiments whether subjects actually maintained this fixation or not. However, we will see from the results that we can backwards-infer that it is highly likely that they did, as there is no other physically plausible explanation for the observed behaviour.

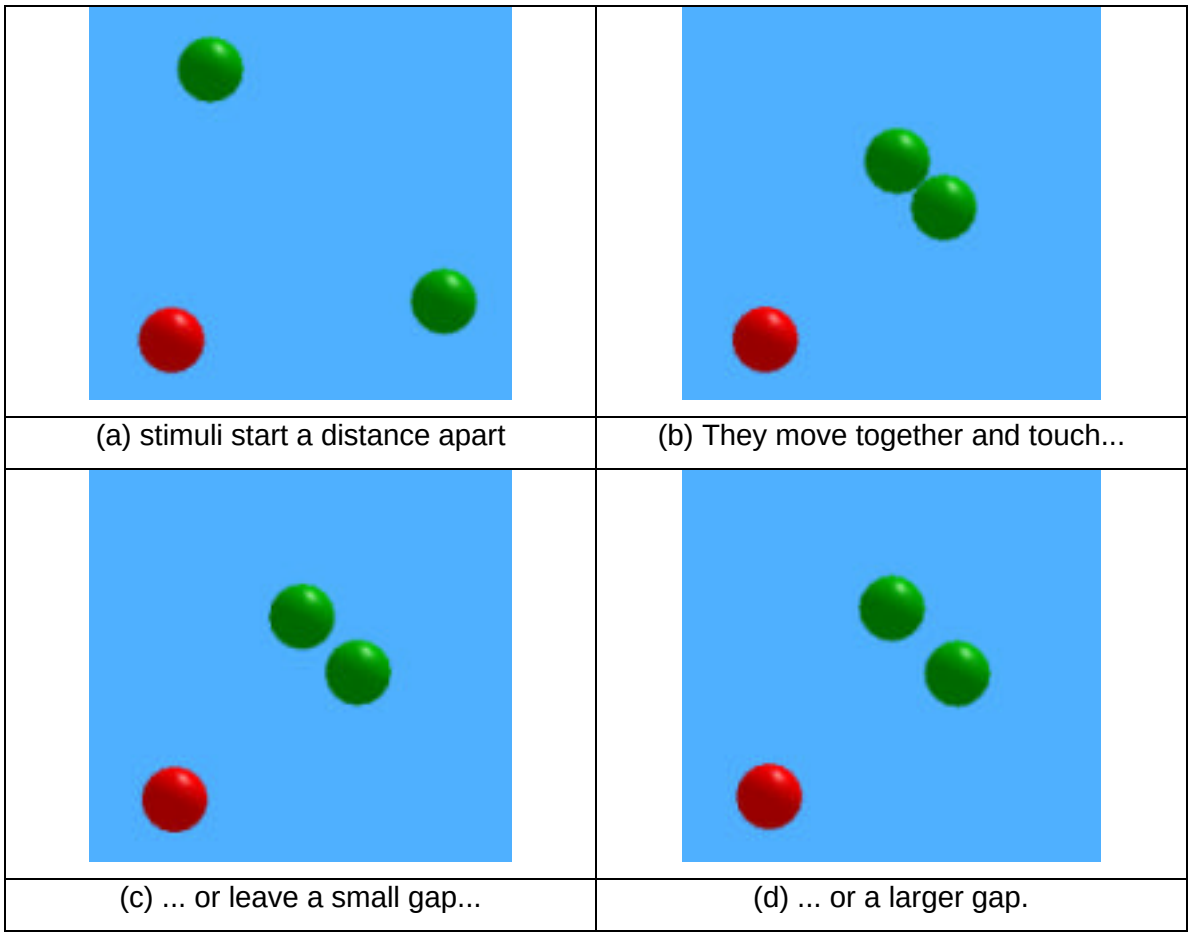


Figure 6.1. The background and stimuli for the pilot experiments

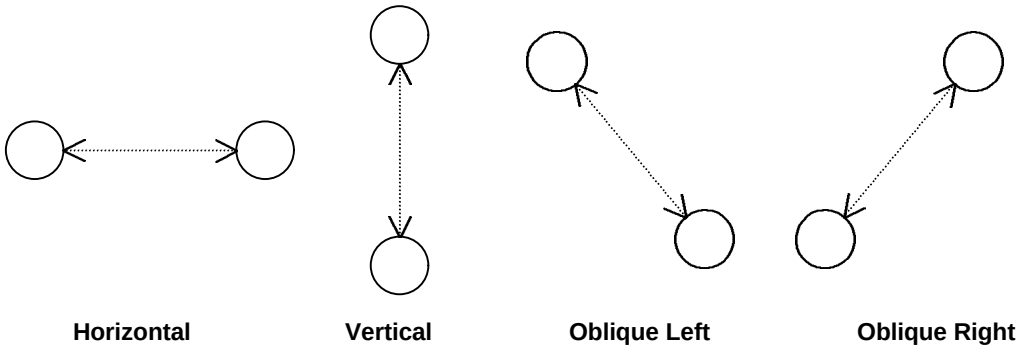
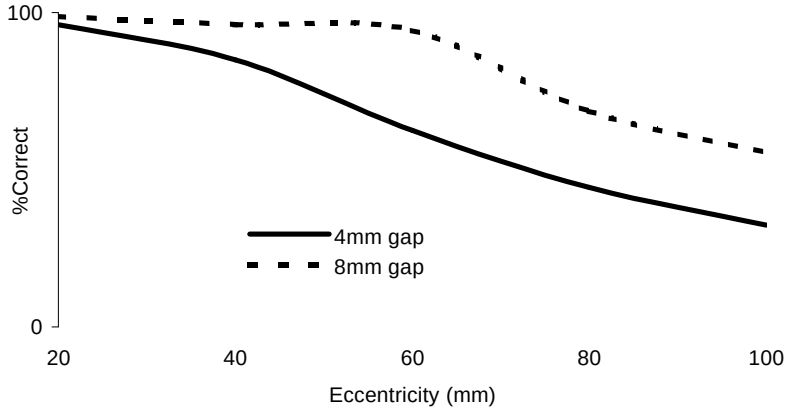


Figure 6.2. The four directions of offset, and also directions of motion

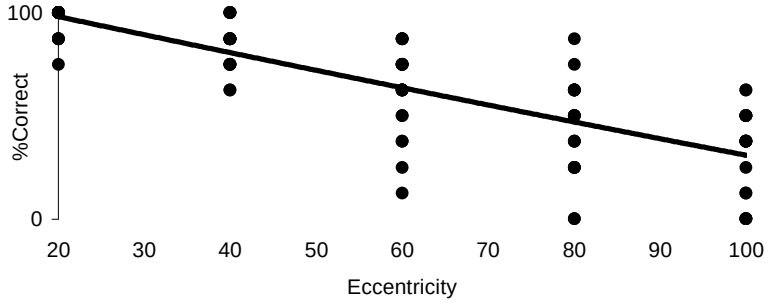
We can see from Figure 6.3(a) that there was a fall-off in detection accuracy with eccentricity, and Figures 6.3(b) and 6.3(c) show that this was evident in all subjects. It is also evident that the size of the gap affected performance, with the smaller gap eliciting fewer accurate results and a faster fall-off with eccentricity than the larger. The results of these experiments were therefore encouraging. However, they are on their own not sufficient to validate our model. Eye-movements to the point of collision would be possible because there was a presentation time of 1 second before collision, and it usually takes about 250 msecs to perform a saccade[1]. We may backwards-infer from the data that subjects behaved appropriately, and fixated on the central sphere. There is no other viable physiological explanation for the observed data. However, stricter control of fixation would be very desirable.

It is also standard practice in psychological research to perform such experiments in a darkened room, to ensure that there are no competing light sources which are competing for visual attention. High contrast stimuli, typically white against a black background, are usually used to maximise the efficiency of the visual system. By limiting the stimuli to flat, uni-colour circles, you are also limiting the factors which could be effecting perception of the collisions, such as specular dots and colour. Results from experiments conducted under such conditions are hence more reliable. It is also essential to have several replications of each combination of factors, leading to a much longer experimental time, and hence a more thorough investigation. Therefore, following promising results from these pilot experiments, these practices were implemented in a more rigourous set of experiments, described in the following sections.

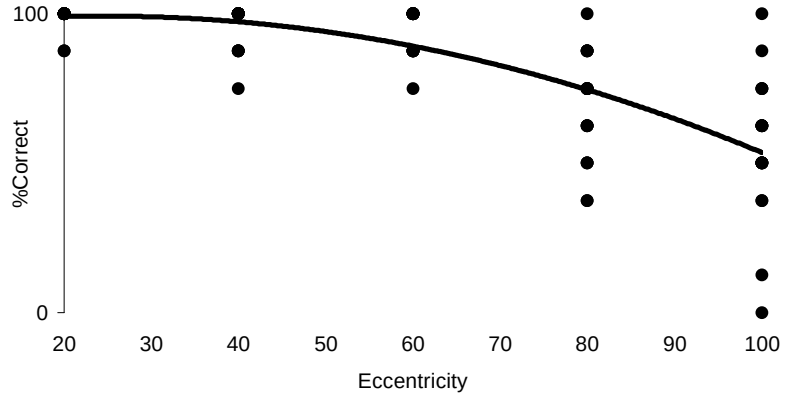
[1] Saccades are high-velocity, short-duration motions of the eyes to a different part of the visual scene. Such movements normally last between 20 and 100 milliseconds. However, there is also a factor called latency, which is the time it takes after a stimulus is presented for the eyes to start moving. It can take between 140-250 msec to initiate a saccade. [Hallett 1986] states that "*very few or no primary saccadic latencies are less than 140-150 msec. Primary latencies are almost always distributed as a single well-defined peak, which is often skewed toward longer latencies*." Hence, the minimum time in which it is possible to perform a saccade is 160 msec, although this would occur in only a small number of cases. 250 msec is commonly observed to be the mean length of saccades. It is a commonly used technique in psychophysical experiments (e.g. [Yap et al. 1987]) to limit the presentation time of stimuli to under 250 msecs, thus ensuring fixation.



(a) mean performance by gap size and eccentricity



(b) 4mm gap, scatter diagram of % correct for all subjects by eccentricity



(c) 8mm gap, scatter diagram of % correct for all subjects by eccentricity

Figure 6.3: Performance in Pilot experiments

6.2 The Psychophysical Experiments

In this section, we will describe a suite of computer-controlled psychophysical experiments that were carried out on human subjects to determine their sensitivity to collision anomalies. The variables, subjects, methods and design of all four experiments are discussed, as is the system that was developed to execute them.

6.2.1 Variables

There is an infinity of possible factors and combinations of factors, and hence potential experiments. It is therefore impossible to test each permutation of factors exhaustively, at least in this study. In fact, it would be foolish to attempt too much without first establishing the usefulness of this approach. Therefore, our strategy is to follow one line in the space of possibilities and to look from time to time to variations to get a feeling for the representativeness of the chosen conditions. The set of variables which we will test are as follows:

- Presence vs. absence of motion
- Separation
- Eccentricity
- Location
- Direction of offset
- Direction of motion
- Presence vs. absence of distractors
- Type of distractors
- Number of distractors

This is not to say that other factors do not affect the perception of collisions. However, the above factors may be determined very rapidly during a frame of the animation, whereas the others would either involve too much computation, or they may need a different experimental set-up (as, for example, in the case of colour and contrast).

We wish to establish the principle that information about the above variables may be used to optimise the realism of the animation. Once this has been established, the approach may be extended to include many other factors and combinations of factors, and indeed many of the concepts may also be applied to other graphical algorithms such as real-time rendering, texture mapping, motion synthesis. It should be noted that all motion in these experiments was linear and two-dimensional, and the objects were also two-dimensional. It could be argued that this would impact upon the generality of our results, but this decision was made for several good reasons:

- We maintain that ALL motion in computer animation systems is 2-dimensional. Due to the fact that each frame is projected onto a plane, and then usually onto a flat screen, the objects are not actually moving towards and away from the viewer. Instead, the illusion of depth of motion is being created by changing the shape, orientation and size of the on-screen projection of the object. Therefore, any effects observed in a 2-dimensional setting, such as the separation effect, may be generalised to a 3-dimensional environment.
- It has been shown in [Schiff and Detwiler 1979] that humans use only two-dimensional visual information to make decisions about three-dimensional collision events. The impression of motion in depth can be given by simply allowing an image to grow in size on the retina. [Rind and Simmons 1999] cite research which maintains that it is this expansion which is the major feature for determining whether an object is on a direct collision course with an observer. Hence, three-dimensional motion, i.e. the growth and expansion of the projected object with motion, is just another interesting factor to be investigated in future extensions of this research, and can be simply incorporated into the current system.
- We reduce the dimensions of our experimental sub-space by restricting ourselves to motion in a plane. Future experiments can then build upon these results, to examine such issues as occlusion and relative depth of objects. Similarly, the effects of more complex motion paths may also be investigated, as can shading and specular techniques, which also contribute to the illusion of depth. Measuring the perceptual contribution of such effects is computationally expensive, and their study belongs with a more comprehensive study of object perception, for such extended purposes as real-time rendering, shading and shadows.

Even after restricting our potential variables to this small subset, we still have a large number of possible permutations. It would be impractical to attempt to test all these in one experiment. We have therefore designed 4 experiments incrementally, using the results of the earlier experiments to reduce the dimensions of the later ones.

