Fast Collision Detection for Deformable Models using Representative-Triangles

Sean Curtis[1] Rasmus Tamstorf[2] Dinesh Manocha[1]

[1]University of North Carolina at Chapel Hill

[2]Walt Disney Animation Studios

http://gamma.cs.unc.edu/RTRI

Figure 1: *Interactive collision detection on a complex, deformable benchmark: an animated sequence of a flamenco dancer with 26 K vertices, 75 K edges and 50K triangles, consisting of 350 frames. We observed up to a 15X reduction in elementary tests and a 4.9X increase in speed on this benchmark. Our new method calculates all self-collisions and inter-object collisions in 200 ms per frame.*

Abstract

We present a new approach to accelerate collision detection for deformable models. Our formulation applies to all triangulated models and significantly reduces the number of elementary tests between features of the mesh, i.e., vertices, edges and faces. We introduce the notion of Representative-Triangles, standard geometric triangles augmented with mesh feature information and use this representation to achieve better collision query performance. The resulting approach can be combined with bounding volume hierarchies and works well for both inter-object and self-collision detection. We demonstrate the benefit of Representative-Triangles on continuous collision detection for cloth simulation and N-body collision scenarios. We observe up to a one-order of magnitude reduction in feature-pair tests and up to a 5X improvement in query time.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types;

Keywords: deformable collisions, continuous collision detection, collision detection, self-collision, bounding-volume hierarhcy

1 Introduction

Collision detection is a ubiquitous task in physical simulation and motion planning algorithms. Broadly, there are two basic types of collision queries: discrete and continuous. Discrete collision detection (CD) tests the mesh configuration at specific time instances. On the other hand, continuous collision detection (CCD) determines if there are any collisions in the interval bounded by two times. The simplest formulation of CCD performs linear interpolation on the vertex positions and determines if corresponding swept triangle features collide during the time interval.

A significant fraction of the collision query time for both types of queries is spent in performing exact intersection tests between the primitives. Each triangle consists of three types of features: vertices, edges and faces. For discrete CD, the exact tests can be reduced to six elementary tests between the edges of one triangle with the face of the other [Tropp et al. 2006]. Each test requires solving a linear equation. Equivalently, for CCD, 15 elementary tests need to be performed. Each entails finding the roots of a cubic polynomial equation [Provot 1997].

In order to accelerate collision queries, many techniques have been proposed that cull away many of the $O(n^2)$ triangle-pairs which don't collide. These include methods based on bounding volume hierarchies (BVHs), built upon the mesh's triangles. The performance of these techniques is governed by their culling efficiency. Poor culling efficiency can produce a high number of candidate triangle pairs, most of which are false positives. In particular, current triangle-based approaches, i.e., techniques that use triangles as the fundamental primitive, may not be able to perform CCD queries at interactive rates on deformable meshes composed of tens of thousands of triangles [Hutter and Fuhrmann 2007; Govindaraju et al. 2005; Tang et al. 2007; Wong 2005].

Main Contributions: We present a novel approach to improve the performance of collision queries on triangulated models. Our approach is applicable to discrete as well as continuous collision detection. The main idea behind our approach is the use of feature-based hierarchies and representations, as opposed to triangle-based hierarchies. In a feature-based hierarchy, the individual features are explicitly stored at the leaf nodes of a BVH. We show that feature-based hierarchies improve performance by:

1. Eliminating duplicate elementary tests that naturally arise in triangle-based formulations.
2. Providing higher culling efficiency.

We introduce the concept of *Representative-Triangles* (R-Triangles) in order to obtain the benefits of feature-based hierarchies by using a single BVH. An R-Triangle is a standard triangle, augmented with mesh feature information. We present a simple algorithm to compute R-Triangles for a mesh and utilize them to eliminate duplicate elementary tests and to improve culling efficiency. The use of R-Triangles is orthogonal to the type of acceleration structure. They can be combined with BVHs of any bounding volume type or any other technique which operates on triangle-pairs.

We use R-Triangles to improve the performance of continuous colli-

sion detection in deformable models and N-body collisions by combining them with a BVH of axis-aligned bounding boxes (AABB). We have applied our algorithm to cloth simulation and N-body collision benchmarks with tens of thousands of triangles and have reduced elementary tests 10-30X and improved query time up to 5X.

Oranization: The rest of the paper is organized in the following manner. We briefly survey prior work on collision detection and introduce the terminology for this paper in Section 2. Section 3 highlights the benefits of feature-based hierarchies for collision detection. We introduce R-Triangles in Section 4 and present algorithms to compute them efficiently. We highlight the performance of our approach in Section 5, analyze the performance in Section 6 and discuss future work in section 7.

2 Related Work and Background

The problem of collision detection has been extensively studied in various fields including computer graphics, robotics, computational geometry and simulation. Good recent surveys on different algorithms are given in [Ericson 2004; Lin and Manocha 2003; Teschner et al. 2005]. In this section, we give a brief overview of hierarchical methods, feature-based algorithms and continuous collision detection.

2.1 Bounding Volume Hierarchies

BVHs are commonly used to accelerate collision queries. These include hierarchies based on simple BVs such as spheres [Hubbard 1993; Bradshaw and O'Sullivan 2004] or AABBs [van den Bergen 1997]. Other hierarchies use tight fitting BVs such as oriented bounding boxes [Gottschalk et al. 1996], discretely oriented polytopes (k-DOPs) [Klosowski et al. 1998], or a hybrid combination of BVs [Sanna and Milani 2004]. There is a trade-off between simple BVs and tight-fitting BVs. In particular, the simple BVs have a lower storage overhead and a fast overlap test, but may result in poor culling efficiency. On the other hand, tight-fitting BVHs have a higher culling efficiency, but are more costly to use.

