Technical Report DIKU-TR-02/04
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 KBH Ø
DENMARK

July 2002

An Introduction to Appoximating Heterogeneous Bounding Volume Hierarchies

Kenny Erleben

Abstract: Approximating heterogeneous bounding volume hierarchies are a data structure, which we believe could be used for efficient and robust collision detection in the context of simulation, animation, virtual reality and robotics. This paper will introduce the concepts and ideas behind this data structure.

1 Introduction

Collision detection is of major importance in simulation, animation, virtual reality and robotics. It is used as a tool, for avoiding interpenetrations.

Simulation, animation, virtual reality and robotics all share some common trademarks. They are all trying to figure out how a configuration of both moving and non-moving objects behave (or should behave) in the near future. In order to do so they are all heavily dependent on collision detection to detect penetration, areas of touching contact and distance computations.

Since we are working with objects that are moving in time, collision detection is no longer just a single query, but a repeated sequence of multiple queries.

Applications in simulation, animation, virtual reality and robotics are often interactive, which means they often are extremely dependent on achieving real time performance. The size and complexity of a configuration and its objects can easily increase dramatically, making collision detection a bottleneck in real time applications.

The fastest methods used today require objects to be based on hierarchies of convex polyhedra. Eventhough most configurations today have adjusted themself to this concept it does not hide the fact that this inhibits the modelling freedom.

The methods that exist today, which are applicable to general shaped objects, are slow compared to the methods based on hierarchies of convex polyhedra. Making it difficult to achieve real-time performance on even small sized configurations.

- Multiple Repeated Queries
- Real-Time Performance
- General Shaped Objects

It would be nice to have competitive real-time methods applicable to general shaped objects. Our main contribution in this paper is the introduction of the concepts and ideas of a data structure, which we call approximating heterogeneous bounding volume hierarchies. We believe this data structure could be a competitive alternative, which solves all these problems.

2 Overview of Collision Detection Algorithms

For the past 20 years or so collision detection has evolved itself tremendeously, from the early simple problems of deciding if two simple geometries intersects or not until todays wide spectrum of methods and algorithms.

The algorithms naturally divides into three groups of algorithms: Spatial data structures, feature based algorithms and simplex based algorithms. Spatial data structures can easily be divided into two separate subgroups of algorithms one group based on subdivision and another based on bounding volume hierarchies. Figure 1 illustrates these groups.

The table below shows how some of the most commonly known algorithms in use today fit into the algorithm groups.

Spatial Subdivisions	Bounding Volume Hierachies	Feature Based	Simplex Based
BSP trees	Sphere trees	Lin-Canny	GJK
octrees	AABB trees	V-Clip	Enhanced GJK
k-d trees	OBB trees	Baraff's algorithm	Rabbitz's algorithm
grids	k-DOP trees		

Algorithms within each of these groups typical share some constraint on the shape of the objects, which they can handle. Let us shortly review some of these.

Feature and simplex based algorithms typical exploit convexity of the objects, which is very limiting in an enviroment of arbitary shaped objects. In most cases this drawback can be helped by allowing an

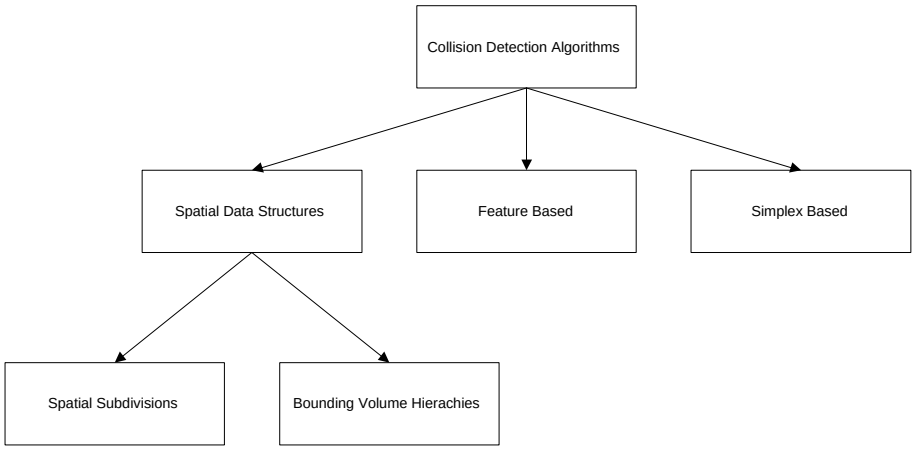


Figure 1: Overview of the groups of collision detection algorithms.

object to be representated as a hierarchy of convex sub-objects. It does however not solve the problem, just imagine how one would de-compose the surface of a torus or bowl into convex pieces (see [38] for an example).

An example of a feature based algorithm which does not suffer from this deficiency is the algorithm proposed by Moore and Wilhelms (see [5]). The algorithm is essential based on Cyrus-Beck clipping (see [1]) and testing points against planes, speeded up with several heuristics and good old fashion data structures. In simple words this algorithm is essential a clipping algorithm for polygon based objects, cable of determining the actual intersection of two objects (self intersection can also be handled), in the context of simulation and animation it is however to slow compared to other algorithms that exploit coherence. Another drawback is that the algorithm does not provide much information if objects are separated.

