Studies on Collision Detection Using Ellipsoidal Bounding Volumes

Leung Yuk Leong, Daniel

A dissertation submitted to
The University of Hong Kong
in partial fulfillment of the requirements for
the degree of Master of Philosophy

August 2000

Abstract of thesis entitled

Studies on Collision Detection Using Ellipsoidal Bounding Volumes

submitted by

Leung Yuk Leong, Daniel

for the degree of Master of Philosophy
at the University of Hong Kong
in August 2000

Collision detection is one of the most fundamental problems in computer animation and virtual reality. These two applications both require short delay and high accuracy.

The problem is usually divided into two phases: the board phase and the narrow phase. In the board phase, techniques of spatial decompositions and bounding volumes are applied. In the narrow phase, exact collision detection is performed. The main aim of the board phase is to avoid the expensive calculations in the exact collision detection. In spatial decomposition, the 3D space is being partitioned into clusters or cells. Polygons or objects falling in two different clusters can be classified as separated immediately. Using the bounding volume technique, objects are enclosed in tightly bound basic geometries like cubes or spheres. These bounding volumes are simple enough so that the collision detection between them is computationally cheap. At the same time, they can be built easily.

Because of the observation that most of the natural moving instance such as human beings and animals are in oval shapes, ellipsoidal bounding volumes seem to be an answer for these kinds of objects. Determining whether two ellipsoids separate or not involves only the detection of the signs of roots of their characteristic equation. Therefore, ellipsoid bounding volumes are used in investigations of collision detection..

The algebraic characteristic equation for any two ellipsoids has at least two negative roots. Further, two ellipsoids are separated if and only if the equation has two positive roots. In this thesis, the techniques for building the bounding ellipsoid of an object and setting up the characteristic equation of two ellipsoids as well as the methods to detect the roots of a quartic equation in this special case are studied, discussed and analyzed.

KEYWORDS: Collision Detection, Characteristic Equation, Spatial Decomposition, Covariance Matrix, Affine Transformation.

Chapter 1 Introduction

Collision detection is a fundamental problem in robotics and computer graphics. In robotics, collision detection helps to construct the path of the moving parts of a machine. This path planning for robotics devices avoiding collisions is also known as motion planning. This is under research for almost thirty years [4][14]. However, the result is too slow to be useful. Faster collision detection can improve performance in motion planning.

In computer graphics, collision detection is essential to many interactive applications such as virtual environment walkthrough, computer animation, 3D games and scientific visualization. With collision detection, objects in the virtual space penetrating each other can be avoided.

Although collision detection is very useful and important, it is often omitted in most systems. The major reason is that it is a very expensive computation in terms of processing time. The consequence of this is other alternatives are used to simulate the effect of collision detection. For example, in 3D games usually pre-computed motion path are applied. However, pre-computed motion path would limit the realism of the motion.

These are the reasons for a fast and accurate collision detection algorithm.

1.1 Background

In practice, geometric models are represented as meshes. The brute force approach to detect the collision between two meshes is by performing intersection tests on polygons. For two meshes with M and N polygons, the time complexity for performing this is O(MN). Complex models usually consist of millions of polygons. It will be too costly to apply this test on all the models in a virtual environment.

figure 1. A skull model with its mesh shown, it consists 215360 polygons

To deal with this, detection process is divided into two phases, the board phase and the narrow phase. Objects are enclosed tightly by bounding volumes, and the intersections among these bounding volumes are detected in the board phases. In the narrow phase, exact collision detection is performed on models only when their bounding volumes intersect.

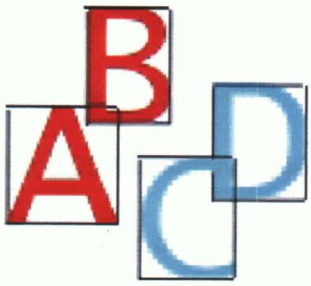

figure 2. Two pairs of intersecting bounding boxes and one pair of object really intersect. Exact collision detection between the red and blue alphabets is unnecessary.

Bounding volume should have simple shape such as sphere, rectangular box or cylinder. It is because detecting collision between these shapes is simple and fast. It should also enclose the model as tight as possible. This can eliminate more unnecessary exact collision detection in the narrow phase. For example, in figure 2, there are four models A, B, C and D. The brute force approach is to detect collision between each pair of them, i.e. A-B, A-C, A-D, B-C and B-D. Instead the detection is divided into two phases. In the board phase, collision is detected between A, B and C, D bounding boxes. Thus, in the narrow phase, exact collision detections are performed on A with B and C with D only. A collision should be detected between C and D.

In summary, the process of collision detection can be more efficient with the help of bounding volume.

1.2 Contribution

In this thesis, we have investigated in using ellipsoid as bounding volume for collision detection. There are two problems we need to solve. We have to understand the condition when two ellipsoids are intersecting and how to build bounding ellipsoids for 3D models.

The contributions of this thesis are listed as follows.

- Collision detection between bounding ellipsoids
- Method of constructing the bounding ellipsoids
- Develop a program to detect collision among objects in 3ds file format
- Compare the performance with other methods

1.3 Motivation

It is common to use human and animal models in animation and virtual reality applications. With the help of a tailor-made bounding volume for collision detection, techniques such as key-framing, motion capture, motion editing and spacetime constraint technique will become more efficient.

There are many kinds of bounding volume, such as OBB[11], bounding sphere[13]. Each of them has it's own advantages and disadvantages. Since human and animal models are usually made up of oval shape sub-objects, ellipsoid seems to give a tighter bound volume than the other. This is the value of using ellipsoidal bounding volume.

figure 3. A human arm with ellipsoid bounding volumes

1.4 General Idea

Two problems needed to be solved in using bounding ellipsoid. The first problem is the way to build this bounding volume. The second problem is the way to detect the collision between two ellipsoids.

1.4.1 Building Ellipsoidal Bounding Volume

There are some difficulties in building a tight bounding volume for an object. There is a trade-off between the tightness of the bounding volume and its complexity. The tighter the bounding volume is required, the more complicate it becomes. The bounding volume can be as complicate as the object itself then it will be meaningless.

Some techniques build multi levels of bounding volume. On the top level, a very simple bounding volume such as sphere or rectangular box is used. Then more complicate and tightest bounding volume is used in lower levels. Collision detection is performed on the top level first. If their bounding volumes collide, the test is made progressively to lower levels. By incremental searching for collision, the performance and accuracy can be adjusted with more flexible.

Challenges were met when building the bounding ellipsoid for an object. For concave polyhedron, many algorithms would try to find its convex hull first [2]. This would decrease the number of points that need to be included in the construction.

1.4.2 Detecting Collision Between Bounding Ellipsoids

It is found that two ellipsoids are separated if their characteristic equation has 2 positive roots [18]. To find the roots of a quartic equation is very time consuming although there are well-known methods to do this. Fortunately, the values of the roots are

not essential, only the signs of them are important in reporting collision. This is an interesting problem in mathematics too.

1.5 Overview of the Thesis

This thesis is organized as a progressive study of my studies on using ellipsoids as bounding volume.

Chapter 1 is an introduction to the collision detection problem. It lays out the usages, constraints and design criteria for a collision detection algorithm, the motivation of the research and studies. It also lays out the reasons for using ellipsoidal bounding volume and the difficulties. Chapter 2 is some previous approaches. Chapter 3 and 4 are the core of the ellipsoidal bounding volume algorithm. Chapter 5 is the experimental results and comparisons with different algorithm.

The core of the ellipsoidal bounding volume algorithm is divided into two parts. The first part is the building of bounding ellipsoid. The second part is the setting up of the characteristic equation of two ellipsoids and detecting the roots of it. Two methods in setting up the characteristic equation and two methods in detecting the signs of the roots are presented.

Chapter 6 is the conclusion of the studies. In appendix, it includes some basic knowledge in matrix operations.

Chapter 2 Related Work

In this chapter, we will give a brief review on the methods for collision detection. They can be generally classified into three categories, they are exact collision detection, bounding volumes and spatial decomposition.

2.1 Exact Collision Detection of Polyhedron

An exact collision detection algorithm is used to detect whether two 3D objects have intersected with each other. An algorithm is devised for detecting collision between two polyhedrons by searching a separating plane between them [5]. The algorithm starts with an initial separating vector, which is defined as the normal of a candidate separating plane, it iteratively refines this vector until a separating plane is found or some termination conditions are reached. The supporting vertices with a separating vector are defined as the vertices of the polyhedron which have either the maximum or the minimum dot product with the separating vector. In each step, the separating vector is modified according to its relation with the supporting vertices. If a separating plane is found, the polyhedrons are declared to be non-colliding. The following figure shows the

separating vector, the supporting vertices and the separating plane of two non-colliding polyhedrons.

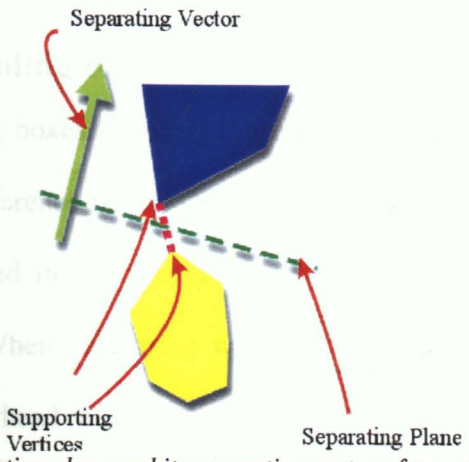

figure 4. One of the separating plane and its separating vector of two objects. The green line connects the two supporting vertices in this case.

Although the searching does converge, it does not have a constant execution time. However, a constant frame rate is required for interactive applications, therefore this algorithm is not suitable for such applications. Although polyhedron is seldom used directly for modeling in virtual reality applications, it can be used as bounding volume which is as tight as the convex hull of the model. We have performed an experiment to analyze the number of iteration required under different situations and the result is shown at the end of this thesis.

2.2 Bounding Volume

Exact collision detection is a very expensive computation, especially for those complex objects. Therefore instead of applying exact collision detection directly on two objects, the usual approach is to detect collision between their bounding volumes first, and apply exact collision detection only when their bounding volumes intersect. This can eliminate the number of exact collision detection significantly. There are several choices

for bounding volumes, the basic requirement is that they should be of simple shape so that we can easily determine whether they have intersected with each other.

2.2.1 Oriented Bounding Box

Oriented bounding boxes (OBB) [11] are very similar to the classical axis-aligned bounding boxes. The difference is that OBB has its three basis vectors aligned with the object. OBBs can be used in many applications, such as ray tracing and interference detection computation. When comparing with classical axis-aligned bounding box and bounding sphere, the overhead in building OBB is greater. However, OBB gives a tighter bounding volume than its counterparts.