6.2.2 Subjects

12 2nd year under-graduate students of Computer Science acted as subjects for all four experiments, and were renumerated upon completion. All were naïve as to the purpose of the experiments, apart from being told that their sensitivity to objects touching or not touching in different parts of their peripheral field of view would be tested. There were 3 female participants and 9 males, aged between 18 and 22 years.

All subjects underwent a vision test, and had normal corrected or un-corrected vision. The hand and eye-dominance of all subjects was also registered. Experiments 1 and 2 lasted about 90 minutes each, experiment 3 lasted 60 minutes, and experiment 4 took 75 minutes, giving a total of 5.25 hours per subject. This means that an overall total of 63 hours of experimentation was carried out to validate our model.

6.2.3 Common Methods and Measurements

The experimental methods were similar for all experiments (see Figure 6.4), and consisted of the following common methods:

- Filled white circles of 14mm diameter presented on a black background served as stimuli. There were no shading or specular effects. Circles are ideal for this purpose, because they have no orientation, and the distance between two of them is independent of their direction of offset.
- The screen was 30 cm high, and 37.5 cm wide. Resolution was 1024 by 768 pixels and viewing was full-screen. From these measurements we can calculate that 1 pixel ≈ 0.4 mm.
- The experiments took place in a darkened room. Subjects were allowed 10-15 minutes to allow their eyes to adapt to the darkness before starting the experiment.

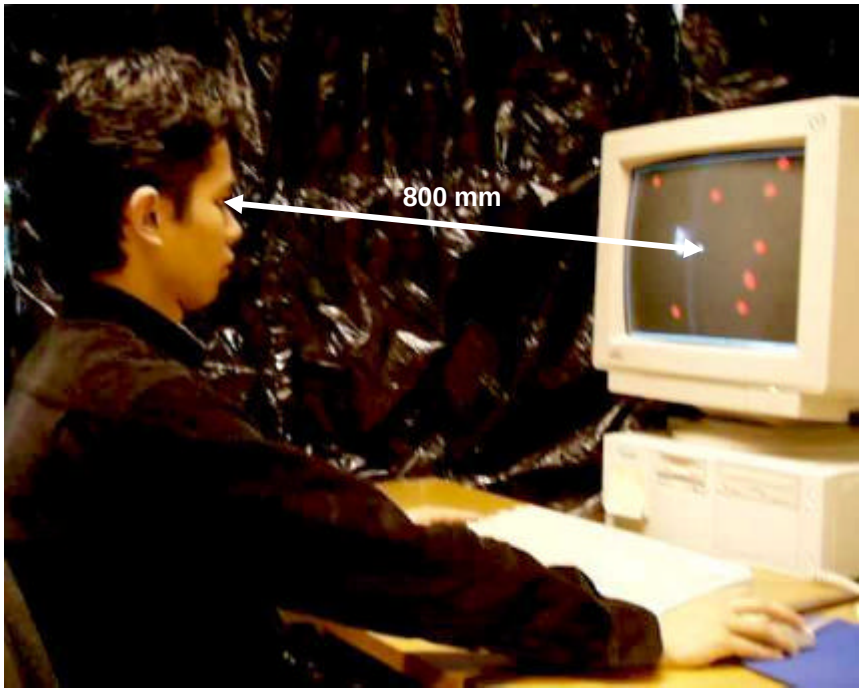


Figure 6.4 The setup for the psychophysical experiments

- In 50% of the runs, the circles touched, in 25% of cases they were separated by a small gap of size 2 mm, and in the remaining 25% of cases, they were separated by a larger gap of size 5mm.
- There was a 2-second delay between the presentation of each stimulus pair. This was discovered in [Yap et al. 1987] to counteract some learning effects.
- The order of the runs was randomised, but the random number generator was given the same seed for every subject.
- A smaller stimulus, (a wire-frame sphere), was displayed in the centre of the screen for the purpose of fixation. Subjects were instructed to look at the centre of the screen at all times, and to use only their peripheral vision to determine the status of each stimulus pair.
- Viewing distance was kept approximately constant at 800mm.
- After every 20 presentations, the screen was cleared. Subjects were encouraged to rest their eyes during this time to avoid eye strain and blurring, by staring at a point far away for a short while. Hitting any key continued the experiment.

- The left and right mouse buttons were used to give responses. Subjects were instructed to hit the left mouse button if they felt that the stimuli had touched, ***or if they were not sure***. They were instructed only to hit the right mouse button if they were very sure that they had perceived a gap between the two stimuli.
- There was a trial run of each experiment, to familiarise the subjects with the methods and stimuli.

Certain measurements are also common to all experiments. Figure 6.5 shows how the eccentricity and location of each stimulus pair is determined. A collision was generated to occur at each intersection between the lines and circles shown. From equation 2 in Section 3.2.1 (i.e. $q = 57.3 \times \frac{h}{d}$) we can express our measurements in degrees of visual angle, with h being the measurement in millimetres, and d being the viewing distance, which is always 800:

Circle diameter = 14 mm = **1**°

Small gap = 2 mm = **0.1**°

Large gap = 5 mm = **0.4**°

Eccentricities = 37.5, 75 and 112.5 mm = **2.7**°, **5.4**°, and **8.1**°

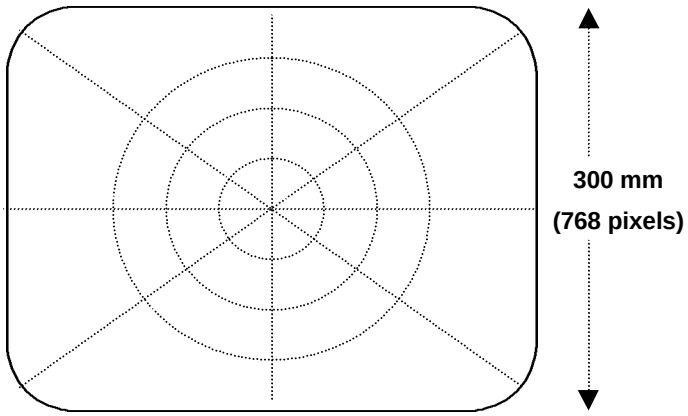


Figure 6.5: Eccentricities and Locations for experiments

6.2.4 The Experimental System

A system for conducting experiments was developed using C++ and OpenGL™. This system provided a framework whereby it was a simple matter to develop experiments by generating whatever type of collision events we wished.

The experimental application consists of an object of type **Experiment**. An object of this type contains a multi-dimensional array of objects of type **Run**, and this array is allocated dynamically at run-time. There is a second array of short integers, of equal dimensions, to hold the user's response to each run. There is one dimension for each variable in the experiment, plus one for the replications. For example, in experiment 1 the four variables are location, eccentricity, gap size and direction of offset. Each combination of variables will be repeated a certain number of times, so number of replications can be viewed as a fifth variable. Hence, two five-dimensional array is used to store all runs and their results. They are declared as:

```
Run        *****runs;
short int  *****results;
```

The memory is then allocated dynamically for these arrays at run-time, allowing the size of each dimension to be set by the user. For example, we can choose to vary the number of eccentricities tested in different experiments (e.g. 2, 3, or 4), and in those cases the size of the dimension corresponding to that variable will change. The Experiment object also contains some viewing information for visualisation purposes. The Run object contains all information necessary for one presentation of the stimuli, such as point of collision, direction of motion or offset, gap-size, radii of spheres, velocity information such as the animation stepsize and number of steps, and the address where the user response will be stored, i.e. the location in the results array of the Experiment object, declared as:

```
short int  *result_ptr;
```

This allows for the randomisation of the runs array, while keeping the results array sorted.

The methods of the Experiment object allow the array of runs to be initialised with all the parameters required by the experimenter, e.g. 8 locations, 3 eccentricities, 4 directions of motion, 3 gap sizes and 3 replications of each combination. The runs are originally created in a sorted sequence, but are then subsequently randomised. However, the results array remains sorted throughout, and is written to a file upon completion of the experiment. The execution of the experiment then consists of a simple loop through the array of runs, calling the draw function of each Run object in turn, and recording the user's response to the array of results.

6.2.5 The Four Experiments

These experiments have been designed so that they answer specific questions, the answers to which should be useful for our specific purpose, i.e. the prioritisation of collisions. For each experiment we will state the following:

- The specific question we wish to answer
- The variables to be included
- Hypotheses
- Methods

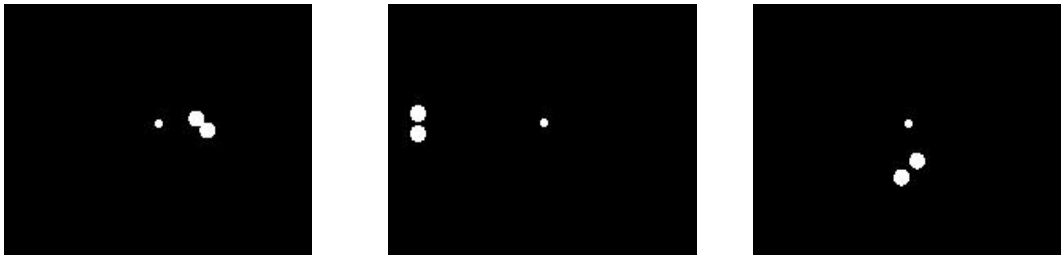
6.2.5.1 Experiment 1

The Question:

> *How is the ability to detect gaps between stationary stimuli affected by eccentricity, region, and the direction of offset?*

Variables:

- 3 eccentricities, of 2.7°, 5.4°, and 8.1°
- 4 directions of offset, horizontal, vertical, oblique left, and oblique right
- 8 locations, right, left, up, down, up-left, up-right, down-left and down-right.
- 50% of runs were with no gap, 25% with gap size 0.1°, and 25% with gap size 0.4°



(a) no gap *(b) small gap 1* *(c) large gap 2*

Figure 6.6: Screen shots of experiment 1

Each combination of variables was repeated 3 times. Hence there were 3 eccentricities X 4 directions X 8 regions X 2 gap sizes = 192 combinations of variables X 3 replications X 2 (to create 50% with no gap for control case) = 1152 runs per subject X 12 subjects = 13824 responses will be recorded in total.

Hypotheses:

- The ability to detect gaps between stationary objects deteriorates with eccentricity.
- Larger gaps are detected more frequently than smaller gaps.
- Some directions of offset elicit more correct responses than others.
- The useful field of view (UFV) is not a perfect circle, and detection in some locations is better than in others.

Methods:

All the common methods were applied. The fixation stimulus, described in Section 6.2.3 remained on continuously for the full duration of the experiment. Stationary stimuli appeared at 2 second intervals for 150 msec before being cleared (see Figure 6.6). This is a technique commonly used to eliminate the possibility of pursuit eye movements, e.g. in [Yap et al. 1987].

6.2.5.2 Experiment 2

Question:

Is the ability to detect gaps between stimuli affected by motion, and the direction of that motion?

Variables:

- 3 eccentricities, of 2.7°, 5.4°, and 8.1°
- 4 directions of motion, horizontal, vertical, oblique left, and oblique right (direction of offset = direction of motion).
- 8 locations, right, left, up, down, up-left, up-right, down-left and down-right.
- 50% of runs were with no gap, 25% with gap size 0.1°, and 25% with gap size 0.4°

Each combination of variables was repeated 3 times. Hence there were 3 eccentricities X 4 directions X 8 regions X 2 gap sizes = 192 combinations of variables X 3 replications X 2 (to create 50% with no gap for control case) = 1152 runs per subject X 12 subjects = 13824 responses recorded in total.

Hypotheses:

- The ability to detect gaps between stimuli is affected by stimulus motion.
- Detection ability is also affected by the direction of motion.
- The useful field of view will not be symmetrical
- Larger gaps will be more easily detected.

Methods:

All common methods applied, and the fixation symbol remained on continuously. In this experiment, the stimuli were visible for 300 msec. Every 2 seconds, two stimuli appeared and moved towards each other at a velocity of 40 mm i.e. 2.9° per second. After 150 milliseconds, they reversed direction after either touching or leaving a small or larger gap. Hence, the collisions occurred at a time when it was impossible for the subjects to have generated a saccade, ensuring that they used their peripheral vision to make the decision. The velocity was chosen to achieve a smooth movement within the time available.

6.2.5.3 Experiment 3

Question:

*Is the ability to detect gaps between stimuli affected by the presence of distractors which are **visually different** from the stimuli, and by the number of distractors present?*

Variables:

- 3 eccentricities, of 2.7°, 5.4°, and 8.1°
- 8 locations, right, left, up, down, up-left, up-right, down-left and down-right.
- 50% of runs were with no gap, 25% gap size 0.1°, and 25% gap size 0.4°
- 3 different levels of 1, 5 and 9 distractors

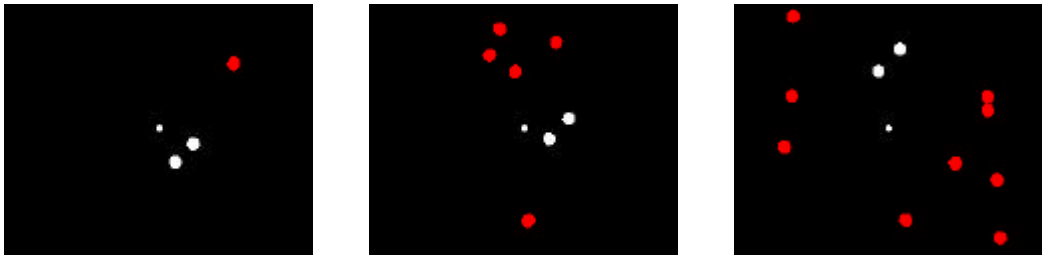
Each combination of variables was repeated 3 times. Direction of motion was randomised for each run. Hence there were 3 eccentricities X 3 distractor levels X 8 regions X 2 gap sizes = 144 combinations of variables X 3 replications X 2 (to create 50% with no gap for control case) = 864 runs per subject X 12 subjects = 10368 responses recorded in total.