In the case of rigid models, the BVHs are computed once, before simulation. For deformable models, the BVHs are recomputed or updated for each frame. These include simple update algorithms to maintain high culling efficiency based on linear-time refitting algorithms [Larsson and Akenine-Möller 2006; Zachmann and Weller 2006], or selective restructuring [Otaduy et al. 2007; Yoon et al. 2007]. Other approaches to improve the culling efficiency of BVH-based algorithms use normal cones for self-collisions [Provot 1997] or GPU-based accelerations [Govindaraju et al. 2005; Sud et al. 2006]. R-Triangles can be combined with all of these acceleration techniques.

2.2 Feature-Based Collision Detection

Most prior acceleration algorithms seek to find potentially colliding triangle pairs; we classify them as triangle-based methods. Many other collision detection algorithms directly utilize vertex, edge and face features [Ericson 2004] and we refer to them as feature-based algorithms. These include efficient algorithms based on external Voronoi regions of the features of convex polytopes [Lin and Canny 1991; Mirtich 1998] and its extension to non-convex models [Ehmann and Lin 2001]. In practice, they are mainly limited to rigid models as recomputing the Voronoi regions for deformable models can be expensive. [Hutter and Fuhrmann 2007] recently proposed using BVs on features for deformable models. We compare our algorithm with their formulation in Sec. 6.

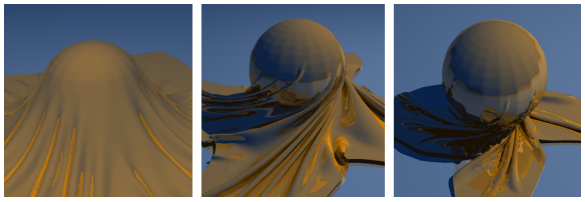

Figure 2: *A cloth simulation with 20 K vertices, 60 K edges and 40 K triangles consisting of 1044 frames. We observed up to a 28X reduction in elementary tests and a 5.1X increase in speed on this benchmark.*

2.3 Continuous Collision Detection

The problem of interactive CCD between complex deformable models is quite challenging. The performance of these algorithms can be accelerated based on continuous normal cone tests [Tang et al. 2007] or GPU-based accelerations [Govindaraju et al. 2005]. Many approaches have also been proposed to reduce the number of elementary tests by taking into account adjacency information [Govindaraju et al. 2005; Wong 2005; Hutter and Fuhrmann 2007; Tang et al. 2007] and randomized marking schemes [Wong and Baciu 2006]. We compare our algorithm with some of these approaches in Section 6.

2.4 Notation and Terminology

Throughout this paper we use the following terms and notation:

- A *feature* is one of the fundamental geometric components of a mesh used in the elementary collision tests–a vertex, an edge or a face.
- A *triangle* is a data structure which includes knowledge of all the features that make up the structure of the triangle. A *face* is the mesh feature used in elementary tests.
- M is a mesh with sets of vertices, edges and faces (V, E and F respectively.)
- Two features are *incident* if either feature includes the other in its construction. Two edges with a shared vertex are incident. A face is incident to all of its vertices. A face is trivially incident to itself.
- A *contact* is a collision between feature pairs–vertex-face (VF) and edge-edge (EE) for CCD and edge-face for CD. In particular, the CCD test for two triangles requires finding the roots of a polynomial for six VF and nine EE pairs.
- A *regular* triangular mesh is a closed-manifold mesh of triangles where all vertices have degree six.

3 Feature-based hierarchies

In this section, we introduce feature-based hierarchies and show that they can considerably improve the performance of collision detection algorithms. We describe a set of feature-based BVHs to improve culling efficiency and eliminate duplicate elementary tests.

Culling Efficiency: We wish to cull triangle pairs that are *clearly* not overlapping. Culling efficiency is measured in terms of the number of false positives – triangle pairs that are tested, but don't actually intersect. Ideal culling efficiency would result in no false positives.

Duplicate Elementary Tests: Fig. 3 shows a single vertex, v_1, incident to six triangles. v_1 makes contact with triangle t_1. The culling algorithm will produce a triangle pair with t_1 and every triangle incident to v_1. In a straightforward implementation, the VF test, VF(v_1, t_1), will be performed six times, each time producing the same result. Also note that all of the EE tests involving the *edges* incident to v_1 will be performed twice, because each edge is shared by two triangles. We wish to cheaply eliminate all such duplicate elementary tests.

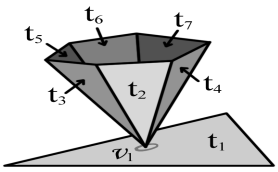

Figure 3: *The elementary test between vertex v_1 and the face of triangle t_1 would naturally be dispatched six times.*

3.1 Feature-based hierarchies

The problem of duplicate queries exists because culling is performed on triangles, but exact tests depend on features, most of which are shared by multiple triangles. Eliminating duplicate queries also mitigates some of the impact of poor culling efficiency. However, we also want to *directly* improve the culling efficiency. The volume a feature occupies can be much smaller than that of its triangle. We can improve culling efficiency by using BVs of the features to cull away non-overlapping features.

Feature-based hierarchies are a set of independent BVHs; one BVH built on each feature type used in the exact elementary collision tests. In order to perform CCD we would require a vertex BVH (V-BVH), edge BVH (E-BVH) and face BVH (F-BVH). To find the contacts we would test for collisions between V-BVH and F-BVH and for self-collisions within the E-BVH.