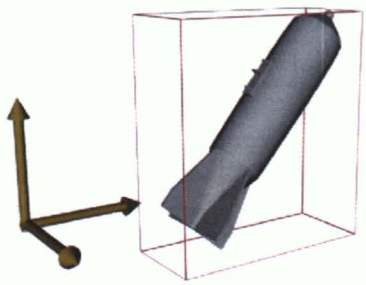

figure 5. An object bounded by an axis-aligned bounding box together with the world coordination axes

2.2.2 OBBTree

An OBBTree decomposes an object into sub-objects hierarchically, where each sub-object is enclosed by an oriented bounding box. Figure 6 shows the steps in constructing the OBBTree for a line segment. First, an oriented bounding box is built to enclose the whole segment. Then the segment is split into two halves by the bisector of the bounding box. Two smaller bounding boxes are built for the two shorter segments. The shorter segment is then further split into smaller segments. This process is repeated

iteratively. Figure 7 shows the corresponding OBBTree for this segment. A smaller segment is represented by a node in the tree. In this way, the tree is built in a top-down manner.

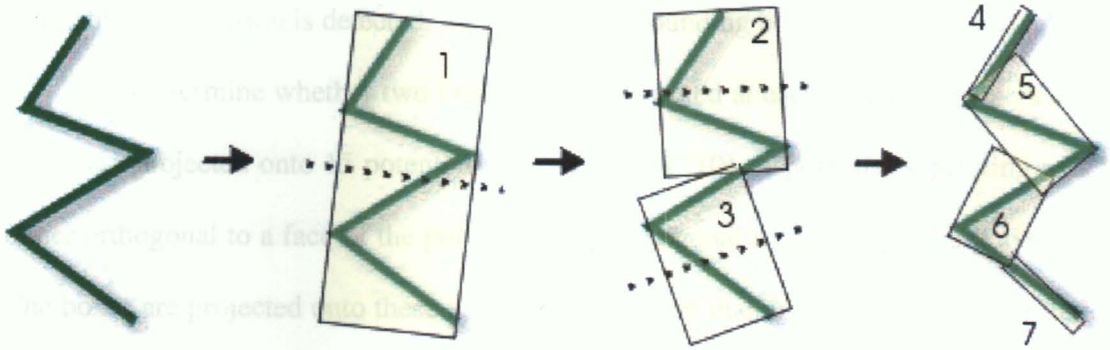

figure 6. 2D line segments being bounded by three oriented-bounding boxes.

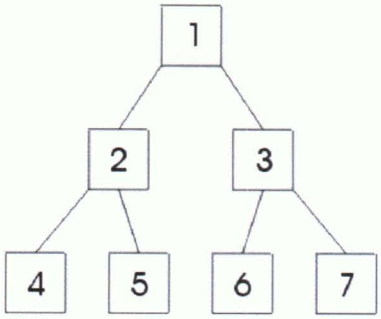

figure 7. The OBBTree structure for the segment

When splitting the segment into smaller segments, we have to determine where to split the segment and how to compute the OBB for the smaller segments. These are decided by the mean and the covariance matrix of the vertices of the object. The center of the OBB is defined as the mean of the vertices. The eigenvectors of the covariance matrix are used as the basis vectors of the OBB. The plane cutting the mean and orthogonal to the longest axis is used to split the object into two smaller sub-objects. The mean and the

covariance matrix of the sub-object are computed and they are used for defining the new OBB.

To perform collision detection, the highest-level bounding boxes of two trees are tested first. If collision is detected, the lower level-bounding boxes are then tested.

To determine whether two OBBs have intersected at each level, the centers of the boxes are projected onto 15 potential separating axes [10]. A potential separating axis is either orthogonal to a face of the polyhedron or orthogonal to an edge of the polyhedron. The boxes are projected onto these axes. Two boxes are disjoint if there is no intersection in every pair of such projections on all potential separating axes. The figure below shows the projections of two OBBs on a separating axis. The translation is the difference between the center of the boxes. The figure also shows that their projections do not intersect when the boxes are separated.

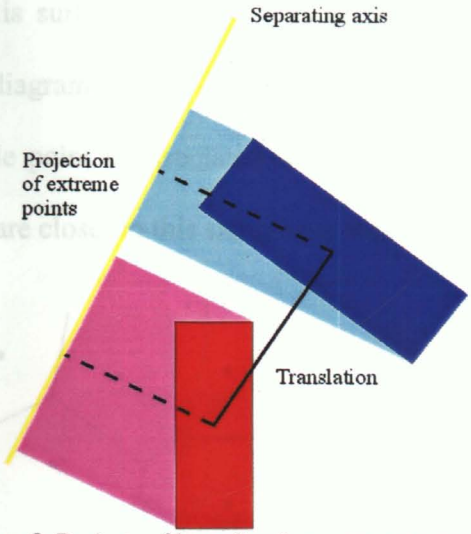

figure 8. Projects of bounding boxes on a separating axis

The advantage of this method is that we can have smaller OBBs for the sub-objects, if the object is split. However, the disadvantage is that we will destroy the

geometric property if the object is to be divided into smaller groups. In order to preserve their properties, we can use a bottom-up approach.

2.3 Bounding Sphere

Two spheres intersect if the distance between their centers is smaller than the sum of their radii. It is simple to find the collision between spheres. However, it is difficult to enclose an object tightly by a sphere. It depends on the aspect ratio of an object. The aspect ratio is the volume of the largest ball that can be contained in an object to that of the smallest ball that can contain the object. It measures how elongated an object is. Bounding spheres are not suitable to enclose objects with small aspect ratio.

2.3.1 Hierarchical bounding sphere

Instead of using a simple bounding sphere, multiple bounding spheres are used in a hierarchical manner [13]. Unlike the OBBTree, it is built in a bottom-up approach.

First, a medial-axis surface [12][13][3] is constructed as the "skeleton" of the object with the Voronoi diagram. A Voronoi diagram of a set of sample points is shown in Figure 9. Every sample point is associated with a Voronoi cell, which is the region containing all points that are closer to this sample point than other sample points.

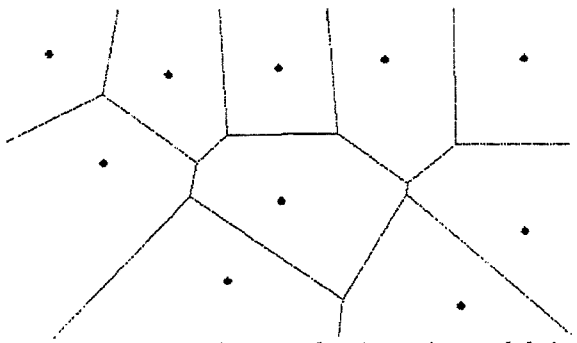

figure 9. A Voronoi diagram showing points and their sites.

In order to find the medial-axis surface of an object, sample points are uniformly placed on the surface of the object. The sample points are distributed according to the point-placement algorithm [9]. Then we find the Voronoi diagram for these sample points. We then use the Voronoi vertices (corners of Voronoi cells) [1] as the centers of bounding spheres.

For each Voronoi vertex, four nearest sample points are used to determine the size of the corresponding bounding sphere. Figure 10 illustrates a 2D example, where three neighboring points lie on the circumference of the bounding circle. It also shows the Voronoi vertices and the spheres centered at two of the Voronoi vertices. The medial-axis surfaces are constructed by connecting the Voronoi vertices inside the object. The Voronoi vertices are the centers of the bounding spheres. This forms the lowest level bounding spheres, as illustrated by Figure 11.

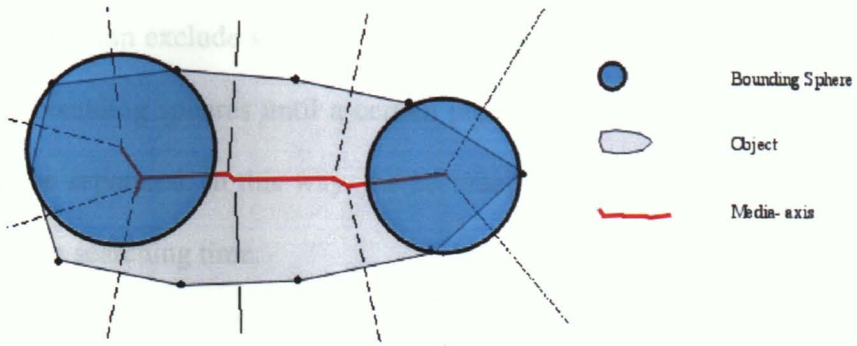

figure 10. Voronoi vertices form the centers of the bounding sphere

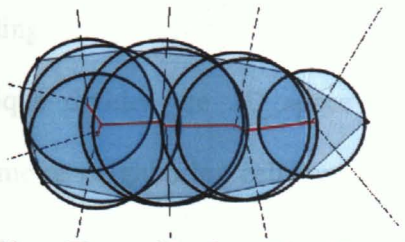

figure 11. The resulting lowest-level bounding spheres

To construct a higher level-bounding sphere, adjacent spheres are merged together, as shown in Figure 12. The new sphere must also cover the points which are originally covered by the smaller spheres. After merging, the number of bounding sphere decreases. At the same time, the Hausdorff distances [17] increase. Merging continues until the Hausdorff distances increase to a predefined threshold value.

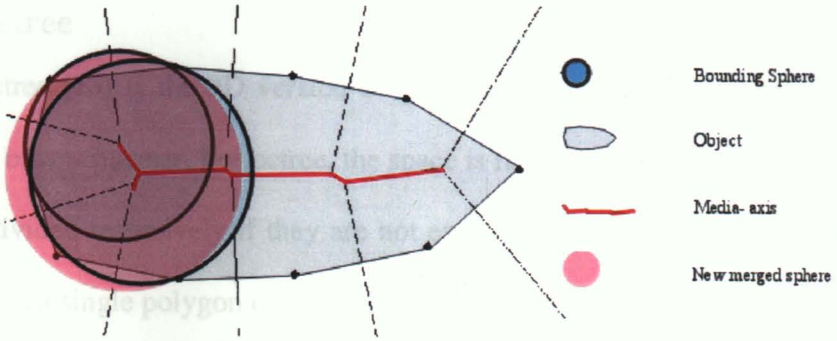

figure 12. Merging of the bounding spheres

To perform collision detection, the top level bounding spheres of two objects are tested first. This can exclude some "obviously" separating objects. The test goes on with lower level bounding spheres until a certain predefined threshold is reached or they are declared to be separated. In this way, the accuracy of the detection can be adjusted as a trade-off of the searching time.

2.4 Spatial decomposition

Besides using bounding volume in the board phase, another technique is spatial decomposition. This technique divides the 3D space into partitions or cells. Only bounding volumes in the same or neighboring cells are tested for collision. In this way, most of the "obviously" non-colliding pairs are excluded so the number of collision detections performed is significantly reduced. Usually these cells are organized in the

form of tree structure. Objects falling into the same partitions are in the same branch of the tree. Objects in different branches will not collide. These trees are built in a preprocessing stage. The time to traverse from the root node to the leave node is bounded by the height of a tree. After reaching the leave nodes, detail intersection tests will be performed among the objects.

2.4.1 Octree

Octree [15] is the 3D version of quadtree. The general idea is to subdivide the space in a binary manner. For octree, the space is first divided into eight equal cells. Cells are then divided recursively if they are not empty. The subdivision continues until each cell contains a single polygon or a single object, or until a certain amount of polygons are left in a single cell. In the stage of dividing cells, polygons may need to be split in order to satisfy these requirements. A 2D case is illustrated by figure 13. In the worst case, the height of the tree can be n, where n is the number of polygons.

