DOI: 10.1111/j.1467-8659.2010.01768.x

Eurographics Symposium on Geometry Processing 2010
Olga Sorkine and Bruno Lévy
(Guest Editors)

*Volume 29 (2010), Number 5*

# Enhancing Bounding Volumes using Support Plane Mappings for Collision Detection

Athanasios Vogiannou[1,2] and Konstantinos Moustakas[2] and Dimitrios Tzovaras[2] and Michael G. Strintzis[1]

[1]Electrical & Computer Engineering Department, Aristotle University of Thessaloniki, Hellas
[2]Informatics and Telematics Institute, Center of Research and Technology Hellas

**Abstract**

*In this paper we present a new method for improving the performance of the widely used Bounding Volume Hierarchies for collision detection. The major contribution of our work is a culling algorithm that serves as a generalization of the Separating Axis Theorem for non parallel axes, based on the well-known concept of support planes. We also provide a rigorous definition of support plane mappings and implementation details regarding the application of the proposed method to commonly used bounding volumes. The paper describes the theoretical foundation and an overall evaluation of the proposed algorithm. It demonstrates its high culling efficiency and in its application, significant improvement of timing performance with different types of bounding volumes and support plane mappings for rigid body simulations.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

## 1. Introduction

Collision detection is a fundamental concept in many scientific fields such as computer graphics, dynamic simulations, robotics and haptics. Every application that involves some kind of interaction between virtual entities requires a suitable method for detecting and handling interpenetrations among them in order to produce a correct and visually appealing result. Due to the importance of collision detection, the respective literature is vast and numerous methods have been proposed. Excellent surveys on prior work can be found in [JTT01, HEV*04] and [TKZ*04].

The various types of applications have different requirements for collision queries, e.g. deformable [SSIF09] or rigid models [TJ08]. Among them, the boolean query between rigid objects is the most studied subject in the field due to the wide applicability of rigid models and the theoretical background from computational geometry. Each BV type has its advantages and drawbacks, depending on the bounding efficiency and the intersection test between the BVs. The bounding efficiency of a BV is defined as the percentage of the volume of the bounded object to the total volume of the BV. It depends on the geometry of the bounded object and the ability of the BV to adapt to it.

The majority of state-of-the-art approaches is based on building a Bounding Volume Hierarchy around the object. The main advantage of Bounding Volumes is the existence of efficient methods to test for intersection between them. The Bounding Volumes (BV) mostly used in practice are Spheres [Hub96, KGS98], Axis Aligned Bounding Boxes (AABB) [van97, Zac02], Oriented Bounding Boxes (OBB) [GLM96, ASC*06] and Discrete Orientation Polytopes (k-DOPs) [KHM*98, Zac98].

The intersection tests between BVs are based on the Separating Axis Theorem for convex objects [GLM96, CS08]. The theorem states that for a pair of disjoint convex objects there exists an axis, such that the projections of the objects on this axis do not overlap. Intersection tests for BVs exploit this theorem by testing the existence of a Separating Axis in a set of candidate axes. The basic difference among different types of BVs is the number of axes needed to be tested. There is a trade-off between the bounding efficiency and the computational cost of the intersection test. BVs with low efficiency, such as spheres, can be tested very fast for

intersection while more efficient bounding volumes, such as the OBBs, require much more computation.

In the collision detection literature every BV is integrated into an hierarchy. The hierarchy can take advantage of the low computational cost of BV-BV intersection tests to detect collisions between much more complex objects. If an intersection is detected in the root of the Hierarchy tree, the algorithm proceeds by checking the volumes in the children of the root and so on, until the leaf nodes are reached and the exact points of collision are found. There are additional issues related to hierarchies like building strategies [ZL03, BO04], hierarchy updating [JP04, WBS07, OCSG07] and other optimization techniques [LC98].

Bounding Volumes and the respective Hierarchies (BVH) have dominated the field of collision detection due to the simple and fast pairwise culling they can achieve. Even though they have been initially used for pairwise collision detection of complex rigid models, there are numerous other methods for extending the hierarchies to other types of applications, such as deformable model simulations [SGG*06, CTM08, TCYM08], continuous collision detection [ZRLK07] and ray-tracing [WBS07]. Most of these methods take advantage of some kind of BVH at an initial culling step before performing more sophisticated algorithms for further collision pruning and contact determination.

In this paper, we present a new method for improving the collision detection performance of Bounding Volume Hierarchies for complex objects. The major advantage of the proposed approach is that it can quickly reject non intersecting objects even when the respective Bounding Volumes intersect, without traversing more levels in the hierarchy. Therefore, it is particularly useful in applications with multiple objects where the BVs of non-intersecting objects overlap frequently on the first levels of the hierarchy. Moreover, the general approach employed by our method can be applied directly to any type of existing hierarchies in rigid body simulations.

The paper begins with the theoretical foundation of the proposed approach in sections 2 and 3. In particular, section 2 describes Support Plane Mappings, a fundamental concept of the presented approach, and section 3 introduces the basic conservative collision rejection theorem that we propose as an extension to the separating axis theorem. The theoretical results of these sections are employed in section 4 where we present the details concerning the culling algorithm for different types of BVs as well as further implementation issues. Finally, the experimental results along with the conclusions are presented in sections 5 and 6 respectively.

### 1.1. Notation

Table 1 summarizes the notation and conventions used throughout the text.

| Expression | Description |
|---|---|
| $E(\mathbf{n},\mathbf{p})$ | Plane with equation $(\mathbf{x}-\mathbf{p})\cdot\mathbf{n}=0$ for $\mathbf{x},\mathbf{n},\mathbf{p}\in\Re^3$ |
| $\mathbf{n}_E$ | Normal vector of plane $E$ |
| $vert(O)$ | The set of vertices of object $O$ |
| $M_O$ | A Support Plane Mapping of object $O$ |
| $H_E^{\{+,-\}}$ | The positive (+) or negative (-) closed half-space of plane $E$ |

**Table 1:** *Notation*

## 2. Support Plane Mappings

Support planes are a well studied subject of computational geometry and have been employed in algorithms for the separation of convex objects [DK85, CW96, vdB03]. From a geometrical perspective, a support plane $E$ of a 3D convex object $O$ is a plane such that $O$ lies entirely on $H_E^-$. Support planes have become useful in previous algorithms based on the concept of support mappings. A support mapping is a function that maps a vector $\mathbf{v}$ to the vertex of $vert(O)$ that is "most" parallel to $\mathbf{v}$ [vdB03, Eri05]. As a direct consequence, a support plane can be defined as the plane that passes through $s_O(\mathbf{v})$ and is parallel to $\mathbf{v}$.