Hypotheses:

- The ability to see gaps between stimuli is affected by visually different distractors.
- Detection ability is also affected by the number of distractors present.
- The useful field of view will not be symmetrical
- Larger gaps will be more easily detected.

Methods:

All common methods applied, and the fixation symbol remained on continuously. In this experiment, the stimuli were visible for 300 msec. Every 2 seconds, two stimuli appeared and moved towards each other at a velocity of 40 mm i.e. 2.9° per second. After 150 milliseconds, they changed direction after either touching or leaving a gap, ensuring that peripheral vision was used to make the decision. However, this time 1, 5 or 9 bright red circles also appeared simultaneously with the normal white stimuli, moving along a random linear trajectory, with a random velocity (see Figure 6.7). They were not allowed to occlude the colliding stimuli.



(a) 1 distractor *(b) 2 distractors* *(c) 3 distractors*

Figure 6.7: Screen shots of experiment 3

6.2.5.4 Experiment 4

Question:

> *Is the ability to detect gaps between stimuli affected by the presence of distractors which are **visually similar** to the stimuli, and by the number of distractors present?*

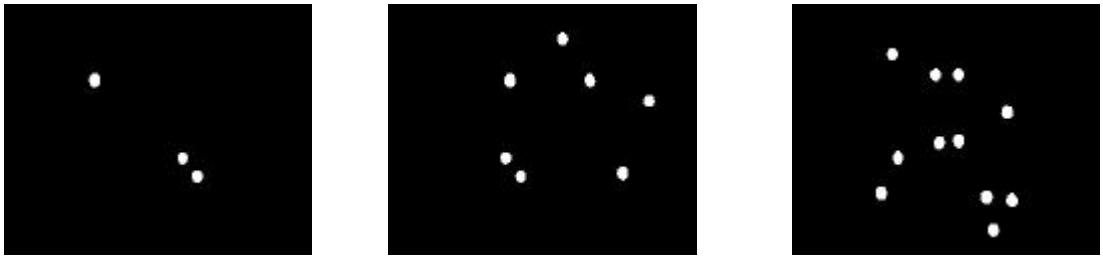
Variables:

- 4 eccentricities, of 0°, 2.7°, 5.4°, and 8.1°
- 8 locations, right, left, up, down, up-left, up-right, down-left and down-right.
- 50% of runs were with no gap, 25% gap size 0.1°, and 25% gap size 0.4°
- 3 different levels of 1, 5 and 9 distractors

> 3 replications, and randomised direction of gave 4 eccentricities X 3 distractor levels X 8 regions X 2 gap sizes = 192 combinations X 3 replications X 2 (no gap controls) = 1152 runs X 12 subjects = 13824 responses recorded in total.

Hypotheses:

- The ability to detect gaps between stimuli is affected by visually similar distractors.
- Detection ability is also affected by the number of distractors present.
- The useful field of view will not be symmetrical
- Larger gaps will be more easily detected.



(a) 1 distractor *(b) 5 distractors* *(c) 9 distractors*

Figure 6.8 Screen shots of experiment 4

Methods:

All common methods applied, and the fixation symbol remained on continuously. In this experiment, the stimuli were visible for 300 msec. Every 2 seconds, two stimuli appeared and moved towards each other at a velocity of 40 mm i.e. 2.9° per second. After 150 milliseconds, they changed direction after either touching or leaving a gap. Hence, the collisions occurred at a time when it was impossible for the subjects to have generated a saccade, ensuring that they used their peripheral vision to make the decision. However, this time 1, 5 or 9 white circles which were identical in appearance to the colliding stimuli also appeared simultaneously with the colliding stimuli, moving along a random linear trajectory, with a random velocity (see Figure 6.8). They were not allowed to occlude the colliding stimuli.

This task was particularly difficult, because it was not always easy to detect which were the colliding stimuli. Subjects were therefore given the added instruction that if they did not identify any circles as moving towards each other and then separating, that they should respond as in the colliding or not-sure cases, i.e. by clicking the left mouse button. Only if they were certain that they saw two circles approach and separate without touching, should they indicate a gap by clicking the right mouse button. Due to the complexity of the task, we also collected information at the point of fixation for this task, hence the fixation cross was hidden when the colliding stimuli and distractors appeared, and re-appeared immediately afterwards. This data was not included in comparisons with other tasks, but simply to see what effect the complexity of this task had on performance at the location of fixation.

6.3 Results and Analysis

The data from the above tests was imported into MS Excel worksheets, and a comprehensive data-analysis was carried out. The aim of this data analysis was to identify which factors or combination of factors influenced collision detection performance most significantly. We will use graphs and statistical functions to illustrate our findings.

In almost all cases, an analysis of variance was carried out to test the significance of any observed results, and the observed value of the F-distribution (see [Milton et al. 1997]) will be presented in these cases, along with the critical value of F for that particular number of observations at 95% significance. Whenever the observed value of F is greater than the critical value of F, we know that our results are statistically significant at that level. The higher the value of F, the more significant is the result, and hence the more robust is the observed effect. The F statistic is hence an excellent metric for the evaluation of factors for inclusion in our model.

There are many potential combinations of factors, so it was necessary to perform the data analysis in a logical and ordered way. We start by looking for broad trends, and then we expand our investigations when we feel that the answers will be useful to us. The main questions to be asked in this analysis are as follows:

1. Did the ***task*** itself affect the results, i.e. did overall performance vary between the four experiments?
2. Did ***gap size*** make a difference?
3. Was there an ***eccentricity*** effect?
4. Did the ***location*** of the collision matter?
5. Did the ***direction of offset*** and/or the ***direction of motion*** affect overall performance?
6. Did the ***type and number of distractors*** have an effect?

The following sections address each of these questions in detail.

6.3.1 Task

To answer question, i.e. whether overall performance varied between experiments, we calculated the overall percentage of correct responses, averaged over all subjects and variables. We can see from Figure 6.9 that the type of task strongly affects collision perception, with performance in the absence of motion being much better than when motion is added. Overall performance in the presence of motion does not seem to depend on the presence or absence of different distractors, but the presence of same distractors does appear to have a negative effect.

The results per subject are shown in Figure 6.10, and we can see that they reflect the overall results. A single-factor analysis of variance was performed on these averages, to ask the question if the observed differences between the tasks with no motion, motion, different distractors, and same distractors were significantly different. The critical value of F is 2.82, and the observed value of F was 52.53 which greatly exceeds the critical value. Hence we can state that the type of task does affect collision perception.

We performed three paired tests to examine these results further. Task 1 (no motion) vs. Task 2 (motion), produced an observed F of 53.45, compared to a critical value of 4.3. Hence, we can deduce that the presence of motion strongly detracts from overall performance. The observed value of F for Task 2 vs. Task 3 (different distractors) was 0.15, which is less than the critical value of 4.3. Therefore, we can state that overall performance was not affected by the addition of distractors which are visually dissimilar from the colliding entities. The value of F for Task 3 vs. Task 4 (same distractors) was 26.93, which is much greater than 4.3, showing that the type of distractors also strongly affects overall performance.

Therefore, at this point of the analysis, we have already learned about two important factors. The presence or absence of motion strongly affects overall collision perception, as does the type of the distractors present. We have not yet detected any effect caused by the presence of visually different distractors, but because performance in the presence of visually similar distractors was significantly worse than in the presence of different distractors, we can transitively infer that the presence or absence of distractors can also strongly affect overall performance, depending on the type of the distractors.

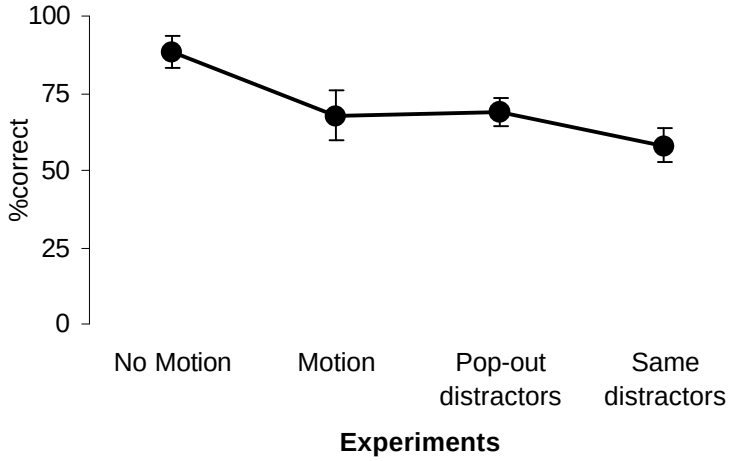


Figure 6.9: Overall results per experiment, averaged over all subjects and variables.

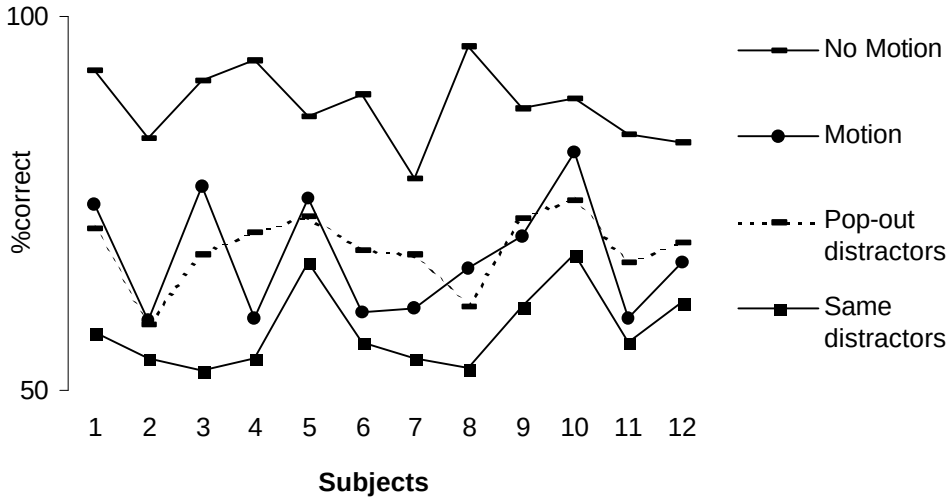


Figure 6.10: Overall performance per experiment by subject, averaged over all other parameters.

6.3.2 Separation

We now turn our attention to question 2, which asks whether separation, i.e. the size of gap left between stimuli, affects the ability to respond correctly. This also includes the question as to whether subjects performed better in the control situation, i.e. in the absence of a gap, or in the presence of a gap. We know from our answer to question (a) that the type of task affects performance, so we look at the results separately for each task. We can see from Figure 6.11 that results varied from task to task. However, on the whole it seems that a separation or gap effect is evident. In experiment 1, the differences seem quite small, but the smaller gap size, gap 1, does elicit a poorer response than the other two cases, i.e. no gap and the larger gap 2. In experiment 2, the difference between performance for the smaller gap and the larger gap is much more marked. However, what was not expected was the relatively poor performance in the control situation for the first two experiments, i.e. no gap, when performance should be close to 100%. In experiment 2, the motion experiment, the percentage of correct responses is actually greater in the case of the larger gap than in the no gap case. This should not be the case if subjects had followed instructions properly. The results in experiments 3 and 4 (with distractors) appear to be more appropriate.

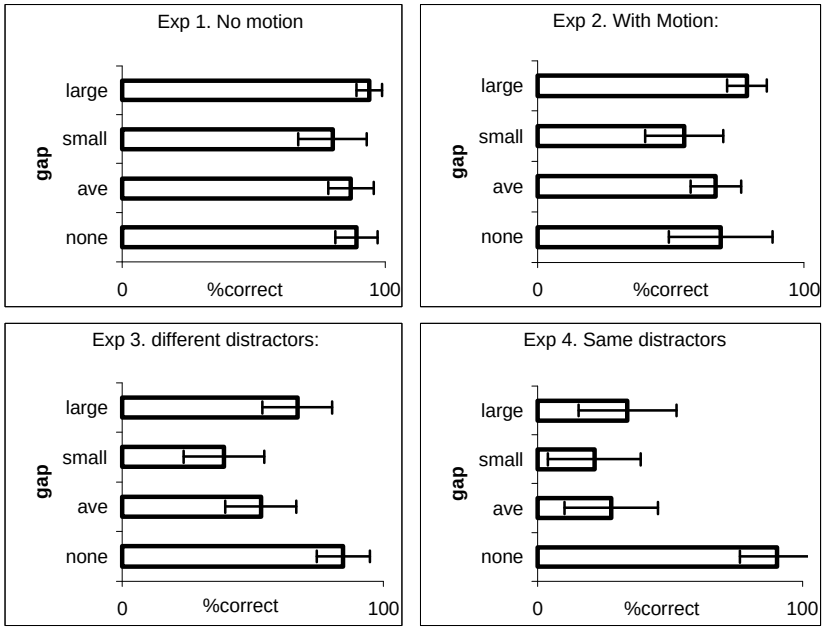


Figure 6.11. Performance per experiment by gap size, averaged over all subjects (error bars show standard deviation).

