VolCCD: Fast Continuous Collision Culling between Deforming Volume Meshes

Min Tang[1], Dinesh Manocha[2], Sung-eui Yoon[3], Peng Du[1], Jae-Pil Heo[3], Ruo-Feng Tong[1]
[1]Zhejiang University, China
[2]The University of North Carolina at Chapel Hill, USA
[3]KAIST, South Korea

We present a novel culling algorithm to perform fast and robust continuous collision detection between deforming volume meshes. This includes a continuous separating axis test that can conservatively check whether two volume meshes overlap during a given time interval. In addition, we present efficient methods to eliminate redundant elementary tests between the features (e.g., vertices, edges, and faces) of volume elements (e.g., tetrahedra, hexahedra, triangular prisms, etc.). Our approach is applicable to various deforming meshes, including those with changing topologies, and efficiently computes the first time of contact. We are able to perform inter-object and intra-object collision queries in models represented with tens of thousands of volume elements at interactive rates on a single CPU core. Moreover, we observe more than an order of magnitude performance improvement over prior methods.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Physically based modeling*

General Terms: Algorithms

Additional Key Words and Phrases: Continuous collision detection, deforming volume meshes, continuous separating axis theorem, assignment culling

1. INTRODUCTION

Many physically-based simulation algorithms use volume meshes to represent deformable objects. Volume meshes (e.g., tetrahedral meshes) are a polygonal representation of the interior volume of an object and correspond to a discrete representation of the internal structure of the object. Such regular or irregular grids are commonly used in finite element analysis and other Lagrangian mesh based simulation algorithms.

In this paper, we address the problem of fast and reliable collision detection between volume meshes. As forces are applied, the objects deform, and the nodes or points inside the object move. Our goal is to compute all the contacts between deforming volume meshes, including self-collisions. In order to perform reliable collision checking, we focus mainly on *continuous collision detection* (CCD) [Provot 1997; Bridson et al. 2002; Redon et al. 2002]. The CCD algorithms typically perform linear interpolation between discrete vertex positions of the volume meshes and check for collisions among swept volumes by performing elementary tests. These CCD methods provide accurate and robust collision results by computing the first time of contact, preventing inter-penetrations, and detecting collisions even between fast moving objects.

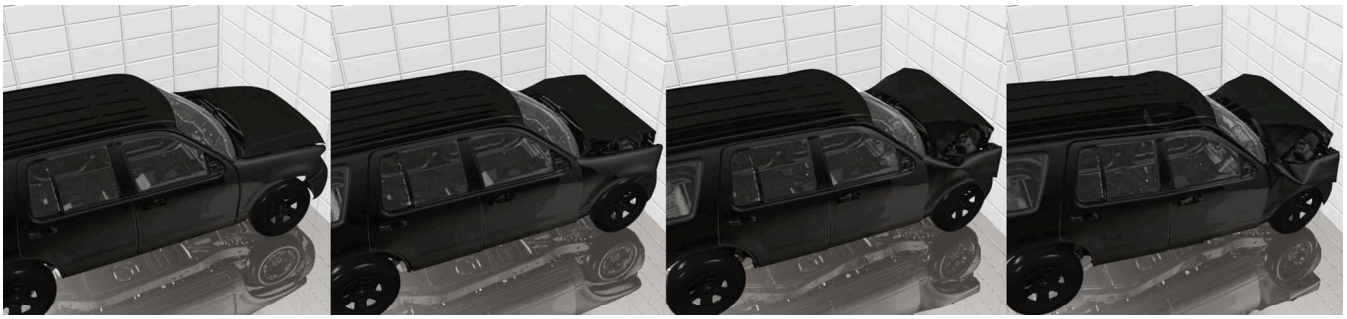
Volume meshes and FEM methods are increasingly used in interactive applications such as games and surgical simulators. These include simulating solid or soft-tissue deformations, fracturing, suturing, cutting, etc. In many applications, it is also important to compute all the contacts among external and internal elements of volume meshes for a high quality simulation [LS-DYNA 2001; O'Brien 2000]. However, it is a major challenge to perform interactive and reliable collision queries in these simulations [Irving et al. 2004; Teschner et al. 2005; Parker and O'Brien 2009; Zhu et al. 2010].

Most of the prior work on CCD algorithms, unfortunately, has been limited to surface meshes represented as triangulated models and thus check for collisions only at the boundaries of the objects. Furthermore, these methods do not work well when the simulation results in topological changes (e.g., fracturing) on the surface meshes [Heo et al. 2010]. Moreover, current techniques for collision checking between volume meshes mainly perform discrete collision checking at fixed time steps using spatial subdivisions, distance fields or image-space techniques. Since these discrete methods can miss collisions, especially when objects deform drastically or move fast, many simulation techniques with volume meshes typically use rather small time steps and thus run very slowly.

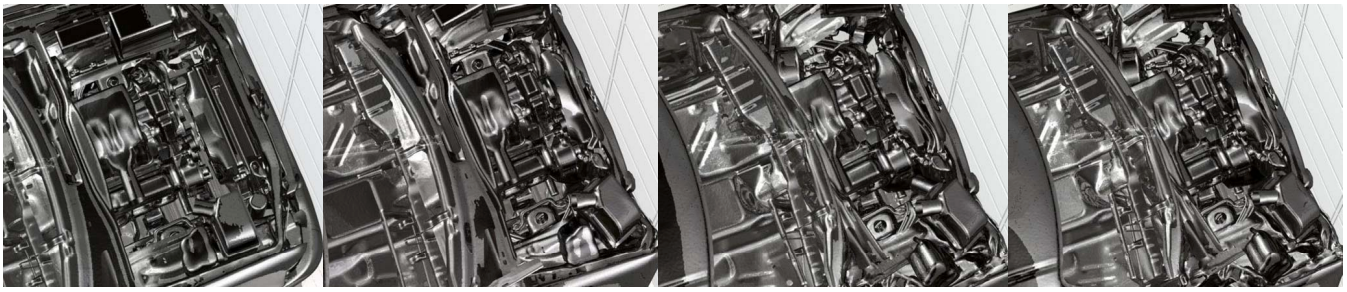
Main results: We present novel algorithms for fast CCD computation between volume meshes. Our approach can handle all regular or irregular meshes, where each volumetric element is a convex polytope. Moreover, the approach is general and makes no assumptions about motion, deformation, or topology.

We describe two new culling methods, feature-level and element-level culling techniques, to accelerate the computation. The first culling technique is based on continuous separating axis tests between the features (e.g., vertices, edges, and faces) and volumetric elements (e.g., tetrahedra, hexahedra, triangular prisms, etc.). Our formulation uses separating axes to conservatively check whether there is a collision between two volume meshes during a given time interval (see Section 4). This test is simple and robust, since it reduces to performing a small number of dot products. In addition, we present techniques to eliminate redundant elementary tests between the features by exploiting mesh connectivity (see Section 5). In particular, we use a feature-level assignment culling that can also handle changes in the mesh connectivity.

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY



(a) Exterior view



(b) Interior view

Fig. 1. **Car Crash:** A Ford Explorer with 1.2M shell elements crashes against a rigid wall and deforms. The underlying finite-element simulation is performed using LS-DYNA package for crash analysis, which performs contact computations between the shell elements. These figures show exterior and interior views during the crash simulation. Average CCD query time for both inter- and intra-collision detection using VolCCD is 3.3 seconds per frame on a single core of 2.4 GHz Intel Q6600 CPU. We obtain 12 times performance improvement over prior collision detection algorithms for this benchmark.

The overall CCD algorithm uses a k-DOP hierarchy and can perform inter-object as well as intra-object collisions. The combination of separating axis tests and feature-level culling can significantly improve the culling efficiency (see Section 6).

We highlight the performance of our CCD algorithm on volume meshes arising from various engineering, medical, and animation simulations with a few thousand to hundreds of thousands of tetrahedra or triangular prisms. In practice, our algorithm can reliably check for collisions in tens to hundreds of milliseconds on a single 2.4 GHz core. We observe up to two orders of magnitude reduction in the number of elementary tests and more than an order of magnitude performance improvement over prior methods (see Section 6). Moreover, we are able to produce realistic simulation results (e.g., fracturing) over collision methods that check collisions only at the boundary at discrete time steps.

Organization: The rest of the paper is organized in the following manner. We survey some related work in Section 2 and give an overview of our approach in Section 3. Section 4 describes our novel culling scheme based on separating axis test. We present the feature-level culling method in Section 5 and highlight the performance in Section 6. We analyze the approach and compare with prior methods in Section 7.

2. RELATED WORK

The problem of collision detection between deformable models has been well studied. We refer the reader to a recent survey on collision detection algorithms [Teschner et al. 2005] and physically-based deformable models [Nealen et al. 2006]. At a broad level, prior methods for collision queries can be classified as surface-based and volume-based methods.

2.1 Collision Queries on Polygonal Models

Most prior techniques have been designed for triangulated models and check for collisions only at the boundary (or surface) of the objects. The simplest algorithms compute bounding volume hierarchies and update the hierarchies using refitting or rebuilding algorithms as the objects deform [Teschner et al. 2005].

CCD provides more accurate collision results than discrete collision detection (DCD). Therefore, CCD has been studied relatively recently for rigid models [Redon et al. 2002], articulated models [Zhang et al. 2007], and deformable models [Provot 1997; Bridson et al. 2002; Govindaraju et al. 2005]. Since CCD require longer computation time than DCD, different acceleration techniques have been proposed. Some of them are based on normal cone culling [Tang et al. 2009], removing redundant elementary tests [Hutter and Fuhrmann 2007; Curtis et al. 2008], and coplanarity-based culling [Tang et al. 2010a]. The elementary tests between the primitives (e.g., triangles) are performed using polynomial root solvers [Provot 1997] or local advancement methods [Tang et al. 2010b].