The importance of support planes is intuitively apparent: they provide an explicit way of deciding whether another object could possibly intersect with the one that the support planes refers to. Based on this simple but important feature of support planes, we present here a slightly more generalized formulation and introduce the concept of support plane mappings by the following definitions:

**Definition 2.1** $E$ is a *Support Plane* (SP) of the object $O$ if

1. $\mathbf{x}\in H_E^-$, $\forall\mathbf{x}\in O$
2. $E$ and $O$ have at least one common point.

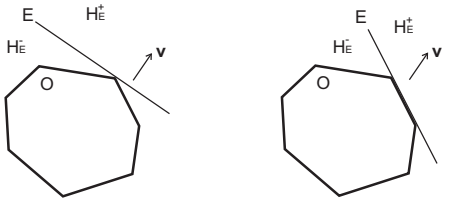
**Definition 2.2** Let the object $O$ and $E_O$ be a set of Support Planes of $O$. A Support Plane Mapping (SPM) of $O$ is defined as

$$M_O(\mathbf{v}) = E\in E_O : \mathbf{v}\cdot\mathbf{n}_E = max\{\mathbf{v}\cdot\mathbf{n}|\mathbf{n}\in\mathbf{n}_{E_O}\}$$

where $\mathbf{n}_{E_O}$ denotes the set of all normals in $E_O$.

The difference between the above definitions and previous work is that the above definitions do not make any assumption about the convexity of $O$ or, concerning only the SPM, about the set of Support Planes $E_O$. For example, a Support Plane Mapping can be constructed using the infinite set of all support planes of object $O$ and the support mapping $s_O$. This kind of SPM will be referred from now on as the *Vertex-based* SPM (Figure 1-(a)) of $O$, since $s_O$ maps to $vert(O)$. The Vertex-based SPM is actually an alternative definition for generating support planes [DK85]. Since there

is no restriction on the used set of Support Planes $E_O$, another approach to construct a SPM would be to use the set of support planes that lie on at least one face of $O$. This kind of mapping will be referred to as the *Face-based* SPM (Figure 1-(b)) of $O$. Note that both Vertex-based and Face-based SPMs are uniquely defined for each object.




(a) (b)

**Figure 1:** *Support Plane Mappings. In (a) the Support Plane E is generated using a vertex-based mapping, i.e. $E = M_O(\mathbf{v})$, and the normal is parallel to the direction of the input vector* $\mathbf{v}$*. In (b) E comes from a face-based mapping and it lies on a face of O.*

In practice we are mostly interested for having enough support planes to surround the given object. Therefore we define the *fully bounding* SPM of an object $O$ as a SPM such that the planes of the respective $E_O$ form a finite sub-space $G = \bigcap_i H^-_{E_i}$ for every $E_i \in E_O$, that fully bounds $O$. This subspace $G$ serves, implicitly, as a convex bounding representation of the object. Note that both Vertex-based and Face-based SPMs are fully bounding SPMs. For the rest of the paper, we shall limit our attention to fully bounding SPMs.

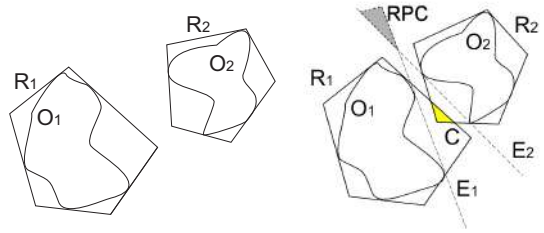
## 3. Conservative Collision Rejection

A BV $R$ of an object $O$ is a conservative representation [Eri05] of the object, meaning that $O \subseteq R$. Consequently, the result of an intersection test between a pair of BVs is always conservative in terms of the bounded object. The combination of a conservative representation with a fast intersection test, which results in conservative collision rejection is the basic formula for the utilization of BVs. In this paper, we present an extension to this concept based on a theorem for quickly rejecting collisions between two objects using the respective BVs and a SP of each object.

Let $E_1$ be a SP for the convex object $O_1$ and $E_2$ be a SP for the convex object $O_2$. We define as the Region of Possible Collision (RPC) the sub-space of $\Re^3$ that lies on the negative half-spaces of both $E_1$ and $E_2$, i.e. $RPC = H^-_{E_1} \cap H^-_{E_2}$. It is trivial to prove that

$$\text{If } C = O_1 \cap O_2 \neq \text{ then } C \subset RPC$$

However, in practice the objects are not always convex. If the respective convex BVs are used instead, we can use the above separation condition to define the following lemma for the separation of two general objects.




(a) (b)

**Figure 2:** *In (a) the convex BVs $R_1$ and $R_2$ of the objects $O_1$ and $O_2$ respectively do not intersect and the two objects do not collide. In (b) the BVs collide but the intersection region C does not lie inside the RPC. Therefore we can detect the absence of collision based on lemma 3.1.*

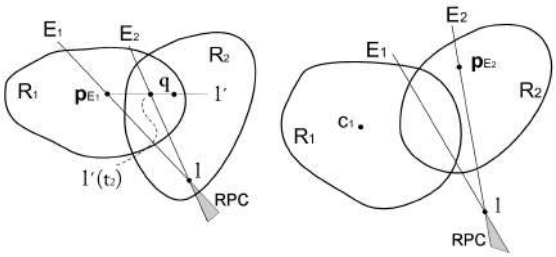
**Lemma 3.1** Let $R_1$, $R_2$ be two convex BVs and $E_1$, $E_2$ be two SPs of the (not necessarily convex) objects $O_1$ and $O_2$ respectively. The two objects are disjoint provided that

$$(R_1 \cap R_2) \cap RPC = \emptyset$$

The above lemma describes the basic idea behind the proposed method by providing a simple relation between BVs and SPs. Indeed, the first component $R_1 \cap R_2$ above, is used in standard BV-BV tests while the additional condition on the RPC represents the proposed enhancement. An example of the above is illustrated in figure 2. The method for performing this additional test is based on the following theorem.

**Theorem 3.2** Let two objects $O_1$, $O_2$ with the respective convex BVs $R_1$, $R_2$ and $\mathbf{c}_1$, $\mathbf{c}_2$ two arbitrary points inside $O_1$ and $O_2$. Let also $E_1$, $E_2$ be SPs of each object and the intersection line be $l = (E_1 \cap E_2)$. If any of the next tests is positive, the objects do not collide.

