Detecting Self-Collisions Using a Hybrid Bounding Volume Algorithm

I. INTRODUCTION

Two major computational bottlenecks in many animation systems are the handling of collisions between the objects under the influence of forces (gravity, user input, etc.), and the self-collision detection which tests the object itself to find colliding regions. Accurate self-collision detection is challenging to perform in real time because many adjacent or nearby primitives of a deforming mesh are always in close proximity.

Some models suffer large deformations which consequently causes self-collisions. The mesh is deformed in such a way that polygons can touch other polygons during the deformation. In this work 3D objects are considered and a hierarchy of spheres is utilised to approximate the shape of the objects.

Valuable surveys detailing collision detection algorithms have been done. Lin et al. [1] classify the methods based on the type of the geometric model used, while Jiménez et al. [2] consider their interference process. Most recently, Teschner et al. [3] surveyed collision detection for deformable objects where five methods are described and applications are shown.

Madera et al. [4] proposed an algorithm to detect collisions, considering the parameters of the BV as part of the mesh in order to follow the deformation, avoiding many computations in the updating process. For a sphere represented by two parameters, only the radius is considered. In this paper an extension to the algorithm is presented which considers self-collisions.

The remainder of the paper is organised as follows: Section II details the basis of the algorithm, the 1-ring concept and the two stages of the algorithm. In Section III the contact determination process is explained. Experiments are shown in Section IV where the Standard Sphere (SV) and the Axis Aligned Bounding Box (AABB) are compared against our approach. Results and plans for future work are given in Section V.

II. THE ALGORITHM DVH (DISTANCE VOLUME HIERARCHY)

In this section the basis of the algorithm proposed is detailed, describing the concepts that support the design and implementation as a well-suited method to detect self-collisions between deformable objects.

A. The 1-ring

A known concept in mesh-based representations is the 1-ring. A 1-ring, \( R \), can be defined as a set of triangles \( R = \{T_1, T_2, ..., T_k\} \), with a set of vertices \( \{v_1, v_2, ..., v_k\} \), where \( v_i \) is the common vertex of \( R \) and the triangles are defined as follows:

\[
T_a = \langle v_1, v_{a-1}, v_a \rangle \quad \text{for} \quad a = 1, 2, ..., k - 1. \quad (1)
\]

\[
T_k = \langle v_k, v_1, v_1 \rangle. \quad (2)
\]

The sphere is a good choice as a BV for three reasons: it is rotationally invariant, the intersection test is a cheap operation, and the centre can be set as the common vertex of the 1-ring. Let \( S \) be a sphere bounding \( R \) with centre \( c \), and radius \( r \). The centre is the average of the vertices in \( R \), and the radius is the maximum distance between \( c \) and the vertices in \( R \). Given two spheres \( A(C_A, r_A) \) and \( B(C_B, r_B) \) from objects \( A \) and \( B \) respectively, an overlap occurs between nodes \( A \) and \( B \) if

\[
\left\| C_A - C_B \right\|_2^2 < (r_A + r_B)^2. \quad (3)
\]

This inequality verifies the squared distance between two spheres, using the 3-vector Euclidean norm, \( \| \cdot \|_2 \), and requires 11 basic operations.
B. The Phases of the Algorithm

The DVH algorithm is comprised of two stages, hierarchy construction and hierarchy traversal. Bounding volumes are created in the hierarchy construction stage whilst self-collisions are determined in the hierarchy traversal stage. One parameter is required to be updated per region during deformation, the radius of the sphere that serves as a bounding volume. A triangulated mesh with a fixed connectivity is employed. This mesh is decomposed into 1-rings.

The phase of hierarchy construction is computed as a pre-processing step, where the top-down method is easily implemented. First of all, the 1-rings are created, these 1-rings are grouped into clusters to form the upper levels of the hierarchy. A cluster contains a group of 1-rings. The algorithm bounds each cluster of geometry with a sphere, which will be utilised for the intersection test and will need to be updated at runtime as deformation occurs.

Given two hierarchies $H_A$ and $H_B$, the intersection test consists of comparing both hierarchies using the breadth-first search method. The method visits all the nodes of a level in order to detect overlaps between the bounded nodes, $BV(H_A)$ and $BV(H_B)$. If an overlap is found, then the pair of nodes will be expanded to the next level, and this continues until the leaves of the tree are reached. For self-collisions the hierarchy must be compared with each bounded polygon.

In rigid bodies, $SBV$ needs to apply the matrix transformations in order to keep the BVs well located, whilst $DVH$ works without any update. For deformable objects, the problem is more difficult since the vertices displaced are independent. When an object deforms, it is not necessary to update all the hierarchy, only the parts containing moving vertices.