2.2 Collision Queries for Volume Meshes

Current techniques for collision checking on volume meshes are mainly limited to discrete collision queries, where they check for contacts at specific time steps. Teschner et al. [2003] presented a spatial subdivision method to efficiently perform collision checking for tetrahedral meshes and integrated it with different FEM based simulation systems [Heidelberger et al. 2004]. Math-

ias and Gu [2007] extended this approach for real-time applications. O'Brien[2000] and Müller et al. [2001] check for collisions between the volume elements using bounding volume hierarchies (BVHs) for fracture simulations. Distance field methods are widely used to check for contacts and penetrations [Sud et al. 2006] and have been applied to volume meshes [Fisher and Lin 2001; Heidelberger et al. 2004; Teschner et al. 2005; Wojtan et al. 2009] to perform queries at a given distance field resolution. Other methods use the rasterization hardware to perform interactive queries at image-space resolution between volume meshes [Lombardo et al. 1999; Heidelberger et al. 2003; Faure et al. 2008; Allard et al. 2010]. However, most of these methods perform discrete collision checking at a specific time instance, as opposed to CCD between volume meshes. Therefore, simulations that employ these discrete methods can miss collisions and thus use small time steps to reduce the chance of missing collisions.

3. OVERVIEW

In this section, we introduce the notation used in the rest of the paper and give an overview of our approach.

3.1 Notation and Definitions

Many deformable and FEM simulations use volumetric elements such as tetrahedra, hexahedra, and triangular prisms to represent the interior volume of objects. We use the symbols of V, E, F, and O to represent the vertices, edges, faces, and volumetric elements of an object, respectively. The lower-cases symbols of v, e, f, and o denote a specific vertex, edge, face, and volumetric element, respectively. We define a volumetric element as a simple polytope consisting of n_v vertices, n_e edges, and n_f triangles. We also use the term of deforming volume meshes for models defined by these volumetric elements. The operators '$*$', '$\cdot$', and '$\times$' denote the product of two scalar values (or a scalar value and a vector), the dot product of two vectors, and the cross product of two vectors, respectively.

Our algorithm does not make any assumption about the motion or the underlying deformation or the topology of the object. The inputs for our algorithm are the initial and final positions of each vertex corresponding to two discrete time instances. We denote the time interval as $[0, 1]$. Similar to prior surface-based CCD algorithms [Provot 1997; Bridson et al. 2002], we assume that the motion of each vertex of the volume mesh can be represented by a constant velocity between the two time steps; we also briefly discuss how our approach can be extended to handle non-constant velocities. Our goal is to check for collisions between all the deforming volumetric elements of the same object (i.e. intra-object collisions) and different objects (i.e. inter-object collisions) during the time interval and compute the first time of contact.

3.2 Collision Checking between Volume Meshes

Most prior work in collision detection can be broadly classified as discrete collision detection (DCD) or CCD methods. In general, CCD methods tend to be more expensive, as they check for collisions along the entire trajectory. However, the CCD methods are more accurate and can guarantee that no collision is missed between the discrete time steps. This makes it possible to employ large time steps during the simulation. On the other hand, simulation algorithms based on DCD tend to choose smaller time steps [Parker and O'Brien 2009], in order to reduce the chances of missing collisions and thus maintain the fidelity of the simulation. However, using a small simulation time step can slow down the overall performance

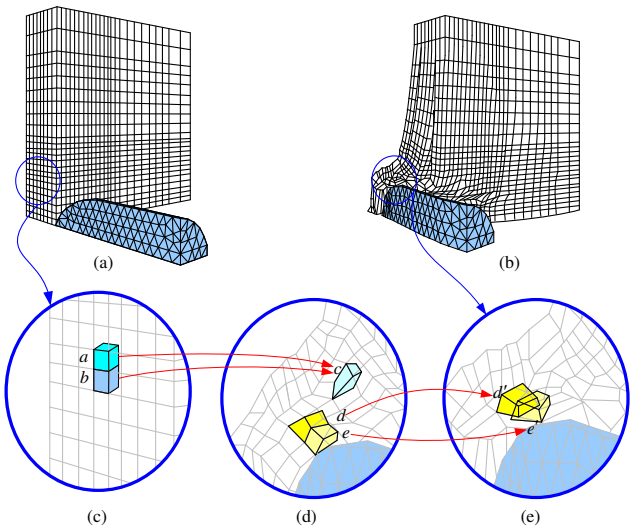


Fig. 2. **Topological changes:** When a high-speed bullet hits a metal plate, topological changes occur (e.g., elements a and b merge into c, and the connectivity of the elements d and e changes). These kinds of simulations with such topological changes can result in a high number of collisions and penetrations among the interior elements of volume meshes. Our algorithm can compute the first time-of-contact among the elements in an efficient manner.

of the simulation. Most of the earlier work in collision detection was based on DCD algorithms. CCD techniques have been shown recently to be useful to perform robust collision detection and handling for many applications [Provot 1997; Redon et al. 2002; Bridson et al. 2002].

One may consider that identifying collisions among the internal elements based on the CCD framework may be unnecessary, since surface-based CCD methods can accurately compute the first time of contact at the boundary. However, contacts can occur even among internal elements especially under strong external forces or high impacts with fast moving objects. Even though the surface-based CCD methods can identify the first time of contact and the associated boundary features, current contact resolution methods may not be able to accurately resolve those contacts and may result in inter-penetrations during subsequent frames of the simulation [Volino and Magnenat-Thalmann 2006]. In this case, it is important to identify all the collisions between the internal elements and resolve them in an appropriate manner.

Many of existing physically-based simulation algorithms check only for collisions at the surface boundary. However, for many CAD/CAM and virtual prototyping applications, such as crash simulation or structure analysis, it is important to check the features of interior volume elements for collisions [Hallquist 2006; ABAQUS 2003]. Therefore, these FEM systems perform various kind of contact and penetration computations [LS-DYNA 2001], and these contact computations can take a significant fraction of the overall simulation time.

In some applications, such as fluid simulation [Fleissner et al. 2007; Plimpton et al. 1998], there are no explicit boundary elements. Such fluid simulations or other complex simulations often undergo topological changes such as explosions or penetrations shown in Figure 2, or cracks [Sifakis et al. 2007]. As a result, the connectivity between the boundary and interior elements changes. It is required to check for collisions among all those elements for

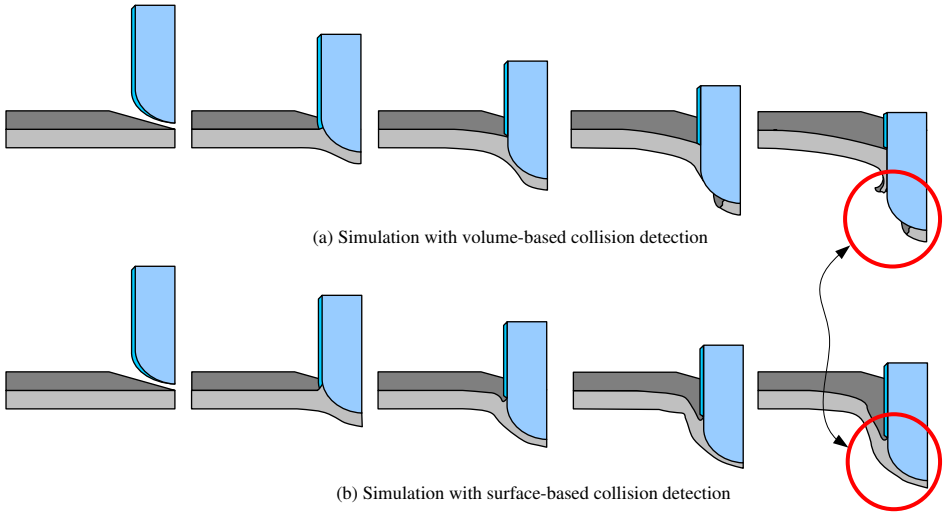


Fig. 3. **Comparison between surface-based and volume-based collision detection:** These two figures show two simulation sequences of a bullet penetration benchmark generated with volume-based collision detection (a) and surface-based collision detection (b). In this simulation, a bullet hits a steel plate. Since the steel plate is thin, the plate should fracture given the high-speed bullet. The highlighted areas stress the difference between the simulation results. For this benchmark, checking collisions between the interior features is essential to compute the interior impact forces that result in cracks and fractures, and thereby generate an accurate simulation. For these simulation results, we use LS-DYNA.

realistic simulation. If there is a severe deformation due to a large external force in simulations, it can result in complex topological changes. This is illustrated in Figure 2.

We give a broad overview of finite-element simulation that checks for collisions between the internal volume elements(Algorithm 1). The overall simulation framework is similar to the ones described in [Müller et al. 2001; O'Brien 2000; O'Brien and Hodgins 1999]. All the volumetric elements in the scene are organized in a hierarchical manner using a BVH. Volume-based collision checking is performed by traversing the BVH in top-down manner and searching all the contacts [O'Brien 2000]. Next, the impact forces are calculated based on these contacts and the nodes are updated with the accumulated forces (internal forces, external forces, etc.). For the nodes undergoing forces that exceed their failure stresses or the merging thresholds, the topology structure around them are updated with splitting and gluing operations [Müller et al. 2001; O'Brien and Hodgins 1999; Wojtan et al. 2009]. Otherwise the nodes are updated based on Newton's second law.

In the example shown in Figure 2, we show that at the end of a specific simulation time step, two volumetric elements a and b merge into c due to the impact force (Line 28-33 of Algorithm 1). Also, two adjacent elements d and e break their connectivity and penetrate with each other (Line 35-38 of Algorithm 1). This kind of topological change results in collisions between the interior features of a mesh. Therefore, we can get deeper inter-penetrations among internal elements of models.

Figure 3 highlights the benefit of performing collision checking between volume elements on the bullet penetration benchmark. In Figure 3(b), we show simulation results achieved by checking for collisions only at the boundary (i.e. only the boundary features). In Figure 3(a), we show the result achieved also by taking into account collision checking between the internal volume elements (i.e. boundary + interior features). The highlighted areas stress the difference between the simulation results. For this benchmark, checking collisions between interior features is essential both to reflect interior impact forces producing fractures and to generate accurate simulation results.