This formulation addresses both issues. First, the separate hierarchies improve culling efficiency by performing culling on the features' BVs. In Fig. 4(a), the two triangles' BVs overlap. Based on this information we would perform nine EE tests. In Fig. 4(b), we see that there are only two pairs of overlapping edge bounding volumes. This tells us that only two EE tests are necessary. This finer granularity of culling leads to fewer elementary tests and fewer false positives.

Feature-based hierarchies also solve the problem of query duplication. The duplications illustrated by the example in Fig. 3 couldn't happen because each vertex is represented in the vertex hierarchy only once. Candidate feature pairs from the traversal of the two hierarchies would be completely unique.

3.1.1 Issues

Although simple in principle and implementation, there is overhead inherent in using multiple hierarchies. In practice, separate feature-based hierarchies can take 4-7X more space than a single, triangle-based BVH. Moreover, each hierarchy requires full maintenance work: updating, refitting, restructuring, etc. At each time step, collision detection would begin with the traversal of two or more sets of hierarchies. As a result, the cost can be high.

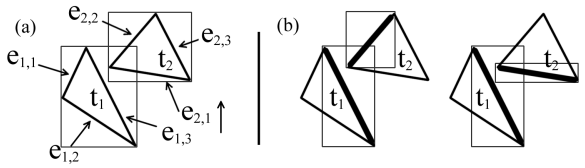

Figure 4: *Fig. (a) shows two triangles', t_1 and t_2, overlapping BVs. With only this knowledge, we would need to perform nine EE tests: $(e_{1,i}, e_{2,j}), i, j = 1...3$, Fig. (b) shows the two pairs of edge BVs that actually overlap. The edge BVs reduce the number of tests to two pairs: $(e_{1,3}, e_{2,2})$ and $(e_{1,3}, e_{2,1})$*

4 Representative-Triangles

We would like to have all the benefits of feature-based hierarchies, namely, automatic elimination of duplicate queries without run-time bookkeeping and improved culling efficiency, but with the cost of a single hierarchy.

Representative-Triangles (R-Triangles) make that possible. An R-Triangle is an augmented triangle. In addition to the basic structural data a triangle carries, an R-Triangle also carries feature assignments and feature bounding volumes. The assignment allows an R-Triangle to represent some subset of its features. This knowledge is used while processing candidate triangle pairs to uniquely dispatch elementary tests. An R-Triangle also carries bounding volume information for the features that it represents. These feature BVs are used to improve culling efficiency. To create valid R-Triangles from a mesh, M, of triangles, we assign each feature of the mesh to a triangle, maintaining the following *properties*:

I Every feature $\lambda \in \{V \cup E\}$, must be represented by a triangle.

II Every feature $\lambda \in \{V \cup E\}$, can be assigned to no more than one triangle.

III If a feature $\lambda \in \{V \cup E\}$ is assigned to triangle t, then t must be incident to λ.

Triangles implicitly represent their own faces. These properties, taken together, insure that every feature is represented by a single, incident triangle.

There are multiple ways to assign features to R-Triangles that satisfy the three properties. Meeting these requirements eliminates the possibility of performing the same elementary test more than once. We call the relationship between a feature and its R-Triangle a *representation assignment*. A *representation assignment schema*, or simply *assignment schema* determines which feature is assigned to which triangle.

The R-Triangles are used in place of the standard triangles in the preferred triangle-based acceleration structure. The acceleration structure follows its normal execution to find potentially colliding triangle pairs. Assignments can be varied across the mesh. Not all R-Triangles will necessarily have symmetric sets of features assigned to them. The set of assigned features can be any combination of the triangle's six features (i.e., three vertices and three edges.)

4.1 Improved Culling Efficiency

We use the feature bounding volumes in the R-Triangle to improve overall culling efficiency. This exactly replicates the functionality of the feature-based hierarchy; the lowest level of culling is on feature BVs. We limit the set of feature-BV overlap tests actually performed for a pair of triangles based on the represented features of that pair. We determine if the R-Triangles represent compatible feature pairs. A candidate triangle pair represents *compatible feature pairs* if it represents features which correspond to an elementary test, e.g., an edge represented by each triangle in the candidate pair would lead to an EE test in CCD. We perform overlap tests on the compatible feature pairs before dispatching the corresponding elementary test in Algorithm 1. Linking the feature BVs to their R-Triangles insures that we won't do any duplicate BV-overlap tests any more than we would do duplicate elementary tests for the reasons given in the next section.

4.2 Eliminating Duplicate Queries

R-Triangles eliminate duplicate queries using a simple idea: for each compatible feature pair represented by the triangle pair, we dispatch the corresponding elementary test. The details of this algorithm are given in Sec. 5 as Algorithm 1.

Theorem 4.1 *For a pair of colliding features in contact, Algorithm 1 guarantees that exactly one elementary test on those features will be dispatched.*

Proof We will prove this by contradiction. Let us assume there is a contact, C between features λ_1 and λ_2 of model M for which the corresponding elementary test is either never performed or performed multiple times.

First we address the possibility of never performing the elementary test. We know that each feature must be assigned to an R-Triangle because of Property I. Let t_1 and t_2 be the R-Triangles of λ_1 and λ_2, respectively. We also know that t_i is in the set of triangles incident to $\lambda_i, i = [1, 2]$ because of Property III. We assume that if two features intersect, the culling algorithm's candidate triangle pairs include the pairs in the product set of the triangles incident to each feature exactly once. So, we know that the pair (t_1, t_2) must be included in the candidate triangle pairs. When Algorithm 1 is passed this pair, it will note that there is at least one compatible feature pair (λ_1, λ_2) and dispatch the corresponding elementary test. We have our first contradiction – at least one elementary test is dispatched for the contact C.