1. In the case that $l$ does not pass through $R_1$:
   - If $(R_1 \cap E_2 \neq \emptyset)$ (Figure 3-(a)), then let $\mathbf{q}$ be an arbitrary point in $(R_1 \cap H^-_{E_2})$, $\mathbf{p}_{E_1}$ be a point in $E_1 \cap R_1$, $l'$ be the line from $\mathbf{p}_{E_1}$ to $\mathbf{q}$ and $l'(t_2)$ be the intersection between $E_2$ and $l'$. Test if
   $$0 < (l'(t_2) - \mathbf{p}_{E_1}) \cdot \mathbf{n}_{E_1} \leq (\mathbf{q} - \mathbf{p}_{E_1}) \cdot \mathbf{n}_{E_1} \quad (1)$$
   - If $(R_1 \cap E_2 = \emptyset)$ (Figure 3-(b)), test if
   $$(\mathbf{c}_1 - \mathbf{p}_{E_2}) \cdot \mathbf{n}_{E_2} > 0 \quad (2)$$
2. In the case that $l$ does not pass through $R_2$:
   - If $(R_2 \cap E_1 \neq \emptyset)$, then let $\mathbf{q}$ be an arbitrary point in $(R_2 \cap H^-_{E_1})$, $\mathbf{p}_{E_2}$ be a point in $E_2 \cap R_2$, $l'$ be the line from $\mathbf{p}_{E_2}$ to $\mathbf{q}$ and $l'(t_1)$ be the intersection between $E_1$ and $l'$. Test if
   $$0 < (l'(t_1) - \mathbf{p}_{E_2}) \cdot \mathbf{n}_{E_2} \leq (\mathbf{q} - \mathbf{p}_{E_2}) \cdot \mathbf{n}_{E_2} \quad (3)$$




(a) (b)

**Figure 3:** *Case (1) of theorem 3.2 where the intersection line l does not pass through $R_1$. In (a) we need to test for the inequality 1 since $R_1 \cap E_2 \neq \emptyset$, while in (b) we need to test for the inequality 2.*

- If $(R_2 \cap E_1 = \emptyset)$, test if

$$(\mathbf{c}_2 - \mathbf{p}_{E_1}) \cdot \mathbf{n}_{E_1} > 0 \tag{4}$$

Figure 3 displays case (1) of the above theorem. The proof is given in the Appendix. Note also that in the case that $l$ passes through both BVs, the theorem cannot be used since neither of the conditions in cases (1) or (2) are satisfied. The underlying concept of the above theorem is based on ray-casting techniques [WBS07]. The tests search for the RPC by calculating line-plane intersections, depending on the relative position of the intersection line between the two SPs. There is no restriction to the type of the used BVs as long as they are convex.

The proposed method employs a conservative representation of the objects along with a conservative test instead of an accurate test. In practice this results in significant culling improvement due to the high bounding efficiency of SPs. Moreover, the following lemma states that it is *always possible* to separate two disjoint objects based on theorem 3.2.

**Lemma 3.3** If two convex objects $O_1$ and $O_2$ are disjoint then there exists a Support Plane for each object such that the corresponding conditions of theorem 3.2 will correctly indicate the absence of collision.

The above lemma can be proved if we note that in the case of two disjoint objects, a pair of SPs parallel to a separating axis (which is guaranteed to exist according to the Separating Axis Theorem [GLM96,CS08]) can be used with the theorem. Therefore, it reduces to testing either of the inequalities 2 or 4 of each case of theorem 3.2 and then it is easy to see that the collision is rejected. A more rigorous proof of the above lemma can be found in the additional material.

### 3.1. Comparison with the Separating Axis Theorem (SAT)

Both the SAT and theorem 3.2 provide a separation condition for a pair of objects and they present quite interesting similarities. If we consider that a SA defines a set of planes in the same direction, the SAT can be described by theorem 3.2 by ignoring the BVs and setting parallel support planes $E_1$ and $E_2$ with opposite normals (the proof of lemma 3.3 is actually based on this property). Therefore, it can be argued that the SAT uses one separation direction while the proposed theorem two, one for each object. Additionally, the SA test is usually applied on BVs to quickly calculate projections of BVs into an axis while the proposed theorem uses BVs to quickly calculate BV-line intersections. The major advantage of the proposed theorem, however, is that it applies even if the used planes are not parallel to a SA and can be considered to be more general than the SAT.

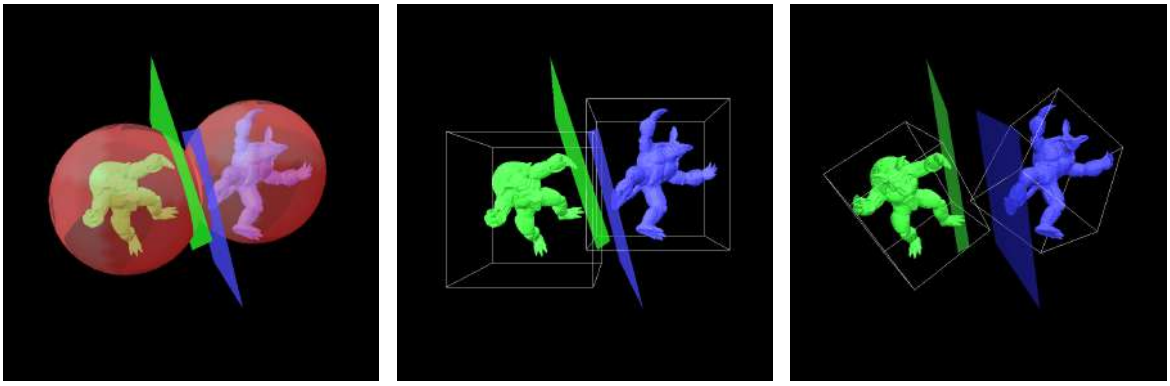
## 4. Enhancing BVs with SPs

The implementation aspects of theorem 3.2 will be discussed in this section. The most important implementation issue of the proposed method is the selection of the SPs. A SPM $M_O$ of each object can be used to extract a pair of SPs which reduces the problem to the selection of a pair of vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ that will be used as input to the SPMs, i.e. $E_1 = M_{O_1}(\mathbf{v}_1)$ and $E_2 = M_{O_2}(\mathbf{v}_2)$. From another point of view, the directions of $\mathbf{v}_1$ and $\mathbf{v}_2$ play the role of the candidate axes used in standard BV-BV tests [van97, GLM96]. Indeed, the SA test can be implemented by theorem 3.2 using a Vertex-based SPM and setting $\mathbf{v}_1 = -\mathbf{v}_2$, as described in section 3.1. The Vertex-based SPM is needed so that the planes will always be parallel to each other (a vertex-based SPM always maps to planes with normals parallel to the input direction). Note again that theorem 3.2 can be used with other types of SPMs, e.g. Face-based SPMs.