3.3 Acceleration using Bounding Volume Hierarchies

We use a k-DOP bounding volume hierarchy to accelerate the computation. The hierarchy is used to cull away pairs of volumetric elements that are not in close proximity. A k-DOP (specifically a 16-DOP in our system) hierarchy B is computed for all the volumetric elements in the scene and updated as the objects move or deform. The 16-DOPs are constructed similar to the 18-DOPs as described in [Klosowski et al. 1998]. The reasons that we use 16-DOPs are two folds: 1) it provides a higher culling ratio over other simple bounding volumes such as axis-aligned bounding box and 2) it can be updated efficiently. For example, the computation can be accelerated with SIMD instructions (e.g., SSE instructions on current CPUs). Furthermore, we use restructuring and refitting algorithms to update the k-DOP hierarchy for deforming volume meshes in an efficient manner. In our benchmarks, updating a k-DOP based BVH takes a minor portion, about 2.45% to 7.56%, of the total query time.

Each leaf node of B encloses the volume swept by a single element o that is deforming during the time interval $[0, 1]$. Collision detection is performed by traversing B recursively in a top-down manner and checking the bounding volumes for overlap. For every overlapping pair of leaf nodes, exact collision checking between the volumetric elements is performed by using elementary tests. In practice, the number of overlapping bounding boxes tend to be rather high for deforming meshes and may result in a very high number of false positives (Figure 4).

3.4 CCD between Volumetric Elements

Let us assume that a given pair of volumetric elements does not overlap at $t = 0$; we will discuss how our method behaves when there are overlaps at the initial position. In that case, the first time of contact between these (internal or external) volumetric el-

Algorithm 1 FEM simulation with volume based collision detection:

```
// Stage 1: Preprocessing.
Build BVH B for the whole scene.

// Stage 2: Solving iteratively.
t = t_0
while t <= t_1 do
  for all volumetric elements o_i ∈ the set of elements elm do
    Calculate internal forces f_j^i at each nodes nd_j.
    Calculate external forces f_j^e at each nodes nd_j.
    // Check volumetric elements for contacts.
    if Any contact detected then
      Compute contact information {t', d_j}.
      // Here t' ∈ [t, t + Δt] is the first-time-of-contact,
      // and d_j is the penetration depth.
      // Calculate impact forces f_j^m of each nodes nd_j
      // based on the contact information {t', d_j}, i.e.,
      f_j^m = ImpactForce(nd_j, {t', d_j})
    else
      f_j^m = 0
    end if
  end for

  for all nodes nd_j ∈ the set of nodes nds do
    // Update position of each node nd_j based on
    // accumulated forces f_j^i + f_j^e + f_j^m, i.e.,
    f_j^a = f_j^i + f_j^e + f_j^m
    // Checking for merging threshold.
    for all nodes nd_k in the near proximity of nd_j do
      if f_j^a > merging_threshold_jk then
        Merge nd_j and nd_k into a new node nd_l,
        and adjust position of nd_l accordingly.
      end if
    end for
    // Checking for failure stress.
    if f_j^a > failure_stress_j then
      Split nd_j into two new nodes nd_p and nd_q,
      and adjust the positions of nd_p and nd_q accordingly.
    end if
    if No topology change then
      pos_j^{t+Δt} = UpdatePosition(pos_j^t, f_j^a)
    end if
  end for

  Update the BVH B.
  t = t + Δt
end while

// Stage 3: Output simulation result.
Output results.
```

ements occurs at the boundary features, corresponding to the vertices, edges, or faces. Prior work on CCD computation has been limited to triangle primitives and performs 15 elementary tests: 9 edge-edge (EE) and 6 vertex-face (VF) tests [Provot 1997; Bridson et al. 2002]. Each elementary test reduces to finding roots of a cubic polynomial.

This formulation can be extended to volumetric elements. Suppose that we have two volumetric elements, o^i and o^j; o^i has n_v^i vertices, n_e^i edges, and n_f^i triangles, and o^j has n_v^j vertices, n_e^j edges, and n_f^j triangles. In the case that o^i and o^j are not adjacent, i.e. do not share any boundary elements between them, computing the first time of contact reduces to performing $(n_v^i * n_f^j + n_v^j * n_f^i)$ VF elementary tests and $n_e^i * n_e^j$ EE tests. For example, a primitive test for two tetrahedra require to perform 32 VF and 36 EE tests. Similarly, a primitive test for two hexahedra require to performing 192 VF and 144 EE elementary tests.

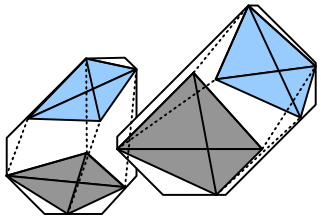


Fig. 4. **Bounding volumes of volumetric elements:** We use tight fitting k-DOPs to enclose the swept volume of each deforming volumetric element. In practice, these k-DOPs can result in a high number of false positives.

3.5 Our Approach

The problem of performing CCD tests between volume meshes is considerably more expensive and complicated than handling surface meshes. This is due to the following reasons:

(1) Bounding volumes of the leaf nodes of a BVH B tend to be more conservative since those bounding volumes enclose the swept volumetric elements in the time interval $[0, 1]$. As a result, the number of false positives given the CCD using a BVH is very high (e.g., $90\% \sim 99\%$ in our benchmarks).

(2) The cost of an exact CCD between volume meshes is much higher, e.g., $4 - 20$ times more expensive as compared to CCD between triangles, because of the higher number of elementary tests.

(3) Culling methods based on normal cones or bounds [Volino and Thalmann 1994; Provot 1997; Tang et al. 2009; Heo et al. 2010] are designed originally for detecting self-collision free regions of surface meshes. These techniques cannot be applied to volume meshes, since it is unclear how to compute normals for internal elements of volume meshes.

In order to accelerate the computation, we present two new culling techniques to reduce the number of false positives and remove redundant elementary tests. These methods are applied to the volumetric elements as elementary-level culling, and to features of each element as feature-level culling. This includes a *continuous separating axis theorem* (CSAT) that can significantly reduce the number of false positives between the volumetric elements. Moreover, we present a feature-level *assignment culling* scheme that is used to remove all duplicate or redundant elementary tests between pairs of volumetric elements.

During each simulation time step, our CCD algorithm proceeds in three stages. The overall pipeline of the CCD algorithm is shown in Figure 5. At the first stage, we perform bounding volume traversal and cull away non-overlapping pairs. We also apply CSAT to the leaf node pairs to reduce the false positives. At the second stage, We perform feature-level culling that handles non-adjacent and adjacent volumetric elements in separate steps. For non-adjacent volumetric element pairs, assignment culling is used to remove redundant elementary tests. Moreover, we present the notion of *O-Set pairs*, which refer to the minimal number of elementary tests between adjacent pairs that are not classified as parts of the tests between the non-adjacent pairs, and can be easily recomputed to handle topological changes. At the third, final stage, exact elementary tests are preformed.

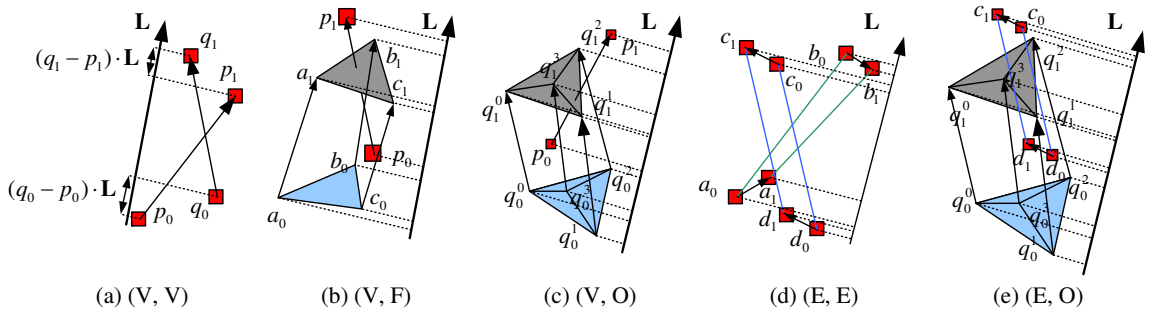


Fig. 6. **CSAT culling algorithm:** the continuous separating axis theorem (CSAT) culling methods (a) for two moving vertices, (b) for a VF test between a vertex and a triangle, (c) for VF tests between a vertex and a volumetric element, (d) for a EE test between two edges, and (e) for EE tests between an edge and a volumetric element. For each case shown in (b) – (e), we can conclude that these deforming feature/element do not overlap with each other during the time interval based on CSAT tests.

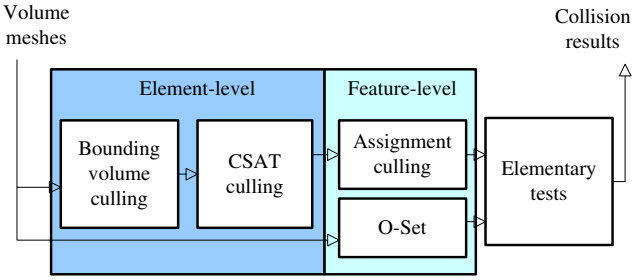


Fig. 5. **CCD algorithm for deforming volume meshes:** By performing element-level and feature-level culling, we drastically reduce the number of false positives and achieve interactive CCD performance on complex volumetric benchmarks.

4. CSAT CULLING ALGORITHM

In this section, we present our culling algorithm based on the continuous separating axis theorem (CSAT). The objective is to derive sufficient conditions for non-overlap elements during the $[0, 1]$ time interval. We first present the CSAT formulation for two vertices undergoing continuous motion during the $[0, 1]$ time interval. Next, we extend it to perform VF and EE tests, and finally to the volumetric elements.

4.1 Continuous Separating Axis Theorem

Given two convex shapes, the separating axis theorem (SAT) states that if there exists a line onto which the projections of two objects do not overlap, then the objects do not intersect. The resulting line with disjointed projections is called the *separating axis*. Separating axis tests are widely used for fast overlap tests between oriented bounding boxes [Gottschalk et al. 1996] and convex shapes. Eberly [2000] extended the SAT formulation to perform overlap tests between rigid convex polytopes that are moving with constant linear velocity and no angular velocity.

We extend SAT designed for rigid convex polytopes to continuous tests between deforming volumetric elements, whose vertices have a constant velocity. Our formulation is conservative, and provides sufficient conditions for non-overlapping during the given time interval. Furthermore, we show that the resulting CSAT between two volume elements can be performed efficiently by a few dot products.