Second, we address the possibility of the elementary test being dispatched more than once. We know by Property II that λ_i can only be represented by $t_i, i = [1, 2]$. That means that no other triangles in the mesh can participate in dispatching the elementary test for the feature pair (λ_1, λ_2). As previously indicated, the triangle-pair culling method produces the candidate triangle pair (t_1, t_2) only once and no other pair can dispatch that elementary test. We have the second contradiction – the elementary test is dispatched only once. QED.

4.3 Optimal Representation

Any assignment schema which satisfies the given properties eliminates duplicate elementary tests. An *optimal* representation assignment would dispatch the absolute minimum number of elementary tests to find all contacts.

Fig. 5 shows two cases of optimal representative assignment. In Fig. 5(a), the two bold-faced edges intersect, so, the culling algorithm will produce four triangle candidate pairs: $(t_1, t_3), (t_1, t_4), (t_2, t_3)$ and (t_2, t_4). These four pairs are processed as follows:

- (t_1, t_4) - t_1 represents no features. t_4, likewise, represents no features. No elementary tasks are dispatched.
- (t_1, t_3) and (t_2, t_4) - One triangle represents no features. One triangle represents one edge. There are no compatible feature pairs. So, no elementary tasks are dispatched.
- (t_2, t_3) - t_2 represents one edge. t_3 also represents one edge. There is one compatible feature pair. One EE test is dispatched.

For two arbitrary triangles, we would ordinarily have to perform 15 elementary tests. In this case, only one of them produces a collision; there are 14 unneeded tests. However, with this representation assignment, a single elementary test is performed. A single contact is found. This is an optimal representative assignment for this scenario.

Similarly, Fig. 5(b) shows an ideal assignment for a vertex-face contact. The culling algorithm produces six candidate triangle pairs, but for reasons similar to those above, only a single VF elementary test is dispatched. Again, this is an optimal assignment for this contact.

The representative assignments in Fig. 5 are optimal for one isolated contact. We avoided performing additional elementary tests by assigning the non-colliding features as far away from the contact as possible. That distance, however, is constrained; the feature *must* be assigned to another incident triangle. If the incident triangle is also involved in a collision, or if, due to culling inefficiency, the culling algorithm includes that triangle in one or more candidate pairs, the tests we avoided in examining the scenarios in Fig. 5 will then be dispatched.

So, although there may exist an optimal representation assignment, it is dependent on the configuration of the mesh and the characteristics of the culling algorithm. For arbitrary collision scenarios, this can't be known *a priori*.

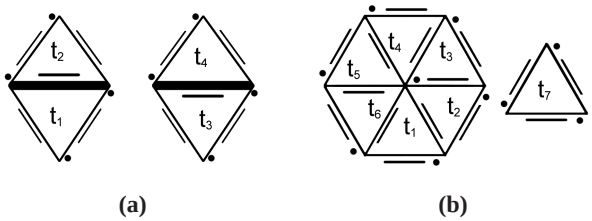

Figure 5: *Fig. (a) shows an optimal representative assignment for an EE contact. Fig. (b) shows the same for a VF contact. Representation is indicated in the following manner: a line close to and parallel with an edge lies inside its R-Triangle. Similarly, a dot near a vertex lies inside the vertex's R-Triangle. The contact is between the two thick edges in (a) and the vertex in the center of the fan on the left (represented by t_3) with t_7 in (b).*

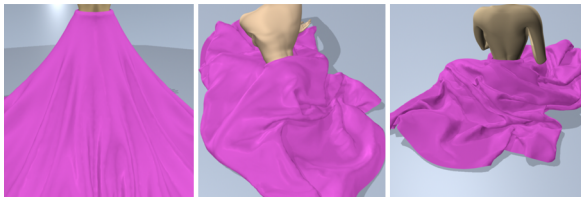

Figure 6: *A cloth simulation with 47 K vertices, 139 K edges and 92 K triangles consisting of 464 frames. We observed up to a 12X reduction in elementary tests and a 4.98X increase in speed on this benchmark using R-Triangles.*

4.3.1 Feasibility of Optimal Representation

Our approach makes no assumptions about the nature of the mesh's deformation. At the same time, we want an assignment schema that will be as close to optimal as possible over the entire sequence. We will show that such a schema does not exist and we are free to adopt any convenient assignment schema.

Wong and Baciu [2006] indicate that the ideal schema would be to assign the features to the smallest set of triangles possible. Or, conversely, it would result in the maximum number of triangles with no assigned features. While they don't provide supporting arguments, their suggestion has certain a intuitive appeal. Concentrating feature representation in a small subset of the triangles *feels* like it would aid performance.

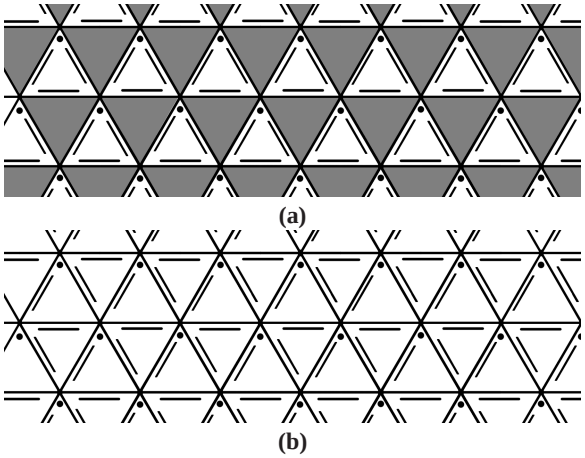

Figure 7: *A sample regular mesh with two different representative assignment schemes. (a) shows the Maximal Schema (unassigned triangles are shaded grey for emphasis.) (b) shows the Uniform Schema.*