Feature and simplex based algorithms do however have some advances, which are particular nice in the context of simulation, animation, virtual reality and robotics (we decribe more about it in the next section). Often one can get more information from these algorithms than a simple yes or no answer to the question of whatever two objects collide with each other.

For instance all the feature based algorithms we have listed represent objects as convex polyhedra. All the algorithms are cable of returning the closest features of two separated objects. From this information one can easily compute the closest points between two objects, the closest distance and in case of near touching contact the features defines a valid contact point which can be used for contact determination.

The simplex based algorithms have allmost the same capabilities as the feature based algorithms. They however tend to be better at handling interpenetration of objects. For instance GJK and enhanced GJK (see [28] and [35]) can give a good measure of the penetration depth, whereas an algorithm like Lin-Can (see [8, 9, 10]) enters an infinite loop when interpeneration happens. V-Clip (see [7]) is far better than Lin-Can, it can handle interpenetration by detecting the penetrating features of the objects. V-Clip can however not give a good measure of the penetration depth like GJK.

The spatial data structures have one great advantage over feature based and simplex based algorithms they can represent arbitary shaped objects. In other words we are no longer restricted to convex shaped objects. Spatial data structures usually work on objects composed of polygon soups, but it need not be so.

There are basically two kinds of spatial data structure: Spatial subdivision data struture and bounding volume hierarchies. Spatial subdivision data strutures are a recursive partitioning of the embedded space, whereas bounding volume hierarchies are based on a recursive partitioning of the primitives of an object.

Spatial subdivision data strutures usually result in larger hierachical data structures than bounding volume hierachies also the way they partition space tend not to tightly cover the objects shape, which make them less suitable for determining contact status between objects.

3 The Problems of Collision Detection

Up until now we have described the collision detection in a very general way. One could be mislead to think that the goal of collision detection is to determine whaterver two objects collide or not. There is however much more involved in collision detection especially in the context of simulation, animation, virtual reality and robotics. We ourselves have a firm background in dynamic simulation of rigid bodies (see [2, 3, 4]), from this we have recognized that there are much more to collision detection than that. Basically there are four main problems [1].

The pair-processing algorithm problem This problem is the clasical problem, which started it all, how do we determine if two objects intersect each other or not in a robust and efficient manner? Our overview in the last section focussed on different families of algorithms, which solve this problem.

The all-pairs weakness problem One does not usually only have two objects. Instead one have hundres or even thousands of objects. This problem is concerned with how one avoid testing every pair of objects against each other, since such a trivial approach would be far to expensive in terms of computation time.

The fixed-timestep weakness problem This problem is very specific to simulation and animation. Running with a fixed timestep makes it possible to miss penetrations and forces one to perform collision detection at each timestep. Introducing adaptive stepsize control can help avoid these disadvantages.

The proxmity query problem In applications such as dynamic simulation it is often not enough to know whatever two objects intersect or not. One often needs to know the closest distance and/or penetration depth, or the parts of the objects surfaces, which are in close contact. All this information are needed in order to figure out how physical bodies interact with each other during a simulation.

It is very common that a collision detection engine are divided into two seperate parts. The part of a collision detection engine, which runs the pair-processing algorithm is called the narrow phase (see [22]). The all-pairs weakness problem is usually handled in the other part, which is called the broad phase (see [22]).

The proxmity query problem can be handled either simultaneously with the narrow phase collision detection or as a post processing step. Algorithms, which solve the fixed-timestep weakness problem, usually use the output from some proxmity query inorder to adaptively change the size of the timestep. These algorithms could be part of the broad phase.

Not many collision detection libraries, which exist today, are concerned with all four problems. They tend to focus on the all-pairs weakness and the pair-processing algorithm. A few have recoginized the fixed-timestep weakness and even less are concerned about the proxmity-query problem.

Approximating heterogeneous bounding volume hierarchies are only concerned with the pair-processing algorithm problem and the proxmity query problem.

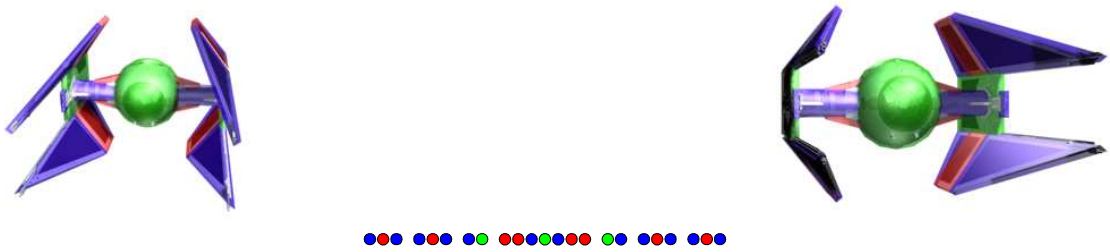
4 Approximating Heterogeneous Bounding Volume Hierachies

Let us try to take a look at an example in order to really understand what an approximating heterogeneous bounding volume hierarchy is. We will start out by showing how an approximating heterogeneous bounding volume hierarchy looks like.