Theorem 1: CSAT for two moving vertices: *Suppose two moving vertices p and q, defined by their positions at the time interval* $[0, 1]$ *(*p_0 *&* p_1 *and* q_0 *&* q_1*, respectively), and a separating axis defined by* $\boldsymbol{L}$ $(\neq 0)$*. If* $(q_0 - p_0) \cdot \boldsymbol{L}$ *and* $(q_1 - p_1) \cdot \boldsymbol{L}$ *have the same sign, q will not overlap with p during the time interval.*

Due to the page limit, we provide proofs of this theorem and other corollaries in the appendix.

Overall, this theorem provides a simple, sufficient condition when two vertices moving along the straight lines have no collisions during the given time interval (as shown in Figure 6(a)). In Section 4.4, we extend this theorem to perform overlap tests between two volumetric elements.

4.2 VF Pairs

Based on the CSAT for two moving vertices, we derive the following culling test for a VF pair.

Corollary 1: CSAT between a vertex and a triangle: *Suppose a vertex p and a triangle f defined by their positions at the time interval* $[0, 1]$ *(*p_0 *&* p_1 *for p, and* a_0, b_0, c_0 *&* a_1, b_1, c_1 *for f, respectively), and a vector* $\boldsymbol{L}$ $(\neq 0)$*. If the following 6 expressions have the same sign:*

$$\begin{array}{lll} (a_0 - p_0) \cdot \boldsymbol{L}, & (a_1 - p_1) \cdot \boldsymbol{L}, & (b_0 - p_0) \cdot \boldsymbol{L}, \\ (b_1 - p_1) \cdot \boldsymbol{L}, & (c_0 - p_0) \cdot \boldsymbol{L}, & (c_1 - p_1) \cdot \boldsymbol{L} \end{array} \quad (1)$$

then p and f will not overlap during the time interval.

This culling condition implies if the projection of a vertex p is above/below all the projections of the three corners of a triangle f at $t = 0$ and $t = 1$, then p and f will not overlap during the interval (as shown in Figure 6(b)). The choice of $\boldsymbol{L}$ is important in terms of performance. In practice, we use the initial normal vector of f, $\boldsymbol{n_0}$, as the projection axis. In such case, $(a_0 - p_0) \cdot \boldsymbol{n_0} = (b_0 - p_0) \cdot \boldsymbol{n_0} = (c_0 - p_0) \cdot \boldsymbol{n_0}$, so we need to perform only 4 dot products for this culling test.

Corollary 2: CSAT between a vertex and a volumetric element: *Suppose a vertex p and a volumetric element o (with* n_v *vertices and* n_f *triangles) defined by their positions during the time interval* $[0, 1]$ *(*p_0 *&* p_1 *for p, and* q_0^i *&* q_1^i *for o,* $i \in [0, n_v - 1]$*), and a vector* $\boldsymbol{L}(\neq 0)$*. If the following* $2 * n_v$ *algebraic expressions have the same sign:*

$$\begin{array}{ll} (q_0^i - p_0) \cdot \boldsymbol{L}, & i \in [0, n_v - 1], \\ (q_1^i - p_1) \cdot \boldsymbol{L}, & i \in [0, n_v - 1] \end{array} \quad (2)$$

then p and all the n_f triangles of o will not overlap during the time interval.

This culling condition provides a simple non-overlapping condition between the moving vertex and the volume swept by o. We randomly pick a face from the volumetric element, and use n_0 of the face as $\boldsymbol{L}$. For a tetrahedron ($n_v = 4$ and $n_f = 4$), we need to perform only 8 dot products for this test (as shown in Figure 6(c)).

4.3 EE Pairs

Corollary 3: CSAT between two edges: *Suppose two edges, e^1 and e^2, defined by their end positions during $t \in [0,1]$ (a_0, b_0 & a_1, b_1 for e^1, and c_0, d_0 & c_1, d_1 for e^2), and a vector $\boldsymbol{L}(\neq 0)$. If the following eight expressions have the same sign:*

$$\begin{aligned}&(a_0 - c_0)\cdot\boldsymbol{L}, (a_1 - c_1)\cdot\boldsymbol{L}, (b_0 - c_0)\cdot\boldsymbol{L}, (b_1 - c_1)\cdot\boldsymbol{L},\\ &(a_0 - d_0)\cdot\boldsymbol{L}, (a_1 - d_1)\cdot\boldsymbol{L}, (b_0 - d_0)\cdot\boldsymbol{L}, (b_1 - d_1)\cdot\boldsymbol{L}\end{aligned} \quad (3)$$

then e^1 and e^2 do not overlap during the time interval.

This implies that if the projections of the two end points of an edge e^1 are above/below the projections of the two end points of another edge e^2 at $t = 0$ and $t = 1$, then e^1 and e^2 will not overlap during the time interval (as shown in Figure 6(d)). All these tests can be performed by using 8 dot products. Again, the choice of $\boldsymbol{L}$ is important in terms of performance. In practice, we use $\boldsymbol{n_0} = (a_0 - b_0) \times (c_0 - d_0)$, as the projection axis. In such a case, $(a_0 - c_0)\cdot\boldsymbol{n_0} = (b_0 - c_0)\cdot\boldsymbol{n_0} = (a_0 - d_0)\cdot\boldsymbol{n_0} = (b_0 - d_0)\cdot\boldsymbol{n_0}$, so we only need to perform 5 dot products to perform this culling method.

Corollary 4: CSAT between an edge and a volumetric element: *Suppose an edge e and a volumetric element o (with n_v vertices and n_e edges) defined by their positions at time interval $[0,1]$ (c_0 & c_1 and d_0 & d_1 for e, and q_0^i & q_1^i for o, $i \in [0, n_v - 1]$), and a vector $\boldsymbol{L}(\neq 0)$. If the following $4 * n_v$ expressions have the same sign:*

$$\begin{aligned}&(q_0^i - c_0)\cdot\boldsymbol{L}, i \in [0, n_v - 1], (q_1^i - c_1)\cdot\boldsymbol{L}, i \in [0, n_v - 1],\\ &(q_0^i - d_0)\cdot\boldsymbol{L}, i \in [0, n_v - 1], (q_1^i - d_1)\cdot\boldsymbol{L}, i \in [0, n_v - 1]\end{aligned} \quad (4)$$

then e and o do not overlap during the time interval.

We randomly pick an edge from the volumetric element and use $\boldsymbol{n_0} = (a_0 - b_0) \times (c_0 - d_0)$ as the choice of $\boldsymbol{L}$. For a tetrahedron ($n_v = 4$ and $n_e = 6$), we need to evaluate only 16 dot products by using this culling method.

4.4 CSAT between Volumetric Elements

Let the two volumetric elements be: o^i with n_v^i vertices, n_e^i edges, and n_f^i triangles, and o^j with n_v^j vertices, n_e^j edges, and n_f^j triangles. In the case that o^i and o^j are not adjacent, i.e. do not share any boundary elements between them, we can compute the first time of contact between o^i and o^j by performing VF tests between the vertices of one element and the faces of another element, and by performing EE tests between the edges of those two elements. Therefore, we have to perform $(n_v^i * n_f^j + n_v^j * n_f^i)$ VF tests and $n_e^i * n_e^j$ EE tests.

Note that our culling method that uses the CSAT between a vertex and an element requires $2 * n_v$ dot products, where n_v is the number of vertices of the element. Therefore, all the VF tests with our CSAT culling method need to perform $4 * n_v^i * n_v^j (= 2 * n_v^i * n_v^j + 2 * n_v^j * n_v^i)$ dot products. Also, our culling method that uses the CSAT between an edge and an element performs $4 * n_v$ dot products. As a result, all the EE tests between two volumetric elements can be performed with our CSAT culling method using $4 * n_e^i * n_v^j$ or $4 * n_e^j * n_v^i$ dot products. A detailed pseudo-code description of CSAT culling between two volumetric elements is given in Algorithm 2 of the appendix.

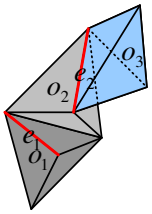


Fig. 8. **Duplicate EE tests between adjacent volumetric elements:** an EE test $\{e_1, e_2\}$ between an adjacent pair $\{o_2, o_3\}$ is covered by the tests of a non-adjacent pair $\{o_1, o_3\}$ by using O-Sets.

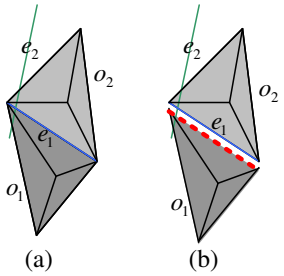


Fig. 9. **Redundant elementary tests:** The edge e_1 is shared by two tetrahedra o_1 and o_2. The edge e_1 will be tested with another edge e_2 twice, if o_1 and o_2 are tested with e_2 independently.

Separating axis, L: Depending on the chosen L, the performance of our culling method varies. In our benchmarks, the chosen L provides a high culling efficiency that reduces the number of operations. Other alternatives, such as selecting L from predetermined features or predicting L by overall velocities of volumetric elements, are possible. However, we found that they do not improve the performance in our experiments.

5. FEATURE LEVEL CULLING

Models represented by volume meshes tend to result in a high number of intra-object collisions. These include overlap tests between adjacent volumetric elements and non-adjacent volumetric elements. After applying CSAT culling, the overall algorithm performs elementary tests between the features of the volumes to compute the first time of contact.

In this section, we present two culling techniques that can eliminate any duplicate or redundant elementary tests. We present two methods: *O-Sets* for adjacent volumetric elements and *assignment culling* for non-adjacent volumetric elements. We have to perform elementary tests with features of all the volumetric elements that pass CSAT culling. However, we drastically reduce the number of tested elementary tests based on the O-Sets and assignment culling. Furthermore, we show how these techniques can also handle topological changes in the volume meshes.

5.1 O-Sets

The CCD tests between the volumetric elements are performed independently. In terms of handling two adjacent volumetric elements, the number of elementary tests turns out to be less than $n_v^i + n_f^j + n_v^j * n_f^i$ VF tests and $n_e^i * n_e^j$ EE tests, because some of the features are shared. Moreover, the algorithm can be proceeded in a more efficient manner by checking the non-adjacent volumetric pairs for overlap first, followed by adjacent volumetric pairs. We extend the orphan set idea proposed for surface meshes [Tang et al. 2009] to the O-Sets for volume elements and compute the minimal number of elementary tests that need to be performed between adjacent volume meshes.