As seen in the hierarchy construction stage, the geometry of the mesh is enclosed with spheres. Given a sphere $S(c, r)$ of $SBV$, a sphere $S'(c', r')$ of $DVH$, and a geometry determined with $p$ points $\{v_1, v_2, ..., v_p\}$ enclosed by $S$ and $S'$, a deformation occurs when one or more points are displaced, and consequently an update is required. To update $S$, $O(p)$ time is required to compute $c$ and $O(p)$ to compute $r$. To update $S'$, only $O(p)$ is required for $r'$. Our approach can reduce the $p$ operations when only a few points of the sphere are moved since the centre is fixed.

We are updating the hierarchy using a bottom-up approach with the aim of minimising the number of spheres created. It is assumed that if a sphere is updated, then its parent is too. This is not always valid because we are dealing with the wrapped approach that takes into account the child geometry rather than the child spheres. The algorithm is valid for all the spheres in the hierarchy, for the geometries of the 1-rings and clusters. A special case occurs when the centre of a child node is moved. At this point the centre is no longer within the threshold of the geometry of its parent, and consequently can be ignored. This is illustrated in Figure 2. We do not know the number of spheres to be updated, this depends on the deformations, but we can make a comparison with the $SBV$ and the AABB to check the number of updates.

III. The Contact Determination Phase

Triangular meshes with a fixed connectivity were employed to represent the models and care has been taken to preserve real deformations. Figure 3 illustrates one of the models used in the experiments, the tubular surface. Adjacent 1-rings are colliding by nature, so they are not tested for overlap. In Figure 3, the 1-ring, in red, is not tested with its neighbouring 1-rings, shown in yellow.

The self-intersection occurring as the mesh deforms may result in two closed intersection paths. Consequently, two
distinct clusters of the mesh intersect. The other self-intersection case, illustrated in Figure 4, results in only one closed intersection path where the mesh deformed, it is formed by a single region that has been folded on itself, forming a loop. This latter case produces unreal deformations and cannot arise for smooth surfaces because it would require the folded mesh to become non-differentiable at the loop vertices, as stated in [5]. In contrast, 2D objects such as cloth, should be checked for intersections in all the regions as the large deformation process takes place.

![Figure 4](image_url)  
**Figure 4.** Unreal deformation occurs when the objects are strongly blended, and subsequently it is not considered in our tests.

The goal is to compute the polygons in collision, but joined polygons should not be considered as polygons in collision because they are placed in the original position. To avoid the problem of joined 1-rings, a 1-ring connectivity is computed in the preprocessing stage in order to determine if two joined 1-rings are colliding. This is supported by the fact that surface based 3D object deformation is applied. The 1-ring is not too small as a polygon, it is 8 times bigger, and it is not too big as the nodes of the first level to collide very often. The hypothesis is that by using 1-rings an efficient method to detect self collisions can be achieved. Once two colliding 1-rings are found, the primitive pair, \( N \) is computed. In the worst case, the DVH reports \( N \) times, the AABB reports similar performance. This is because the AABB contains fewer pairs of BVs tested and therefore fewer pairs of triangles must be computed. In the worst case, the DVH reports 11 ms, the SBV 17 ms, and the AABB 11 ms. Such total time is divided into two parts as eq. (4) states, the intersection test and the updating process. In the updating process the DVH and the AABB take 1 ms whilst the SBV takes 2 ms in the maximum workload.

**IV. EXPERIMENTS**

The algorithm was run on a PC with a 2.4 GHz Intel Quad Core CPU and an NVIDIA 8800 GTX. Three objects were employed, a dinosaur with 28K polygons, a tubular surface with 8K polygons and a pawn with 7K polygons. The experiments were set up to determine the times and the number of self-intersections during simulations. The experiments were run using an octree for each one of the BVs approaches: DVH, SBV, and AABB.

According to [7], [8], the total cost of intersecting two hierarchies, \( TC \), is given as:

\[
TC = N_V C_V + N_P C_P + N_U C_U, \tag{4}
\]

where \( N_V \) is the number of BV pairs tested for overlap, \( C_V \) is the cost of testing a pair of BVs for overlap, \( N_P \) is the number of primitive pairs tested, \( C_P \) is the cost of testing a primitive pair, \( N_U \) is the number of nodes that need to be updated, and \( C_U \) is the cost of updating such a node.

In the first experiment, the tubular surface is animated using control splines. This method determines the points of the curve according to some chosen parameters. 16 points were defined as the basis to create the tubular surface. 15 segments were divided into 8x20 parts to form a circle around each pivot point. The number of 1-rings is 1,400 and the number of bounding volumes is 580. The surface slides during the animation and collides with itself. Figure 6 illustrates the total time of the algorithm to detect collisions during 1,000 steps using an octree. The DVH method becomes faster on average, but in the worst case when the number of collisions is maximum, the AABB reports similar performance. This is because the AABB contains fewer pairs of BVs tested and therefore fewer pairs of triangles must be computed. In the worst case, the DVH reports 11 ms, the SBV 17 ms, and the AABB 11 ms. Such total time is divided into two parts as eq. (4) states, the intersection test and the updating process. In the updating process the DVH and the AABB take 1 ms whilst the SBV takes 2 ms in the maximum workload.