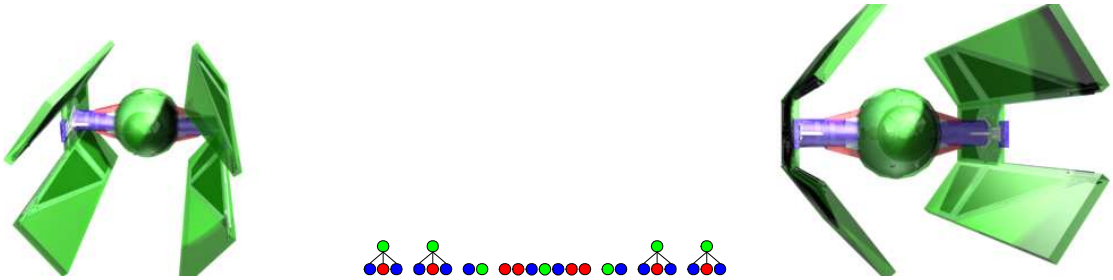
Below you can see a tie interceptor polygon soup with little more than 80.000 faces. At the left and right colums you can see two different views of the same tie interceptor in the midle colum we have shown a structural view of the bounding volume hierarchy as we construct it.

[1] Hubbard [22, 23, 24, 25] named and identified the first three problems.

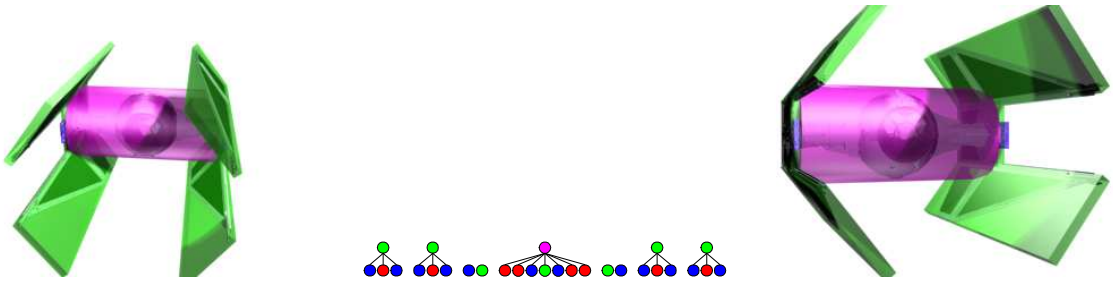
First we find the bounding volumes at the highest level of detail. Since humans are extremely good at recognizing patterens and shapes they can almost immediately with the naked eye see what kind of simple geometry that approximates a more complex shape. In our little example we will use a "human computer" together with our common sense and good intuition in order to find the bounding volumes. Note the color correspondence between the bounding volumes and the nodes in the structural view.



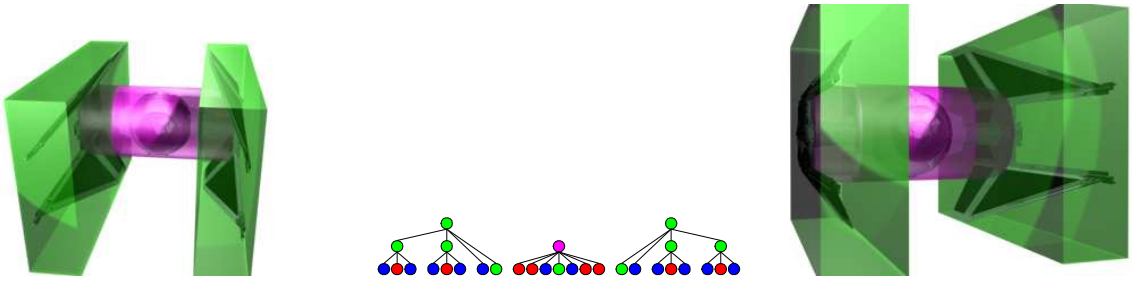
We have used four different kinds of bounding volumes, oriented bounding boxes, oriented cylinders, spheres and prism. Now we will approximate the wings with lower detail bounding volumes. Note that the higher level detail bounding volumes becomes the children of the new lower detail bounding volumes.



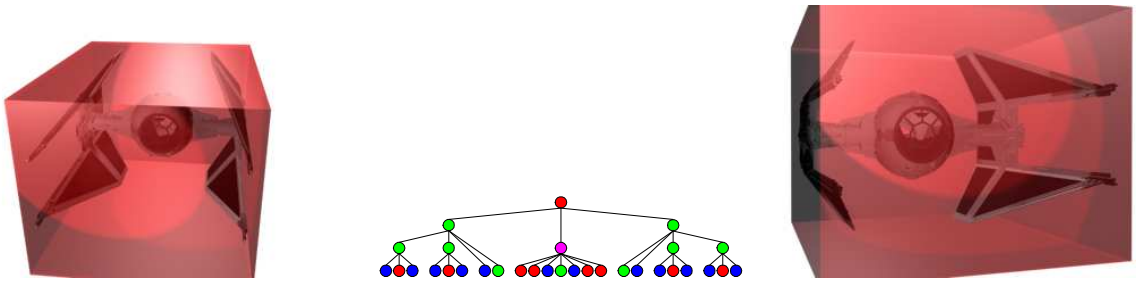
Now we repeat the last step, but this time for the hull of the tie interceptor.



Once again we find even lower detailed bounding volumes for the wings.



Finally we find a single bounding volume for the entire tie interceptor. This completes our boundig volume hierarchy.



From the example it is quite obvious that our approximating heterogeneous bounding volume hierarchy would be more than accurate in a wide range of applications. Especially in game applications, animations, virtual reality world and motion planning. It is very common to attribute each leaf bounding volume with the faces they enclose.