In order to explain the O-Set, consider the example shown in Figure 8. Some of the EE tests (e.g., $\{e_1, e_2\}$) corresponding to

Fig. 7. **Octopi Benchmark:** Three deforming octopi with 24 tentacles consist of 17.6K tetrahedra. The simulation has multiple inter-object and intra-object collisions. The average CCD query time is 75ms per frame.

adjacent elements are same as the EE tests that need to be tested for collisions among the non-adjacent elements. We define the O-Set to be the feature pairs between adjacent element pairs that are not explicitly checked for collisions as part of elementary tests between the non-adjacent pairs. Therefore, if we first check for collisions between the non-adjacent element pairs, we only need to perform elementary tests between the feature pairs that belong to O-Sets, and not all the feature pairs among adjacent elements.

Given an object, its O-Set is computed as part of a preprocess by analyzing the connectivity information among the volume meshes. A pseudo-code description of construction algorithm for O-Set computation is given as Algorithm 3 in the appendix.

Incrementally updating the O-Sets: As the model undergoes topology changes, the connectivity information changes. In this case, the O-Sets are incrementally updated to reflect the topology changes that can occur during splitting, tearing, or object breaking. For a volume object O' is generated from O with topology changes, it can be described as $O' = O + A_s - A_t$, where A_s is the set of volumetric elements that are added to O', A_t is the set of deleted volumetric elements from O that are no longer in O'. The O-Sets are updated incrementally by removing feature pairs related to A_t and adding new feature pairs in the proximity of A_s.

5.2 Assignment Culling

A large number of features (vertices, edges, and faces) are shared between volumetric elements. If we perform elementary tests between the volumetric element pairs independently, many redundant elementary tests are performed. This is illustrated in Figure 9(a).

These redundant tests can be eliminated using an assignment mechanism that ensures that a feature is assigned to only one volumetric element that is incident to the feature. As shown in Figure 9(b), the redundant EE test is removed by assigning e_1 only to o_1.

The assignment mechanism has been used for triangle meshes [Curtis et al. 2008]. We extend this idea for volumetric elements as *static feature assignment*. The assignment is not unique as long as it satisfies the condition that a feature is assigned to a single volumetric element. A simple, yet effective solution is to use a greedy algorithm to ensure that a feature is assigned to only one element (as shown in Algorithm 4 in the appendix).

The main problem with static feature assignment is that it is not compatible with O-Sets. By using O-Sets, all the duplicate tests between adjacent volumetric elements are skipped. When using O-Sets with static feature assignment, some collisions between feature pairs may be missed when those feature pairs are assigned to adjacent volumetric elements. These feature pairs will be skipped when adjacent element pairs are skipped.

In order to overcome this problem, a non-adjacent constraint for the feature assignment is taken into account by the assignment scheme. We present this modification as the *dynamic feature assignment* method.

Dynamic feature assignment: For a feature pair $\{elm1, elm2\}$ that is tested for CCD, we assign them in a dynamic manner. The pair $\{elm1, elm2\}$ may be assigned to any volume elements among $\{o_i, o_j\}$, where o_i is incident to $elm1$ and o_j is incident to $elm2$. There must be at least one element pair within $\{o_i, o_j\}$ that o_i is not adjacent o_j. Otherwise $\{elm1, elm2\}$ correspond to adjacent element pairs and will be proceeded by the O-Set computation algorithm. So we directly assign the pair $\{elm1, elm2\}$ to the first non-adjacent volumetric element pair $\{o_i, o_j\}$. The dynamic feature assignment algorithm for a VF/EE test between a feature elm_1 from o_1 with another feature elm_2 from o_2 is performed in a similar manner. Also, we can use our culling method based on O-Sets without any modification. A detailed pseudo-code description is given in Algorithm 5 of the appendix.

Another benefit of the dynamic assignment over the static assignment is that it naturally works for volume meshes with topology changes, since the feature assignment is performed during the runtime process. As new feature pairs are added or deleted, we change their assignment in a dynamic manner. This makes it possible to make our algorithm work well on scenes with topological changes.

6. IMPLEMENTATION AND PERFORMANCE

In this section, we describe our implementation and highlight the performance of our CCD algorithm on several benchmarks arising from engineering, medical, and animation simulations that have different characteristics.

6.1 Implementation

We have implemented a CCD system for volume meshes (VolCCD) based on our algorithm on a standard PC (Windows/XP 32 bits, Intel Q6600 CPU 2.4GHz, 2GM RAM) by using C++. The timings are generated using a single thread running on a single core.

We compare the performance of our system (VolCCD), with the following two approaches:

(1) **NavCCD:** This algorithm use the k-DOP hierarchy, but performs all the elementary tests between the volumetric elements with overlapping bounding volumes. This way, we can evaluate the benefit of our culling schemes.

(2) **HashCCD:** This algorithm is based on optimized spatial hashing [Teschner et al. 2003] as opposed to bounding volume hierarchies. Instead of discrete collision checks, we perform CCD between volume meshes using the elementary tests.

Table I. **Performance and Speedup:** This table shows the average query time of VolCCD, performance improvements over NavCCD and HashCCD, and culling ratios (the ratios of the number of elementary tests of VolCCD over the number of NavCCD).

Model	Avg. Query	Speedup over NavCCD	Speedup over HashCCD	Culling ratio
Octopi	75ms	12x	18.3x	0.01%
Bullet	48ms	21x	28x	0.69%
Airbag	88ms	10.2x	13x	0.58%
Liver	53ms	10.3x	14.1x	0.72%
Ford	3.3s	12.4x	17.4x	3.54%

Table II. **Culling Efficiency on the Octopi Benchmark:** This table shows the number of VF and EE elementary tests performed for the deforming octopi benchmark without performing any culling (A), with feature-level culling only (B), and with element-level culling and feature-level culling (C). Each of them uses the 16-DOP BVH.

Culling	VF tests	EE tests	total	ratio
A	$26M$	$312M$	$339M$	100%
B	$1.1M$	$6.4M$	$7.5M$	2.23% (B/A)
C	$7.9K$	$24K$	$32K$	0.42% (C/B)

Table III. **Culling Efficiency on the Car Crash Benchmark:** This table shows the number of VF and EE elementary tests performed the benchmark Car Crash without performing any culling (A), with feature-level culling only (B), and with element-level and feature-level culling (C). Each of them uses the 16-DOP BVH.

Culling	VF tests	EE tests	total	ratio
A	$8.24E+09$	$1.6E+10$	$2.43E+10$	100%
B	$9.8E+07$	$3.8E+08$	$4.76E+08$	1.96% (B/A)
C	$2.3E+07$	$1.2E+08$	$1.41E+08$	29.61% (C/B)

6.2 Benchmarks and Performance

In order to test the performance of our algorithm, we use 5 different benchmarks, arising from different simulations with different characteristics.

(1) **Octopi:** Three deforming octopi with 24 tentacles contact with each other and generate a lot of inter-object and intra-object collisions (Figure 7).

(2) **Bullet Penetration:** A high speed copper bullet hits a steel target and achieves penetration (Figure 10). During the penetration, the topology changes result in a high number of inter-object collisions.

(3) **Airbag:** The deforming airbag has self-collisions as well as other collisions with the steering wheel (Figure 12).

(4) **Car Crash:** A Ford Explorer crashes against a rigid wall and deforms (Figure 1). This benchmark is designed for car crash analysis.

(5) **Cutting Liver:** A liver is cut during operation (Figure 13). This benchmark comes from surgery simulation.

Table I shows the average query time of VolCCD and the improvement over NavCCD and HashCCD, and culling ratios (the ratios of the number of elementary tests of VolCCD over the number of NavCCD), respectively. Overall our method shows more than one order of magnitude performance improvement over NavCCD and HashCCD across all the tested benchmarks. This overall performance improvement is mainly caused by the drastic reduction (about 100:1 reduction) of the tested number of elementary tests.

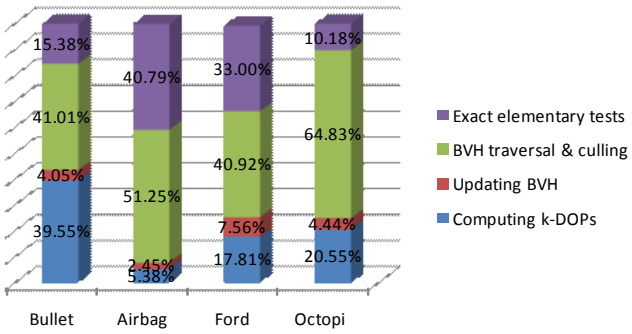


Fig. 11. **Portions of the steps of our CCD algorithm:** This figure provides a breakdown of computational cost of our CCD method (e.g., computing k-DOPs, updating k-DOP based BVH, BVH traversal & culling, and exact elementary tests.)

Tables II and III highlight the efficiency of each culling technique. The culling ratio for the feature-level culing only is computed as a ratio of the number of elementary tests without any culling to that with the feature-level culling only. The culling ratio for the element-level culling and feature-culling is computed as a ratio of the number of elementary tests performed with both elementary-level and feature-culling to that with the feature-level culling only. In Table II, only $0.01\%(= 2.23\% * 0.42\%)$ tests of those passed culling have to be performed. In Table III, only $0.58\%(= 1.96\% * 29.61\%)$ tests of those passed culling have to be performed.

Figure 11 provides a breakdown of computational cost (computing k-DOPs, updating k-DOP based BVH, BVH traversal & culling, and exact elementary tests) of our CCD method. For all the benchmarks, BVH updating & culling and exact elementary tests are the two most time-consuming steps ($56\% \sim 91\%$).

6.3 Continuous Trajectory with Topological Changes

In some benchmarks like the Cutting Liver and Bullet Penetration, the connectivity and topology changes. As a result, the number of vertices in the model may change and we cannot perform linear interpolation between the vertices. Instead, we use the previous position, p', of a vertex p by subtracting its movement, i.e. $p' = p - s * v_p$, where s is the simulation time step and v_p is the average velocity of p that is computed from the simulation. We perform linear interpolation between p and p', and perform the CCD tests between them to compute the first time of contact.