We can investigate this situation further by examining the numbers in Table 6.1. We can see that in all experiments, overall performance in the presence of the smaller gap was poorer than in the case of the larger gap. This difference was also statistically significant in all cases, although in the case of experiment 4, where the distractors were visually similar to the colliding stimuli, the difference was not so marked and was not significant at the 95% level, but at just below 90%. In the comparisons of Gap vs. No Gap, the difference in performance in the presence or absence of a gap is very insignificant in experiments 1 and 2 (no distractors), but very significant in experiments 3 and 4 (with distractors).

Subjects were instructed to right-click only in those cases where they were certain that there was a gap between the objects. If unsure, they were instructed to always left-click, i.e. indicate a touch. If they had followed these instructions fully, then we should observe an average detection performance of close to 100 percent. However, in experiment 1 (no motion) and experiment 2 (with motion), the performance was poorer in the no gap case than in the case of the large gap. A possible reason for this could be that, when unsure, the subjects did not always default to left-click as instructed, but that they resorted to guessing. It may even be possible that in their desire to “get it right”, some subjects defaulted to right-click, i.e. indicated a gap, when in doubt. If this was the case, then their performance in the detection of gaps may not be as good as they appear. To investigate which subjects behaved in this way, we need to look at the results for each individual subject, plotted in Figure 6.12.

	Exp1 (no motion)	**Exp2 (motion)**	**Exp3 (same distractors)**	**Task4 (different distractors)**
Gap1	80.03	55.09	39.00	21.49
Gap2	94.04	78.53	67.09	33.99
Gap	87.04	66.81	53.05	27.74
No Gap	89.08	68.76	84.65	90.57
Observed F Gap1 vs Gap2	12.15	24.56	22.80	2.91
Observed F Gap vs no Gap	0.35	0.10	41.27	92.70
F crit	4.30	4.30	4.30	4.30

Table 6.1. Results for overall performance by Gap size

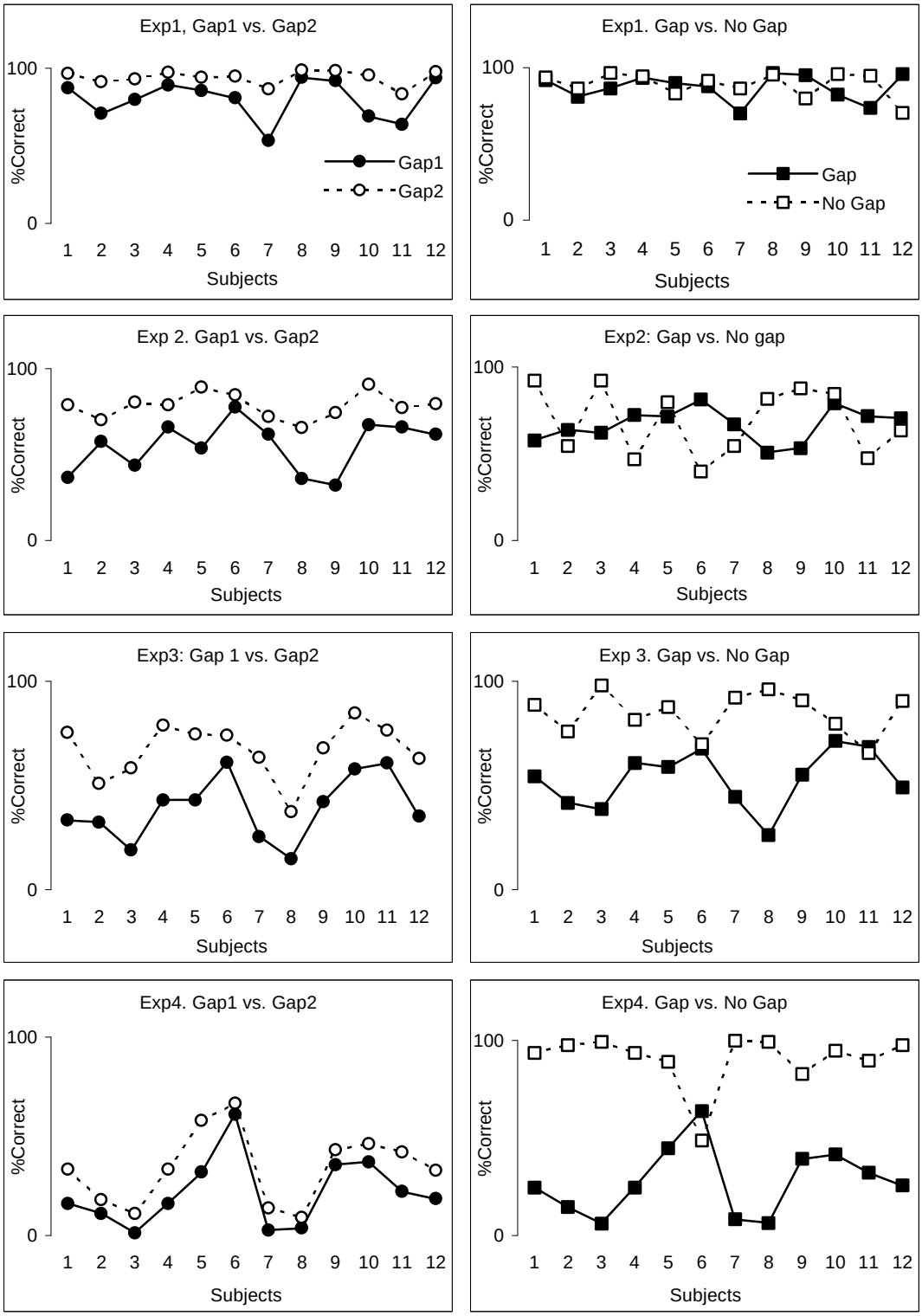


Figure 6.12: Overall performance by gap per subject for all experiments

We can see that in experiment 2 (motion), erroneously reporting a gap when none was present was a phenomenon which was particularly prevalent, and to a lesser extent in experiment 1. There is no physiological explanation for such a phenomenon, and hence it can be concluded that this behaviour is as a result of an as yet un-identified subject-related factor in the perception of collisions, i.e. task-specific semantics. In several cases, what we instructed the subjects to do, and what they thought they had to do, were obviously mismatched. In hindsight, we probably should have provided more training and feed-back at these early stages, and replaced certain subjects who proved to be incapable of following instructions properly. This is particularly true in the case of subject number 6, who persisted in his aberrant behaviour througout the whole suite of experiments. Subjects 4 and 11 also behaved in a similar way, but did improve in the later experiments. (It may be interesting to note that these subjects were all young males.)

However, this confusion also highlights the artificial nature of the first two experiments. Our work is not concerned with collisions between two objects, when only they are visible. Even if we were, in the real-world scenario, subjects would not be trying to identify collision anomalies, they would be concentrating on some other task, and only accidentally would they notice such gaps. Experiment 4 (same distractors) most closely represents the situation with which we are concerned. Many objects, which all look very similar, are moving around in close proximity to each other, some of them colliding. Hence, it is encouraging to note that, apart from one exception, subjects behaved appropriately in this experiment, the results of which particularly interest us. It is those results which will be used to validate our model of collision perception introduced in Chapter 3. Hence, we will continue with our analysis, having learned a valuable lesson for any future experiments, and treating the first two experiments as a useful comparison and training base for the later two, more important, tests.

6.3.3 Eccentricity

Now we turn our attention to question 3, i.e. did eccentricity affect performance? By now we know that the task, and the size of gap, determined detection ability to some extent, so we will examine the effects of eccentricity separately in each case.

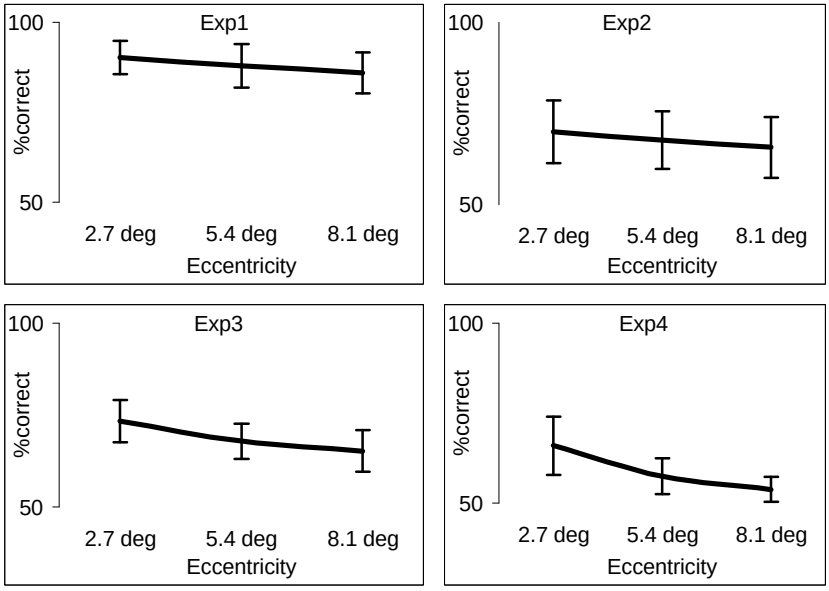


Figure 6.13: Overall eccentricity by experiment

	Exp1 (no motion)	Exp2 (motion)	Exp3 (diff. distractors)	Exp4 (same distractors)
2.7°	90.21	69.97	73.41	65.91
5.4°	87.96	67.71	67.91	57.41
8.1°	86.00	65.69	65.22	53.73
Observed F	1.74	0.80	7.13	13.73
F crit	3.28	3.28	3.28	3.28

Table 6.2: Results for overall eccentricity by experiment

We can see from Figure 6.13 and from Table 6.2, that there appears to be a slight overall eccentricity effect in experiment 1 (but only at an 80% significance level), and little or no effect in experiment 2 (less than 55% significance). However, the eccentricity effect is very strong in experiment 3, and even stronger in experiment 4. We know that gap size affects performance also, so we must now look deeper at the results of the experiments, and determine what the eccentricity effect is at different gap sizes.

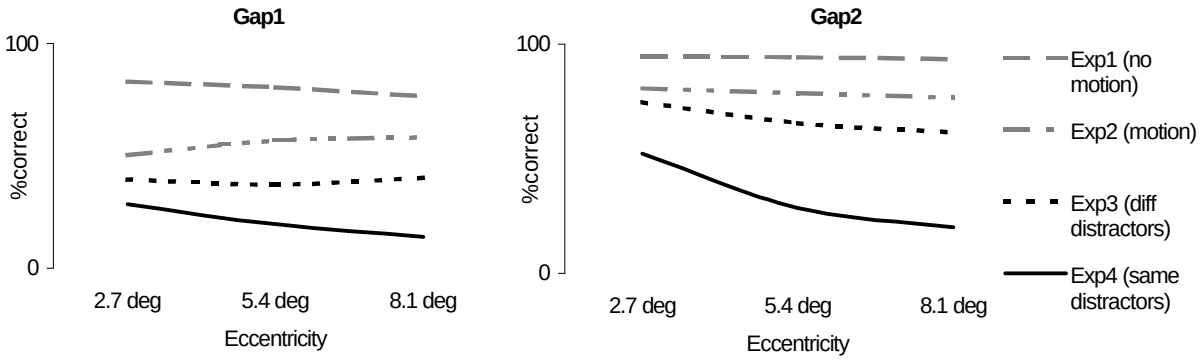


Figure 6.14: Comparison of performance by eccentricity and gap size.

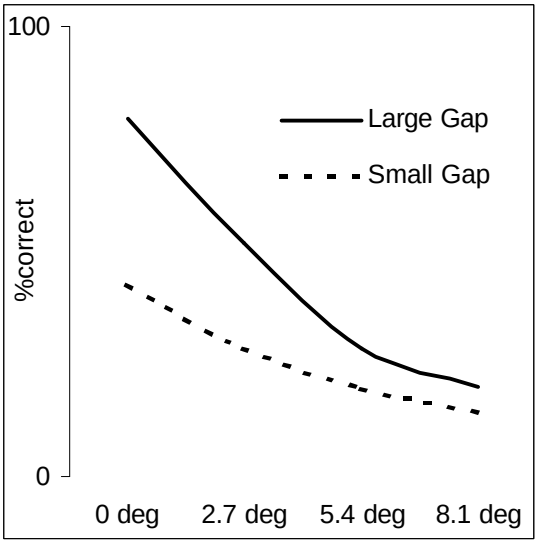


Figure 6.15 Exp. 4 (same distractors), performance by eccentricity and gap size.

Figure 6.14 shows us the breakdown in performance results by experiment and gap size, and the values are shown in Table 6.3, along with the F statistics. We can see that none of the eccentricity effects were particularly significant for the smaller gap size, although the actual values are much lower than with the larger gap size. Only in experiment 4 is it anyway significant, at the 80% level. However, the eccentricity effects are stronger in the case of the larger gap, with eccentricity effects present in experiment 3 (90% significant), and strongly present in experiment 4.

Figure 6.15 shows the eccentricity effect by gap size for experiment 4 (same distractors), this time with the performance at the fixation point added. This is the situation which is closest to the real-world scenario we are interested in. This diagram is very like the diagram of perceptual function f_2, shown in Chapter 4, Figure 4.11, with C=100. An analysis of variance with this added data indicates a stronger eccentricity effect, of 4.08 for the small gap, and 25.42 for the larger gap, compared with a smaller critical value of 2.82. This indicates that we were on the right track with our model, and that it is a reasonable representation of human behaviour in the circumstances we are considering. It also indicates that we should have recorded correct levels at the fixation point for the previous three experiments also, and this would have most likely given us more significant results for the eccentricity effect in those cases also. We incorrectly assumed that there would be perfect performance at the fixation point in those simpler tasks, but results so far have shown us that this type of assumption is not a safe one to make. We will again put this down as a valuable lesson learned for the future.