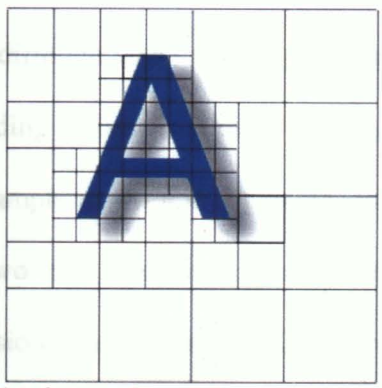

figure 13. The letter A is being split and the space is being divided equally recursively

2.4.2 BSP-tree

A Binary Space Partitioning Tree (BSP-tree) [16] divides the 3D space into partitions called clusters. Instead of subdividing the space equally, BSP tree subdivides

the space according to the orientation of the polygons. A polygon in the scene is chosen as the root of the BSP tree. This polygon is then used as a separating plane to split the space into two half-spaces. All other polygons are grouped in the leave nodes according to the half-space they fell into. Any polygon crossing the separating plane is split in order to make sure that no polygon can fall into both half-spaces. Then a new polygon in each half-space is chosen as the new separating plane. The final structure of the tree will greatly depend on the polygon chosen when dividing the space. A 2D case is illustrated by Figure 14.

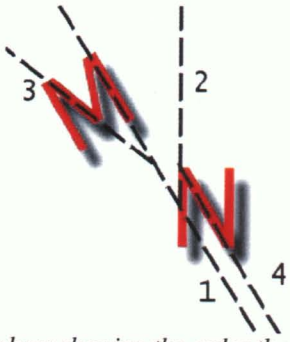

figure 14. A BSP-tree with numbers showing the order the space being divided, note that a polygon of N is split.

The situation is a bit different when applying spatial decomposition on bounding volumes. In this case, a bounding volume is allowed to stay in two cells if it is intersected by a separating plane. An example is given in Figure 15. We have four spheres falling in four cells. Sphere C falls in two cells, one with sphere B and the other one with sphere D. During the board phase collision detection, sphere C will be tested against D and B, but against not A.

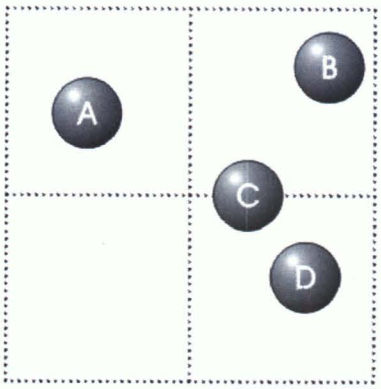

figure 15. Applying spatial decomposition on bounding spheres

Chapter 3 Building Bounding Ellipsoid

In this chapter, we will discuss a method to build a bounding ellipsoid. We use meshes to represent objects. In our implementation, the objects are in 3D Studio (.3ds) file format.

Finding the minimum bounding ellipsoid is not obvious. Instead of building the minimum bounding ellipsoid, we find an optimal bounding volume whose size is as close to that of the minimum ellipsoid as possible.

Building bounding ellipsoids is a preprocess for collision detection and there are two criteria. Firstly, an ellipsoid should enclose the object as tight as possible in order to eliminate more objects in the board phase of the detection. Secondly, the preprocessing should be as automatic as possible because building bounding volume without user interaction is a must for complicated models and scenes in virtual environment.

An ellipsoid can be uniquely determined by its center, the lengths of its three major axes and its orientation. To construct an optimal ellipsoid for enclosing an object, its center should be close to the center of the object. The lengths of its major axes should also be as short as possible in order to minimize the volume of the ellipsoid.

Chapter 3 : Building Bounding Ellipsoid

Given an object in mesh representations, we first triangulate all the polygons in the mesh. Then we compute the mean and the covariance matrix for the vertex coordinates. The mean vertex is given by dividing the sum of vertices by the number of triangles:

$$\text{Mean vertex} = \mu = \frac{1}{3n}\sum_{i=0}^{n}\left(v_1{}^{i} + v_2{}^{i} + v_3{}^{i}\right),$$

where n is the number of triangles and v^i_1, v^i_2 and v^i_3 are the three vertices of the i^{th} triangle.

This mean vertex is taken to be the center of the input vertices. To set the basis of the three major axes, we first construct the covariance matrix, and then find its eigenvectors. The covariance matrix is given by

$$Cov(c,r) = \frac{1}{3n}\sum_{i=0}^{n}\left(\overline{v_1{}^{i}}(c)\overline{v_1{}^{i}}(r) + \overline{v_2{}^{i}}(c)\overline{v_2{}^{i}}(r) + \overline{v_3{}^{i}}(c)\overline{v_3{}^{i}}(r)\right), \qquad 1 \le c,r \le 3,$$

where n is the number of triangles, $\bar{v} = v - \mu$, and Cov(c,r) is the entry of the c^{th} *column* and *the* r^{th} row. Also, $v(1)$, $v(2)$, $v(3)$ means the x, y and z components of the vertex v respectively. The resulting covariance matrix is a 3×3 symmetric matrix.

To determine the length of the three major axes of the bounding ellipsoid, the vertices of the meshes are projected onto each of the 3 major axes. The maximum length of projection on each axis is taken as the length of that major axis of the ellipsoid. However, there may still be some vertices that are not enclosed. Then we resize the generic ellipsoid to include these exterior vertices. The vertices are substituted into the equation of the ellipsoid to check if they are inside the ellipsoid after the resize. Figure 16 shows the result of enclosing a clone with an ellipsoid.

vertex is closer to the head of the missile, which results in the excess space of the bounding ellipsoid near the head of the missile.

To improve this drawback, instead of using the vertices of the mesh, we use the vertices of the convex hull of the object in finding the mean vertex. This can resolve the problem caused by the interior of the object because interior vertex will not affect the convex hull. Another advantage of using convex hull is that the number of vertices can be reduced unless the object itself is a polyhedron. To moderate the influence on the center because of the uneven distribution of vertices, we can resample the convex hull by adding more vertices to regions with low vertices density [2][6]. The determination of the center of the ellipsoid is now improved.

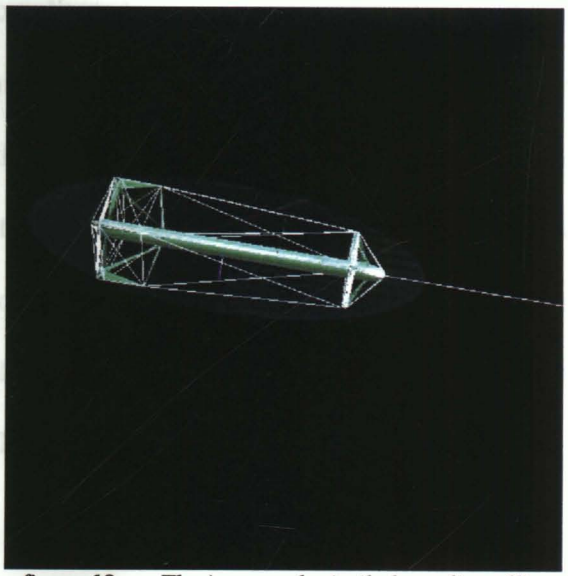

figure 18. The improved missile bounding ellipsoid.

Human parts such as heads are in oval shape. We try to enclose a skull model with an ellipsoid as an example. The model that we use consists of 215360 triangles. By directly finding the mean vertex by the vertices of the meshes, we obtained the result as shown in figure 19.

figure 19. A bounding ellipsoid for a skull

Due to the complicated interior of the model near the jaw, the center of the bounding ellipsoid shifted a long way towards it. This results in the large empty space introduced between the jaw and the ellipsoid. To overcome the problem of complex interior structure of the model, we find the convex hull for the skull and build the bounding ellipsoid according to it. The larger empty space near the jaw has disappeared. However, the relatively more complicated patches near the top of the skull now shift the center towards them. This is shown in figure 20.

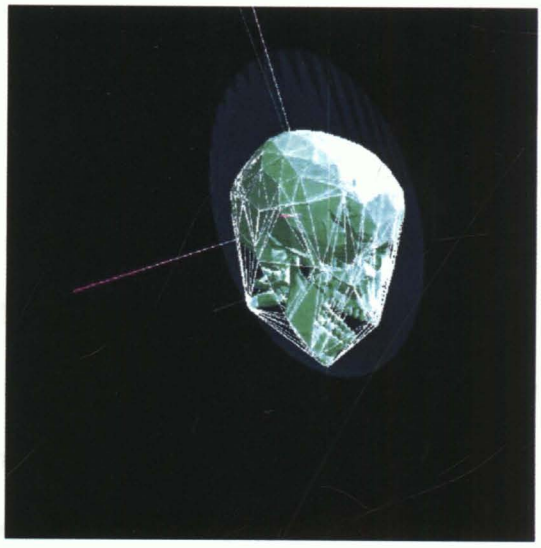

figure 20. The ellipsoid center now shifts to the top of the skull.

By redistributing the vertices of the convex hull, we finally obtain a satisfactory bounding ellipsoid. Redistribution normally means adding more vertices to the model. However, the method that we use does not produce more vertices. We notice that if a region contains more vertices, it implies that there will be more triangles and the triangles are smaller. So instead of redistributing additional vertices in a certain region, we assign weighting to the vertices. By this weighting, some vertices contribute more to the position of the mean than the other. The weighting is assigned according to the area of the triangle. The greater the area of a triangle, the higher the weighting its vertices will be assigned. The calculation of the mean vertex now becomes,

$$\text{Mean vertex} = \mu = \frac{1}{3n \times totalweighting} \sum_{i=0}^{n} \left({v_1}^i + {v_2}^i + {v_3}^i \right) \times weighting^i$$

where $weighting^i = \dfrac{area\ of\ i^{th}\ triangle}{area\ of\ the\ smallest\ triangle}$

In the left image shown in figure 21, the vertices at the top of the skull pull the center along the indicated direction while the vertices in the jaw pull the center in the other direction. The difference in the number of blue arrow has a net effect in shifting the center higher. The green arrows show the cancellation of the horizontal pulling effect and hence the center does not shift horizontally.

In the right image of figure 21, by assigning different weighting to the vertices according to the area of the triangles, it minimizes the effect of the vertices at the top. The result is shown in figure 22.