The standard SA test is employed with BVs by testing a set of candidate axes. Employing theorem 3.2 with the same approach would cause significant computational overhead since testing line – BV intersections is not as fast as calculating BV projections into an axis. Therefore, in the proposed implementation we will limit the test in a single pair of axes which, intuitively, approximate a SA. Several efficient methods for computing line-plane and line-BV intersections are given in the literature [vdB03, Eri05].

### 4.1. BV Types

In general, any BV type can be employed by theorem 3.2 as long as it can provide a way to extract the necessary information to test for the inequalities, i.e. the points $\mathbf{c}_1, \mathbf{c}_2, \mathbf{q}, \mathbf{p}_{E_1}$ or $\mathbf{p}_{E_2}$ and the previously mentioned SPM direction vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. In the following, we describe the implementation of algorithm 3.2 using Spheres, AABBs and OBBs (Figure 4). In all cases, the centroids of the objects are set as the points $\mathbf{c}_1$ and $\mathbf{c}_2$. Also, the points $\mathbf{p}_{E_1}$ or $\mathbf{p}_{E_2}$ can be directly retrieved if we store each SP as a point-vector pair $(\mathbf{p}_{SP}, \mathbf{n}_{SP})$ such that $\mathbf{p}_{SP} \in R$.



(a) (b) (c)

**Figure 4:** *3D example of the proposed algorithm using different Bounding Volumes: (a) Spheres, (b) AABBs and (c) OBBs. The shown planes are generated from a discretized face-based SPM. In all examples, even though the BVs intersect, the proposed method rejects the possibly colliding pair at the first level of the BVH.*

#### 4.1.1. Bounding Spheres

Let $T_1(\mathbf{c}_1, r_1)$, $T_2(\mathbf{c}_2, r_2)$ (where $\mathbf{c}_i$ the center and $r_i$ the radius) denote the bounding spheres of $O_1$ and $O_2$ respectively. We set $\mathbf{v}_1 = \mathbf{d}$ and $\mathbf{v}_2 = -\mathbf{d}$ where $\mathbf{d} = \mathbf{c}_2 - \mathbf{c}_1$. The simple motivation behind this choice is that for a pair of spherical objects the separating axis is always along the direction of $\mathbf{d}$. The point $\mathbf{q}$ can be easily calculated as, e.g., $\mathbf{q} = \mathbf{c}_1 - r_1\mathbf{n}_{E_2}$.

#### 4.1.2. AABBs

In the case of AABBs, we will take advantage of the observation that the axis is affected by the relative position of the common area between the two AABBs. In particular, let $A$ be the intersection AABB between the AABBs $B_1$ and $B_2$ of the two objects. Then $\mathbf{v}_1 = \mathbf{c}_A - \mathbf{c}_{B_1}$ and $\mathbf{v}_2 = \mathbf{c}_A - \mathbf{c}_{B_2}$, where $\mathbf{c}_k$ refers to the center of the AABB $k$. In order to extract the point $\mathbf{q}$ of algorithm 3.2 we exploit the fact that when an AABB intersects with a plane then at least one of its vertices lies on the negative halfspace and so we can select the point $\mathbf{q}$ so that $\mathbf{q} \in \{vert(B_1), vert(B_2)\}$.

#### 4.1.3. OBBs

Although OBBs are boxes like AABBs, the computation of the intersection volume between a pair of OBBs is not as simple as AABBs. To avoid this additional computational overhead, we employ the same approach as with spheres, i.e. $\mathbf{v}_1 = \mathbf{d}$ and $\mathbf{v}_2 = -\mathbf{d}$. As far the rest of the algorithm parameters are concerned, OBBs are implemented similarly to AABBs.

### 4.2. Discretized SPMs

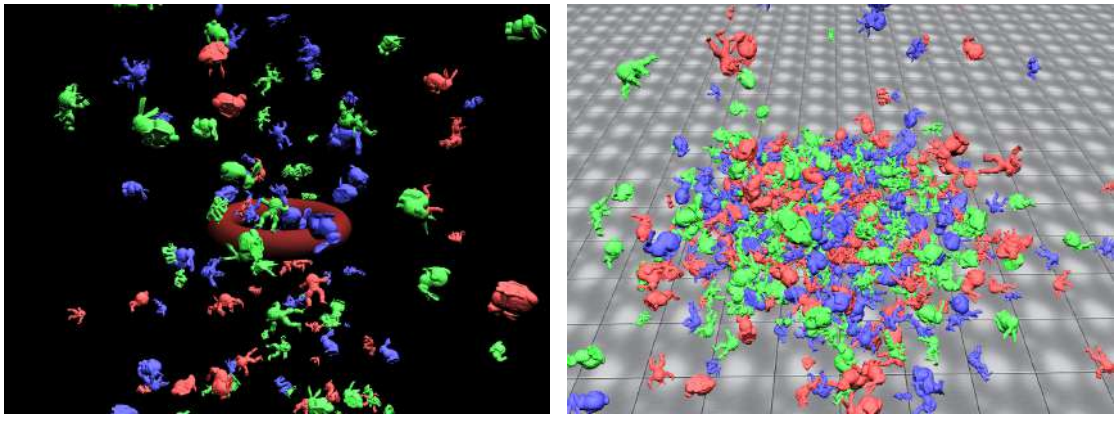
In section 2 we presented two basic types of SPMs, namely the Vertex-based SPM (VSPM) and the Face-based SPM (FSPM). Both VSPM and FSPM of an arbitrary polyhedron require an iterative implementation approach which can be optimized using the Dobkin-Kirkpatrick hierarchy [DK85]. However, because of the conservative nature of the proposed method, we can rely on a less accurate but faster approach for the SPMs. Specifically, we propose the use of Discretized VSPMs and FSPMs by sampling on the angular coordinates $(\phi, \theta)$ of the unit input vector. Therefore, during real-time execution the computational cost of extracting a SP is reduced to retrieving data from a lookup table. It is important to note that the validity of theorem 3.2 is completely independent from the accuracy of the SPM. The only issue influenced by the discretization of the SPM is the culling efficiency of the test and not the correctness of the result.