Fig. 7 shows two different assignment schema for a regular mesh. Fig. 7(a) shows the result of applying the schema in which we maximize the number of unassigned triangles. We'll refer to this schema as the *Maximal Schema*. In this particular case, the size of the set of unassigned triangles is $\frac{|F|}{2}$. This is possible because of the regularity of the mesh; we can efficiently alternate between assigned and unassigned triangles. If we were to add one more triangle to the unassigned set, we would have two adjacent, unassigned triangles – leading to an edge, their shared edge, being unassigned. So, for a regular mesh, the size of the set of unassigned triangles cannot exceed $\frac{|F|}{2}$. If the mesh has any vertices with odd degree or the mesh is open, the maximum for that set is strictly less. So, a regular mesh would have the largest unassigned set (relative to $|F|$).

Fig. 7(b) shows an alternative schema; every triangle is an R-Triangle with the same number of assigned features. We'll refer to this schema as the *Uniform Schema*. If we re-examine the scenarios from Fig. 5 with the schema from Fig. 7 we get the assignments shown in Fig. 8. Evaluating these scenarios would yield the results shown in Table 1.

For an EE contact the total numbers of elementary tests that are performed for each assignment schema are equal. For the VF contact, there is a marginal difference. The Maximal Schema has the best possible performance when the vertex contacts an unassigned triangle, with only three elementary tests dispatched. However, in the worst case, when the vertex contacts an assigned triangle, it performs 36 tests. In this case, there is an even distribution of unassigned triangles and assigned triangles. Thus, the best-case and worst-case

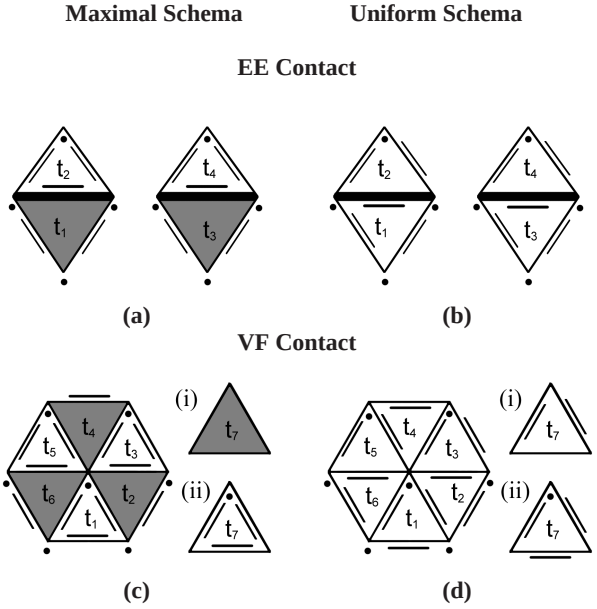

Figure 8: *The two scenarios from Fig. 5 with two assignment schema from Fig. 7 applied. Note that in the VF tests it was necessary to consider both types of faces for the contact face.*

Fig. Scenario	Number of VF Tests	Number of EE Tests	Total Elementary Tests	Total Contacts
Fig. 8a	4	9	13	1
Fig. 8b	4	9	13	1
Fig. 8c.i	3	0	3	1
Fig. 8c.ii	9	27	36	1
Fig. 8d.i	9	9	18	1
Fig. 8d.ii	9	9	18	1

Table 1: *The number of elementary tests dispatched for a single contact based on the two assignment schema illustrated in Fig. 7.*

collisions have equal probability of occurring and the expected cost of using this schema would be 19.5 elementary tests. The Uniform Schema has a consistent cost of 18 elementary tests for both types of vertex-face contacts.

We draw several conclusions from this analysis. First, although there may exist a locally optimal representative assignment, there is no globally optimal assignment schema because optimality depends on the actual contacts. Second, the efficacy of a locally optimal assignment is quickly diffused by even mediocre culling efficiency, i.e., the value of a locally optimal assignment is only realized if none of the surrounding triangles appear in triangle candidate pairs; this is an unlikely outcome for a system without perfect culling efficiency. Third, taken as a reasonable sample set, these two schemas indicate that different global assignment schemas are functionally equivalent. We conclude that there is no globally optimal assignment schema and we can select an assignment schema based on alternative criteria.

4.3.2 Representation Assignment Schema

We favor the simplest and most efficient means of assigning representatives possible. We use a greedy algorithm, which simply scans through the triangles one at a time. For each triangle, it determines which of its features haven't been assigned yet and assigns them to

that R-Triangle. With this schema, there will be some triangles with six features assigned, some with none and the vast majority with between one and three assigned features.

Additionally, in simulation systems where the mesh might undergo topological changes, such as tearing or fracturing, we want to update representative assignments in an efficient fashion. This greedy algorithm is compatible with scenarios where meshes undergo topological changes. When a mesh is fractured or torn, new triangles may be introduced, old triangles may be removed and the boundary of the mesh may change. For each deleted triangle we would do the following:

- Delete unsupported features, (i.e., features that are not used by the remaining triangles.)
- Assign the supported features whose R-Triangle was deleted to one of the remaining incident triangles.

For new triangles, we simply provide the newly created triangles as the set of triangles to iterate over.

5 Implementation and Results

5.1 Implementation Details

We implemented the system described here in C/C++. The BVH simply partitions triangles across the mid-point of the longest axis of a BV. The tests were run on an Intel Xeon 3 Ghz machine with 3 GBytes of RAM running 32-bit Windows XP.