7. ANALYSIS AND COMPARISON

In this section, we analyze the performance of our algorithm and compare its features with prior methods.

7.1 Analysis and Discussions

The performance improvement in our CCD algorithm for volume mesh comes from two factors: 1) significantly reducing the number of false positives based on CSAT tests and 2) eliminating redundant elementary tests based on the O-Sets and assignment culling.

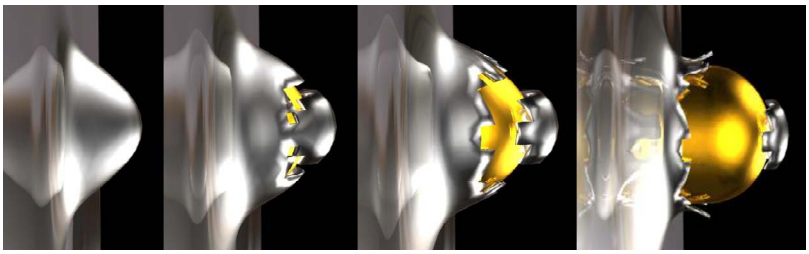


Fig. 10. **Bullet Penetration:** A high speed copper bullet (with $11.8 - 12$K hexahedra) hits a steel target and results in topological changes.

The CSAT algorithm for a VF pair $\{v, f\}$ only needs to perform 15 addition and 12 multiplication floating-point operations, when we use the normal vector of f as the separating axis. On the other hand, an exact elementary test between a VF pair takes roughly 155 addition, 217 multiplication, and 6 division operations, on average. As a result, the cost of the conservative CSAT culling method is about 6% to 8% of the total cost of the exact elementary test. Since the k-DOP hierarchy results in a high number of false positives, the CSAT culling method can be quite effective. Assignment culling is also effective in terms of removing redundant elementary tests between adjacent volumetric elements and non-adjacent volumetric elements with a very low overhead.

Extension to non-constant velocities: Currently, our approach has been applied to deforming volumetric elements whose vertices have a constant velocity between the discrete instances. Note that the constant velocity assumption is widely used in many prior CCD algorithms (including surface-based algorithms), as mentioned in Section 3.1. It leads to a simple formulation to interpolate the motion between two discrete instances. In the case that the velocities of the vertices are not constant (e.g. the vertices are moving with acceleration a_v), we have to use an appropriate function to model the trajectory of each vertex, as opposed to the linear trajectories shown in Figure 6. The CSAT culling method (Section 4) would be modified accordingly, but is still applicable. Since our approach is conservative, we expect that the culling efficiency would reduce in this case. If we model the motion using the constant acceleration, the resulting trajectory would correspond to quadratic polynomial functions of time t. The resulting elementary tests would involve computing the roots of degree 6 polynomials.

Relative performance of surface-based vs. volume-based C-CD algorithms: Our method performs CCD to check for collisions that may occur on the surface boundary as well as among the internal volume elements. In some applications, only performing collision checking between surface-based boundary primitives may be sufficient. As a result, we compare the performance of a surface-based and volume-based algorithm. For the two cases (a) and (b) of the bullet penetration scene in Figure. 3, the performance of volume-based collision detection is about 3.2X slower than the surface-based collision detection when running with a single thread. However, with the help of SIMD instructions and multi-threads (on a Intel Q6600 GPU with 4 cores), we observed that volume-based method is just about 1.3X slower than the surface-based method by exploiting the parallelism.

Collisions at the initial simulation time: We have derived our culling method based on the assumption that there are no overlapping elements at the initial simulation time $t = 0$. However, our culling methods work even in this case, since our CCD algorithm reduces to discrete collision checking at t=0.

7.2 Comparisons

In this section, we compare features of our algorithm with prior methods.

Optimized spatial hashing: Teschner et al [2003] used optimized spatial hashing to accelerate discrete collision detection among deforming volume meshes. This method can be easily extended to perform CCD. We have also extended this method to HashCCD to perform CCD and tested it against our method. We can also combine our culling algorithms with it. However, in our benchmarks, we observed that k-DOP hierarchies offer much higher performance over spatial hashing, since some of our tested benchmarks have very dense geometry in localized regions.

Flip methods: Some researchers have advocated the use of flip methods to check for self-collisions among volume meshes, and these have been used for tetrahedral meshes [Erleben et al. 2005]. The flip method only checks whether a specific element of the mesh undergoes self-collision. It can be used as a sufficient condition to check for self-collisions, but it is not a necessary condition. In other words, even if there is no flip, self-collisions can also appear if we fail to resolve contacts among the surface boundary and thus have inter-penetrations among the internal elements of the mesh.

Normal cones: Normal cones and continuous normal cones [Provot 1997; Tang et al. 2009] are used for self-collisions and work well on the portions of the boundaries that are relatively flat or have low variation in curvature. That approach is limited only to surface-based collision checking. In other words, it can only be applied to the boundary of the objects, and not to the interior elements. Since each volume mesh is a convex polytope, it is unclear whether normal cone culling can show meaningful culling.

Orphan sets & representative triangles: Orphan sets [Tang et al. 2009] and representative triangles [Curtis et al. 2008] are quite effective for removing redundant elementary tests between adjacent triangles and non-adjacent triangles of meshes. Our feature-level culling extends these ideas to volume meshes, and integrates them together with the dynamically assignment scheme.

Image-space algorithms: Many image-space algorithms that use GPU rasterization methods [Faure et al. 2008; Allard et al. 2010] can be used for interactive collision checking. However, the use of image-space methods can result in missed collisions, by using or relying on discrete image-space representations. Moreover, since different elements are connected to each other, it is unclear whether we can get a high culling ratio for volume meshes based on these image-space techniques.

Coplanarity-based culling: Our CSAT tests are complementary to coplanarity-based culling [Tang et al. 2010a]. Coplanarity-based culling checks for penetration along the interpolation trajectory of the deforming features. For example, to test a VF pair $\{v, f\}$, coplanarity-based culling computes the deforming normal vector

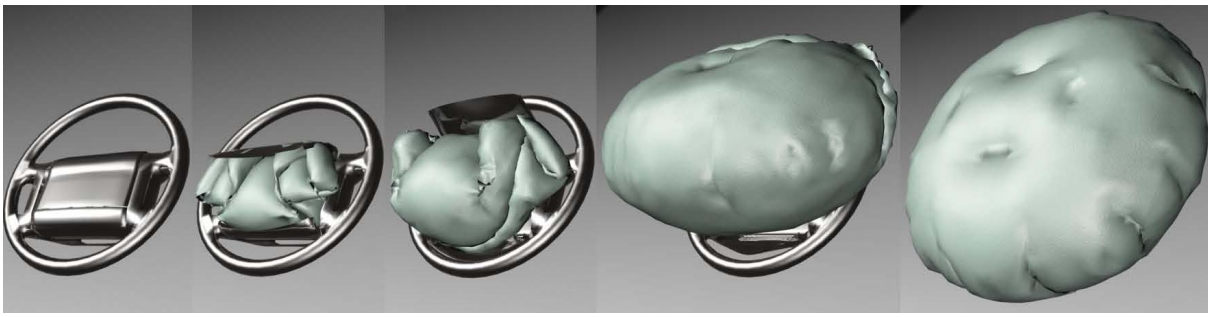


Fig. 12. **Airbag:** This deforming airbag has 9.1K shell elements. VolCCD performs inter-object and intra-object CCD queries at 88ms on average.

n_t of a triangle, and tests the relative positions of v and f with respect to the plane defined by n_t. On the other hand, CSAT is used to check for separate directions and also extends to volumetric elements. Moreover, CSAT involves fewer arithmetic operations. In practice, for a single VF pair, the cost of CSAT is about two times lower than coplanarity-based culling. In terms of performing four VF pairwise tests between a tetrahedron and a vertex, CSAT is about six times faster than coplanarity-based culling.

Continuous OBB test: Redon et al. [2002], Levine [2000], and Eberly [2000] used continuous OBB tests to deal with (moving) rigid models. OBB trees are computed once for these models, and checked for overlap among objects undergoing rigid motion based on a "continuous OBB overlap test". However, for deformable objects, OBBs may not be a good choice for bounding volume, as the updating cost is relative high ($15 \sim 25$X slower) over k-DOPs. For example, the OBB overlap test includes about 180 operations: 15 comparisons, 60 add/sub, 81 mult, and 24 abs operations, which are much more expensive than our CSAT test. Therefore, we use k-DOPs because they provide a good balance between the culling efficiency and the updating cost.

7.3 Limitations

Our culling techniques have several limitations. The culling efficiency of assignment culling depends on the connectivity information between volumetric elements. Also, if there are high numbers of topological changes, then the cost of updating O-Set can be high. The efficiency and effectiveness of CSAT depends on the choice of the separating axis. Our current method uses a simple heuristic to estimate a good separating axis, and it may not work well in all cases. The CSAT culling algorithm has an additional overhead. If the number of false positives after the BVH culling are relatively low, the use of the CSAT culling algorithm can result in reduced performance, although the slowdown would be minor because of the small overhead of the CSAT culling method. Also, in some simulations where we do not have strong external forces, contacts occurred in constrained geometric configurations, or high impacts with fast moving objects, detecting contacts based only on the surface boundary may be sufficient to generate realistic results.

8. CONCLUSION AND FUTURE WORK

We have presented a novel algorithm for CCD between volume meshes. Our BVH based approach is applicable to the contact handling of various kinds of volumetric elements that is needed in different FEM applications, physically-based simulation, engineering analysis, and medical applications. Moreover, our CCD algorithms accurately compute the first time of contact and hereby make it possible to use a larger simulation time step while providing a high-quality contact information. We have highlighted its performance on many benchmarks and observed more than an order of magnitude performance improvement. To the best of our knowledge, VolCCD is the first interactive algorithm that can perform accurate collision checking for FEM or other simulations that use a few thousands to hundreds of thousands of tetrahedra or triangular prisms.