4.1 The Construction Algorithm

The construction algorithm we used in our example belongs to a group called bottom-up methods. From the example it should be pretty obvious why they are called so. There exist two other groups of construction algorithms these are called top-down and incremental methods.

Using a "human computer" approach as we did is very time consuming and extremely error prone due to human errors. It took us several hours just to get the little example right. It would be nice if a computer could construct the bounding volume hierarchy automatically. Unfortunately this is not easy. The majority of construction algorithms today uses top-down methods, which recursively partition the object into separate sets based on variance and not the actual shape of the objects.

As our example illustrates we believe that a bottom-up method would be the best way to construct the kind of bounding volume hierarchy we want. Those bottom-up methods we know of, only considers one kind of bounding volume, which result in homogeneous bounding volume hierarchies. Considering several different kind of bounding volumes (heterogeneous) therefore introduces several other problems than just those of homegeneous bounding volume hierarchies.

In our opinion constructing a bounding volume hierarchy in the way we want to, leaves us with several problems.

Shape decomposition How should we decompose the shape of an object into subsets?

Bounding volume fitting How do we find the best way to fit a given kind of bounding volume to some subset of the shape of the object?

Redundancies If we have chosen to apply a fully conservative coveraging strategy[2]to the shape of an object then it is very likely that redundant bounding volumes could exist. How do we eliminate such redundancies?

[2]This basical means that at any level in the hierarchy the entire shape is covered by the bounding volumes belonging to that level. The opposite is called sample-based coverage.

Bounding volume selection What kind of bounding volume should we select for a subset of the shape of the object?

Approximation control How should we control the approximation? Both outer and inner approximations could be mixed as in our example.

The three first problems are general problems, which also applies to homogeneous bounding volume hierarchies. The last two problems are specific to heterogeneous and approximating bounding volume hierarchies respectively.

4.2 The Queries Algorithms

As described previously we are interested in three different kinds of queries, these are:

- Penetration detection
- Contact determination
- Separation distance

The algorithms for performing these kinds of queries and other queries (for instance span distance) have not changed much, since hierarchical bounding volume hierarchies began their evolution.

All these algorithms are basically nothing more than a simple traversal of a hierarchical data structure, which in most cases is a tree structure as in our example. Below we have listed such an algorithm.

```
 0: Q: Queue

 1: algorithm simple-traversal(A:BVH,B:BVH)
 2:   let a be root of A
 3:   let b be root of B
 4:   push (a,b) onto Q
 5:   while Q not empty
 6:     pop (a,b) from  Q

 7:     update position and orientation of a and/or b

 8:     if a and b do not overlap then
 9:       continue with next loop

10:     if a and b both leaves then
11:        treat collision between a and b

12:     pick either a and/or b
13:     if a was picked then
14:       for all children c of a do
15:         push (c,b) onto Q
16:     if b was picked then
17:       for all children c of b do
18:         push (a,c) onto Q
```

This algorithm can be used for penetration detection by reporting a penetration in line 11. In our case with heterogeneous bounding volumes the only difficulties we have are to determine a fast, efficient and robust way to update and detect overlap between all the kinds of bounding volumes (line 7 and 8 in the pseudocode).

Coordinate system update How to update the different kinds of bounding volumes according to the current position and orientation of the objects.

Overlap detection How to detect overlaps between the different kinds of bounding volumes in a fast, efficient and robust way.

When the test in line 8 succedes this is sometimes called culling or pruning, because the entire sub hierarchies, which are below the two bounding volumes, are totally disregarded in the rest of the traversal.

The pseudocode in lines 12-18 covers what is usually called the traversal rule or the descent rule. This part of the algorithm determines how we walk through the bounding volume hierarchies when we perform our queries. There exist several different variations and heuristics for these traversal rules.

Traditionally the same algorithm can be modified to collect exact information about the overlapping parts of the shapes of the objects. The small modification happens again in line 11. This time we simply test the attributed faces of the bounding volumes for overlap, and report those, which overlap.

We can however not use this kind of approach for contact determination, because we want to detect near touching contact between shapes. That is those parts of the shapes where the separation distance is below some predefined threshold value. In other words we want to be capable of detecting contact points before penetration occurs. The reason for this desire is due to our background in dynamic simulation. In most simulator paradigms penetrations of objects are totally forbidden[3].

Another problem with the traditional approach for contact determination is redundant contact points. Imagine that a leaf bounding volume b from one hierarchy overlaps two leaf bounding volumes a_1 and a_2 from another hierarchy. If for instance a_1 and a_2 share some common edge then redundant contact points will be computed along this edge. Our experience indicate that the less number of faces a leaf bounding volume encloses the greater the chance for more redundant contact points. Removing redundant contact points can be extremely computational expensive for large polygon models.

Eventhough our approximating bounding volume hierarchy is cabable of attributing the leaf bounding volumes with faces, which makes it possible to perform exact contact determination it is not what we always want. First of all a leaf bounding volume could enclose several thousands of faces, making it a computational difficult problem to compute contact points between the faces of two leaf bounding volumes. Second of all we would like to exploit the approximation idea, which means that the contact determination should be done between the geometries of the approximating bounding volumes and not the underlining geometry they approximate.

Let us summarize the contact determination problems.

Near touching contact Performing contact determination before penetration happens.