Representation Encoding: We encode representation assignment in a four-bit mask. The first two bits indicate the number of vertices an R-Triangle represents, the last two, the number of edges. To make this compact representation work, we re-order the features so that if triangle t represents n features, they are the first n features in its ordered feature list. This re-ordering is benign because it merely changes the local indexing of the features from the specific triangle's perspective, but the mesh remains unchanged.

Four-bit encoding means that the representation data can be placed directly into the triangle identifier with minimum impact on the size of the identifier space. The single greatest advantage of placing this information into the identifier is that all of the information required to decide if two triangles have compatible features is immediately available; no extra fetches to memory are required.

We use this simple encoding to easily determine if two triangles have compatible feature pairs. We have at least one compatible feature pair if either triangle represents at least one vertex or both represent at least one edge.

Processing Candidate Triangle Pairs: We observe that if a triangle pair has no compatible feature pairs, then it is unnecessary to even test if the triangles' BVs overlap. To exploit this fact, we push the R-Triangle functionality into the culling algorithm's code which processes pairs of leaf nodes. The code is shown in Algorithm 1

Element BV Type: In our implementation we chose AABBs for the BVs of the swept faces and edges. However, any type of BV would serve the purpose.

Edges in CD and swept vertices in CCD are simple line segments. Instead of creating poorly fitting bounding volumes on these line segments, we simply perform an intersection test between the line segment and the bounding volume of the triangle. Table 2 illustrates the relative culling efficiency of using line-BV intersection tests over BV-BV intersection tests. Line segment-BV tests improved culling over BV-BV tests by 8-34%.

Algorithm 1: *Using R-Triangles and feature bounding volumes to process candidate triangle pairs*

Algorithm: processLeafPair(Node $node1$, Node $node2$)

```
/* Determines if the two triangles represent compatible feature pairs */
if HasCompatible(node1.tri, node2.tri) then
    if Overlaps(node1, node2) then
        /* Elementary test construction */
        foreach Vert v represented by node1.tri do
            if Overlaps( getBound(v), node2 ) then
                testVF( v, node2.tri)
            end
        end
        foreach Vert v represented by node2.tri do
            if Overlaps( getBound(v), node1 ) then
                testVF( v, node1.tri)
            end
        end
        foreach Edge e1 represented by node1 do
            foreach Edge e2 represented by node2 do
                if Overlaps( getBound(e1), getBound(e2) ) then
                    testEE( e1, e2)
                end
            end
        end
    end
end
```

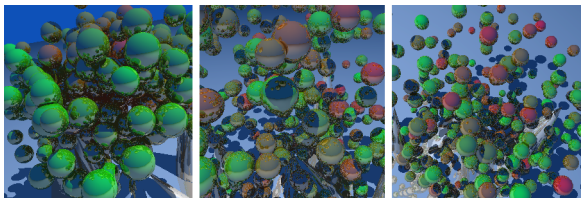

Figure 9: *An n-body simulation with 18 K vertices, 51 K edges and 34 K triangles consisting of 374 frames. We observed up to a 9X reduction in elementary tests and a 5.3X increase in speed on this benchmark.*

Memory Requirements: As already indicated, representation assignment incurs no additional memory costs on the collision detection system. However, storing feature BVs does. For typical simulation meshes, $|V| + |E| \approx 3|F|$. There are already roughly $2|F|$ bounding volumes in a binary BVH. So, storing feature BVs increase the storage requirements of the BVH by roughly 2.5 times. Because we use line segment-BV intersection tests for VF pairs, we eliminate the need to store BVs for swept vertices, so, in practice, the memory overhead only increases by the amount needed to store $|E|$ additional BVs. For our benchmarks, storing edge BVs increased the memory overhead by 75% over a typical triangle-based BVH.

If memory is particularly tight, edge BVs can be computed on demand and then discarded. This has the added benefit of eliminating the need to update all feature BVs at each time step. The cost wouldn't be particularly onerous. Creating the BV for a swept edge is an operation on only four vertices.

Obviously, the cost of dynamically instantiating edge BVs increases with both the complexity of the BV type and with the number of tests in which a particular edge is involved. In choosing this route, this trade-off should be carefully weighed. For multi-layered cloth, or any simulation where many features are in close proximity, the likelihood of a feature being used in multiple tests is high and could justify the memory cost to maintain a persistent copy of the edge BV.

5.2 Results

Benchmarks: We tested our system on four benchmarks:

1. N-body Balls. Hundreds of balls colliding in a small space – Fig. 9
2. Cloth-ball: A cloth drapes over a rotating ball – Fig. 2
3. Princess: A woman in a dress sits on the ground – Fig. 6
4. Flamenco: A flamenco dancer – Fig. 1

Performance Comparison: We compare our algorithm against three other algorithms. All three algorithms use the exact same BVH data structure and maintenance schemes. They only differ in how they process leaf-node pairs.

The first algorithm is a basic, straightforward application. It performs all 15 elementary tests for every leaf-node pair it produces (except for tests between incident features.) It is noted as "BASIC" in the data.

The second algorithm uses a derivative of the approach described in [Tang et al. 2007]. It processes the non-adjacent triangle pairs first and then uses the results of the non-adjacent stage to cull the tests between adjacent pairs. This algorithm is noted as "ADJ" in the data. This algorithm uses a run-time database to eliminate duplicate queries and feature BVs similar to the techniques in [Hutter and Fuhrmann 2007].

The last algorithm uses the concept of representation for duplicate queries but does not use feature BVs to perform extra culling. This comparison illustrates the impact of the feature BVs in performance. It is noted as "NO-DUPL" in the data (i.e, it has NO DUPLicates.). This algorithm shares the significant properties of [Wong and Baciu 2006]. Finally, the R-Triangle algorithm is labeled as "R-TRI" in the data.