There are many avenues for future work. In addition to addressing the limitations of our method mentioned before, we would like to improve the performance of our method by employing multiple CPU cores or GPUs [Tang et al. 2011]. We expect that all the proposed culling methods map very well with parallel implementation. Even though we have drastically reduced the number of false positives, we still perform a high number of elementary tests for complex benchmarks, such as car crash simulation composed of 1.2M shells. We would like to design exact and less conservative culling methods to improve the performance of our method. Finally, we would also like to integrate our algorithm with FEM simulators and design interactive simulation techniques.

ACKNOWLEDGMENTS

We would like to thank Jeremie Allard and SOFA team for providing Octopi Benchmark [Allard et al. 2010] and Liver Cutting Benchmark, and helping us with the rendering. We thank Miguel Otaduy for useful discussions and thoughtful comments on an earlier draft. The Car Crash Benchmark and Airbag benchmarks from the Finite Element Model Archive provided by NCAC [National-Crash-Analysis-Center 2010]. We also want to thank Daniel Heiserer from BMW for many useful discussions.

This research is supported in part by National Basic Research Program of China (No. 2011CB302205), ARO Contract W911NF-10-1-0506, NSF awards 0917040, 0904990 and 1000579, and Intel. Tang is supported in part by Natural Science Foundation of China (No. 60803054), Natural Science Foundation of Zhejiang, China (No. Y1100069), Important Science and Technology Specific Project of Zhejiang, China (2008C01059-4). Yoon is supported in part by MKE/MCST/IITA [2008-F-033-02], KRF-2008-313-D00922, KMCC, MSRA, BK, DAPA/ADD (UD080042AD), MKE/KEIT [KI001810035261], MEST/NRF/WCU (R31-2010-000-30007-0), and VFX simulation techniques.

REFERENCES

ABAQUS. 2003. ABAQUS 6.4. analysis user's manual. *Hibbitt, Karlsson & Sorensen, Inc.*

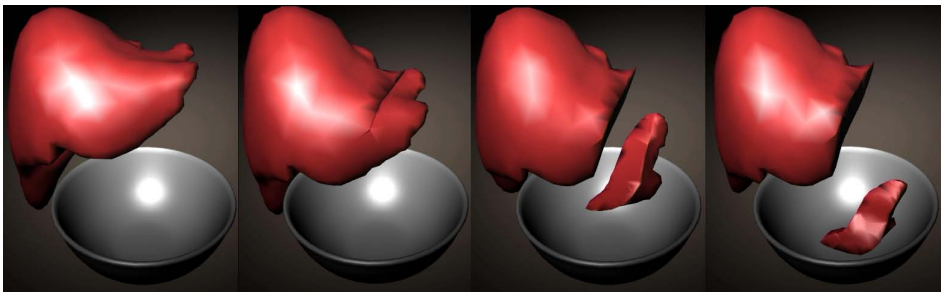


Fig. 13. **Cutting Liver:** A liver is cut during a surgical operation. The model has 3874 tetrahedra initially and is deformed to have 4338 tetrahedra because of the cutting operation. The average CCD query time is 53.2ms per frame. We achieve more than 10 times improvement over prior methods.

ALLARD, J., FAURE, F., COURTECUISSE, H., FALIPOU, F., DURIEZ, C., AND KRY, P. G. 2010. Volume contact constraints at arbitrary resolution. In *ACM SIGGRAPH 2010*. ACM, 1–10.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21,* 3, 594–603.

CURTIS, S., TAMSTORF, R., AND MANOCHA, D. 2008. Fast collision detection for deformable models using representative-triangles. In *Symp. on Interactive 3D graphics and games*. 61–69.

EBERLY, D. H. 2000. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann.

ERLEBEN, K., DOHLMANN, H., AND SPORRING, J. 2005. The adaptive thin shell tetrahedral mesh. In *WSCG (Journal Papers)*. 17–24.

FAURE, F., BARBIER, S., ALLARD, J., AND FALIPOU, F. 2008. Image-based collision detection and response between arbitrary volume objects. In *Symp. on Computer Animation*. 155–162.

FISHER, S. AND LIN, M. C. 2001. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Proceedings of the Eurographic workshop on Computer animation and simulation*. Springer-Verlag New York, Inc., New York, NY, USA, 99–111.

FLEISSNER, F., EBERHARD, P., BISCHOF, C., BCKER, M., GIBBON, P., JOUBERT, G. R., MOHR, B., (EDS, F. P., FLEISSNER, F., AND EBERHARD, P. 2007. Load balanced parallel simulation of particle-fluid dem-sph systems with moving boundaries. In *Proceedings of Parallel Computing: Architectures, Algorithms and Applications*. 37–44.

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

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

HALLQUIST, J. 2006. LS-DYNA theory manual. *Livermore Software Technology Corporation*.

HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2003. Real-time volumetric intersections of deforming objects. In *Proceedings of Vision Modeling Visualization (VMV'03)*. 461–468.

HEIDELBERGER, B., TESCHNER, M., KEISER, R., MÜLLER, M., AND GROSS, M. 2004. Consistent peneration depth estimation for deformable collision response. In *Proceedings of Vision Modeling Visualization (VMV'04)*. 330–346.

HEO, J.-P., SEONG, J.-K., KIM, D., OTADUY, M. A., HONG, J.-M., TANG, M., AND YOON, S.-E. 2010. FASTCD: Fracturing-aware stable collision detection. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*.

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

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Symp. on Computer animation*. 131–140.

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.

LEVINE, R. 2000. Collisions of moving objects. `http://realtimecollisiondetection.net/files/levine_swept_sat.txt`.

LOMBARDO, J.-C., PAULE CANI, M., AND NEYRET, F. 1999. Real-time collision detection for virtual surgery. In *In Computer Animation*. 26–28.

LS-DYNA. 2001. Contact modeling in LS-DYNA. `http://www.dynasupport.com/tutorial/contact-modeling-in-ls-dyna/contact-types`.

MATHIAS, E. AND GU, L. 2007. Hierarchical spatial hashing for real-time collision detection. In *IEEE International Conf. on Shape Modeling and Applications 2007*. 61–70.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*. New York, NY, USA, 113–124.

NATIONAL-CRASH-ANALYSIS-CENTER. 2010. Finite element model archive. Website. `http://www.ncac.gwu.edu/vml/models.html`.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25,* 4, 809–836.

O'BRIEN, J. F. 2000. Graphical modeling and animation of brittle fracture. Ph.D. thesis, Georgia Institute of Technology, Atlanta, Georgia.

O'BRIEN, J. F. AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '99. New York, NY, USA, 137–146.

PARKER, E. G. AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Symp. on Computer Animation*. 165–175.

PLIMPTON, S., ATTAWAY, S., HENDRICKSON, B., SWEGLE, J., VAUGHAN, C., AND GARDNER, D. 1998. Parallel transient dynamics simulations: Algorithms for contact detection and smoothed particle hydrodynamics. *Journal of Parallel and Distributed Computing 50,* 1-2, 104–122.

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

REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. *Proc. of Eurographics (Computer Graphics Forum) 21,* 3, 279–288.

SIFAKIS, E., DER, K. G., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 73–80.

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., CURTIS, S., YOON, S.-E., AND MANOCHA, D. 2009. ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics 15,* 4, 544–557.

TANG, M., KIM, Y. J., AND MANOCHA, D. 2010b. Continuous collision detection for non-rigid contact computations using local advancement. *Proceedings of International Conference on Robotics and Automation*.

TANG, M., MANOCHA, D., LIN, J., AND TONG, R. 2011. Collision-streams: Fast GPU-based collision detection for deformable models. In *I3D '11: Proceedings of the 2011 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 63–70.

TANG, M., MANOCHA, D., AND TONG, R. 2010a. Fast continuous collision detection using deforming non-penetration filters. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. ACM, New York, NY, USA, 7–13.

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of Vision Modeling Visualization (VMV'03)*. 47–54.

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.

VOLINO, P. AND MAGNENAT-THALMANN, N. 2006. Resolving surface collisions through intersection contour minimization. *ACM Trans. Graph. 25,* 3, 1154–1159.

VOLINO, P. AND THALMANN, N. M. 1994. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum (Eurographics) 13,* 3, 155–166.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2009) 28*, 76:1–76:10.

ZHANG, X., REDON, S., LEE, M., AND KIM, Y. J. 2007. Continuous collision detection for articulated models using taylor models and temporal culling. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2007) 26,* 3, 15.

ZHU, Y., SIFAKIS, E., TERAN, J., AND BRANDT, A. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Trans. Graph. 29,* 2, 1–18.

Appendix

Proof of Theorem and Corollaries

Theorem 1: CSAT for two moving vertices: *Suppose two moving vertices p and q, defined by their positions at the time interval* $[0, 1]$ *(p_0 & p_1 and q_0 & q_1, respectively), and a separating axis defined by* $\boldsymbol{L}$ *($\neq 0$). If* $(q_0 - p_0) \cdot \boldsymbol{L}$ *and* $(q_1 - p_1) \cdot \boldsymbol{L}$ *have the same sign, q will not overlap with p during the time interval.*

PROOF. We define $\boldsymbol{v_q}$ and $\boldsymbol{v_p}$ as $\boldsymbol{v_q} = q_1 - q_0$ and $\boldsymbol{v_p} = p_1 - p_0$ respectively. We also let $\boldsymbol{L'}$ be the normalized vector of the $\boldsymbol{L}$, i.e., $\boldsymbol{L'} = \frac{\boldsymbol{L}}{||\boldsymbol{L}||}$.