		Exp1	Exp2	Exp3	Exp4
Small gap	2.7°	82.99	50.26	39.47	28.47
	5.4°	80.64	56.86	37.27	19.56
	8.1°	76.48	58.16	40.28	14.12
	Observed F:	0.68	0.92	0.09	1.80
Large gap	2.7°	94.53	80.64	74.54	52.20
	5.4°	94.18	78.47	65.39	28.36
	8.1°	93.40	76.48	61.34	19.91
	Observed F:	0.15	0.15	2.48	8.49
	F Critical:	3.28			

Table 6.3: Results for eccentricity by gap size and experiment

6.3.4 Location

We have seen that our proposed model of collision perception, based on eccentricity and gap-size is an appropriate one. We now wish to check if it is complete, i.e. whether there are other factors which need to be incorporated in order to make it more general. The next question we address is question 4, i.e. does the location of a collision matter? Figure 6.16 shows performance by location, averaged over both gap sizes. There appears to be a slight bias to the left in experiment 4, but the field of view in the other experiments appears to be quite symmetric. We show the performance by gap and location, averaged over the 3 eccentricities, in Figure 6.17. We can see that the field of view is not symmetric in experiments 2, 3 and 4. However, different biases appear under different circumstances. Table 6.4 displays the results of some tests we performed to test the significance of these asymmetries. We can see that there is some statistical significance for better detection of gaps between stimuli above the fixation point in experiments 2 and 3. However, these are not significant at the 95% level, and there is no evidence to support this trend in experiment 4. All other asymmetries are very insignificant.

	Small gap				**Large gap**			
	Exp1 (no motion)	Exp2 (motion)	Exp3 (diff. dists)	Exp4 (same dists)	Exp1 (no	Exp2 (motion)	Exp3 (diff. dists)	Exp4 (same dists)
Top	78.86	60.36	41.77	19.65	93.36	80.17	70.58	34.67
Bottom	79.78	45.31	31.69	20.68	94.37	70.09	59.67	31.69
Obs. F	0.03	3.42	2.46	0.02	0.17	2.04	3.44	0.15
Crit. F	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30
Right	81.40	54.36	40.12	21.19	94.21	74.44	65.74	32.61
Left	79.71	53.45	38.37	20.27	94.29	74.88	67.49	36.52
Obs. F	0.10	0.01	0.07	0.02	0.00	0.00	0.07	0.24
Crit. F	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30
Hori	82.18	54.99	45.83	22.38	94.56	72.88	72.99	34.41
Vert	78.47	51.77	38.27	20.68	93.40	74.28	68.52	30.25
Obs. F	0.47	0.18	1.02	0.05	0.30	0.04	0.65	0.35
Crit. F	4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.30

Table 6.4: Results of comparisons of locations by gap size

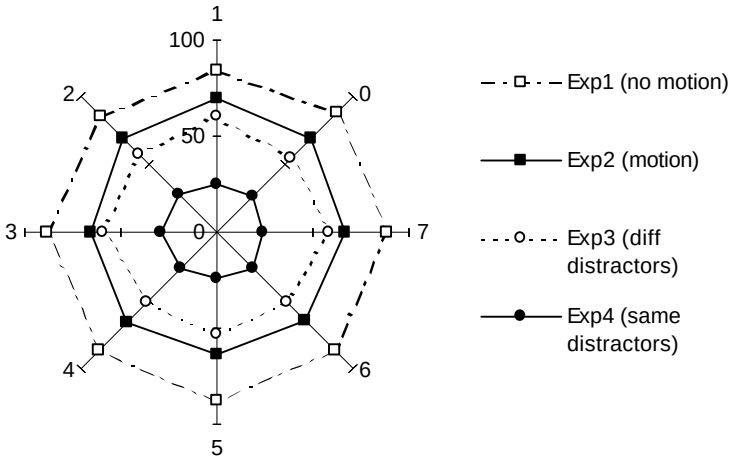


Figure 6.16: performance by location, averaged over both gap sizes

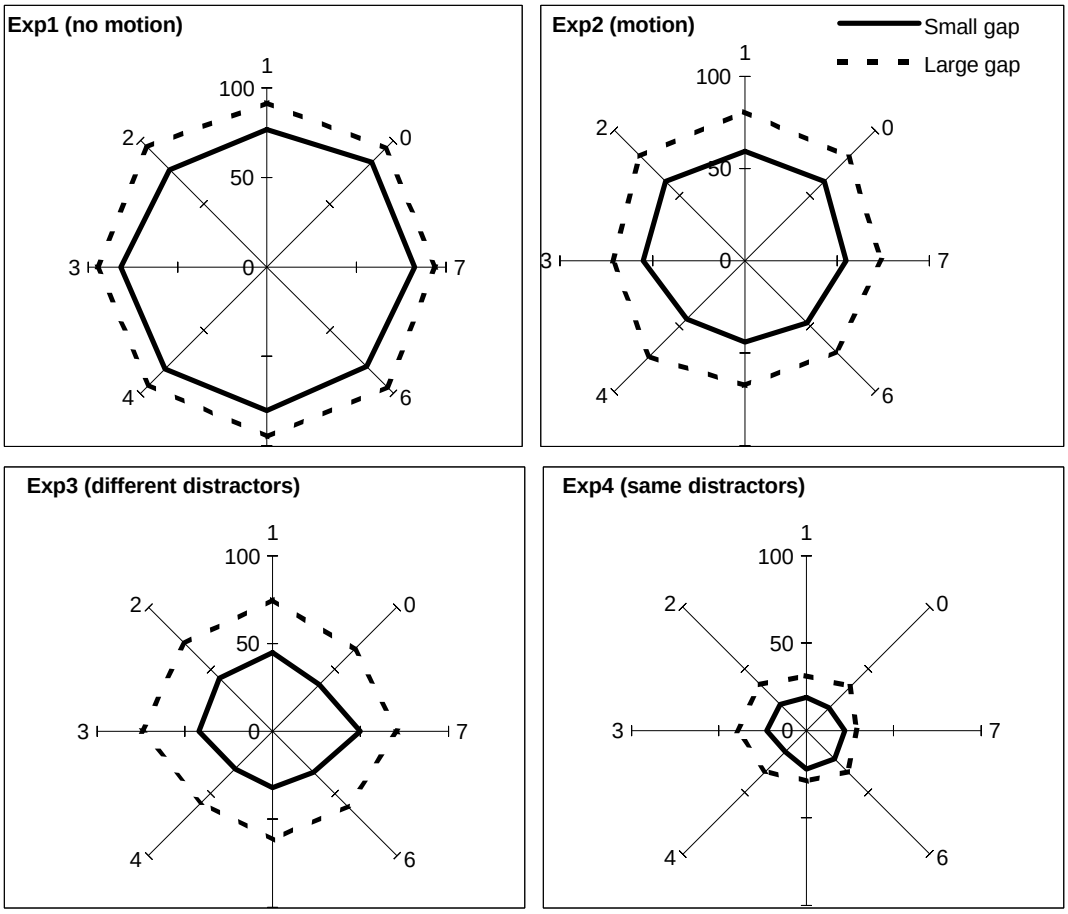


Figure 6.17: Performance by location and gap size

6.3.5 Direction of Offset and Direction of Motion

In experiment 1, we presented stationary stimuli at 4 different directions of offset, horizontal, vertical, oblique-left, and oblique right, as in the pilot experiments (see Figure 6.2). An overall analysis of the direction of offset collapsed over all variables yielded no significant difference. Looking at the values by gap size, performance was slightly poorer for the horizontal for both the small and the larger gap size, but the statistical significance was low (<60% for the small gap, and <80% for the larger gap). We then looked at each gap size by location, and performed a two-factor analysis of variance with replications. The significance of the difference between the directions was more significant for the larger gap this time, (>95% significance), but the absolute differences were quite small, and for some locations, the horizontal direction did not elicit the worst performance. We also tested the hypothesis that there is an anisotropy, i.e. a difference between detection ability in the isoeccentric (both stimuli at the same eccentricity) vs radial (stimuli oriented away from the point of fixation) offsets, but found no evidence to support this theory.

In experiment 2, we added motion, and allowed stimuli to move towards and away from each other along the same paths defined by the directions of offset from experiment 1. Again, an analysis of direction of motion averaged over all other variables per subject indicated no preferences. When looking at the data per gap size, averaged over all other variables, directions along the horizontal provided slightly worse results than in other directions (see Figure 6.18), but these results were below 80% significance for the smaller gap, and were insignificant for the large gap.

We then performed a single-factor analysis on the data expanded by location, the results were more significant for the smaller gap (>95%), but were below 70% significance for the large one. A two-factor analysis with replications for the smaller gap yielded significant evidence for both location and direction effects, but with low evidence of interaction between the factors (<60%). A similar analysis for the large gap yielded strong evidence of a location effect, but the direction and interaction effects achieved low significance (<75%). Looking at the results for the small gap per location (Figure 6.19a) we can see that the horizontal direction of motion achieved the worst results in almost all locations. Such a result is not evident for the large gap.

However, the diagram shows us something much more interesting. Performance is particularly bad for the small gap in the horizontal direction in both the horizontal locations, i.e. right and left. This is the radial direction for those locations. Examining Figure 6.19b, we can see that this is also the case here. We compare the horizontal and vertical directions in 6.19c and 6.19d. We can see that the horizontal direction is worse than the vertical when it is the radial direction, but the opposite is true when it is the isoeccentric direction. We compare the two diagonal directions in 6.19e and 6.19f, and find that, on the whole, the isoeccentric direction is better than the radial in these cases also. A two-factor analysis of variance was carried out on the data for both gaps, comparing the isoeccentric direction with the radial, and this yielded significance values of above 95% both for the difference between the directions, and for an interaction effect with location.

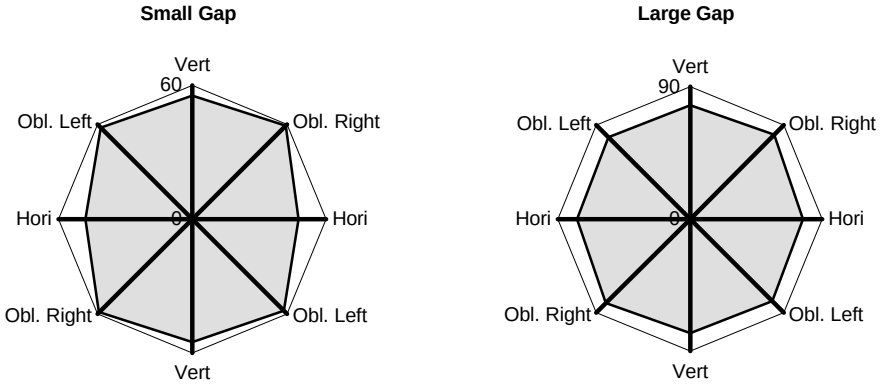


Figure 6.18: Performance by direction of motion

6.3.6 Distractors

Figure 6.20 compares the overall performance by number of distractors in experiment 3, where distractors were added which were visually dissimilar to the colliding entities, and experiment 4, where distractors were added which were visually identical. It is obvious from these diagrams that performance in the latter case was much poorer than in the former. The effect of the number of distractors was also different in both cases. The addition of increasing numbers of visually similar distractors appears to cause a deterioration of performance in experiment 4, whereas the addition of bright red ones seems to have actually improved performance in experiment 3.