Exact determination Detecting contact points between the real shapes of the objects.

Approximate determination Detecting contact points between the approximating bounding volumes.

Having dicussed the two first kind of queries let us now turn our attention toward the last kind of query, the separation distance. The simple traversal pseudocode query can easily be changed to accomodate this kind of query. The idea is to keep track of the currently best known estimate and then prune away those subparts of the hierarchies, which have a higher separation distance than the estimate.

[3]Penalty methods are the only exception.

```
 0: Q: Queue

 1: algorithm distance-traversal(A:BVH,B:BVH)
 2:   estimate = infinity
 3:   let a be root of A
 4:   let b be root of B
 5:   push (a,b) onto Q
 6:   while Q not empty
 7:     pop (a,b) from  Q

 8:     update position and orientation of a and/or b

 9:     if distance between a and b greater than estimate then
10:       continue with next loop

11:     estimate = distance between a and b

12:     if a and b both leaves then
13:       compute closest distance between attributed faces
14:       assign computed distance to estimate

15:     pick either a and/or b
16:     if a was picked then
17:       for all children c of a do
18:         push (c,b) onto Q
19:     if b was picked then
20:       for all children c of b do
21:         push (a,c) onto Q
22:   return estimate
```

Again we have the same problems with exact and approximate computation as we did with contact determination. Note that the approximation can be achieved simply be omitting lines 12-14.

Distance Computation How to compute the distance between different kinds of bounding volumes.

Exact computation Computing the separation distance between the real shapes of the objects.

Approximate computation Computing the separation distance between the approximating bounding volumes.

Note that if an approximate computation is chosen then the computed estimate would proberly be a very tight lower bound to the true value. It is even possible to derive an upper bound for the distance if one knows how well the leaf bounding volumes approximate the true shape of the objects (see [43]).

4.3 Comparison with Traditional Hierarchies

Let us try to compare our approximating heterogeneous bounding volume hierarchy with a traditional hierarchy. By traditional we mean a homogeneous boundig volume hierarchy constructed by a binary partitioning top-down method. Such a traditional hierarchy would be a binary tree. The leaf bounding volumes would enclose exactly one face.

Since the tie interceptor have little more than 80.000 faces a traditional hierarchy would contain $280.000 - 1 = 159999 \approx 160000$ nodes. Our approximating bounding volume hierarchy contain only 31

nodes. One immediate benefit is an extremely huge memory saving. Another indirect benefit from the smaller size is a huge performance improvement.

Note that our approach of finding bounding volumes are different from the traditional methods in the way that traditional methods converges towards the shape of the objects, approximating bounding volumes converges towards the volume (solid based) of the objects.

Previous work with bounding volume hierarchies has shown that tighter fitting bounding volumes and minimizing interpenetration of bounding volumes at the same level of detail greatly improve on the ability to prune away huge subparts of the hierarchies thereby improving the overall performance.

If we take a look at our tie interceptor example we notice that the bounding volumes we use are very tight fitting and interpenetration of the bounding volumes are definitely at a minimum at the highest level of detail it is allmost totally eliminated. We therefore speculate that approximating heterogeneous bounding volume hierarchies have an improved pruning ability over traditional hierachies. This will also contribute to a performance improvement.

As a final point our requirements and ideas for contact determination are not accomodated by the traditional hierarchies. Figuring out a way to perform contact determination as we have described would prove itself extremely useful especially for simulation purposes.

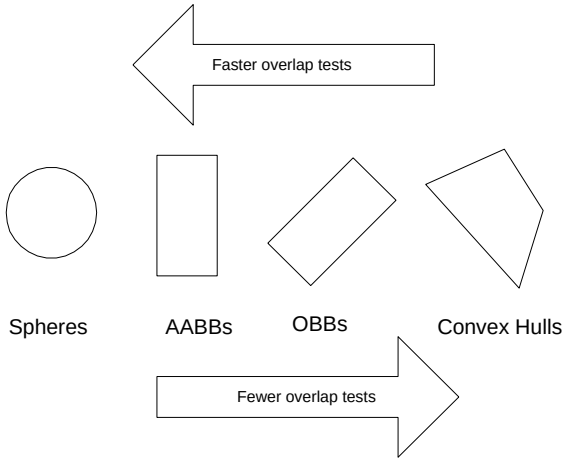
We will end this subsection, by listing the benefits we believe approximating heteregeneous bounding volume hierarchies possess.

- Memory savings
- Improved prunning capabillities
- Improved performance
- New approach for contact determination with bounding volume hierarchies.

5 Previous Work

Bounding volume hierarchies have been around for a long period of time. Consequently there is a huge wealth of literature about boudning volume hierarchies. Most of the literature addresses homogeneous bounding volume hierarchies and top-down construction methods. A great variarty of different kinds of bounding volumes have been tried out. Hubbard [22, 23, 24, 25], Palmer [48] and Palmer and Grimsdale [47] have used spheres, whereas van den Bergen [27] have worked with axed aligned bounding volumes (AABBs). Klosowski, Held, Mitchell, Sowizral and Zikan [40] and Zachman [32] have done a lot of work with discrete orientation polytypes (k-DOPs), Gottschalk, Lin and Manocha [13] and Gottshcalk [14, 15, 16] have used oriented bounding boxes (OBBs). Krishnan, Pattekar, Lin and Manocha [11] looked at a higher order bounding volume, which they called spherical shells. Larsen, Lin, Manocha and Gottshcalk [45] tried out swept sphere volumes (SSVs), which are an unification of several different kinds of shapes.