The discretized version of the SPMs can be interpreted as an implicit BV with fixed directions such as the k-DOPs. However the directions of the surrounding planes depend, in general, on the object and are not extracted from a finite set for all the objects. Furthermore, theorem 3.2 allows us to perform tests between non-parallel planes which is not the case for k-DOPs. Therefore, we can say that discretized SPMs are the equivalent of "object oriented DOPs" of arbitrary degree k.

## 5. Experimental Results

The proposed approach has been implemented as an enhancement of standard BV tests in BVHs of complex rigid objects and has been evaluated in simulations with both 2-objects and multiple objects situations. The results illustrate the usage of SPMs on different number of hierarchy levels and the comparison among different types of SPMs and BVs. We also provide results about the efficiency of the proposed Discretized SPMs.

The proposed method and the experimental setup were implemented using C++ on a Core2 6600 2,4GHz CPU



(a) (b)

**Figure 5:** *Screenshots of the simulations. (a) Torus simulation, (b) Plane Simulation.*

PC with 2GB of RAM and a GeForce 7600 GS Graphics Card. Three different 3D models have been used, namely the armadillo (Figure 6-(a)), the bunny (Figure 6-(b)) and the dragon (Figure 6-(c)). The virtual environment in every simulation contains more than 10 millions faces in total.

(a) (b) (c)

**Figure 6:** *3D Models used in the simulations: (a) Armadillo (8650 vertices, 17296 faces), (b) Bunny (8821 vertices, 17490 faces) and (c) Dragon (8082 vertices, 16380 faces).*

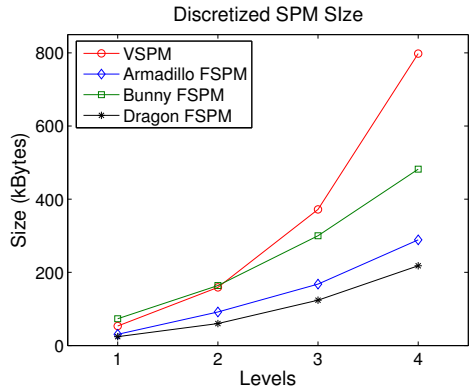



**Figure 8:** *Memory Requirements of the Discretized SPMs for the three models using a* $32 \times 32$ *sampling array. The VSPM size is the same for all models since it depends only on the sampling array.*

### 5.1. Evaluation of the Discretized SPMs

Figure 7 illustrates the bounding efficiency of the discretized SPMs (DVSPM and DFSPM) for different sampling densities in comparison with the efficiency of other bounding volumes. SPMs show a clear advantage over other bounding volumes even for relatively low sampling densities. As the sampling density increases, the SPMs converge to the efficiency of the convex hull, which would be the ideal convex BV. FSPMs converge faster than VSPMs since the used planes are based on the faces of the convex hull. Note also that VSPMs "behave" similar to k-DOPs since the planes are distributed along the directions of the discrete samples, such as the axes of DOPs. For example, the bounding efficiency of a VSPM with sampling density of 6 is equivalent to the efficiency of a 36-DOP.

Based on the results shown in figure 7, the sampling density of 32 was selected to be used throughout the testing. Figure 8 displays the memory requirements of the two discretized SPM types for different levels of the hierarchy using this density. In order to reduce the memory requirements, the Discretized SPM was stored as a set of samples and a set of SPs, so that each sample maps to one of the planes. This approach works very well for the FSPM which is restricted to a finite number of planes by definition (see Section 2). On the other hand, the Discretized VSPM maps to a different plane for each sample so this optimization is not possible. This also means that the size of a Discretized VSPM depends only on the total number of samples and not on the model. Therefore Figure 8 displays the VSPM size for all models.

We have also evaluated the culling efficiency of the discretized SPMs and compared to the standard SPMs, according to the number of levels that contain BVs enhanced with SPMs. Since the proposed algorithm is an enhancement of

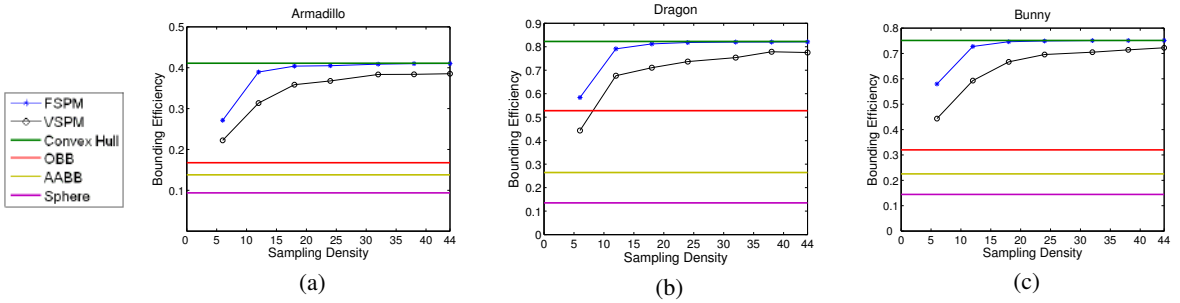



**Figure 7:** *Bounding Efficiency of the BVs and the Discretized SPMs for the armadillo (a), the dragon (b) and the bunny (c). The Bounding Efficiency is defined as the ratio $V_O/V_{BV}$ where $V_i$ denotes the volume of i. For the SPMs the volume $V_{BV}$ is defined as the volume of the finite sub-space $G = \bigcap_i H_{\overline{E_i}}$ for every $E_i \in E_O$. The sampling density corresponds to the number of samples along each angular coefficient $(\phi, \theta)$. The maximum efficiency that we can achieve with a convex bounding volume is equal to the bounding efficiency of the convex hull. In the diagrams, the SPMs display much better efficiency than other bounding volumes, and converge to the efficiency of the convex hull as the sampling density increases.*