Elementary Tests: Figs. 10(a) and (b) show the results of the four algorithms on the benchmarks. The primary goal of R-Triangles is to efficiently reduce the number of elementary tests performed. In Fig. 10(a) we can see several things. First, a basic culling algorithm based on triangle-based BVH produces an order of magnitude more elementary tests than does R-Triangle, most of them duplicates. In fact, in the worst case (the Cloth-ball benchmark), the BASIC algorithm performs 28X more elementary tests. This illustrates the scope of duplicate queries. The NO-DUPL algorithm uses the same culling algorithm as BASIC, but it uses the assignment property of R-Triangles to eliminate duplicate elementary tests. BASIC performs approximately five times more elementary tests than NO-DUPL. Duplication is clearly a significant issue.

R-TRI and NO-DUPL both use representation assignment to eliminate duplicate elementary tests. Despite that, NO-DUPL still performs more tests (typically three times as many tests.) The difference is that R-TRI also uses feature BVs to further cull elementary tests. ADJ uses feature BVs to cull and also uses a run-time database to prevent duplicate elementary tests. Unsurprisingly, it performs the same number of tests as R-TRI.

Query Time: Reducing the number of elementary tests performed is desirable, but it is not the final metric of success. The ADJ algorithm performs as few elementary tests as R-TRI, but examining Fig. 10(b) shows that it spends a great deal of time in culling those tests. Although it performs one third to one sixth of the tests as NO-DUPL, NO-DUPL's average frame time is as much as half of that of ADJ. This is particularly true in the Flamenco benchmark. The Flamenco benchmark is a complex model with seven layers of cloth on a body. Self-collision and collisions are computed between layers of cloth as well as with the body. Every layer of cloth lies in very close proximity with the other layers. This produces a massive set of candidate triangle pairs and a correspondingly large set of duplicate queries. R-TRI and NO-DUPL, because they eliminate duplicate queries at the source, carry the advantage in this scenario. ADJ must perform extensive searches in its database to determine which feature tests have already been performed.

Benchmark	BV Cull Rate	LineSeg-BV Cull Rate	Pct. Improvement of LineSeg-BV over BV
Balls	72.3%	77.9%	7.8%
Cloth-ball	77.4%	87.9%	13.6%
Princess	49.l%	65.8%	34%
Flamenco	61.2%	72%	17.6%

Table 2: *Improved culling efficiency of Line-Segment-BV intersection tests over bounding volume intersection tests for culling swept vertices against swept triangles.*

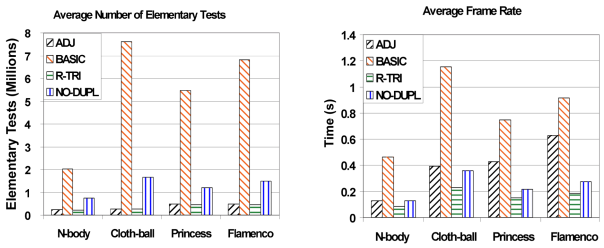

Figure 10: *(a) shows the average number of elementary tests per frame for each of the benchmarks across all collision detection algorithms. (b) shows the average time per frame for each of the benchmarks across all collision detection algorithms.*

R-TRI and NO-DUPL are almost identical; they both use representation assignment to prevent duplicate elementary tests. R-TRI further uses feature BVs to cull other tests. R-TRI performs the minimum number of possible elementary tests. The improved culling leads to a savings in the number of elementary tests dispatched. It carries the cost of extra BV-overlap tests. However, the savings is greater than the cost and, for our benchmarks, R-Triangles routinely out-performs NO-DUPL by approximately 33%.

6 Analysis and Limitations

Analysis: Representative-Triangles clearly improve collision detection query time. They eliminate duplicate queries without expensive memory accesses or unwieldy run-time data structures. This directly leads to performance improvement. They further improve culling efficiency by going beyond candidate triangle pairs and resolve candidate feature pairs. Taken together, these two attributes can take an existing, triangle-based collision acceleration technique and provide an increase in performance.

It is worth noting that these two ideas have been investigated earlier. Hutter and Fuhrmann [2007] address the issue of culling in their paper. They explicitly keep track of all features within the leaf nodes of the BVH. They recognize the culling improvement inherent in this finer granularity. However, like the ADJ algorithm, they must use a run-time database to eliminate duplicate feature BV-overlap tests and elementary tests. For large scenes with many triangles in close proximity, this quickly breaks down and can become inefficient.

Wong and Baciu [2006] presented an algorithm for randomly marking up triangles and using those marks to filter features during col-

lision detection to also prevent duplicate queries. Our formulation, R-Triangles, shares a common philosophical origin with Wong and Baciu. However, there are some key differences.

1. By coupling feature assignments and feature BVs we exploit the same advantage of duplicate elimination but also improve culling efficiency.
2. We present, what we feel is, a much simpler assignment algorithm and provide a theoretical justification for the algorithm.
3. Our lighter assignment algorithm is more amenable to simulations in which the mesh undergoes topological changes. When the topology changes, new assignments must be made as efficiently as possible.

Limitations: Representative-Triangles still leave a great deal of room for improvement. Even with duplicate query elimination and improved feature-based BV culling, the percentage of tests that prove to be false positives is immense (above 90%.)

Second, any scene made up of a triangle soup would gain no benefit from R-Triangles. Every triangle, by its very nature as part of a triangle soup, would represent all of its features. Admittedly, in this case there would be no duplicate queries, per se. The only benefit would be from feature BV culling.