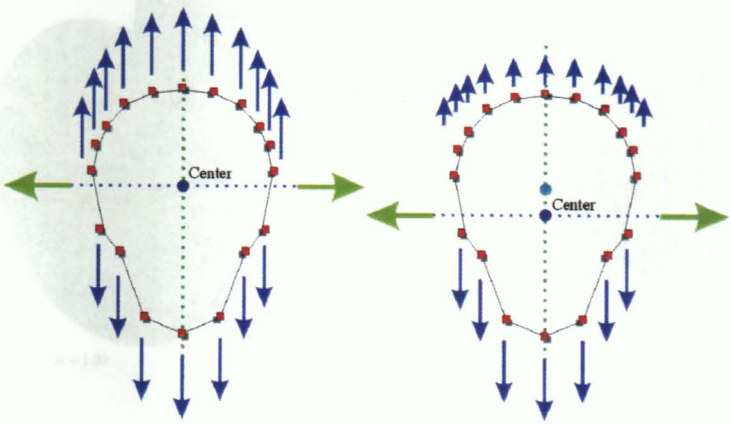

figure 21. Weighting of vertices on the skull surface

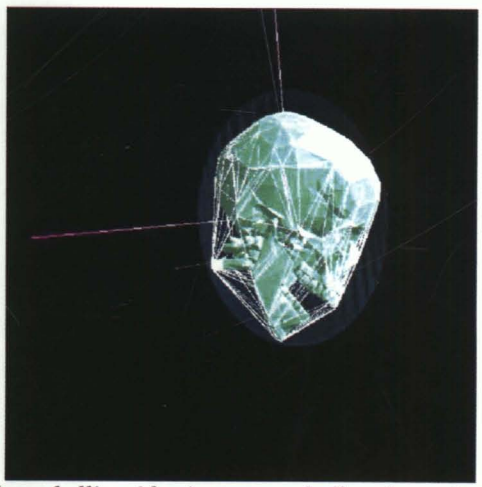

figure 22. A tight bound ellipsoid using convex hull and reassigning weighting to vertices.

Building convex hull has time complexity $O(n \log n)$. The time complexity for resampling the convex hull depends on the density of the vertices. However, by reassigning the weighting of the vertices according to the area, the time complexity is $O(n)$, which is the time complexity needed to find out the minimum area. Both the computation of the mean and the covariance matrix have complexity $O(n)$.

The effect of the tightness of the bounding ellipsoids is shown in figure 23.

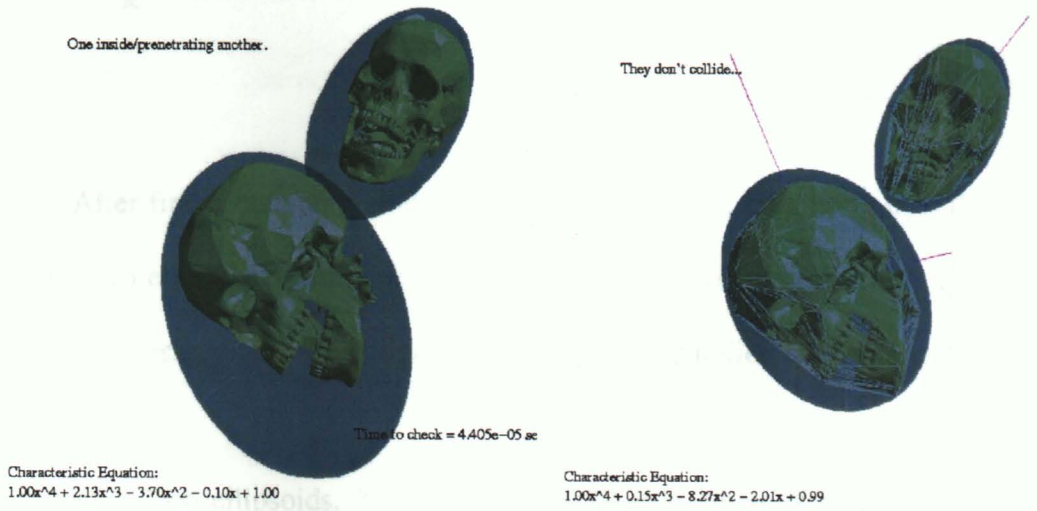

figure 23. ***A real example of two skulls that are not colliding. The tightness of the bounding ellipsoids can reject them from exact collision detection.***

Chapter 4 Collision detection using Ellipsoid bounding volume

After find a bounding ellipsoid for each object, the next thing is to determine whether two ellipsoids collide or not. The following theorem is given by [18] which states the necessary and sufficient conditions for the separation of two ellipsoids:

Theorem 1 Two ellipsoids, $X^TAX = 0$ and $X^TBX = 0$, where X is the homogeneous coordinate, do not intersect (and neither ellipsoid contains the other) if and only if their characteristic equation, $f(\lambda) = \det(\lambda A+B) = 0$, has two positive roots. Moreover,

(1) they are separate if and only if the two positive roots are distinct, and

(2) they externally touch each other if and only if the two positive roots are a double root

Therefore, there are two main steps in detecting collisions between two ellipsoids. We have to first construct the characteristic equation, and then find the roots of the equation. We introduce two methods in finding the characteristic equation:

- Expanding the determinant by cofactors and reusing duplicate calculations.

- Rearranging the transformation matrices before direct expansion.

We also introduce two methods in detecting the roots of the characteristic equation:

- By applying Sturm's sequence.
- By applying Descartes' theorem.

4.1 Finding the Characteristic Equation

Finding the characteristic equation is the most essential part of the whole algorithm as it is most time consuming. The performance of the collision detection by ellipsoidal bounding volume is greatly determined by the efficiency of the calculations involving the determination of the characteristic equation.

A characteristic equation is a quartic equation representing the relationship of two conic sections. The signs and magnitude of the roots indicate the relative geometric locations of the two ellipsoids. In real-time applications, the positions and the orientations of the bounding ellipsoids are updated frequently and cannot be pre-computed. The frequencies may be even higher than the output frame rates. Time performance is a critical concern for collision detections in real-time applications, while in simulation, accuracy is the main concern.

Time performance is not only related to the time complexity of the algorithm. If the constant of an algorithm with time complexity O(1) costs a few minutes in computation in each frame, it can hardly be applied to a real-time system.

4.1.1.1 Properties of the roots of the characteristic equation

Before going into the details of the characteristic equation, here are some properties of the roots of the characteristic equation of two ellipsoids:

(1) It has at least two negative roots;

(2) Any imaginary roots are in the form of conjunctive pair;

(3) The two ellipsoids touch each other if and only if there is a positive double root.

4.1.2 Expanding the determinant by the use of cofactors

Let us first examine the characteristic equation $f(\lambda) = \det(\lambda A + B) = 0$ of two ellipsoids A and B. The two ellipsoids A and B are represented by two 4×4 homogeneous matrices. The values in the 16 entries of the matrix are related to the algebraic equation of an ellipsoid. A general ellipsoid can be described as the solution to an equation in the form of $S(x,y,z) = Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyx + 2Fxz + Hx + Iy + Jz + K = 0$ or in a symmetric 4×4 matrix form, called the quadric form

$$S(x,y,z) = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} A & D & F & H \\ D & B & E & I \\ F & E & C & J \\ H & I & J & K \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (1)$$

Equation (1) is an ellipsoid with axis lengths $2A$, $2B$ and $2C$ along the principal directions. By applying an affine transformation M, the ellipsoid can be transformed into the standard form $S'(x,y,z) = A'x^2 + B'y^2 + C'z^2 + K = 0$, where the center of the ellipsoid is at the origin and the three principal directions are lying on the x, y, z-axis. In our implementation, ellipsoids are stored in this standard form together with the transformation M. The standard form of an ellipsoid is

$$S'(x,y,z) = \begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} A' & & & \\ & B' & & \\ & & C' & \\ & & & K' \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

4.1.2.1 Finding the standard form and the orthogonal transformation

As described in earlier sections, in constructing a bounding ellipsoid for an object, the 3 eigenvectors of the covariance matrix formed by the vertices of the object are chosen to be the 3 principal directions of the ellipsoid and the mean vertex is taken as the center of the ellipsoid. If we want to store an ellipsoid in its standard form, we also need to find out the transformation M. The transformation M is a rotation by R which rotates the three major axes of the bounding ellipsoid to that of the standard form, followed by a translation T which takes the center of the bounding ellipsoid to the center of the standard ellipsoid at the origin.

Let A be the bounding ellipsoid and A^S be its standard form. The 3 principal axes of A^S can be represented by a 3×3 identity matrix I. Since R will bring the three axes of A to that of A^S, we have $I = R \begin{pmatrix} ev_1 & ev_2 & ev_3 \end{pmatrix}$ or $R = \begin{pmatrix} ev_1 & ev_2 & ev_3 \end{pmatrix}^{-1}$, where ev_1, ev_2 and ev_3 are the three principle axes of A.

Then we need to find T that defines the translation from A to A^S. T is actually the negation of the coordinates of the center of A. Combining the two components R and T, we have the 4×4 orthogonal transformation,

$$M = \begin{pmatrix} R & T \\ & 1 \end{pmatrix}$$

Storing the ellipsoid in standard form has the advantage that 12 of the entries of the matrix in the quadric form are zero. This will also ease some calculations if the ellipsoid undergo further transformations.

Denote two standard ellipsoids by A^S and B^S, then the resulting characteristic equation will be $\det(\lambda(M_A^{-T} A^S M_A^{-1}) + (M_B^{-T} B^S M_B^{-1}))$, where M_A and M_B are the orthogonal transformation applied to A and B respectively. We have

$$M_A^{-1} = \begin{pmatrix} R^T & -T_R \\ & 1 \end{pmatrix} \tag{2}$$

where R is the 3×3 rotational transformation of the standard ellipsoid to the new orientation, and T_R is the translation column vector of the standard ellipsoid to the new position (center of the ellipsoid) multiplies by R. If the center of the current ellipsoid is $(C_x, C_y, C_z)^T$ we have

$$R = \begin{pmatrix} r_0 & r_3 & r_6 \\ r_1 & r_4 & r_7 \\ r_2 & r_5 & r_8 \end{pmatrix}$$

$$T_R = \begin{pmatrix} C_x r_0 + C_y r_1 + C_z r_2 \\ C_x r_3 + C_y r_4 + C_z r_5 \\ C_x r_6 + C_y r_7 + C_z r_8 \end{pmatrix}$$

Using this way in computing M_A^{-1} and M_B^{-1}, there are only 9 multiplications, 6 additions and some shuffling for each of the inverses. (2) is a faster way to find the inverse of an orthogonal transformation when compared with applying the classical inverse computation. Finding $(M_A^{-T} A^S M_A^{-1})$ and $(M_B^{-T} B^S M_B^{-1})$ can be performed using 80 multiplications and 48 additions.

4.1.2.2 Reusing terms in cofactors

Let the resulting matrix $(M_A^{-T} A^S M_A^{-1})$ and $(M_B^{-T} B^S M_B^{-1})$ be

$$\begin{pmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{pmatrix} \text{ and}$$

$$\begin{pmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{pmatrix} \text{ respectively.}$$

We now need to expand the characteristic equation $\det(\lambda A+B) = 0$ to the form $a\lambda^4+b\lambda^3 + c\lambda^2 + d\lambda + e = 0$ where a, b, c, d and e are all real integers.

In fact, the coefficient of λ^4, a, is equal to $\det(A)$ and the constant term, e, is equal to $\det(B)$. The method to find values of b, c and d are as follows.