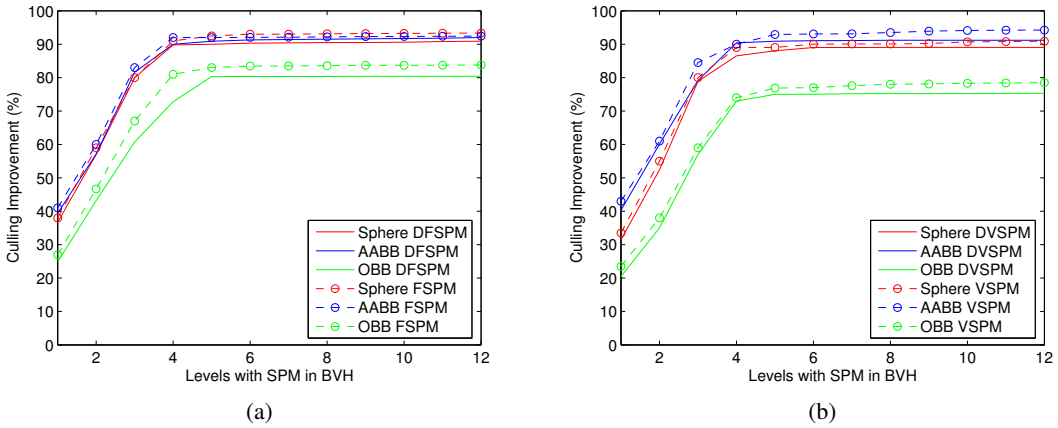



**Figure 9:** *Culling improvement using SPMs for different number of levels in the BVH. The culling improvement is defined as the percentage of the tests between non-colliding objects that the proposed algorithm managed to reject, even though the respective BVs overlapped, i.e. a value of 100 % culling improvement means that all non-colliding pairs being rejected at the given level of the BVH. The reference value of the standard BVHs is 0%. The figure shows that the culling improvement reaches a maximum at 5-6 levels in the BVH which is something expected due to the fact that, for the given models, BVs in level 6 and higher contain small numbers of faces.*

previous methods, we refer here to the *culling improvement*, which is defined as *the percentage of the tests between non-colliding objects that the proposed algorithm managed to reject, even though the respective BVs overlapped.* This means that a value of 100 % culling improvement corresponds to perfect culling, i.e. all non-colliding pairs being rejected at the given level of the BVH. Figure 9 displays the results for the average culling rate improvement of all the simulations that we performed.

The first thing to note in the figure is that the culling improvement reaches a maximum at 5-6 levels in the BVH. This is something expected and is caused due to the fact that BVs in level 6 and higher contain small numbers of faces and therefore the BV fits much tighter to them. In essence, the SPM does not provide much more information about the bounded geometry for the given models at that levels. Even so, it is clear that there is significant culling improvement up to this level, getting very close to 100%. Note also the Discretized versions of the SPMs are practically as efficient as the standard implementations.

### 5.2. Timing Performance

In order to make a comparison based on previous benchmarks, we have conducted a technical experiment where a pair of objects were placed randomly in a bounded space and tested for intersection, similar to the tests described in

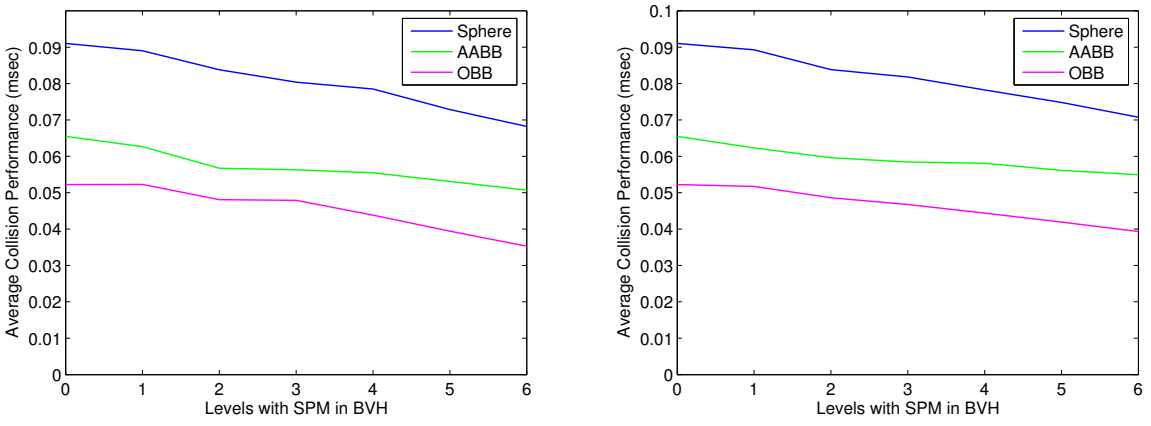



**Figure 10:** *Average collision performance for tests with 2 objects (DVSPM left, DFPSM right) for different number of levels with SPM in the BVH and different types of BVs. The performance is measured as the time needed to detect collisions between the 2 objects. Results for level 7 and higher are omitted for brevity since there is not any significant culling improvement and therefore no more performance gain (see figure 9). The performance for the zero SPM level refers to the standard BVH that does not use the proposed method.*

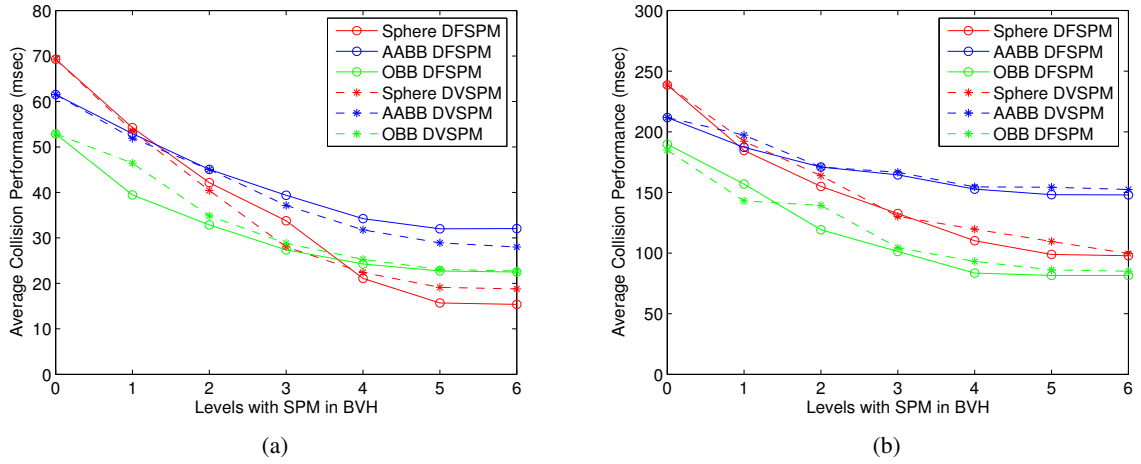



**Figure 11:** *Average collision performance in tests with multiple objects (simulation with the torus left, simulation with plane right) for different number of levels with SPM in the BVH and different types of BVs and SPMs. The performance is measured as the time needed to detect all the collisions in the scene. Results for level 7 and higher are omitted for brevity since there is not any significant culling improvement and therefore no more performance gain (see figure 9). The performance for the zero SPM level refers to the standard BVH that does not use the proposed method. The results show that the proposed method achieves almost an order of magnitude better performance than standard BVs.*

[BO04, van97, Zac02]. In particular, 2,000,000 different position configurations were tested and the collision percentage (i.e. the possibility that the objects intersect for a position configuration) was set to 60%. Figure 10 displays the results for 12 levels of the BVH.