Third, if storing feature BVs in memory, using R-Triangles increases the memory requirements of the collision detection algorithm. Typically, $|V|+|E| \approx 3|F|$. This would obviously be significant for very large models which may fit in main memory without R-Triangles, but which could result in out-of-core problems with R-Triangles. In addition to having larger memory requirements, the feature BVs must be updated at each time step. This increases the cost of BVH maintenance.

7 Future Work

There are many avenues for future work.

Integration into Simulation: In the future, we would like to fully apply our method into a system which simulates topological changes. We would also integrate it into a production-quality simulation system to evaluate its impact on the entire simulation pipeline. There are additional characteristics of our approach that bear further investigation.

Element Bounding Volumes: The choice of AABB as the feature bounding volume is arbitrary. We selected the AABB because of its simplicity and cost. More advanced bounding volume types, such as object-aligned bounding volumes (OBBs) or k-DOPs would certainly provide superior fit and culling efficiency. It is worth investigating to see if these more advanced BV types, used as feature BVs, can justify their greater cost through increased culling efficiency. Tang et. al [2007] indicate that replacing AABBs with kDOPs increased their overall performance for CCD.

Dynamic Representative Re-assignment: We argued that it is not possible to create a globally optimal representation assignment for a mesh undergoing unknown deformations. We could exploit temporal coherence to try and create locally optimal representation assignments which dynamically change based on the current mesh configuration. The idea is to identify features which were involved in elementary tests but didn't produce a collision. We would reassign them to an incident triangle which didn't appear in a triangle candidate pair in the previous time step. Based on temporal coherence, we assume that if the triangle wasn't involved in a collision in the previous step, it won't be involved in this step either.

Acknowledgements

This research was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043, and Intel. Part of this work was done while the primary author was an intern at Walt Disney Animation Studios.

References

BRADSHAW, G., AND O'SULLIVAN, C. 2004. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. on Graphics 23*, 1.

EHMANN, S., AND LIN, M. C. 2001. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proc. of Eurographics'2001) 20*, 3, 500–510.

ERICSON, C. 2004. *Real-Time Collision Detection*. Morgan Kaufmann.

GOTTSCHALK, S., LIN, M., AND MANOCHA, D. 1996. OBB-Tree: A hierarchical structure for rapid interference detection. *Proc. of ACM Siggraph'96*, 171–180.

GOVINDARAJU, N., KNOTT, D., JAIN, N., KABAL, I., TAMSTORF, R., GAYLE, R., LIN, M., AND MANOCHA, D. 2005. Collision detection between deformable models using chromatic decomposition. *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH) 24*, 3, 991–999.

HUBBARD, P. M. 1993. Interactive collision detection. In *Proceedings of IEEE Symposium on Research Frontiers in Virtual Reality*.

HUTTER, M., AND FUHRMANN, A. 2007. Optimized continuous collision detection for deformable triangle meshes. In *Proc. WSCG '07*, 25–32.

KLOSOWSKI, J., HELD, M., MITCHELL, J., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Trans. on Visualization and Computer Graphics 4*, 1, 21–37.

LARSSON, T., AND AKENINE-MÖLLER, T. 2006. A dynamic bounding volume hierarchy for generalized collision detection. *Computers and Graphics 30*, 3, 451–460.

LIN, M., AND CANNY, J. F. 1991. Efficient algorithms for incremental distance computation. In *IEEE Conference on Robotics and Automation*, 1008–1014.

LIN, M., AND MANOCHA, D. 2003. Collision and Proximity Queries. In *Handbook of Discrete and Computational Geometry: Collision detection*

MIRTICH, B. 1998. V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics 17*, 3 (July), 177–208.

MOLLER, T. 1997. A fast triangle-triangle intersection test. *Journal of Graphics Tools 2*, 2.

OTADUY, M., CHASSOT, O., STEINEMANN, D., AND GROSS, M. 2007. Balanced hierarchies for collision detection between fracturing objects. In *IEEE Virtual Reality*.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garment. *Graphics Interface*, 177–189.

SANNA, A., AND MILANI, M. 2004. CDFast: an algorithm combining different bounding volume strategies for real time collision detection. *SCI Proceedings 2*, 144–149.

SUD, A., GOVINDARAJU, N., GAYLE, R., KABUL, I., AND MANOCHA, D. 2006. Fast Proximity Computation Among Deformable Models using Discrete Voronoi Diagrams. *Proc. of ACM SIGGRAPH*, 1144-1153.

TANG, M., YOON, S., CURTIS, S., AND MANOCHA, D. 2007. Interactive continuous collision detection between deformable models using connectivity-based culling. *UNC Chapel Hill, Technical Report*.

TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., AND VOLINO, P. 2005. Collision detection for deformable objects. *Computer Graphics Forum 19*, 1, 61–81.

TROPP, O., TAL, A., SHIMSHONI, I. 2006. A fast triangle to triangle intersection test for collision detection. *Computer Animation and Virtual Worlds 17*, 5, 527–535.

VAN DEN BERGEN, G. 1997. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools 2*, 4, 1–14.

WONG, W. S.-K., AND BACIU, G. 2006. A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. *Proc. of ACM VRCIA*.

WONG, W. S.-K. 2005. Dynamic interaction between deformable surfaces and nonsmooth objects. *IEEE Tran. on Visualization and Computer Graphics*.

YOON, S., CURTIS, S., AND MANOCHA, D. 2007. Ray tracing dynamic scenes using selective restructuring. *Proc. of Eurographics Symposium on Rendering*.

ZACHMANN, G., AND WELLER, R. 2006. Kinetic bounding volume hierarchies for deforming objects. In *ACM Int'l Conf. on Virtual Reality Continuum and its Applications*.