We first write the matrix A in the form $(CA_1 \quad CA_2 \quad CA_3 \quad CA_4)$ where CA_i are the column vectors, and matrix B in the form $(CB_1 \quad CB_2 \quad CB_3 \quad CB_4)$. The coefficient of λ^3, b, is the sum of the determinants of four matrices. Each matrix is formed by a combination of three column vectors from A and one column vector from B, i.e.

$$b = \det(CB_1 \quad CA_2 \quad CA_3 \quad CA_4) + \det(CA_1 \quad CB_2 \quad CA_3 \quad CA_4) + \det(CA_1 \quad CA_2 \quad CB_3 \quad CA_4) + \det(CA_1 \quad CA_2 \quad CA_3 \quad CB_4).$$

The coefficient of λ^2, c, is the sum of the determinants of six matrices. Each matrix is formed by a combination of two column vectors from A and two column vectors from B, i.e.

$$c = \det(CB_1 \quad CB_2 \quad CA_3 \quad CA_4) + \det(CB_1 \quad CA_2 \quad CB_3 \quad CA_4) + \det(CB_1 \quad CA_2 \quad CA_3 \quad CB_4) + \det(CA_1 \quad CB_2 \quad CB_3 \quad CA_4) + \det(CA_1 \quad CB_2 \quad CA_3 \quad CB_4) + \det(CA_1 \quad CA_2 \quad CB_3 \quad CB_4).$$

Similarly, the coefficient of λ, d, is again the sum of the determinants of four matrices. Each matrix is formed by combining one column vector from A and three column vectors from B, i.e.

$$d = \det(CA_1 \quad CB_2 \quad CB_3 \quad CB_4) + \det(CB_1 \quad CA_2 \quad CB_3 \quad CB_4) + \det(CB_1 \quad CB_2 \quad CA_3 \quad CB_4) + \det(CB_1 \quad CB_2 \quad CB_3 \quad CA_4).$$

By this method, it takes altogether 16 matrix determinant calculations to set up the characteristic equation. However, because of the coherency of the patterns of the elements in these 16 matrixes, some of the intermediate results in the in-between multiplications and additions can be reused. For a 4×4 matrix, its determinant is the sum of the product of a row vector and their cofactors. For example,

$$det(A) = a_0 * \text{cofactor of } a_0 - a_4 * \text{cofactor of } a_4 + a_8 * \text{cofactor of } a_8 - a_{12} * \text{cofactor of } a_{12}$$

Each cofactor is a 3×3 matrix, which can be calculated by the product of its row vector and its 2×2 cofactors again. Most 2×2 cofactors appear twice in the 16 4×4 matrices and can be reused. For example, the cofactor $\begin{pmatrix} a_{10} & a_{14} \\ a_{11} & a_{15} \end{pmatrix}$ appears both in the calculations of $\det(CA_1 \quad CA_2 \quad CA_3 \quad CA_4)$ and $\det(CB_1 \quad CA_2 \quad CA_3 \quad CA_4)$.

Because of the repetition for these cofactors, we can reduce the number of mathematical operations performed by reusing the results of the cofactor calculations. For a 4×4 matrix, it requires 28 multiplications and 17 additions to compute its determinant by reusing repetitive cofactors. In other words, finding a, the coefficient of λ^4, needs 28 multiplications and 17 additions, finding the coefficient b of λ^3 needs 112 multiplications and 71 additions, finding the coefficient c of λ^2 needs 168 multiplications and 105 additions, finding the coefficient d of λ needs 112 multiplications and 71 additions and finding the constant term e need 28 multiplications and 71 additions.

Given A and B, by directly expanding the determinant of $\lambda A+B$ without reducing the redundancy, 344 multiplications and 195 additions are required. By breaking down the calculation into 16 matrices and reducing the redundancy by reusing the cofactors, only 202 multiplications and 136 additions are required and the coding is much simpler.

4.1.3 Simplify $f(\lambda) = \det(\lambda A+B)$ to $f(\lambda) = \det(\lambda I+A^{-1}B)$

Apart from reducing the calculations by reusing the cofactors, another approach is to first simplify $f(\lambda) = \det(\lambda A+B)$ into $f(\lambda) = \det(A)\det(\lambda I+A^{-1}B)$. As $\det(A) \neq 0$, the characteristic equation is simplified to $f(\lambda) = \det(\lambda I+A^{-1}B)$. A can be written as $M_A{}^T A_S M_A$, where A_S is the standard ellipsoid and M_A is the resulting orthogonal transformation matrix applying to A_S to produce A. By a little shuffling of the terms, the characteristic equation becomes $f(\lambda) = \det(\lambda I + A_s^{-1}(M_B M_A^{-1})^T B_S(M_B M_A^{-1}))=0$.

Advantages

The identity matrix has 12 zero entries. This reduces the complexity in calculating the determinant if we group the matrix multiplications into terms. By (2), M_A^{-1} can be computed by 9 multiplications and 6 additions from M_A. Reciprocating the diagonal entries of A_S form A_S^{-1}. The major time consuming computation is used in the five matrix multiplications. Calculate $\left(M_B M_A^{-1}\right)$ needs 64 multiplications and 48 additions while the transpose of a matrix only need entries shuffling. We first group the multiplications of the matrices into two terms $A_s^{-1}(M_B M_A^{-1})^T$ and B_s

$^1(M_B M_A^{-1})$, where both A_S and B_S are diagonal matrix. Now, these two terms can be computed by 80 multiplications and 48 additions. The product of these two resultant matrices takes another 64 multiplications and 48 additions. Altogether it takes 142 multiplications and 96 additions. By direct expansion, because of the elimination by the zero entries in the identity matrix, 62 multiplications and 46 additions are required to find $\det(\lambda I+N)$ where N is any 4×4 matrix. Thus, using this method, the total computation requirement is only 204 multiplications and 152 additions. It is 78 multiplications and 48 additions fewer than the method discussed before by reusing the cofactors.

4.2 Detecting Positive Roots of a Characteristic Equation

Now that we have calculated the characteristic equation for two ellipsoids, the next step is to detect the roots of the equation so that we can check whether the two ellipsoids collide or not by using the conditions established in Theorem 1. In this section, we will discuss two methods for detecting the roots of the characteristic equation, which is also a quartic equation.

4.2.1 Sturm Sequence

Sturm's Theorem

There exits a set of real polynomials $f_1(x), f_2(x), \ldots, f_m(x)$ whose degrees are in descending order, such that if $b > a$, the number of distinct real roots of $f(x) = 0$ in the interval $x = a$ to $x = b$ is given by the difference $V(a) - V(b)$, where $V(a)$ and $V(b)$ are the number of sign changes in the sequence $f_1, f_2, \ldots, f_m$ at $x = a$ and $x = b$ respectively.

There are many ways to construct the Sturm's sequence. We chose the way by assigning $f'(x)$ as $f_2(x)$ and the remainder of their division as $f_3(x)$. Let $f(x) = 0$ be an equation with real coefficients, and $f'(x)$ be the first derivative of $f(x)$. Divide $f(x)$ by $f'(x)$ we get q, the quotient, and $r(x)$, the remainder. i.e. $f = qf' + r$. Writing $f_2 = -r$, we have

$$f = qf' - f_2$$

Similarly, divide f' by f_2, and writing f_3 = the remainder with its sign alter. Continues to divide the quotient with the remainder until remainder is equal to a constant. i.e.

$$f' = q_2 f_2 - f_3, \quad f_2 = q_3 f_3 - f_4, \quad \ldots\ldots$$

This is the Sturm sequence we use, by substituting values x into these functions f and comparing the sign of the resultant value, the range of the roots can be found. The following calculation is an example.

Example:

$f(x) = x^3 + 6x^2 + 3x - 3$. Then $f' = 3x^2 + 12x + 3$,

$$f = \left(\frac{1}{3}x + \frac{2}{3}\right) f' - f_2,$$

$$f_2 = 10x - 6,$$

$$f' = \left(\frac{3}{10}x + \frac{138}{100}\right) f_2 - f_3,$$

$$f_3 = -\frac{162}{100}$$

For x = 0, the signs of f, f', f_2, f_3 are - + - -, showing two variations of consecutive signs. For x = 1, the signs are + + + -, showing one variation of signs. By Sturm's theorem, it means that there is a single real root between 0 and 1 because the difference between the number of changes of signs for f, f', f_2, f_3 is 1. The following table shows the change of signs when for x = -1, -2, -3 and –4.

X	Signs				Variations
-1	-	+	-	-	2
-2	+	+	-	-	1
-3	+	-	-	-	2
-4	-	+	-	-	2

By Sturm's theorem, we know that there is a single real root between –1 and –2, and one between –2 and -3. However, since there is no change of signs for x = -3 and x = -4, there is no real root in this interval.

By substituting zero and positive infinity to the Sturm sequence, the number of positive roots can then be found.

The Sturm sequence of a quartic equation can be pre-evaluated. The evaluation can be used by substituting the coefficients of the characteristic equation at run time. Since the characteristic equation of two ellipsoids must be a quadric equation, we can make use of this result. The following is the procedure to apply Sturm sequence to a quartic equation. Let the quartic equation be

$$f = ax^4 + bx^3 + cx^2 + dx + e,$$

then $$f' = 4ax^3 + \frac{3}{4}bx^2 + 2cx + d$$

$$f = \left(X + \frac{b}{4a}\right)f' - f_2 \qquad \text{Where } a \neq 0 \tag{3}$$

$$f_2 = \left(\frac{3b^2}{16a} - \frac{1}{2}c\right)x^2 + \left(\frac{bc}{8a} - \frac{3d}{4}\right)x + \left(\frac{db}{16a} - e\right)$$

$$f' = \left[\frac{a}{p}x + \left(\frac{3}{4}b - \frac{aq}{p}\right)\frac{1}{p}\right]f_2 - f_3 \qquad \text{Where } p \neq 0 \tag{4}$$

Where $p = \frac{3b^2}{16a} - \frac{1}{2}c$, $q = \frac{bc}{8a} - \frac{3d}{4}$, $r = \frac{db}{16a} - e$

$$f_3 = \left[\left(\frac{3}{4}b - \frac{aq}{p}\right)\frac{q}{p} - \left(\frac{1}{2}c - \frac{ar}{p}\right)\right]x + \left[\left(\frac{3}{4}b - \frac{aq}{p}\right)\frac{r}{p} - \frac{d}{4}\right] \tag{5}$$

Then $$f_2 = \left[\frac{p}{s}x + \left(q - \frac{pt}{s}\right)\frac{1}{s}\right]f_3 - \left[\left(q - \frac{pt}{s}\right)\frac{t}{s} - r\right]$$

Where $s = \left(\frac{3}{4}b - \frac{aq}{p}\right)\frac{q}{p} - \left(\frac{1}{2}c - \frac{ar}{p}\right)$, $t = \left(\frac{3}{4}b - \frac{aq}{p}\right)\frac{r}{p} - \frac{d}{4}$

$$f_4 = \left(q - \frac{pt}{s}\right)\frac{t}{s} - r = u \qquad \text{and } s \neq 0 \tag{6}$$

There is no need to substitute x as positive infinity and zero to this Sturm sequence to evaluate the change of signs. Since only the sign of the sequence f, f', f_2,

f_3 and f_4 is important, we only need to know their dominating factors. The dominating factors of these equations when x is equal to positive infinity are a, p, s and u. The dominant factors when x is zero are e, d, r, t and u. The number of operations needed for evaluation these dominating factors are as follows:

Value	Multiplication	Division	Addition
a	0	0	0
e	0	0	0
d	0	0	0
P	2	1	1
R	2	1	1
S	5	3	3
T	4	2	2
U	2	2	2
Total	16	9	9

By 16 multiplications, 9 divisions and 9 additions, we can find the number of positive roots of the characteristic equation.