All this work have resulted in valuable knowledge about the different kinds of choices of bounding volumes. It have been discovered that there is a tradeoff between the complexity of the geometry of a bounding volume and the speed of its overlap test and the number of overlap tests in a query. This tradeoff is perhaps best illustrated by a figure.



There have not been written much about approximating bounding volume hierarchies, Hubbard's work [25] are to our knowledge the latest contribution to this subject.

There have been written even less about heterogeneous bounding volume hierarchies to our knowledge SSVs [45] are the most recent work in this area.

The general belief is however that heterogeneous bounding volumes does not change the fundamental algorithms but merely introduces a raft of other problems. We have ourself argumented for this viewpoint in this paper.

It is also believed that heterogeneous boundning volumes could provide better and more tightly fitting bounding volumes resulting in higher convergence towards the true shape volume of the objects. This could mean an increase in the prunning capabilities and a corresponding increase in performance.

Most of the work with bounding volume hierarchies has addressed objects that are representated by polygonal models many people have stated that they believe OBBs (and other retangular volumes) provides the best convergence for this kind of shape, and many experiments also indicate this[4][13, 32, 45, 16].

In our opinion one of the most amazing thing about bounding volume hierachies is that the underlying query algorithms for penetration detection, separation distance and contact determination have not changed much. In its basic form these algorithms are nothing more than simple traversals. Instead most of the recent work with bounding volume hierarchies have taken the following directions.

New volumes Trying out new kinds of bounding volumes.

Faster tests Finding faster and better overlap test methods.

Fitting methods Figuring out better methods for fitting a bounding volume to a subset of an objects underlying geometry.

Volume Comparisions Comparing homogeneous bounding volume hierarchies of different bounding volume types with each other.

There have also been a lot of attemps to try out different methods and heuristics, which could improve on the performance of these traversal algorithms.

Depth control In time critical applications it can sometimes be beneficial to set a limit on the depth a traversal is allowed to proceed in a bounding volume hierarchy. Thereby trading accuracy for performance [22].

Layered bounding volumes Having recoginized the tradoffs between the complexity of the geometry of the bounding volumes and their overlap test speed. Simpler geometries are tried out first in order to get a quick rejection test. Spheres are most commonly used [17, 28, 16].

[4]Note the difference retangular volumes are believed to converge best towards the shape, whereas heterogeneous volumes are believed to converge best towards solids.

Caching bounding volumes Caching bounding volumes from previous invokations creates withness that can exploit spatial and temporal coherence [17, 28, 16].

Shared bounding volumes This changes a hierarchy from a tree into a DAG (see [25]).

Eventhough there exist several ways for constructing boundig volume hierarchies. There have been a tendency to use top-down methods. Many people believe that bottom-up methods would be superior to top-down methods in the sense that smaller and tighter bounding volume hierarchies can be constructed. To our knowledge the work of Hubbard [25] and Palmer [47, 47] are the latest on bottom-up construction methods. Palmer uses a method based on octrees, whereas Hubbard uses a method based on the medial axe surfaces. Hubbard have shown that his method is better than the one Palmer uses. Hubbards method is however not very fast nor very easy to implement.

There are a substantial amount of literature about contact determination, but in our opinion it seems to focus on different kinds of problems. Like generating all possible contact formations between polygon models, handling uncertainties in relative position of objects or planing of contact transitions. When we described contact determination we especially had rigid body simulation in mind. Most literature about collision detection does not treat the specific needs of this area, but merely limit itself to only be concerned about overlapping faces. The work of Bouma and Vanecek [46] does however address this specific topic. To our knowledge there does not exist any literature on how to compute contact points based on bounding volume hierarchies in the way we want them. We are aware of attempts to apply hysterese [5]to the traditional hierarchies when determining contact points, but it limits simulators to be based on backtracking control algorithms and it is believed to be extremely computational expensive.

5.1 Our Preliminary Thesis

It is our belief that approximating heterogeneous bounding volume hierarchies would be superior compared to traditional hierarchies in terms of size, pruning ability and performance.

We believe our ideas and concepts for extending approximating heterogeneous bounding volume hierarchies with contact determination would result in a performance improvement over traditional hierarchies. The work would definitely be valuable to the graphics community, because it will solve a practical unpleasent problem.

There have not been much attention towards bottom-up construction methods of hierarchies, and since it is general believed that hierarchies constructed in this way are smaller and/or tighter fitting and therefore superior in terms of prunning capability and convergence, it is perhaps about time to find a more simple and effective solution for a bottom-up method, which could initiate further work in this area and perhaps even make bottom-up methods a practical alternative to the top-down methods used today.

The ideas of using approximating bounding volumes for time critical collision detection was pioneered by Hubbard. Our ideas in this area is merely an extension of Hubbards work, they are however original and any results would be valuable to the graphics community both in terms of usage and future work.

6 Conclusion

We have shown how collision detection algorithms are classified into groups and thereby we have shown how approximating heterogeneous bounding volume hierarchies are related to other kinds of collision detection algorithms.

We have explained the ideas and concepts behind approximating heterogeneous bounding volume hierarchies. Hopefully giving the reader a good intuition of what we mean by the phrase approximating heterogeneous bounding volume hierarchy.