We have also created two simulation setups involving a set of 600 objects falling towards a torus in the first (Figure 5-(a)), and falling on a plane in the second (Figure 5-(b)). In the first simulation the number of collisions is moderate (approximately 14,000 for 14 secs, not including collisions with the static torus) however the objects are in close proximity in every frame as they keep falling together. The second simulation generates a significant number of collisions

(approximately 130,000 for 14 secs not including collisions with the static plane) due to the stacking of the objects on the plane. In both simulations there is a large number of objects which are in close proximity but do not collide, e.g. the first and the third object in a stack of the second simulation.

The results are shown in figure 11 and demonstrate a significant timing performance gain, especially in the case of multi objects simulations. This makes clear that the proposed method has to offer more as a mid-phase in the collision detection pipeline by further reducing the potentially colliding objects that pass from broad phase tests [JTT01, vdB03, Eri05]. This is an important contribution for simulations that the objects have a rather complicated geometry with many concavities, like the ones used in our tests (Figure 6), where simple BVs have very low bounding efficiency (Figure 7). Especially in the torus simulation, where many pairs of objects remain at close proximity but do not collide, the usual BVs continuously fail to cull such pairs and the performance gain is noticeably better than the plane simulation, reaching almost to an order of magnitude.

The results also confirm two major factors that affect the performance. The first is the computational cost of the implementation of the proposed algorithm. For spheres, the cost of the proposed test is much lower (see section 4.1) and so the performance gain is higher than for the other BVs, performing even better than OBBs in the simulation with the torus. The second factor is related to the computational cost of a BV-BV intersection test and the respective gain of the proposed algorithm by reducing the total number of these tests. For example, in the case of OBBs, a BV-BV test has higher computational cost than when AABBs or Spheres are used. Therefore we expect that reducing OBB tests with the proposed method will be more beneficial to the timing performance than reducing the same number of tests between other BVs. The results confirm our analysis, since the performance gain for OBBs is higher relatively to the culling rate, when compared with AABBs or Spheres.

*In all, the proposed method achieves high culling rates and significant timing performance improvement, depending on the algorithm's implementation and the computational cost of the BV – BV tests.*

## 6. Conclusions & Future Work

This paper presented a new approach for employing Support Planes in order to improve the culling efficiency of Bounding Volumes in collision detection queries. The major contribution of the paper is a novel conservative collision rejection theorem that can be used as an alternative to the Separating Axis test for non-parallel axes. We have also presented the concept of Support Plane Mappings which provides a necessary theoretical tool in the implementation of the proposed theorem. The presented method has been integrated into standard Bounding Volume Hierarchies and has been tested in two simulations. Different aspects of the algorithm involving different types of Bounding Volumes and Support Plane Mappings have been demonstrated. The results show that the usage of additional SPM in the first levels of the hierarchy is very beneficient in terms of culling efficiency and timing performance, especially in the case of multi object simulations.

The presented method has shown promising results for further extending theorem 3.2 to the more challenging problems of deformable models and continuous collision detection or even in different fields such as ray tracing. A significant issue that needs to be solved for employing the proposed approach is the generation of SPs, i.e. the extension of SPMs for non rigid models. Our plan is to work on these problems in the near future and examine the possible generalizations of the proposed theorem.

## Appendix A: Proof of Theorem 3.2

The following two lemmas will be used for proving the validity of the theorem (the proofs of the lemmas can be found in the additional material).

**Lemma A.1** Let $L$ be the line of intersection between the planes $E_1$, $E_2$ and $B$ a convex region. If $L$ does not pass through $B$ ($L \cap B = \emptyset$) then the relative position between $E_1$ and $E_2$ does not change inside $B$, i.e. for $i \neq j \in [1,2]$

$$\text{either } \forall \mathbf{x} \in (B \cap E_i) \Rightarrow \mathbf{x} \in {H_{E_j}}^-$$
$$\text{or } \forall \mathbf{x} \in (B \cap E_i) \Rightarrow \mathbf{x} \in {H_{E_j}}^+$$

**Lemma A.2** Let a convex region $B$ and two planes $E_1, E_2$. If the conditions of lemma A.1 apply as $\forall \mathbf{x} \in (B \cap E_1) \Rightarrow \mathbf{x} \in {H_2}^+$ and $\forall \mathbf{x} \in (B \cap E_2) \Rightarrow \mathbf{x} \in {H_1}^+$ then

$$B \cap (H_{E_1}^- \cap H_{E_2}^-) = \emptyset$$

*Proof of Theorem 3.2*

*Proof* We will prove that the result in case (1) is correct as the extension to the other case is trivial. At first, consider the case that $R_1 \cap E_2 \neq \emptyset$ (figure 3-(a)). If the test is positive, then

$$(l'(t_2) - \mathbf{p}_{E_1}) \cdot \mathbf{n}_{E_1} > 0 \Rightarrow l'(t_2) \in H_{E_1}^+ \tag{5}$$

and

$$l'(t_2) \in R_1 \tag{6}$$

since the line segment $(\mathbf{p}_{E_1}, \mathbf{q})$ lies inside the convex $R_1$. Let $\mathbf{u}$ denote the unit vector such that $l'(t) = \mathbf{p}_{E_1} + t\mathbf{u}$ for $t \in \Re$.

$$\mathbf{q} \in H_{E_2}^- \Rightarrow (\mathbf{q} - l'(t_2)) \cdot \mathbf{n}_{E_2} \leq 0 \Rightarrow \mathbf{u} \cdot \mathbf{n}_{E_2} \leq 0$$

Therefore

$$(\mathbf{p}_{E_1} - l'(t_2)) \cdot \mathbf{n}_{E_2} = -t_2 \mathbf{u} \cdot \mathbf{n}_{E_2} \geq 0 \Rightarrow \mathbf{p}_{E_1} \in H_{E_2}^+ \tag{7}$$