Theoretically, the Sturm sequence works well in detecting the signs of the roots of an equation. However, the conditions a, p and s must be non-zero. Otherwise, the divisions in equations (3), (4) and (6) will result in overflow error. To examine these situations, we need to go further and understand the situation for the Sturm sequence of the characteristic equation of two ellipsoids.

First of all, by directly finding $\det(\lambda\alpha+\beta)$, where α and β are two quadrics, a is equal to the determinant of α. The determinant of an ellipsoid can be zero only if it is a degenerate ellipsoid, such as a plane or a line. The determinant of a non-degenerating ellipsoid will not be zero. It is because any ellipsoid can be transformed from a unit sphere. As the determinant of a unit sphere is 1, unless these transformations are irreversible, they should have their inverse. The only

transformations that are irreversible are those that will degenerate the unit sphere. So a non-degenerate ellipsoid has a non-zero determinant and thus a is non-zero.

We next examine the case when p is equal to zero. If $p = 0$, only the term x and the constant term will be left in f_2.

$$f_{2(p=0)} = qx + r \qquad \text{where } q = \frac{bc}{8a} - \frac{3d}{4} \text{ and } r = \frac{db}{16a} - e$$

Here, if q is equal to zero, we have $f_2 = r$, which is a constant, and the Sturm sequence ends here. However, if q is not equal to zero, we have

$$f'_{(p=0)} = \left[\frac{a}{q}x^2 + \left(\frac{3}{4}b - \frac{ar}{q}\right)\frac{x}{q} + \left(\frac{c}{2} - \left(\frac{3b}{4} - \frac{ar}{q}\right)\frac{r}{q^2}\right)\right] f_{2(p=0)} - f_{3(p=0)}$$

Where $q \neq 0$

$$\text{and } f_{3(p=0)} = \frac{d}{4} - \left(\frac{c}{2} - \left(\frac{3b}{4} - \frac{ar}{q}\right)\frac{r^2}{q^2}\right) \tag{7}$$

As the result of equation (7) is a constant, the sequence ends here. If p is not equal to zero in equation (4), we need to check if s is equal to zero in equation (5). If s is equal to zero, we have

$$f_{3(s=0)} = \left[\left(\frac{3}{4}b - \frac{aq}{p}\right)\frac{r}{p} - \frac{d}{4}\right] \tag{8}$$

As the result of equation (8) is a constant, the sequence ends here. All the cases result in fewer computations than when p and s are not equal to zero.

If we find $\det(\lambda I + A^{-1}B)$ instead of $\det(\lambda A + B)$ as discussed in the previous section, the analysis is very similar. Now a is equal to $\det(I)$ that is 1. The same will hold for p and s as the arguments discussed above.

The following tables show the operations required in the two special cases:

If $p = 0$,

Value	Multiplication	Division	Addition
a	0	0	0
e	0	0	0
d	0	0	0
p	2	1	1
r	2	1	1
$f_{3(p=0)}$	4	2	2
Total	8	4	4

If $s = 0$,

Value	Multiplication	Division	Addition
a	0	0	0
e	0	0	0
d	0	0	0
p	2	1	1
r	2	1	1
s	5	3	3
t	4	2	2
Total	13	7	7

In conclusion, using Sturm sequence to detect the roots of the characteristic equation only require 16 multiplications, 9 division and 9 additions.

4.2.2 Descartes' Rule of Signs

Besides using the Sturm sequence that is able to find out the range of the solution of a quartic equation, there is another method called *Descartes' rule of signs* that determines the number of positive roots of a quartic equation. By its definition, the Descartes' rule of signs can only determine the upper bound of the number of positive roots. However, because of the first property of the characteristic equation of two ellipsoids (section 4.1.1.1), we can detect the exact number of roots by applying the Descartes' rule of signs. First of all, let us take a look at the definition of the Descartes' Rule of Signs.

Descartes' rule of signs

*The number of positive roots of an equation with real coefficients either equals the number of V^+ variations of signs in the series of coefficients or is less than V^+ by an even integer. A root of multiplicity **m** is counted as **m** roots.*

Corollary

The number of negative roots of $f(x) = 0$ is either the number of V^- variations of signs in the coefficients of $f(-x)$ or is less than that number by an even integer.

As there are at least 2 negative roots for a characteristic equation, there are only 4 cases for the other 2 roots:

- Both roots are positive
- Both are negative
- One positive and one negative
- Both are imaginary roots

Note that the two imaginary roots would come in as conjugate pair for an equation with real coefficients.

We will first prove that the remaining two roots cannot be the third case. By simple algebraic rule, we know that the product of roots of a polynomial $ax^4+bx^3+cx^2+dx+e = 0$ is equal to $\frac{e}{a}\times(-1)^4$. a and e as stated in an earlier section, is equal the det(A) and det(B). The proof below shows that the determinant of an ellipsoid must be negative. As a result the product of the roots of the characteristic equation must be positive that implies the third case is not possible.

Proof

For any ellipsoid xAx^{-1}, A can be written as MEM^{-1}, where M is the transformation applying on the standard ellipsoid E. Then

$$\begin{aligned}\det(MEM^{-1}) &= \det(M)\det(E)\det(M^{-1}) \\ &= \det(M)\det(M^{-1})\, det(E) \\ &= \det(I)\det(E) \\ &= \det(E)\end{aligned}$$

For a standard ellipsoid $ax^2 + by^2 + cz^2 = d$, its matrix representation is

$$\begin{pmatrix} a & & & \\ & b & & \\ & & c & \\ & & & -d \end{pmatrix}$$

where a, b, c and d are positive real integers for an ellipsoid with real coefficients. Its determinate is always negative. Then product of roots of the characteristic equation must be positive. By this, the 2 remanding roots cannot be one positive and one negative.

We have shown that the roots of the characteristic equation can only be one of the 3 combinations. By closer examination on the properties of V^+ and V^-, we notice some relationships between them. The relationships between V^+ and the roots of the characteristic equation are listed below.

Roots	V^+	Discriminant of characteristic equation
Both positive	2	> 0
Both negative	0	> 0
One positive and one negative	NA	NA
2 imaginary number	NA	< 0

Proof

First of all, it can be shown that the sum of V^+ and V^- is always equal to the highest order of the polynomial.

Let the highest order for an equation is i. For $i = 1$, that is,

$$f(x) = ax + b$$

If both a and b have the same sign, $V_1^+ = 0$ and $V_1^- = 1$. Otherwise, $V_1^+ = 1$ and $V_1^- = 0$.

Assume when $i = k$, the sum of V_k^+ and V_k^- is equal to k.

For $i = k + 1$, we cut the last k^{th} order term and left with a k order polynomial. We then have,

$f(x) = ax^{k+1} + k^{th}$ order polynomial

where V_k^+ and V_k^- is equal to k for the k^{th} order polynomial. Assume a has the same sign as the coefficient of the k^{th} coefficient. Then $V_{k+1}^+ = V_k^+$ and $V_k^- = V_{k+1}^- + 1$. On the other hand, if a has a different sign with the coefficient of the k^{th} coefficient, $V_{k+1}^+ = V_k^+ + 1$ and $V_k^- = V_{k+1}^-$.

In conclusion, the sum of V^+ and V^- is always equal to the highest order of a polynomial, and is equal to 4 for the quartic characteristic equation. Hence, when V^- equals to 4, V^+ must be equal to 0. When there are two negative roots and two positive roots, both V^- and V^+ must at least equal to or greater than two, since their sum is equal to four, and V^+ and V^- must be two. However, if there are imaginary roots, V^+ and V^- can equal to any number. To avoid this, the discriminant of the characteristic equation is checked before evaluating V^+.

For any polynomial f, $f = 0$ has imaginary roots if and only if its discriminant is less than zero. Its discriminant is also equal to zero if and only if there are double roots. Before applying the Descartes' Rule of Sign to detect the roots, we first calculate its discriminant. It is because if the discriminant is greater than zero, V^+ equal to two if and only if there are two positive roots.

The resolvent cubic of a quartic equation $x^4 + bx^3 + cx^2 + dx + e = 0$ is given by $y^3 - cy^2 + (bd - 4e)y - b^2e + 4ce - d^2 = 0$. The discriminant of this resolvent cubic is $-4P^3 - 27Q^2$, where $P = bd - 4e - \frac{1}{3}c^2$ and $Q = -b^2 + e + \frac{1}{3}bcd + \frac{8}{3}c - d^2 - \frac{2}{27}c^3$. The discriminants of a quartic equation and its resolvent cubic are the same. By this, we can get the discriminant of the characteristic equation by 19 multiplications and 10 additions. This can ensure the characteristic equation does not have two imaginary roots before applying the Descartes' Rule of Sign.

Advantages

Given the characteristic equation, using Descartes' Rule of Sign to detect the existence of two positive roots only requires calculations in the step of finding out the discriminant. It saves the divisions required in applying the Sturm sequence.

Disadvantages

Precise result in calculating the discriminant is required. Moreover, the discriminant of the equation will be smaller than zero if the equation has imaginary roots. There are methods to calculate the discriminant of a quartic equation. However,

they are very sensitive to floating point precision. Consider the definition of the discriminant, where it is equal to the product of the square of the differences between roots. For a quartic equation with four roots, x_1, x_2, x_3 and x_4, the discriminant is

$$(x_1-x_2)^2(x_1-x_3)^2(x_1-x_4)^2(x_2-x_3)^2(x_2-x_4)^2(x_3-x_4)^2$$

From the above definition, if there are double roots, the discriminant should be equal to zero. However, due to floating point precision, the resulting error can be very huge. For example, if the difference of the double roots is 10^{-12} instead of zero and the differences of the other roots are within a range of 100. Then the resulting discriminant will be $10^{-12} * 10^{20} = 10^8$ instead of zero.

Chapter 5 Result

This section is the studies on the run time performance of using ellipsoidal bounding volumes. The studies are mainly focus on the case of the collision detection of ellipsoid shape models. The reason for this is that well-known collision detection method such as AABB and OBB are more suitable for bounding box shape polyhedra. Comparisons have been made on the collision detection between spheres, between approximated polyhedrons and between ellipsoids. The reason for chosing spheres for comparsion is the promising high performance in detecting collision detection between spheres. However there may have accuracy penalty in using sphere to bound ellipsoidal sphere object. On the other hand, polyhedrons can approximate some ellipsoidal shapes much better. By comparison with these two kinds of shapes, a spectrum of speed versus accuracy can be constructed.

5.1 Collision detection between ellipsoids

5.1.1 Generating ellipsoids

In generating an ellipsoid, we need to generate the lengths of the three major axes, its +position and orientation. So there are totally 9 variables need to be randomly generated. The lengths of the three major axes and the x, y, z coordinate of the center are 6 floating-point numbers. The 3×3 rotational matrix only needs 3 floating-point numbers for the 3 Euler Angles along the three principal axes. Assume the Euler Angles along x, y and z-axis are α, β and γ in radian respectively, then the 3×3 rotational matrix will be

$$\begin{pmatrix} \cos\beta\cos\gamma & \cos\beta\sin\gamma & -\sin\beta \\ \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\cos\beta \\ \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{pmatrix}$$

Two sets of ten thousand ellipsoids are generated. One set of them contains five thousand pairs of ellipsoids that collide. The other set contains five thousand pairs of ellipsoids that do not collide. Throughout this chapter, we will refer to these two sets of ellipsoids as the generic ellipsoids sets.

figure 24. Ten thousands of ellipsoids generated in 3D space

By combining one of the two methods in expanding the discriminant and one of the two methods in detecting the signs of roots, we form four different tests. These four tests are then applied on the generic ellipsoids sets.