We have described some of the obvious problems that need to be solved. Some of these problems are not specific for approximating heterogeneous bounding volume hierarchies, but are general to bounding volume hierarchies.

[5]The basic theory of hysterese also known as contact tracking is described in [6].

We have discussed what others have done and thereby explained why approximating heterogeneous bounding volume hierarchies are original in themselves.

Especially we have explained why we believe approximating heterogeneous bounding volume hierarchies would prove themself useful, by comparing them to traditional hierarchies.

References

[1] James D. Foley, Andries van Dam, Steven K. Feiner and John F. Hughes: *Computer Graphics: Principles and Pratice*, 2nd ed. in C, Addison-Wesley, 1996.

[2] Kenny Erleben: *Et studie af fysisk baseret simulation af stive legemer*, Skriftlig projekt, Datalogisk Institut Københavns Universitet. April 2000 (nr. 99-12-12).

[3] Kenny Erleben: *Et studie af grovkornet kollisionsbestemmelse*, Skriftlig projekt, Datalogisk Institut Københavns Universitet. Juli 2000 (nr. 00-08-1).

[4] Kenny Erleben: *En introducerende lærebog i dynamisk simulation af stive legemer*, Speciale, Datalogisk Institut Københavns Universitet. Maj 2001 (nr. 00-09-1).

[5] M. Moore and J. Wilhelms, *Collision Detection and Response for Computer Animation*, Computer Graphics (proc. SIGGRAPH), vol. 22, pp 289-298, 1988.

[6] Brian Mirtich: *Rigid Body Contact: Collision Detection to Force Computation*, MERL, Technical Report, TR-98-01, March 1998.

[7] Brian Mirtich: *V-Clip: Fast and Robust Polyhedral Collision Detection*, ACM Transactions on Graphics. Vol. 17. No. 3. July 1998. Pages 177-208.

[8] J. D. Cohen, M. K. Ponamgi, D. Manocha and M. C. Lin: *Interactive and Exact Collision Detection for Large-Scaled Enviroments*, Technical report TR94-005, Department of Computer Science, University of N. Carolina, Chapel Hill. http://www.cs.unc.edu/ dm/collision.html.

[9] D. Manocha and M. C. Lin: *Efficient Contanct Determination Between Geometric Models*, International Journal of Computational Geometry and Applications, 1995. http://www.cs.unc.edu/ dm/collision.html.

[10] M. K. Ponamgi, D. Manocha and M. C. Lin: *Incremental algorithms for collision detectin between solid models*, IEEE Transactions on Visualization and Computer Graphics , 1997. http://www.cs.unc.edu/ dm/collision.html.

[11] S. Krishnan, A. Pattekar, M. Lin and D. Manocha.: *Spherical shell: A higher order bounding volume for fast proximity queries*, In Proc. of Third International Workshop on Algorithmic Foundations of Robotics, pages 122-136, 1998.

[12] M. Lin and S. Gottschalk: *Collision Detection between Geometric Models: A Survey*, Appeared in the Proceedings of IMA Conference on Mathematics of Surfaces 1998. http://www.cs.unc.edu/ dm/collision.html

[13] S. Gottschalk, M. C. Lin and D. Manocha: *OBB-Tree: A Hierarchical Structure for Rapid Interference Detection*, Technical report TR96-013, Department of Computer Science, University of N. Carolina, Chapel Hill. Proc. of ACM Siggraph'96. 1996. http://www.cs.unc.edu/ geom/OBB/OBBT.html

[14] S. Gottschalk: *Seperating axis theorem*, Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill, 1996 ftp://cs.unc.edu/pub/users/Gottsch.

[15] S. Gottschalk: *Good-Fit Box for 3D Triangles in O(n lg(n)) Time*, erratta for [14]. http://www.cs.unc.edu/ geom/OBB/OBBT.html.

[16] S. Gottschalk: *Collision Queries using Oriented Bounding Boxes*, Ph.d. thesis, Department of Computer Science, UNC Chapel Hill, 2000. http://www.cs.unc.edu/ stefan/.

[17] David Eberly: *Dynamic Collision Detection using Oriented Bounding Boxes*, Magic Software, Inc. http://www.magic-software.com.

[18] David Eberly: *Intersection of Objects with Linear and Angular Velocities using Oriented Bounding Boxes*, Magic Software, Inc. http://www.magic-software.com.

[19] David Eberly: *Centers of Simplex*, Magic Software, Inc. http://www.magic-software.com.

[20] David Eberly: *Least Squares Fitting of Data*, Magic Software, Inc. http://www.magic-software.com.

[21] David Eberly: *Intersection of Cylinders*, Magic Software, Inc. http://www.magic-software.com

[22] P. M. Hubbard: *Interactive Collision Detection*, Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality, October 25-26, 1993, pp. 24-31. http://siesta.cs.wustl.edu/ pmh/research.html.

[23] P. M. Hubbard: *Collision Detection for Interactive Graphics Applications*, IEEE Transactions on Visualization and Computer Graphics , 1(3), September 1995, pp. 218-230. http://siesta.cs.wustl.edu/ pmh/research.html.

[24] P. M. Hubbard: *Real-Time Collision Detection and Time-Critical Computing*, Proceedings of the First ACM Workshop on Simulation and Interaction in Virtual Environments, July 1995, pp. 92-96. http://siesta.cs.wustl.edu/ pmh/research.html.