Then the projected distance of q and p along $\boldsymbol{L}$, d_{qp}, is given as:

$$\begin{aligned} d_{qp} &= (q - p) \cdot \boldsymbol{L'} \\ &= [(q_0 - p_0) + t * (\boldsymbol{v_q} - \boldsymbol{v_p})] \cdot \boldsymbol{L'} \\ &= \frac{(q_0 - p_0) \cdot \boldsymbol{L} + t * (\boldsymbol{v_q} - \boldsymbol{v_p}) \cdot \boldsymbol{L}}{||\boldsymbol{L}||}, \end{aligned} \tag{5}$$

where t is a time parameter and $t \in [0, 1]$. As a result, d_{pq} is within a bound as follows:

$$d_{pq} \in [\frac{(q_0 - p_0) \cdot \boldsymbol{L}}{||\boldsymbol{L}||}, \frac{(q_0 - p_0) \cdot \boldsymbol{L} + (\boldsymbol{v_q} - \boldsymbol{v_p}) \cdot \boldsymbol{L}}{||\boldsymbol{L}||}]. \tag{6}$$

This is equivalent to:

$$d_{pq} \in [\frac{(q_0 - p_0) \cdot \boldsymbol{L}}{||\boldsymbol{L}||}, \frac{(q_1 - p_1) \cdot \boldsymbol{L}}{||\boldsymbol{L}||}] \tag{7}$$

Since $(q_0 - p_0) \cdot \boldsymbol{L}$ and $(q_1 - p_1) \cdot \boldsymbol{L}$ have the same sign, d_{pq} will not be zero. This implies that the projections of q and p on $\boldsymbol{L}$ are disjoint, and thus q and p do not overlap. □

Corollary 1: CSAT between a vertex and a triangle: *Suppose a vertex p and a triangle f defined by their positions at the time interval* $[0, 1]$ *(p_0 & p_1 for p, and a_0, b_0, c_0 & a_1, b_1, c_1 for f, respectively), and a vector* $\boldsymbol{L}$ *($\neq 0$). If the following 6 expressions have the same sign:*

$$\begin{aligned} &(a_0 - p_0) \cdot \boldsymbol{L}, \quad (a_1 - p_1) \cdot \boldsymbol{L}, \quad (b_0 - p_0) \cdot \boldsymbol{L}, \\ &(b_1 - p_1) \cdot \boldsymbol{L}, \quad (c_0 - p_0) \cdot \boldsymbol{L}, \quad (c_1 - p_1) \cdot \boldsymbol{L} \end{aligned} \tag{8}$$

then p and f will not overlap during the time interval.

PROOF. Let q be a point inside f, and it can be expressed as a convex combination of a, b, c, i.e., $q = u * a + v * b + w * c$. Here, $u, v, w \in [0, 1]$, and $u + v + w = 1$. The projected distance of p and q along $\boldsymbol{L}$ is given as:

$$\begin{aligned} d_{qp} &= (q - p) \cdot \boldsymbol{L'} = (u * a + v * b + w * c - p) \cdot \boldsymbol{L'} \\ &= (u * (a - p) + v * (b - p) + w * (c - p)) \cdot \boldsymbol{L'} \end{aligned}$$

We build on CSAT for two vertices. Since all the scalar values in Equation (8) have the same sign, $(a - p) \cdot \boldsymbol{L'}$, $(b - p) \cdot \boldsymbol{L'}$, and $(c - p) \cdot \boldsymbol{L'}$ will have the same sign as well. Therefore, d_{qp} will not be zero. It means that p and f will not overlap during the time interval. □

Corollary 2: CSAT between a vertex and a volumetric element: *Suppose a vertex p and a volumetric element o (with* n_v *vertices and* n_f *triangles) defined by their positions during the time interval* $[0, 1]$ *(p_0 & p_1 for p, and q_0^i & q_1^i for o, $i \in [0, n_v - 1]$), and a vector* $\boldsymbol{L}$*($\neq 0$). If the following* $2 * n_v$ *algebraic expressions have the same sign:*

$$\begin{aligned} &(q_0^i - p_0) \cdot \boldsymbol{L}, \quad i \in [0, n_v - 1], \\ &(q_1^i - p_1) \cdot \boldsymbol{L}, \quad i \in [0, n_v - 1] \end{aligned} \tag{9}$$

then p and all the n_f *triangles of o will not overlap during the time interval.*

PROOF. Since all of n_f triangles are defined by these n_v vertices, all the scalar values in Equation (8) have the same sign for all the triangles of o. As a result, all the VF tests corresponding to different triangles of o can be culled away. □

Corollary 3: CSAT between two edges: *Suppose two edges,* e^1 *and* e^2*, defined by their end positions during* $t \in [0, 1]$ *(a_0, b_0 &*

Algorithm 2 CSAT culling between two volumetric elements o_i and o_j.

```
for all vertex v1 ∈ oi do
    Perform CSAT culling between v1 and oj with Corollary 2
end for
for all vertex v2 ∈ oj do
    Perform CSAT culling between v2 and oi with Corollary 2
end for
for all edge e1 ∈ oj do
    Perform CSAT culling between e1 and oi with Corollary 4
end for
```

a_1, b_1 for e^1, and c_0, d_0 & c_1, d_1 for e^2), and a vector $\boldsymbol{L}(\neq 0)$. If the following eight expressions have the same sign:

$$\begin{aligned}&(a_0 - c_0)\cdot\boldsymbol{L}, (a_1 - c_1)\cdot\boldsymbol{L}, (b_0 - c_0)\cdot\boldsymbol{L}, (b_1 - c_1)\cdot\boldsymbol{L},\\ &(a_0 - d_0)\cdot\boldsymbol{L}, (a_1 - d_1)\cdot\boldsymbol{L}, (b_0 - d_0)\cdot\boldsymbol{L}, (b_1 - d_1)\cdot\boldsymbol{L}\end{aligned} \quad (10)$$

then e^1 and e^2 do not overlap during the time interval.

PROOF. Based on CSAT for two vertices, if the first four expressions have the same sign then the projections of a and b are below/above the projection of c along $\boldsymbol{L}$. If the last four expressions have the same sign, then the projections of a and b are below/above the projection of d along $\boldsymbol{L}$. All the points on e^1, e^1_t, can be expressed as a convex combination of a and b. Similarly, all the points on e^2, e^2_t, can be expressed as a convex combination of c and d. So the projection of e^1_t is below/above the projection of e^2_t. If their projections on the axis do not overlap, we can conclude that the original edges will not overlap. □

Corollary 4: CSAT between an edge and a volumetric element: *Suppose an edge e and a volumetric element o (with n_v vertices and n_e edges) defined by their positions at time interval $[0, 1]$ (c_0 & c_1 and d_0 & d_1 for e, and q^i_0 & q^i_1 for o, $i \in [0, n_v - 1]$), and a vector $\boldsymbol{L}(\neq 0)$. If the following $4 * n_v$ expressions have the same sign:*

$$\begin{aligned}&(q^i_0 - c_0)\cdot\boldsymbol{L}, i \in [0, n_v - 1], (q^i_1 - c_1)\cdot\boldsymbol{L}, i \in [0, n_v - 1],\\ &(q^i_0 - d_0)\cdot\boldsymbol{L}, i \in [0, n_v - 1], (q^i_1 - d_1)\cdot\boldsymbol{L}, i \in [0, n_v - 1]\end{aligned} \quad (11)$$

then e and o do not overlap during the time interval.

PROOF. Since all the n_e edges are defined by these n_v vertices, the Equations (10) holds for all these edges. As a result, all the EE pairs between e and o are culled away. □

Algorithms

The pseudo-code for performing CSAT culling between two volumetric elements, o_i and o_j, is shown as Algorithm 2. It is executed as 3 stages. We first perform CSAT culling between every vertex $v_1 \in o_i$ with o_j, then perform CSAT culling between every vertex $v_2 \in o_j$ with o_i, finally perform CSAT culling between every edge $e_1 \in o_i$ with o_j.

The O-Set is constructed by using Algorithm 3. In this algorithm, all the adjacent volumetric element pairs are traversed. For each one of these pair $\{o_i, o_j\}$, we check all the feature pairs $\{elm_1, elm_2\}$, here $elm_1 \in o_i$ & $elm_2 \in o_j$. If one of the volumetric elements in the incident set of elm_1, o_k, is not adjacent to o_j, then the feature pair $\{elm_1, elm_2\}$ is checked based on testing a non-adjacent element pair $\{o_k, o_j\}$. As a result, we do not need to store $\{elm_1, elm_2\}$ in the O-Set. Similarly, if one of the volumetric elements in the incident set of elm_2, o_l, is not adjacent to o_i, then the feature pair $\{elm_1, elm_2\}$ is checked for collision while handling the non-adjacent element pair $\{o_i, o_l\}$. As a result, we do not include the pair in the O-Set. Otherwise, the feature pair $\{elm_1, elm_2\}$ is inserted in the O-Set.

Algorithm 3 O-Set Construction: Compute the O-Set based on connectivity information.

```
for all adjacent element pair {oi, oj} do
    for all feature pair {elm1 ∈ oi, elm2 ∈ oj} do
        for all elements ok ∈ the incident set of elm1 do
            if ok is not adjacent to oj then
                return
            end if
        end for
        for all elements ol ∈ the incident set of elm2 do
            if ol is not adjacent to oi then
                return
            end if
        end for
    end for
    add {elm1, elm2} into O-Set
end for
```

Algorithm 4 Static Feature Assignment: Assign each feature to and only to a volumetric element.

```
for all volumetric elements oi do
    for all feature elmj ∈ oi do
        if elmj has not been not assigned then
            assign elmj to oi
        end if
    end for
end for
```

Algorithm 5 Dynamic Feature Assignment: assignment for a test between a feature elm_1 from o_1 and another feature elm_2 from o_2.

```
for all volumetric elements oi ∈ the incident set of elm1 do
    for all volumetric elements oj ∈ the incident set of elm2 do
        if oi is not adjacent to oj then
            if oi = o1 AND oj = o2 then
                ElementaryTest(elm1, elm2)
            else
                return
            end if
        end if
    end for
end for
```

Algorithm 4 highlights the details of static feature assignment computation. A greedy approach is used by traversing all the volumetric elements. For each volumetric element o_i, we check its features. For each feature elm_j of o_i, if it has not been assigned to any volume element, then it is assigned to o_i.

Algorithm 5 shows the details of the dynamic feature assignment algorithm. The dynamic feature assignment is performed before we perform an exact elementary test for a feature pair $\{elm_1, elm_2\}$, where elm_1 belongs to o_1 and elm_2 belongs to o_2. We traverse each element pair $\{o_i, o_j\}$, where o_i is incident to elm_1, and o_j is incident to elm_2. In the case that o_i is not adjacent to o_j, we perform the exact test between elm_1 and elm_2 only if $o_i = o_1$ and $o_j = o_2$.