5.1.2 Result

The following table states the results of the eight tests. The program is run on an SGI Onyx workstation with 195MHz R10000 CPU, 512MB RAM.

Time	Without collision				With collision			
	By cofactors		By rearrangement		By cofactors		By rearrangement	
	Sturm	Descartes'	Sturm	Descartes'	Sturm	Descartes'	Sturm	Descartes'
Total /sec	0.116926	0.0989934	0.110841	0.0933326	0.111946	0.104921	0.100297	0.082785
Max /μs	141.962	71.9772	83.0154	455.948	490.986	80.0331	79.9838	87.024
Min /μs	15.9634	12.9677	15.9408	11.9509	15.9415	12.982	14.943	6.96473
Average /μs	23.3385	19.7987	22.1682	18.6665	22.3892	20.9842	20.0594	16.557

Table 1. Time performance of the 8 tests in ellipsoid-based system

The result shows that for 5000 collision detections, it takes about 0.1 second. As predicted, the fastest way is by rearranging the terms before finding the discrimiant and using the Descartes' rule of sign to check for the positive roots. It is about 30 percent faster than using direct expansion of the discriminant and using Sturm sequence. The time taken for detecting separated ellipsoids and collided ellipsoids are nearly the same. This is the advantage of using algebraic method.

5.2 Collision detection between polyhedrons

5.2.1 Generating polyhedrons

The approximating polyhedrons we generated are rather regular. Vertices of the polyhedron are regularly distributed on the surface of the corresponding ellipsoid. The number of vertices used is control by the surface parameters u and v, the dimensions a_1, a_2, a_3, and the roundness shape parameters n, e.

The coordination of a vertex is given by x, y and z, where

$$x(u,v) = a_1 c(v,n)c(u,e),$$
$$y(u,v) = a_2 c(v,n)s(u,e),$$
$$z(u,v) = a_3 s(v,n),$$
$$\frac{-\pi}{2} \le v \le \frac{\pi}{2},$$
$$-\pi \le u < \pi$$

This is also known as the parametric surface representation of an ellipsoid.

figure 25. Three polyhedron with different complexities approximating the same bounding ellipsoid. Using 6 vertices, 50 vertices and 170 vertices from left to right

Polyhedrons are used to approximate the two generic ellipsoids sets. We want to compare the performance in collision detection between polyhedron-based system and ellipsoid-based system.

5.2.2 Result

The following chart shows the differences between ellipsoid and polyhedron-based system. In the first chart, the average time used for testing a pair of polyhedrons is plotted against the number of vertices of one polyhedron. The result shows that the detection time for polyhedron-based system is much longer. Collision detection of 6-vertices approximating polyhedrons takes 10 times longer than ellipsoidal-based system. In the second chart, the average numbers of iterations are plotted. The time and iterations used in detecting colliding polyhedrons are two times more than that of detecting non-

colliding polyhedrons. Furthermore the time and iterations used are proportional to the number of vertex of an approximating polyhedron.

These results imply that time performance of the collision detection in a polyhedron-based system is depended on the positions and the complexities of the objects.

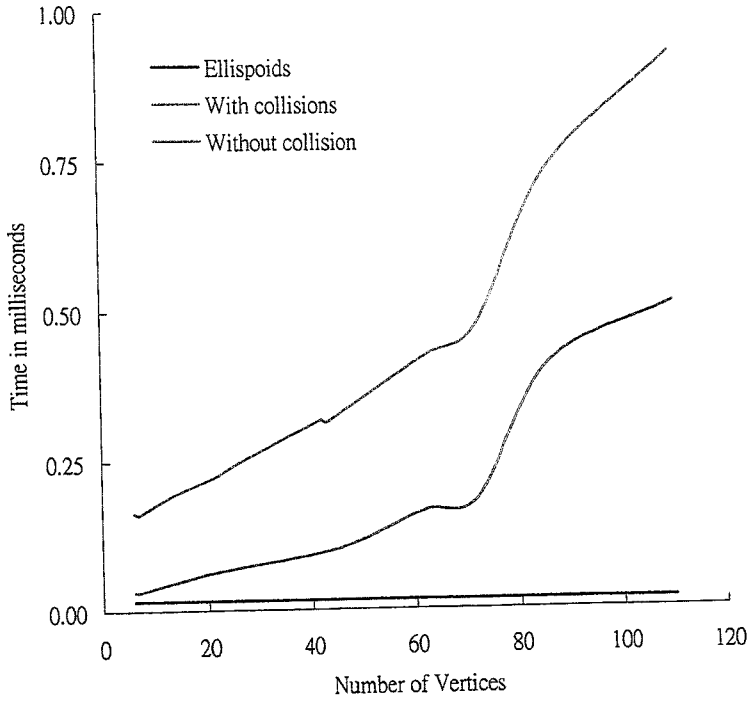

Chart 1 Difference in time used in detecting collide and non-collide polyhedrons

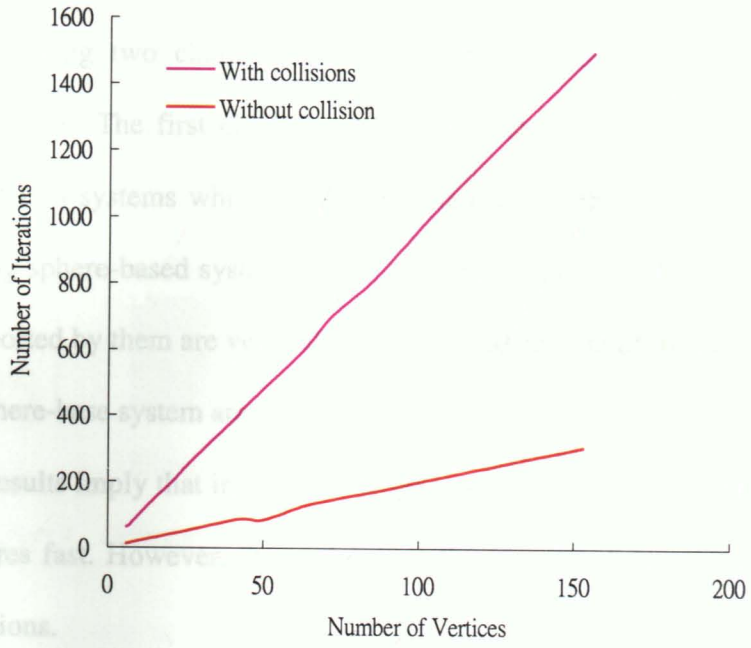

Chart 2 Difference in iteration used in detecting collide and non-collide polyhedrons

5.3 Collision Detection between Spheres

5.3.1 Generating sphere

In this comparison, we generate two different kinds of spheres to approximate the two sets of generic ellipsoids. They are spheres that contain the ellipsoids and spheres that can be contained in the ellipsoids. They are also called the enclosing spheres and the core spheres.

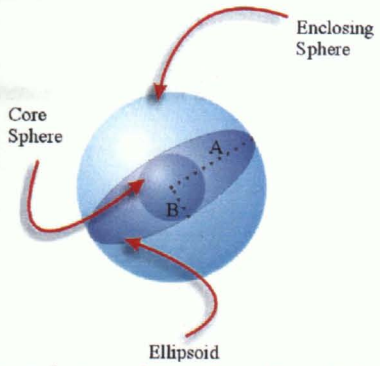

figure 26. Enclosing sphere and core sphere of an ellipsoid. A and B are the longest and shortest major axis of the ellipsoid.

Chapter 6 Conclusions

6.1 Conclusions

Collision detection procedure can be divided into two phases. They are the board phase and the narrow phase. During the board phase, all pairs of intersecting bounding volumes need to be found. During the narrow phase, exact collision detection is performed for each intersecting pair found in the broad phase.

For the board phase, spatial decomposition is one of the techniques that can minimize the number of collision testing on bounding volumes. However, the splitting involved destroys the general geometric properties of an object, and increases the number sub-objects. Using ellipsoids to enclose human parts, the oval shapes of them can be preserved. Other bounding volumes such as bounding spheres, axis-aligned bounding boxes and object-oriented bounding boxes may not serve this purpose.

Two ellipsoids are separated if and only if their characteristic equation has two positive roots. Studies show that in collision detection between ellipsoids, the most expensive computation is the matrix multiplications required in setting up the characteristic equation. Detecting the signs of the roots is much cheaper.

From experimental results, collision detection in ellipsoid-based system is faster than in polyhedron-based system and only a bit slower than in sphere-based system.

6.2 Future Work and Discussion

As mentioned in section 5.1.2, the fastest way is by rearranging the terms before finding the discrimiant and using the Descartes' rule of sign to check for the positive roots. However, the way we use to determine the sign of the discriminant may produce precision error when the discriminant is very close or equal to zero. In the case of computing the collision detection of moving objects, their bounding volume may have an instance that they just hit each other where the discriminant of the characteristic equation just equal to zero. Further studies should be made in solving this precision problem.

Chapter 7 Reference

[1] Goldak, J. A., Yu, X., Knight, A., and Dong, L. 1995. Constructing discrete medial axis of 3-D objects. *Int. J. Comput. Geometry Appl. 1*, 3, 327-339.

[2] B. Barber, D. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hull. Technical Report GCG53, The Geometry Center, MN, 1993.

[3] A. Bowyer and J. Davenport; How to construct the skeleton of CSG objects. In *The Mathematics of Surfaces IV.* Eds., Oxford University Press, Oxford. Available as Tech. Rep. CSD-TR-1014, Computer Sciences Department, Purdue University.

[4] S.Cameron. Obstacle avoidance and path planning. *Industrial Robot,*21(5):9-14,1994.

[5] T.L. Chung and W.Wang, Quick Collision Detection of Polytopes in Virtual Environments. *Proceedings of the 3rd ACM Symposium on Virtual Reality Software and Technology, 125-132, 1996.*

[6] A. M. Day; The implementation of an algorithm to find the convex hull of a set of three-dimensional points; *ACM Trans. Graph.* 9, 1 (Jan. 1990), Pages 105 - 132

[7] Wilf, I., and Manor, Y., Quadric-surface Intersection Curves: Shape and Structure, Computer-Aided Design, Vol. 25, No. 10, pp. 633-643, 1993.

[8] A. S. Glassner; An efficient bounding sphere. In *Graphics Gems.* Ed. Academic Press, Boston, MA, 301-303

[9] Turk, G. 1991; Generating textures on arbitrary surfaces using reaction-diffusion. In *Proceedings of SIGGRAPH '91*, published as *Comput. Graph. 25,* 4 (Aug.), 289-298

Reference

[10] S. Gottschalk. Separating axis theorem. Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill, 1996.