[25] P. M. Hubbard: *Approximating Polyhedra with Spheres for Time-Critical Collision Detection*, ACM Transactions on Graphics , 15(3), July 1996, pp. 179-210. http://siesta.cs.wustl.edu/ pmh/research.html.

[26] S. Suri, P. M. Hubbard and J. F. Hughes: *Analyzing Bounding Boxes for Object Intersection*, http://siesta.cs.wustl.edu/ pmh/research.html.

[27] Gino van den Bergen: *Efficient Collision Detection of Complex Deformable Models using AABB Trees*, Department of Mathematics and Computer Science Eindhoven University of Technology, 1998. http://www.win.tue.nl/ gino/solid.

[28] Gino van den Bergen: *A Fast and Robust GJK Implementation for Collision Detection of Convex Objects*, Department of Mathematics and Computer Science Eindhoven University of Technology, 1999. http://www.win.tue.nl/ gino/solid.

[29] Gabriel Zachman: *Exact and Fast Collision Detection*, Diploma thesis Technical University Darmstadt, Dept. of Computer Science, 1994. http://www.igd.fhg.de/ zach.

[30] Gabriel Zachman: *The BoxTree: Exact and Fast Collision Detection of Arbitrary Polyhedra*, SIVE95 (First Workshop on Simulation and Interaction in Virtual Environments), University of Iowa, July 1995. http://www.igd.fhg.de/ zach.

[31] Gabriel Zachman: *Real-time and Exact Collision Detection for Interactive Virtual Prototyping*, Proc. of the 1997 ASME Design Engineering Technical Conferences, September 14-17, 1997, Sacramento, California. Paper # CIE-4306. http://www.igd.fhg.de/ zach.

[32] Gabriel Zachman: *Rapid Collision Detection by Dynamically Aligned DOP-Trees*, Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS '98. Atlanta, Georgia; March 1998. http://www.igd.fhg.de/ zach.

[33] C. Qin, S. Cameron and A. Mclean: Towards Efficient Motion Planning for Manipulators with Complex Geometry, ISATP'95, August 1995. ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Stephen.Cameron.

[34] S. Cameron and R. K. Culley: *Determining the Minimum Translational Distance between Two Convex Polyhedra*, IEEE Conf. Robotics and Automation, San Francisco, April 1986. ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Stephen.Cameron.

[35] S. Cameron: *Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra*, Int Conf Robotics and Automation, April 1997. ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Stephen.Cameron.

[36] S. A. Ehmann and M. C. Lin: *Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching*, In Proc. International Conf. on Intelligent Robots and Systems, 2000. http://www.cs.unc.edu/ geom/SWIFT/

[37] S. A. Ehmann and M. C. Lin: *SWIFT: Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching*, Technical Report, Computer Science Department, University of North Carolina at Chapel Hill, 2000. http://www.cs.unc.edu/ geom/SWIFT/

[38] S. A. Ehmann and M. C. Lin: *Accurate and Fast Proximity Queries Between Polyhedra Using Convex Surface Decomposition*, EUROGRAPHICS 2001, volume 20 (2001), Number 3. http://www.cs.unc.edu/ geom/SWIFT++/

[39] James Thomas Klosowski: *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments*, Ph.D. Dissertation, State University of New York at Stony Brook, May 1998. http://www.ams.sunysb.edu/ jklosow/publications

[40] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral and K. Zikan: *Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs*, IEEE Transactions on Visualization and Computer Graphics, March 1998, Volume 4, Number 1. http://www.ams.sunysb.edu/ jklosow/publications

[41] Thomas MÖller: *A Fast Triangle-Triangle Intersection Test*,

[42] E. Welzl: *Smallest enclosing disks (Balls and ellipsoids*, New Results and New Trends in Computer Science. Lecture Notes in Computer Science, vol 555, pages 359-370. Springer-Verlag, 1991.

[43] David E. Johnson and Elaine Cohen: *A Framework For Efficient Minimum Distance Computations*, Proc. IEEE Intl. Conf. Robotics & Automation, Leuven, Belgium, May 16-21, 1998, pp. 3678-3684

[44] Tim Culver, John Keyser, and Dinesh Manocha: *Accurate Computation of the Medial Axis of a Polyhedron*, In Proc. of ACM Solid Modeling, 1999. UNC Chapel Hill Computer Science Technical Report TR98-034, 1998 http://www.cs.unc.edu/ geom/MAT

[45] Eric Larsen, Stefan Gottschalk, Ming C. Lin, Dinesh Manocha: *Fast Proximity Queries with Swept Sphere Volumes*, Technical report TR99-018, Department of Computer Science, University of N. Carolina, Chapel Hill. 1999. http://www.cs.unc.edu/ geom/SSV

[46] William J. Bouma and George Vanecek Jr.: *Modelling Contacts in a Physical Based Simulation*, Proc. Solid Modeling, 409-419, 1993, 1, 3.

[47] I.J.Palmer and R.L.Grimsdale: *Collision detection for animation using sphere-trees*, Computer Graphics Forum, 14(2), 1995, pp105-116.

[48] I.J.Palmer: *Collision detection for animation: the use of the sphere-tree data structure*, presented at The Second Departmental Workshop on Computing Research, University of Bradford, June 1995.