From equations 5, 6, 7 and lemma A.1

$$\forall \mathbf{x} \in (R_1 \cap E_1) \Rightarrow \mathbf{x} \in H_{E_2}^+ \text{ and } \forall \mathbf{x} \in (R_1 \cap E_2) \Rightarrow \mathbf{x} \in H_{E_1}^+$$

and so, from lemma, A.2 $R_1 \cap (H_1^- \cap H_2^-) = \emptyset$. However,

$$(R_1 \cap R_2) \subseteq R_1 \Rightarrow (R_1 \cap R_2) \cap (H_{E_1}^- \cap H_{E_2}^-) = \emptyset$$

which means, according to lemma 3.3, that the objects do not collide. In the case that $(R_1 \cap E_2) = \emptyset$, it is easy to prove that all points in $R_1$ lie on the same half-space of $E_2$. If the test is true then

$$(\mathbf{c}_1 - \mathbf{p}_{E_2}) \cdot \mathbf{n}_{E_2} > 0 \Rightarrow \mathbf{c}_1 \in H_{E_2}^+$$

Consequently, $R_1 \in H_{E_2}^+ \Rightarrow R_1 \cap H_{E_2}^- = \emptyset$ and again the objects do not collide. □

## References


[ASC*06] ALBOCHER D., SAREL U., CHOI Y.-K., ELBER G., WANG W.: Efficient continuous collision detection for bounding boxes under rational motion.

[BO04] BRADSHAW G., O'SULLIVAN C.: Adaptive Medial-Axis Approximation for Sphere-tree Construction. *ACM Transactions on Graphics 23*, 1 (2004), 1–26.

[CS08] COMING D., STAADT O.: Velocity-Aligned Discrete Oriented Polytopes for Dynamic Collision Detection. *Visualization and Computer Graphics, IEEE Transactions on 14*, 1 (2008), 1–12.

[CTM08] CURTIS S., TAMSTORF R., MANOCHA D.: Fast Collision Detection for Deformable Models using Representative-Triangles. In *SI3D '08: Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games* (2008), pp. 61–69.

[CW96] CHUNG K., WANG W.: Quick Collision Detection of Polytopes in Virtual Environments. In *ACM Symposium on Virtual Reality Software and Technology 1996* (July 1996), pp. 1–4.

[DK85] DOBKIN D. P., KIRKPATRICK D. G.: A linear algorithm for determining the separation of convex polyhedra. *Journal of Algorithms 6*, 3 (1985), 381 – 392.

[Eri05] ERICSON C.: *Real-Time Collision Detection*. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, 2005.

[GLM96] GOTTSCHALK S., LIN M. C., MANOCHA D.: OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics 30*, Annual Conference Series (1996), 171–180.

[HEV*04] HADAP S., EBERLE D., VOLINO P., LIN M. C., REDON S., ERICSON C.: Collision Detection and Proximity Queries. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes* (2004), ACM, p. 15.

[Hub96] HUBBARD P. M.: Approximating Polyhedra with Spheres for Time-Critical Collision Detection. *ACM Transactions on Graphics 15*, 3 (1996), 179–210.

[JP04] JAMES D. L., PAI D. K.: BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models. *SIGGRAPH (ACM Transactions on Graphics) 23* (2004), 393–398.

[JTT01] JIMÉNEZ P., THOMAS F., TORRAS C.: 3D Collision Detection: A Survey. *Computers and Graphics 25*, 2 (Apr. 2001), 269–285.

[KGS98] KIM D.-J., GUIBAS L., SHIN S.-Y.: Fast collision detection among multiple moving spheres. *Visualization and Computer Graphics, IEEE Transactions on 4*, 3 (1998), 230–242.

[KHM*98] KLOSOWSKI J. T., HELD M., MITCHELL J. S. B., SOWIZRAL H., ZIKAN K.: Efficient Collision Detection using Bounding Volume Hierarchies of *k*-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (1998), 21–36.

[LC98] LI T.-Y., CHEN J.-S.: Incremental 3D Collision Detection with Hierarchical Data Structures. In *VRST '98: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (1998), pp. 139–144.

[OCSG07] OTADUY M., CHASSOT O., STEINEMANN D., GROSS M.: Balanced Hierarchies for Collision Detection between Fracturing Objects. In *Virtual Reality Conference, 2007. VR '07. IEEE* (2007), pp. 83–90.

[SGG*06] SUD A., GOVINDARAJU N., GAYLE R., KABUL I., MANOCHA D.: Fast Proximity Computation among Deformable Models using Discrete Voronoi Diagrams. *ACM Transactions on Graphics 25*, 3 (2006), 1144–1153.

[SSIF09] SELLE A., SU J., IRVING G., FEDKIW R.: Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics 15*, 2 (2009), 339–350.

[TCYM08] TANG M., CURTIS S., YOON S.-E., MANOCHA D.: Interactive Continuous Collision Detection between Deformable Models using Connectivity-based Culling. In *SPM '08: Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (2008), pp. 25–36.

[TJ08] TWIGG C. D., JAMES D. L.: Backward steps in rigid body simulation. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (2008), pp. 1–10.

[TKZ*04] TESCHNER M., KIMMERLE S., ZACHMANN G., HEIDELBERGER B., RAGHUPATHI L., FUHRMANN A., CANI M.-P., FAURE F., MAGNETAT-THALMANN N., STRASSER W.: Collision Detection for Deformable Objects. In *Eurographics State-of-the-Art Report (EG-STAR)* (2004), pp. 119–139.

[van97] VAN DEN BERGEN G.: Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools: JGT 2*, 4 (1997), 1–14.

[vdB03] VAN DEN BERGEN G.: *Collision Detection in Interactive 3D Environments*. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, 2003.

[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. *ACM Transactions on Graphics 26*, 1 (2007).

[Zac98] ZACHMANN G.: Rapid collision detection by dynamically aligned dop-trees. In *VRAIS '98: Proceedings of the Virtual Reality Annual International Symposium* (1998).

[Zac02] ZACHMANN G.: Minimal hierarchical collision detection. In *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology* (2002), pp. 121–128.

[ZL03] ZACHMANN G., LANGETEPE E.: Geometric Data Structures for Computer Graphics. In *ACM SIGGRAPH*. ACM Transactions of Graphics, 27–31 2003.

[ZRLK07] ZHANG X., REDON S., LEE M., KIM Y. J.: Continuous Collision Detection for Articulated Models using Taylor Models and Temporal Culling. *ACM Trans. Graph. 26*, 3 (2007), 15.