The second experiment was carried out using an animated dinosaur, employing morphing by point specification. Figure 7 illustrates the total time of the algorithm to detect collisions during 200 steps. The object is deformed in several parts of the body to simulate a jump, causing the legs to collide with the stomach. The DVH is the faster approach with 38 ms in the maximum workload followed by the SBV with 45 ms and finally the AABB with 55 ms. The intersection test time is more expensive than the updating process. In this sense, the time given by the total cost is

![Figure 5](image_url)  
**Figure 5.** The three BVs utilised in the contact determination phase.

**Triangles — Intersection** function is called in order to obtain the pair of triangles in collision. Bounding volumes were applied to the polygons such that the process remains the same as in the broad phase, for performing overlapping tests. The process is different from the one used for the 1-rings where a predefined centre is computed. When using the SBV and the AABB approaches, each polygon is bounded to be prepared for the intersection test. In the case of the DVH, triangle \( T_a \) (1) is bounded by the sphere with diameter \( v_i, v_{i-1} \). Since \( T_a \) is part of a 1-ring, then the sphere bounding the neighbouring polygon with diameter \( v_i, v_{i-1} \) also covers part of \( T_a \), as depicted in Fig. 5. When two bounded triangles are overlapping, the interval overlap method proposed by Møller [6], is applied.
Figure 6. Scene 1. (left) A tubular surface deformed using control splines. Coloured triangles indicate the self-collisions detected. (right) Total time of the collision detection algorithm for the three methods AABB, DVH, and SBV. This time includes the intersection test for all the bounding volumes of the hierarchy and the updating process.

Figure 7. Scene 2. (left) A dinosaur deformed using morphing. The left object is the original pose and the right object is the deformed object. Black triangles indicate the self-collisions detected on the surface. (right) Total time of the collision detection algorithm for the three methods AABB, DVH, and SBV. This time includes the intersection test for all the bounding volumes of the hierarchy and the updating process.

divided into two parts, the intersection test/updating process as follows. 35/3 ms for the DVH, 41/4 ms for the SBV, and 48/7 ms for the AABB. The contact determination phase is the most expensive computation in the intersection test because the triangles are tested for intersection.

The third experiment recorded 1,000 time steps of the animated pawn. Forces in the top of the object were applied as illustrated in Figure 8. The DVH is the faster method with 10.5 ms, followed by the AABB of 11.5 ms and the SBV with 19 ms. Most of the total time is consumed by the intersection test and few milliseconds in the updating process, that is, 0.5 ms in the DVH, 1 ms in the SBV, and 1.5 ms in the AABB. The performance of the AABB and the DVH are similar.

V. CONCLUSIONS

An algorithm for efficiently computing self-collisions has been presented, the design and implementation issues were discussed as well as some experiments to validate its performance. The algorithm involves two main stages, the construction of the hierarchy in the precomputation stage, and the comparison of the hierarchy against the primitives for detecting collisions during the simulation.

According to the experiments, the most expensive operation of the algorithm is the intersection test, requiring approximately 90% of the total time. This function is formed by several procedures, BV tests that include the BVs of the clusters and 1-rings, and the contact determination phase which includes the triangle-triangle intersection computation. Even when the BVs tested in the spheres approaches is greater than in the AABB, the time is less. However, when applying the contact determination phase where the triangle-triangle intersection takes place, the timing in the AABB becomes faster.

A contact determination phase was chosen to obtain more accuracy in the collisions. This function returns the features of the polygons in collision, which can be the face, the edge or the vertices.

The algorithm works for both, collisions and self-collisions, but in this work the focus is on self-collisions. Three deformation schemes were employed, the mass-spring model, morphing deformation by points specification and the
control splines.

There are some avenues for future work. One is to use the algorithm in other applications including avatars, and cloth; and the other is the implementation of a collision response approach to evaluate the performance of the algorithm when using forces. Furthermore, the idea of using the feature-based and the BVH can be extended to other bounding volumes to speed up and save operations.

REFERENCES


Figure 10. Scene 2. The number of pairs of BVs tested (left) and the number of pairs of triangles tested (right) in the second experiment for the three methods AABB, DVH, and SBV.

Figure 11. Scene 3. The number of pairs of BVs tested (left) and the number of pairs of triangles tested (right) in the third experiment for the three methods AABB, DVH, and SBV.