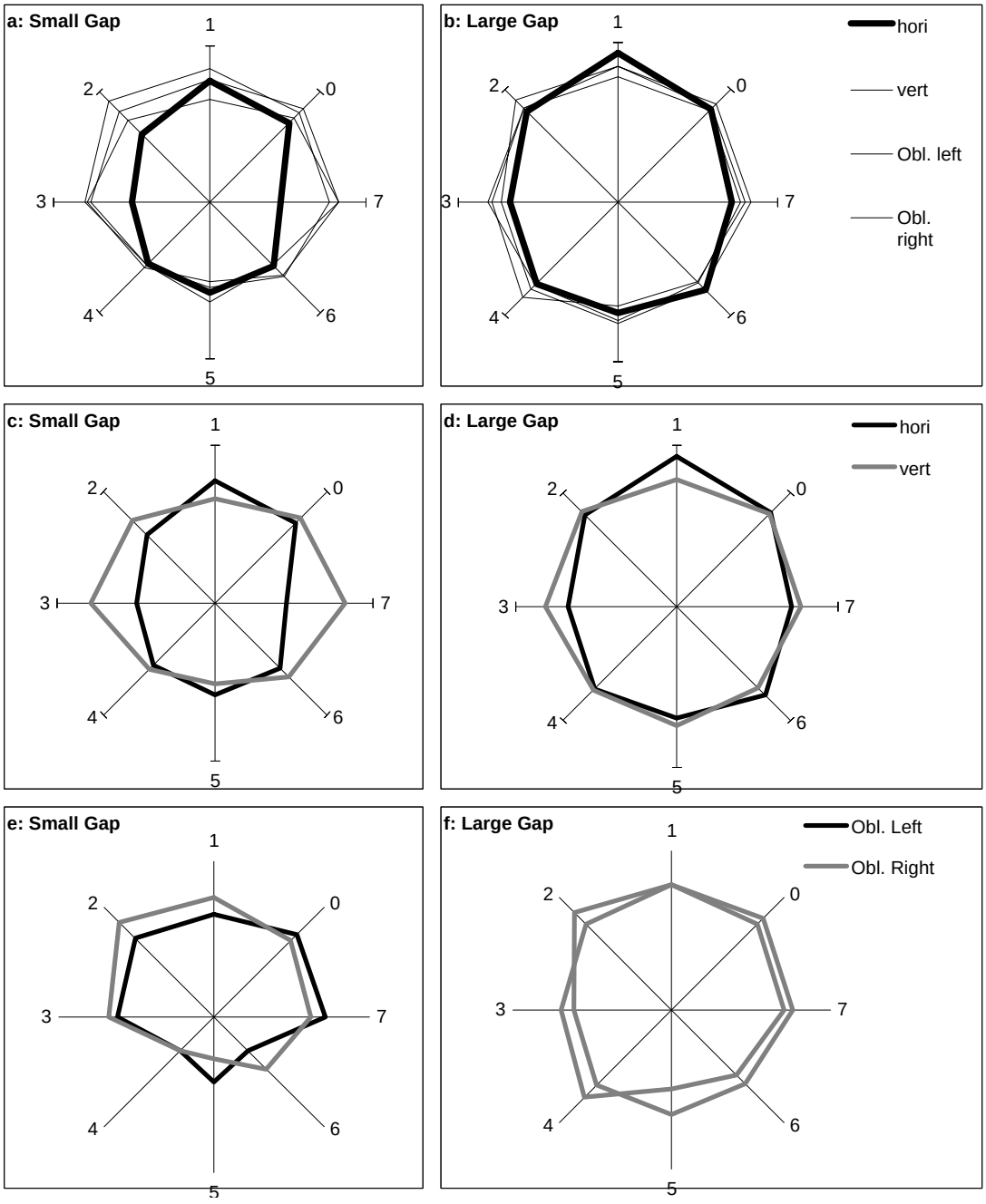


Figure 6.19: Performance per direction of motion by location

Analysis of the data averaged over all other factors yields a significant effect for the number of visually similar distractors in experiment 4, and no evidence for any effect for the number of visually different ones in experiment 3. A similar analysis by gap size yielded no evidence of any effect in experiment 3 (different), and in experiment 4 (same distractors) a strong effect was evident for the large gap, and a weaker effect (only 60% significance) for the smaller gap size. Experiment 4 most closely resembles the real-world situation with which we concern ourselves, i.e. the animation of large numbers of visually homogeneous objects. To analyse the effects present in more detail, we have plotted the effect of eccentricity by number of distractors in Figure 6.21, and vice-versa in Figure 6.22. We can see from Figure 6.21 that the fall-off in performance with eccentricity is much more severe in the case of 5 and 9 distractors than with only one distractor for both gap sizes. The difference is not so severe between 5 and 9 distractors, but there is a difference at the further eccentricities, especially for the larger gap.

Figure 6.22 illustrates this by inverting the graphs. Now we can see that the fall-off in performance with an increase in distractors is most pronounced at the furthest eccentricity. Hence, it appears the the number of visually similar distractors does have an effect, but this effect is not necessarily a continuous function. Performance in the presence of a very small number of distractors (in our case, only one) is much better than in the presence of a larger number of distractors, but the larger number of distractors do appear to also provide a “pop-out” effect at closer eccentricities. This is due to the fact that the distractors, although visually similar to the colliding entities, are ***functionally dissimilar***, in that they do not change direction but continue along a linear trajectory. At further eccentricities the difficulty of the task increases and it becomes more of a serial search. At this level, the number of distractors becomes more important as the strength of the “functional pop-out” becomes less strong.

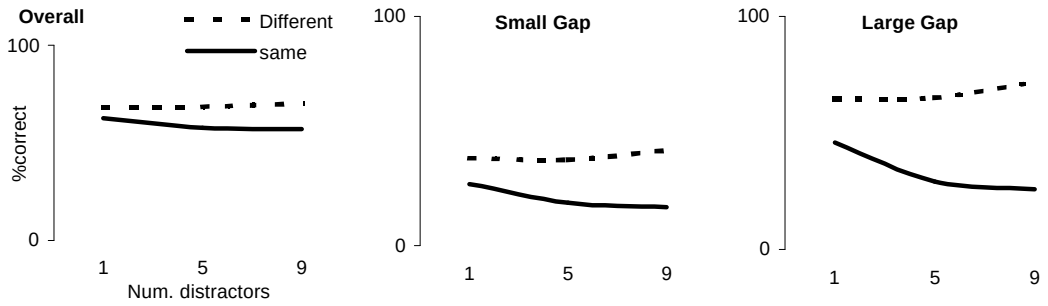


Figure 6.20, performance by number of distractors in experiments 3 and 4

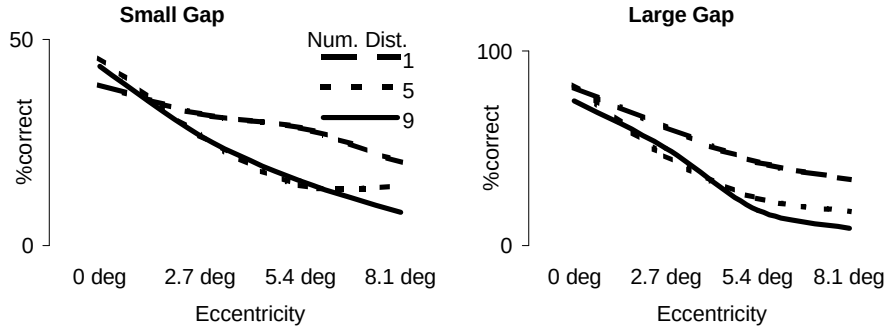


Figure 6.21: The effect of eccentricity by number of distractors in Experiment 4

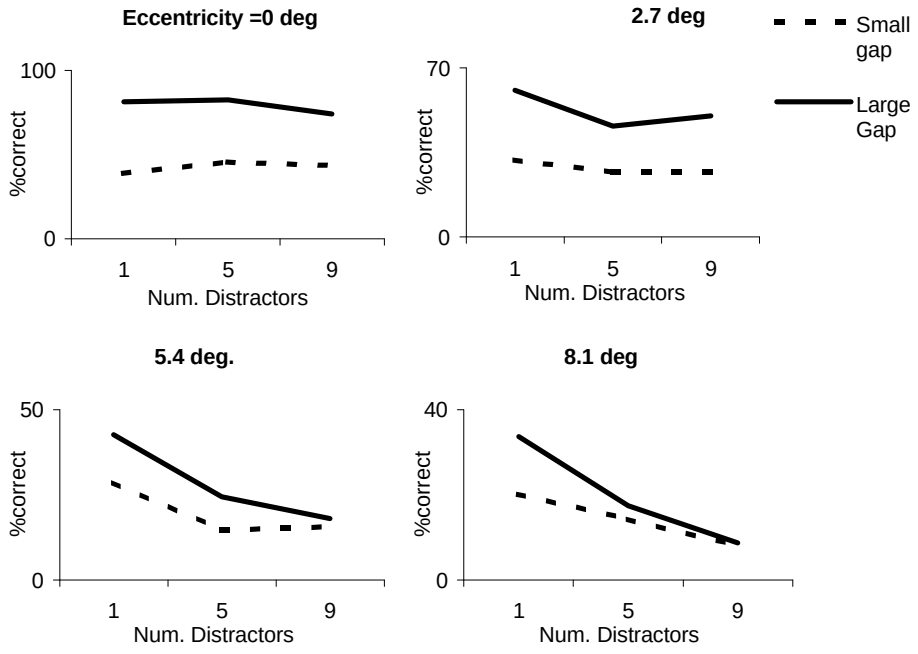


Figure 6.22: The effect of number of distractors by eccentricity in Experiment 4

6.4 Validation and Discussion

The main questions we asked ourselves at the beginning of this chapter were:

(a) Is the model based on perceptual function f_2 a valid one?

(b) Under what circumstances is the model invalid, or incomplete?

We have conducted a comprehensive suite of experiments to examine human perception of collision anomalies, and we are now in a position to provide answers to these questions. Perceptual function f_2, shown in Figure 6.23a for constant C=100, provides a model of collision perception based on eccentricity and separation, which shows an exponential fall-off in performance with eccentricity, with performance also worsening with smaller gap sizes. We stressed that we were modelling perception of collision anomalies with large numbers of visually homogeneous objects. This was the situation in the tests of our 3-d application, the results of which were presented in Chapter 5. Figure 6.23b shows the results of psychophysical tests under such circumstances (i.e. in the presence of 9 visually similar distractors). We can see that the model is a very good approximation to the observed behaviour.

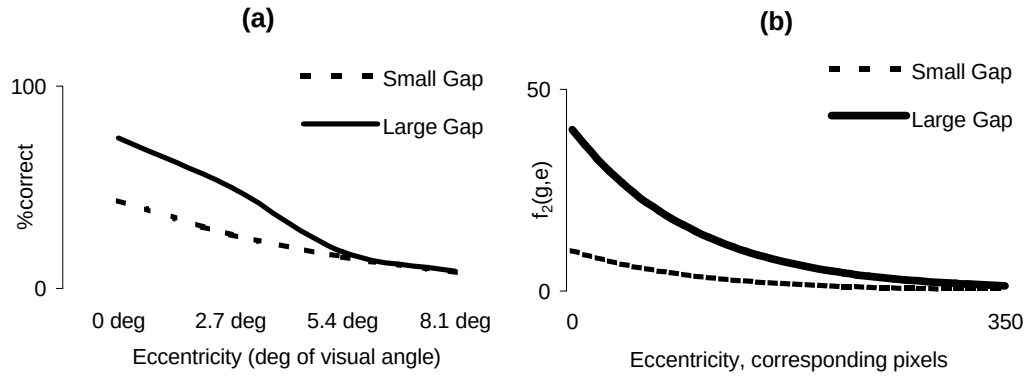


Figure 6.23: Performance by eccentricity and gap size; observed (a), modelled (b)

To answer the second question, our model is only valid in the presence of visually similar distractors. We have seen in Section 6.4.5 that direction of motion had a strong effect in the absence of distractors, and that this effect was more important than the eccentricity effect. We also saw in Section 6.4.4 that location was an important factor under certain circumstances. However, incorporating these effects into a prioritisation algorithm would be computationally expensive for many objects, and hence our approach is probably not suitable under these circumstances. Nevertheless, the effect of eccentricity and gap size are so over-powering in the situation which we are modelling, i.e. many visually homogeneous objects, that this over-shadows any slight effect that other factors may have. Hence, we are justified in omitting these factors from our model.

It could be argued that there are many other functions which could be used to model the same behaviour, and this is true. However, in the most successful tests described in Chapter 5, we used eccentricity alone for the prioritisation algorthm, i.e. priority queue scheduling. This function was only used to approximate perceived inaccuracy. Finding another, better-fitting curve would not alter the results significantly, because the importance is in the relative importance of collisions which are near, to those which are further away. It would be possible to incorporate the effect of the number of distractors into model f_2, perhaps by changing the value of the constant C. From Figure 4.11 in Chapter 4, we can see that higher value of C (1000) give a more gradual fall-off than the lower value(100). We can see from Figure 6.23 that a value of C=100 corresponds to approximately 10 objects. By reducing the value of C in inverse proportion to the increasing number of distractors, we can make a steeper fall-off with eccentricity. In addition to this extension to the model, we could adapt our priority-queue scheduling algorithm to reduce the size of the circle within which collisions are added to the high-priority queue. This would handle the type of "tunnel-vision" effect that is caused by increasing numbers of objects.

However, we do not yet know the effect of larger numbers of distractors, or indeed the effect of the location and patterns of such distractors can have. For example, a bunch of distractors closely clumped together beside two colliding entities will make it very difficult for the viewer to observe any anomaly. The same bunch of distractors located at a distance from the colliding entities will have a much less serious effect. Further studies should consider such situations.

Nevertheless, statistically we can expect that with larger numbers of objects, the probability of observing a collision anomaly decreases. Hence, we can suggest another plausible model (see Figure 6.24) which incorporates the effect of visually homogeneous distractors. It also includes a scaling of size 2 to extend the range from 0 to 100 and not 0 to 50:

$$f\left(g_i^{'}, e_i, d\right) = \frac{g_i^{'}}{\exp\left(\frac{e_i}{C}\right)} \qquad \text{where} \quad C = \frac{d}{100} + \frac{1000 - d}{10}$$

where $g_i^{'}$ is the estimated 2-dimensional error gap size between the entities in collision i, e_i is the eccentricity of the centre of collision i from the fixation point **F**, and d is the number of visually homogeneous objects being animated ($d \geq 10$, the minimum number of objects for which our model is valid).