[11] S. Gottschalk, M.C.Lin, D.Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection, In *Proceedings of the 23rd annual conference on Computer graphics*, 1996, Pages 171 - 180

[12] Blum, H. 1967 W. Wathen-Dunn; A transformation for extracting new descriptors of shape. In *Models for the Perception of Speech and Visual Form* Ed., MIT Press, Cambridge, MA, 362-380.

[13] Philip M. Hubbard; Approximating polyhedron with spheres for time-critical collision detection; ACM *Trans. Graph.* 15,3 (Jul. 1996), Pages 179-210

[14] Y.K. Hwang and N. Ahuja. Gross motion planning-a survey. *ACM Computing Surveys,* 24(3):219-291, Sept. 1992.

[15] M. Moore and J. Wilhelms, Collision Detection and Response for Computer Animation, In *Comput., Graphics (SIGGRAPH88 Proc.),* Vol. 22, pp. 289-298, Aug 1988.

[16] B. Naylor, J. A. Amatodes, and W. Thibault, Merging BSP Tree Yields Polyhedron Set Operations, In *Comput. Graphics (SIGGRAPH '90 Proc.)*, Vol. 24, pp.115-124, Aug 1990.

[17] Preparata, F.P. and Shamos, M. I. 1985; *Computational Geometry: An Introduction* Springer-Verlag, New York.

[18] Wenping Wang, Jieye Wang, Myung-Soo Kim. Algebraic Condition for the Separation of Two Ellipsoids. *Manuscript.*

[19] Yunhong Zhou and Subhash Suri; Analysis of a bounding box heuristic for object intersection; *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, 1999, Pages 830 – 839

[20] Levin, J Z, A parametric algorithm for drawing pictures of solid object composed of quadric surfaces; *Commun. ACM* Vol 19 (1976) pp. 555-563.

Chapter 8 Appendix

8.1 Algebraic Condition for the Separation of Two Quartic

The analysis is to base on the Segre characteristics [7]; it classifies the intersection curve of two arbitrary quadrics. Decomposing the intersection curve by factorization of the parameterization polynomials forms the Segre characteristics. The analysis also can check to see if the resultant intersection curve can be further reduced to components and can be parameterized or not.

Quadric surfaces are represented in the matrix form $X^TMX = 0$, where $X = (x, y, z, w)^T$ in homogeneous coordinate and M is a 4×4 symmetric discriminant matrix. The analysis of the intersection of two quadrics M_1 and M_2 are based on solving $C = M_1-\lambda M_2$. In order to find λ satisfying $\det(M_1-\lambda M_2)$, i.e. when the set of linear equation is singular. We called the equation $\det(M_1-\lambda M_2) = 0$, the characteristic equation, which is a quartic equation with four roots.

The Segre characteristic is a string related to the exponents and bases of the roots of the characteristic equation. By Levin's method [20] the Segre characteristic can be found. And in the reduced form the quadric-surface intersection curve can be analyzed.

By the number of roots together with the sign of the roots and their values, the intersection can be classified. Whether it is planar or non-planar, cone, elliptic, parabolic, hyperbolic, etc, can all be determined.

8.2 Matrix and determinant

An identity matrix, denoted by I, is a square matrix that the diagonal entries all equal to one, at the same time all the other entries equal to zero. Any other square matrix, when multiplying with an identity matrix remains unchanged, i.e. $MI = IM = M$. An example of a 4×4 matrix is

$$I=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}.$$

The transpose of a matrix, denoted by M^T is a matrix with its column and row interchange. If a matrix has its transpose equal to itself, then it is called a symmetric matrix. Furthermore, the transpose of a matrix product is equal to the product of the transpose of the matrix before multiplication and multiply in a reverse order, i.e. $(MR)^T=R^TM^T$.

The inverse of a $n\times n$ matrix M is denoted by M^{-1} and M^{-1} is also a $n\times n$ matrix. The product of a matrix and its inverse is an identity matrix. Not all the square matrix precedes an inverse of itself. However, if it has one, it is unique. Nonsingular matrix is matrix that has inverse; singular matrix is matrix that does not have inverse. The inverse of a product is equal to the product of the inverse of the matrix before multiplication but multiply in a reverse order, i.e. $(MR)^{-1}=R^{-1}M^{-1}$

A determinant of a matrix is a scalar. The determinant of an identity matrix is always 1. So by $\det(MM^{-1}) = \det(I) = \det(M)\det(M^{-1})$.

The determinant of the inverse of a matrix is equal to the reciprocal of the determinant of the original matrix. Furthermore, a matrix is nonsingular and has an inverse if and only if its determinant is non-zero.

8.3 Finding inverse

There are many ways to calculate the inverse of a matrix; one of the ways is the Gauss-Jordan elimination. The idea of the Gauss-Jordan elimination is to apply elimination to the matrix, and at the same time the same elimination is applied on an identity matrix. When the elimination is applied on the original matrix and reduced it to an identity matrix, the same elimination will transformed the identity matrix to the inverse of the matrix.

For a linear equation $Mx = y$, if a square matrix is multiplied to it then we have

$GjMx = Gjy$. Where Gj is the Gauss-Jordan elimination

If Gj is to equal to the inverse of M, then $x = M^{-1}y$. Therefore, if the Gauss-Jordan elimination, which reduced M on the left into an identity matrix, is applied to the identity matrix, the identity matrix on the right can be transformed to M^{-1}.

8.4 Finding determinant

The determinant of a matrix can be found by using the cofactors and minors. Consider a $n \times n$ matrix,

Appendix

$$M = \begin{pmatrix} m_{11} & \cdot & \cdot & m_{1n} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ m_{n1} & \cdot & \cdot & m_{nn} \end{pmatrix}$$

The minor of an entry is the determinant of the matrix with the column and row of that entry being omitted. For the above M the minor of m_{11} is

$$\underset{m_{11}}{\text{minor}} = \det \begin{pmatrix} m_{22} & \cdot & \cdot & m_{2n} \\ \cdot & & & \cdot \\ \cdot & & & \cdot \\ m_{n2} & \cdot & \cdot & m_{nn} \end{pmatrix}$$

The cofactor of an entry is the product of it minor and the term $(-1)^{r+c}$ where r and c are the row and column of the entry.

The determinant of M is finally given by $\sum_{c=1}^{n}(-1)^{r+c}m_{rc}M_{rc}$ or $\sum_{r=1}^{n}(-1)^{r+c}m_{rc}M_{rc}$ depends on whether choosing the entries in the same row or the same column to expand their cofactors.

8.5 Homogeneous coordinates and matrix representation

In computer graphics, we usually use geometric transformation to change the position and orientation of objects. Points in the 3D space can be move to new positions by adding translation to them. Let a point p(x,y,z) moves to a new position p'(x',y',z'), then the point can be considered as moving in three direction by d, where

$$d_x = x' - x \qquad d_y = y' - y \quad \text{and} \quad d_z = z' - z$$

Using column vectors to represent the translation, we will have

$$p = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \qquad p' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}, \qquad ts = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix}, \qquad \text{having } p' = p + ts$$

Appendix

We say $p' = p + ts$ as the translation that can be applied to the points of an object in order to move the whole object by the x, y, z component of d in the three x, y, z-axis directions.

Similarly, multiplying a rotation matrix to 3D points can rotate them about the origin. Let a point $p(x,y,z)$ rotate through an angle θ counterclockwise from x toward y about the origin along the z axis. Then the point will rotate to a new position $p'(x',y',z')$, where

$$x' = x\cos\theta - y\sin\theta \quad y' = x\sin\theta + y\cos\theta \quad z' = z$$

Using matrix form to represent the rotation, we will have

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \text{ simply we have } p' = r \cdot p$$

Similarly, we have

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \text{ when the points are rotate along } x \text{ axis and}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \text{ when the points are rotate along } y \text{ axis.}$$

We say $p' = r \cdot p$ as the rotation that can be applied to the points of an object in order to rotate the whole object. It can be easily shown that both translation and rotation are reversible.

Appendix

The matrix representations for translation and rotation are thus represented by $p' = p + ts$ and $p' = r \cdot p$. In order to have both transformations can be calculated by multiplication of matrix instead of addition, the homogeneous coordinates are introduced.

The homogeneous coordinates are simply adding a fourth coordinate to a point in 3D space. Points are considered to be the same point if are the coordinate are different by the same ratio. For example, (1,4,7,1) and (2,8,14,2) are representing the same point. The fourth coordinate can be any number besides zero. Points with the fourth coordinate equal to zero are representing point at the infinity. When the fourth coordinate equal to 1, the other 3 coordinates have the same value as if they are representing in the Cartesian coordinates. We can always have the fourth coordinate has the value 1 by dividing all coordinates by the fourth coordinate. In this way, we are homogenizing the point.

In Cartesian coordinates, transformation in 3D space is represented by a 3×3 matrix. After using homogeneous coordinates, transformation is represented by a 4×4 matrix. For example translation now become

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & & & d_x \\ & 1 & & d_y \\ & & 1 & d_z \\ & & & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Expanding the multiplication we will have $(x',y',z',1)^T = (x+d_x,\ y+d_y,\ z+d_z,\ 1)^T$. So, instead of addition of column vectors in Cartesian coordinates, translation can be represented by matrix multiplication in homogenous coordinates. Rotation can still be represented by matrix multiplication in homogenous coordinates as if in Cartesian coordinates. However, it becomes the multiplication of a 4×4 matrix and a 1×4 column vector. The three rotations along the 3 axes are shown below.

Appendix

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & \cos\theta & -\sin\theta & \\ & \sin\theta & \cos\theta & \\ & & & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$ when the points are rotate along x axis,

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & & \sin\theta & \\ & 1 & & \\ -\sin\theta & & \cos\theta & \\ & & & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$ when the points are rotate along y axis and

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & & \\ \sin\theta & \cos\theta & & \\ & & 1 & \\ & & & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$ when the points are rotate along z axis.

The sign of θ is defined as if it is in Cartesian coordinates. The counterclockwise rotation along an axis by looking from a positive axis location toward the origin is considered to be positive. Or the right-handed rule can be applied here. The thumb is in the direction along the rotational axis, and the direction of the other fingers contract is the direction of the positive angle.

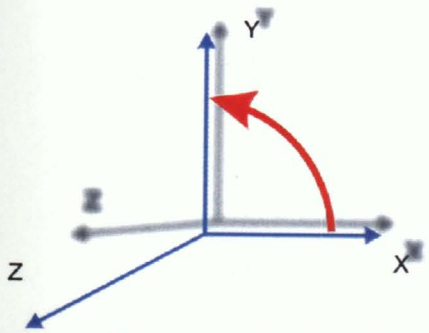

figure 27. The z-axis is point out of the paper, and defining the positive angle of rotation along z-axis.

By homogenous coordinates, sequence of rotations and translations can be multiplied together and form a resultant matrix to represent the final transformation. This becomes very convenience to represent any location and orientation of an object in the

3D space. Also, these primitive transformations have their inverse, and the inverse for translation is obtained by negating the d, while the inverse for rotation is by putting the opposite direction angle to the 9 entries of the matrix. The actual meaning of the inverse of the primitive transformation is the transformation that reverses back the object before applying it.