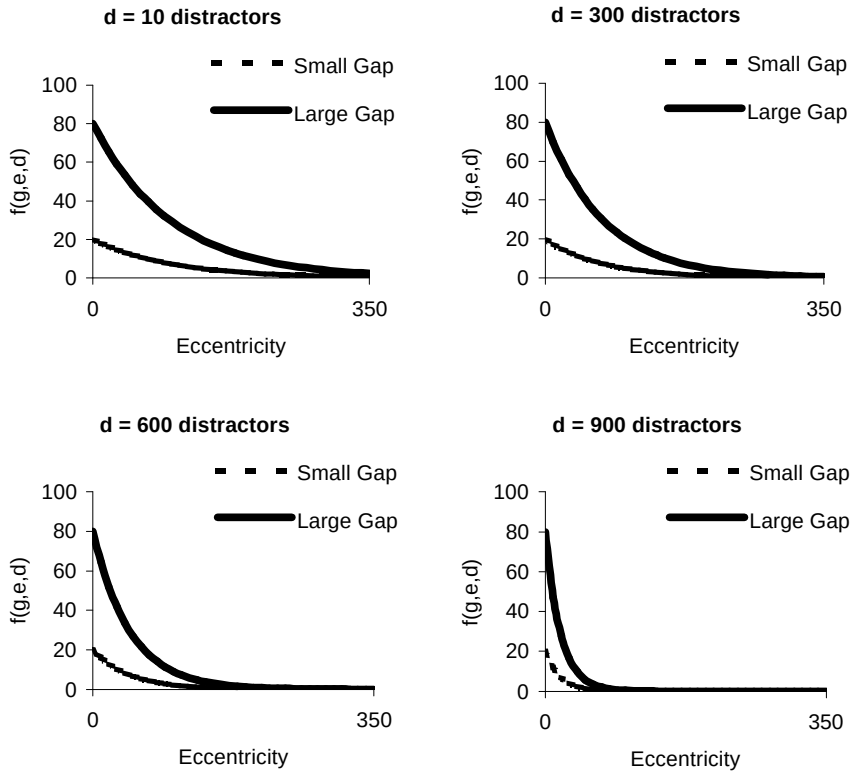


Figure 6.24 Perceptual function f(g=gap size,e = eccentricity,d = num. distractors)

7 Conclusions and Future Work

7.1 Contributions

In this thesis we have developed *the first perceptually-adaptive collision detection algorithm*. *New collision scheduling strategies* were also presented and evaluated, along with a *new interruptible algorithm* to test for the intersection between two sphere trees. The algorithms are described in Chapter 4, along with a *new model of human visual perception of collisions*. In Chapter 5 we showed the feasibility of using this model as the basis for perceptual scheduling of collisions in a real-time animation of large numbers of homogeneous objects. It has been demonstrated that by using a priority queue scheduling algorithm, perceived inaccuracy can be significantly improved*. In the first study of its kind*, described in Chapter 6, we validated this model psychophysically.

This work is not only relevant for the problem of Collision Detection, but also for other applications where the processing of fine detail leads to a computational bottleneck. Our model could easily be adapted to accomplish perceptually-sensitive real-time shadows, shading, and other such techniques. Our psychophysical experiments can provide a starting point for other computer graphics practitioners who wish to enhance the realism of their real-time animations. We have also contributed to the fields of Human Computer Interaction, and eye-movement research, by proposing a potential *new application of eye-tracking technology and eye-movement analysis* for interactive computer systems.

7.2 Future Work

At the moment, the collision time-step in our animation system is equal to the rendering time-step, which can lead to objects interpenetrating or tunnelling through each other if the time-step is too large. We are working on adapting the time-step for high-priority collisions by approximating the collision point through backtracking or interpolation. However, it may be that some level of interpenetration must be accepted as a trade-off, so we must also study the perceptual response of the human visual system to this anomaly also.

The model of collision perception which we use is very specific. It represents typical human reactions to collision anomalies, where large numbers of homogeneous objects are being animated. Work is in progress to make it applicable in more general cases. Other factors, such as location and direction of motion, velocity, acceleration, colour and luminance can have very strong effects under certain circumstances, and if necessary must also be included if the model is to be truly representative of human behaviour. The psychophysical investigations conducted to date have largely been a test of how collision detection can be prioritised effectively. Similar tests might be done to determine how sensitive viewers are to approximations of particular responses to collisions and implement this in our prioritisation scheme.

A more realistic collision response needs to be generated, using the laws of physics. Further behavioural detail still needs to be added to the system to increase the realism and the general applicability of the system. For instance, friction forces during collisions, gravity and elasticity of collisions have been experimented with but are not fully implemented in the current system. Such factors would undoubtedly add further complexity to the system and we need to investigate how much more complexity we can afford and how we might also cull these behaviours in a fully adaptive real-time system. The effects on this process of reduced information about points of contact is also being investigated [O'Sullivan and Dingliana 1999]. It would also be interesting to implement the real-time animation of rigid objects which break or disintegrate in some way upon contact, e.g: plates or glasses smashing when dropped on the ground, as was achieved in non-real-time in [O'Brien and Hodgins 1999].

A fully time-critical system has never been developed in which each of the sub-systems, i.e. rendering, motion synthesis, and collision handling, have all been interruptible. Work is ongoing is several areas to build such a system. We have stated that the approach taken in this thesis is also applicable in other areas of computer graphics. For example, real-time animation systems employ shadows to more clearly illustrate object interaction, and to improve realism. However, shadow generation imposes a significant penalty in terms of the time required to render a scene, especially as the complexity of the scene and the number of polygons needed increases. A "heuristical" approach to real-time shadow generation, using different levels of detail (LOD) of shadows to achieve optimum realism within a target time was used in [Meaney and O'Sullivan 1999].

It would be useful to evaluate the feasibility of a client/server approach to collision handling. One processor, the server, will perform fully accurate collision detection and motion synthesis for the objects in a virtual environment, while any number of clients will interact with this environment, from completely different view-points. The server will not be responsible for any rendering, as each client updates their own view of the VE. In addition, each client should implement interruptible collision detection, so that it is not over-dependent on the connection with the server for information about the current state of the objects and their interactions. It is envisaged that there would be one centralised scheduler located on the server. The client will therefore need to correct any errors in its approximations when the accurate data is received from the server. The implementation of parallel algorithms for collision handling on a cluster of PCs is also planned.

Bibliography

[Albright 1989] Albright, T.D. (1989). Centrifugal directional bias in the middle temporal visual area (MT) of the macaque. Vis. Neurosci. 2, 17, pp 7-188.

[Aubert and Foerster 1857] Aubert, H. Foerster, 0. (1857). Beiträge zur Kentniss des indirekten Sehens (I). Untersuchungen über den Raumsinn der Retina. Arch. Ophthamology, 3, pp 1-37.

[Barzel et al. 1996] Barzel, R. Hughes, J.F. Wood, D.N. (1996) Plausible Motion Simulation for Computer Graphics Animation. Computer Animation and Simulation '96. 183-197.

[Beard et al. 1997] Beard, B.L. Levi, D.M. Klein, S.A. Vernier Acuity with Non-simultaneous Targets: The Cortical Magnification Factor Estimated by Psychophysics. Vision Research, Vol. 37, No. 3, pp. 325-346.

[Borgolte et al. 1993] U.Borgolte, H.Hoyer, F.Wrosch. Online Collision Avoidance for two robots in 3D-space. Proc. of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan. 1919-1926

[Cameron 1990] Cameron, S.A. Collision Detection by Four-Dimensional Intersection Testing. IEEE Transactions on Robotics and Automation. 6(3) 291-302.

[Cameron 1997] Cameron, S.A. Enhancing GJK: Computing Minimum Penetration Distances Between Convex Polyhedra. Proceedings of the Int. Conf. On Robotics and Automation. 3112-3117.

[Carlson and Hodgins 1997] Carlson D.A. Hodgins, J.K. Simulation Levels of Detail for Real-time Animation. Proceedings Graphics Interface ’97. 1-8.

[Carrasco and Frieder 1997] Carrasco, M. Frieder, K.S. Cortical Magnification Neutralizes the Eccentricity Effect in Visual Search. Vision Research. 37(1) 63-82.

[Chande et al. 1993] Chande, P.K. Shrivastava, M. Sharma, G.N. Neural Assisted Robot Arm Collision Avoidance. In SICE 93 Proc. of the 32nd SICE Ann. Conf. Inter. Sess. Kanusawa, Japan. 1547-1550

[Chenny and Forsythe 1997] VIew-dependent Culling of Dynamic Systems in Virtual Environments. ACM Symposium on Interactive 3D Graphics 1997

[Chung et al. 1996] Chung, S.T.L. Levi, D.M. Bedell, H.E. (1996) Vernier in Motion: What Accounts for the Threshold Elevation? Vision Research, Vol 36, No. 16, pp. 2395-2410.

[Cohen et al. 1995] Cohen, J.D. Lin,M.C. Manocha, D. Ponamgi,M.K.(1995) I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scaled Environments. Proceedings of ACM Int. 3D Graphics Conference. 189-196.

[Deering 1992] Deering, M. (1992). High Resolution Virtual Reality. Computer Graphics, 26, 2, 195-202

[DeValois and DeValois 1988] DeValois, R.L, and DeValois, K.K. (1988). Spatial Vision. New York: Oxford University.

[Dobkins and Albright 1993] Dobkins, K.R. Albright, T.D. What Happens if it Changes Color when it Moves?: Psychophysical Experiments on the Nature of Chromatic Input to Motion Detectors. Vision Research 33(8) 1019-1036.

[Duchowski 1998] Duchowski, A.T. Incorporating the Viewer's Point-Of-Regard (POR) in Gaze-Contingent Virtual Environments. Proceedings of The Engineering Reality of Virtual Reality, SPIE.

[Fahle 1986] Fahle, M. Curvature detection in the visual field and a possible physiological correlate. Experimental Brain Research, 63, 113-124.

[Ferwerda et al. 1996] Ferwerda, J.A. Pattanaik, S.N. Shirley, P. Greenberg, D.P. (1996) A Model of Visual Adaptation for Realistic Image Synthesis. SIGGRAPH '96. 249-258.

[Ferwerda et al. 1997] Ferwerda, J.A. Pattanaik, S.N. Shirley, P. Greenberg, D.P. (1997) A Model of Visual Masking for Computer Graphics. SIGGRAPH '97. 143-152.

[Funkhouser and Sequin 1993] Funkhouser,T.A. Sequin, C.H. (1993) Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. SIGGRAPH '93 247-254.

[Gilbert et al. 1988] A fast procedure for computing the distance between complex objects in three-dimensional space. IEEE Journal of Robotics and Automation. Vol 4, 3, 193-203.

[Glenstrup and Engell-Niellsen 1995] Glenstrup, A.J. Engell-Nielsen, T. Eye-Controlled Media: Present and Future State. Under-graduate thesis, Laboratory of Psychology, University of Copenhagen.

[Gordon 1996] I.E.Gordon. *Theories of Visual Perception*. Wiley, U.K.

[Gossweiler 1994] Gossweiler, R. A System for Application-Independent Time-Critical Rendering. ACM Proceedings of SIGGHI'94 (short paper). 261-262

[Gottschalk et al. 1996] Gottschalk, S. Lin, M.C. Manocha, D. (1996) OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. SIGGRAPH '96.

[Graziano et al. 1994] Graziano, M.S.A. Anderson, R.A., Snowden, R.J. (1994) Tuning of MST Neurons to Spiral Motions. J. Neuroscience, 14(1): 54-67.

[Greenberg et al. 1997]. Greenberg, D.P. Torrance, K.E. Shirley, P. Arvo, J. Ferwerda, J.A. Pattanaik, S. Lafortune, E. Walter, B. Sing-Choong, F. Trumbore, B. A Framework for Realistic Image Synthesis. SIGGRAPH'97. 477-494.

[Hallet 1986] Hallet, P.E. Eye Movements, in Boff, Kaufman, Thomas (Eds.) Handbook of Perception and Human Performance, Vol. 1 10.1 - 10.112.

[Hariyama et al. 1994] Hariyama, M. Hanyu, T. Kameyama. M. A Collision Detection Multiprocessor for Intelligent Vehicles Using a High-Density CAM. In Proc. of the Intelligent Vehicles '94 Symp. Paris, France. 143-148

[Hodgins et al. 1998] Hodgins, J.K. O'Brien, J.F. Tumblin, J. Perception of Human Motion With Different Geomentric Models. IEEE Transactions on Visualization and Computer Graphics. 4(4) 307-316.

[Hoppe 1996] Hoppe, H. Progressive meshes. SIGGRAPH 96. 99-108.

[Hoppe 1997] Hoppe, H. View-dependent refinement of progressive meshes. Siggraph'97, 189-198

[Hoppe 1998] Hoppe, H. Smooth view-dependent level-of-detail control and its application to terrain rendering. IEEE Visualization '98. 35-42

[Hubbard 1995] Hubbard, P.M. (1995) Collision Detection for Interactive Graphics Applications. IEEE Trans. on Vis. and Comp. Graphs. 1(3) 218-230.

[Hubbard 1996] Hubbardm P.M. Approximating Polyhedra with Spheres for Time-Critical Collision Detection. ACM Transactions on Graphics, 15(3) 179-210

[Hubel and Wiesel 1968] Hubel, D.H. Wiesel, T.N (1968). Receptive fields and functional architecture of monkey striate cortex. J.Physiology, 195, 215-243

[Jacob 1993] Jacob, R.J.K. What You Look At is What You Get: Eye Movement User Interfaces. IEEE Computer, 26(7) 65-67

[Jacob 1994] Jacob, R.J.K. New Human-Computer Interaction Techniques. In Human-Machine Communication for Educational Systems Design. Brouwer-Janse, M.D. and Harrington, T.L. 131-138.

[Jacob 1995] Jacob, R.J.K. (1995) Eye Tracking in Advanced Interface Design. In Barfield,W. and Furness, T.A. (Eds) Virtual Environments and Advanced Interface Design. 258-288.

[Janott and O'Sullivan 1999] Janott, M. O'Sullivan, C. Interactive Perception of Multiresolution Meshes. ECEM'10, 10th European Conference on Eye Movements, (To Appear).

[Kitamura et al. 1994] Kitamura, Y. Takemura, H. Ahuja, N. Kishino, F. Efficient Collision Detection Among Objects in Arbitrary Motion Using Multiple Shape Representations. Proceedings 12th IAPR Int. Conf. On Pattern Recognition, Jerusalem, Vol.1. 390-396.

[Klosowski et al. 1998] Klosowski, J.T. Held, M. Mitchell, J.S.B. Sowizral, H. Zikan, K. (1998) Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. IEEE transactions on Visualization and Computer Graphics 4(1).

[Krishnan et al. 1998, a] Krishnan, S. Pattekar, A. Lin, M. Manocha, D. Spherical Shell: A higher order bounding volume for fast proximity queries. In Proc. Of Third International Workshop on Algorithmic Foundations of Robotics.

[Krishnan et al. 1998, b] Krishnan, S. Gopi, M. Lin, M. Manocha, D. Pattekar, A. Rapid and Accurate Contact Determination between Spline Models using ShellTrees. Eurographics Association, 1998.

[Lin 1993] Lin, M.C. Efficient Collision Detection for Animation and Robotics. PhD thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, December 1993.

[Lin and Canny 1991] Lin, M.C. Canny J.F. Efficient algorithms for incremental distance computation. IEEE Conference on Robotics and Automation, pp 1008-1014.

[Lin and Gottschalk 1998] Lin, M.C. Gottschalk, S. Collision Detection between Geometric models: A Survey. Proceedings of IMA conference on Mathematics of Surfaces.

[Luebke and Erikson 1997] Luebke, Erikson, View dependent simplification of arbitrary polygonal environments, SIGGRAPH'97

[Mahoney and Ullman 1988] Mahoney,J.V. Ullman,S. (1988) Image chunking defining spatial building blocks for scene analysis. Z.Pylyshyn (Ed.) Computational processes in human vision: An interdisciplinary perspective.

[McConkie 1990] G.W.McConkie "Where vision and congnition meet". In Proceedings of the HFSP Workshop on Object and Scene Perception, Leuven, Belgium.

[McNamara et al. 1998] McNamara, A. Chalmers, A. Troscianko, T. Reinhard, E. Fidelity of Graphics Reconstructions: A Psychophysical Investigation. In Proceedings of the 9th Eurographics Rendering Workshop, 237--246

[Meaney and O'Sullivan 1999] Meaney, D. O'Sullivan, C. Heuristical Real-time Shadows. In Magnenat-Thalmann, N., Thalmann, D. (eds.) Computer Animation and Simulation '99, 167-176.

[Milton et al. 1997] Milton, J.S. McTeer, P.M. Corbet, J.J. Introduction to Statistics. McGraw Hill.

[Mirtich 1998] Mirtich, B. V-Clip: Fast and Robust Polyhedral Collision Detection. ACM Transactions on Graphics. 17(3) 177-208

[Myskowski et al. 1999] Myszkowski, K., Rokita, P., Tawara, T. Perceptually-informed Accelerated Rendering of High quality Walkthrough Sequences. Rendering Techniques 1999.

[Nies et al. 1998] Nies, U. Bedenk, B., Heller, D. Radach, R. (1998). Eye movement patterns during search in a homogeneous background. In Becker, W., Deubel, H. and Mergner, T. (Eds). Current Oculomotor Research: Physiological and Psychological Aspects. New York: Plenum Publishers.

[O'Brien and Hodgins 1999] O'Brien, J. F., Hodgins, J. K., (1999) "Graphical Modeling and Animation of Brittle Fracture". Proceedings of ACM SIGGRAPH 99.

[O'Rourke and Badler 1979] Decomposition of three-dimensional objects into spheres. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(3) 295-305.

[O'Sullivan and Dingliana 1999] Realtime Collision Detection and Response using Sphere-trees. Proceedings of the Spring Conference on Computer Graphics, Bratislava, 83-92.

[O'Sullivan and Radach 1999] The Functional Field of View for Collision Perception in Real-time Animation Systems. ECEM'10, 10th European Conference on Eye Movements, (To Appear).

[O'Sullivan and Reilly 1997] O'Sullivan, C. Reilly, R. REACT: REal-time Adaptive Collision Testing, an Interactive Vision approach. In Thalmann, D. van de Panne, M. (eds.) Computer Animation and Simulation '97. 163-175.

[O'Sullivan et al. 1999] O'Sullivan, C. Radach, R. Collins, S. A Model of Collision Perception for Real-Time Animation. In Magnenat-Thalmann, N., Thalmann, D. (eds.) Computer Animation and Simulation '99, 67-76.

[Ohshima et al. 1996] Ohshima, Yamamoto, Tamura, Gaze-directed adaptive rendering for interacting with virtual space. Proceedings IEEE VRAIS'96

[Olzak and Thomas 1986] Olzak, L.A. Thomas, J.P. *Seeing Spatial Patterns*, in Boff, Kaufman, Thomas (Eds.) Handbook of Perception and Human Performance, Vol. 1 pp. 7.1 - 7.56.

[Palmer and Grimsdale 1995] Palmer, I.J. Grimsdale, R.L.(1995) Collision Detection for Animation using Sphere-Trees. Computer Graphics Forum, 14(2) 105-116

[Ponamgi et al 1997] Ponamgi, M.K. Manocha, D. Lin, M.C. Incremental Algorithms for Collision Detection Between Polygonal Models. IEEE Transactions on Visualization and Computer Graphics. Vol. 3. No. 1. pp 51-64.

[Preperata and Shamos 1985] Preperata, F.P. Shamos, M.I. Computational Geometry. An Introduction. Springer-Verlag.

[Quinlan 1994] Quinlan, S. (1994) Efficient Distance Computation between Non-Convex Object. Proceedings International Conference on Robotics and Automation. 3324-3329.

[Rabbitz 1994] Rabbitz, R. Fast Collision Detection of Moving Convex Polyhedra. Graphics Gems IV, P.S. Heckbert, Ed. Academic Press, Cambridge, MA, 83-109.

[Rao and Ballard 1997] Rao, R.P.N. Ballard, D.H. Dynamic Model of Visual Recognition Predicts Neural Response Properties in the Visual Cortex. Neural Computation, 9(4) 721-763

[Reddy 1998] Reddy, M. Specification and Evaluation of Level of Detail Selection Criteria. Virtual Reality: Research, Development and Application, 3(2): 132-143.

[Reynolds 1987] Reynolds, C.W. Flocks, Herds, and Schools: A Distributed Behavioral Model. Siggraph '87. 25-34.

[Rind and Simmons 1999] Rind. F.C. Simmons, P.J. Seeing What is Coming: Building Collision-Sensitive Neurones. Trends in Neuroscience 22(5) 215-220.

[Rohlf and Helman 1994] IRIS Performer: A High Performance Multiprocessing Toolkit for Realtime 3D Graphics. Proceedings SIGGRAPH'94, 381-393.

[Rossignac and Borrel 1993] Rossignac, J. Borrel, P. Multi-resolution 3D approximations for rendering complex scenes., in Geometric Modeling in Computer Graphics, Springer Verlag, Eds. B. Falcidieno and T.L. Kunii, 455-465

[Rovamo and Virsu 1979] Rovamo, J. Virsu, V. An estimation and application of the human cortical magnification factor. Experimental Brain Research, 37, 495-510.

[Saarinen 1994] Saarinen, J. Visual search for global and local stimulus features Perception, 23, 237-243

[Saarinen et al. 1987] Saarinen,J. Rovamo, L. and Virsu, V. Texture Discrimination at different eccentricities. Optical Society of America, 4, 1699-1703.

[Sammet and Webber 1988] Sammet, H. and Webber, R. Hierarchical Data Structures and Algorithms for Computer Graphics. 1988 in IEEE Comp. Graphics and Applications. Vol.8 No. 3 48-68

[Schiff and Detwiler 1979] Schiff, W. Detwiler, M.L. (1979) Information used in judging impending collision. Perception 8, 647-658

[Shaffer and Herb 1992] Shaffer, C.A. Herb, G.M. A Real-Time Robot Arm Collision Avoidance System. IEEE Trans. on Robotics and Automation. 8(2) 149-160

[Shene and Johnstone 1991] Shene, C. Johnstone, J. On the planar intersection of natural quadrics. Proceedings of ACM Solid Modeling, 234-244

[Sillitoe et al. 1994] Sillitoe, I.P.W. Batersby, A. Edwards, J. A Parallel Architecture for Efficient Clash Detection. Progress in Transputer and Occam Research. Miles, R. Chalmers, A. (Eds.) IOS Press, 1994. 32-39

[Singh et al. 1995] Singh, K. Ohya, J. Parent, R. Human Figure Synthesis and Animation for Virtual Space Teleconferencing. Proceedings IEEE Viertual Reality Annual Int. Symp. 118-126.

[Skerjanc and Pastoor 1997] Skerjanc, R. and Pastoor, S. (1997) New generation of 3-D desktop computer interfaces, SPIE Proceedings 3012 - Stereoscopic Displays and Virtual Reality Systems IV (The Engineering Reality of Virtual Reality), 439 – 447

[Snyder et al. 1993] Snyder, J.M. Woodbury, A.R. Fleischer, K. Currin, B. Barr, A. Interval methods for Multi-Point Collisions between Time-Dependent Curved Surfaces. SIGGRAPH’93 321-334.

[Stiefelhagen et al. 1997] Stiefelhagen, R. Yang, J. Waibel, A. Tracking Eyes and Monitoring Eye Gaze. PUI'97, Workshop on Perceptual User Interfaces.

[Strasburger et al. 1994] Strasburger, H. Rentschler, I. Harvey, L.O. Jr. Cortical Magnification Theory Fails to Predict Visual Recognition. European Journal of Neuroscience, Vol. 6, pp. 1583-1588.

[Swift et al. 1993] Swift, L.K. Johnson, P.E. Lividas, P.E. Parallel Creation of Linear Octrees from Quadtree Slices" in Communicating with Virtual Worlds, Lausanne Switzerland. 519-522

[Thalmann and Thalmann 1995] Virtual Actors Living in a a Real World. Proceedings Computer Animation 1995. 19-29.

[Tootell et al. 1982] Tootell, R.B.H, Silverman, M.S. Switkes, E, De Valois, R.L. (1982). Deoxyglucose analysis of retinotopic organization in primate striate cortex. Science, 218, 902-904.

[Treisman 1982] Treisman, A. (1982). Perceptual Grouping and Attention in Visual Search for Features and for Objects. Journal of Experimental Psychology: Human Perception and Performance, 8, 194-214.

[Tseng and Wu 1995] Tseng, C. Wu, C. Collision Detection for Multiple Robot Manipulators by using Orthogonal Neural Networks. in Journal of Robotic Systems, 12, 479-490

[Von Den Bergen 1997] Efficient Collision Detection of Complex Deformable Models using AABB Trees." Journal of Graphics Tools, 2(4):1-13.

[Von-Herzen et al. 1990] Von-Herzen, B. Barr, A.H. Zatz, H.R. Geometric Collisions for Time-Dependent Parametric Surfaces. SIGGRAPH'99, 39-48.

[Wampers and van Diepen 1999] Wampers, M. van Diepen, P. M. J. The use of coarse and fine peripheral information during the final part of fixations in scene perception. In W. Becker, H. Deubel, & T. Mergner (Eds.), Current oculomotor research: Physiological and psychological aspects, 257-267.

[Wandell 1995] Wandell B.A. *Foundations of Vision*. Sinauer Associates, Inc.

[Ware and Mikaelian 1987] Ware, C. Mikaelian, H.H. An Evaluation of an Eye Tracker as a Device for Computer Input. Proceedings of ACM Human Factors in Computing Systems and Graphics Interface. 183-188.

[Watson et al. 1997] Managing level of detail through peripheral degradation: effects on search performance in a head mounted display. ACM Trans. Computer-Human Interaction, 4(4)

[Welsh 1996] Welsh, G.F. SCAAT: Incremental Tracking with Incomplete Information. TR-96-051 Oct. 1996. Department of Computer Science, University of North Carolina, Capitol Hill.

[Weymouth 1958] Weymouth, R.W. (1958). Visual sensory units and the minimal angle of resolution. American Journal of Ophthamology, 46, pp 102-113.

[Yap et al. 1987] Yap, Y.L. Levi, D.M. Klein, S.A. (1987) Peripheral hyperacuity: isoeccentric bisection is better than radial bisection. J. Opt. Soc. Am. A, Vol 4, number 8, pp. 1562-1567.

[Youn and Wohn 1993] Youn, J.H. Wohn, K. Realtime Collision Detection for Virtual Reality Applications. Proceedings IEEE Virtual Reality Annual Int. Sym. 18-22.

[Yuan 1995] Yuan, J. A Neural Network Measuring the Intersection of m-dimensional Convex Polyhedra. In Automatica. 31. 517-29

[Zeki 1974] Zeki, S.M. (1974) Functional Organization of a Visual Area in the Posterior Bank of the Superior Temporal Sulcus of the Rhesus Monkey. J. Physiology. 236, 549-573.

[Zeki 1993] Zeki, S. *A Vision of the Brain*. Blackwell Scientific Publication, Oxford.