2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Continuous Collision Detection for Non-rigid Contact Computations using Local Advancement

Min Tang, Young J. Kim and Dinesh Manocha

Abstract— **We present a novel algorithm to perform continuous collision detection(CCD) between non-rigid, deformable models using local advancement. Given the initial and final configurations of a deformable model, our algorithm computes linear deformation by interpolating the vertices from the initial to the final configurations with a straight line path and checks for collision along that path. Our approach is applicable to polygon-soup models with arbitrary topology, handles self-collisions and makes no assumption about the underlying non-rigid motion. We accelerate the algorithm by computing motion bounds on the primitives and their bounding volumes. These bounds are combined with hierarchical culling techniques and used for fast collision checking. In practice, we have observed up to four times improvement in running time because of local advancement.**

I. Introduction

Fast and reliable collision detection is an important problem in robotics, computational geometry, virtual environment, and computer graphics. Due to its importance and wide applicability, collision detection has been extensively studied in different fields for more than three decades. In this paper, we primarily focus on accurate and continuous collision detection (CCD) between non-rigid and deformable models. Given two discrete instances or configurations of an object, the CCD algorithm computes a continuous trajectory between those instances, and checks for collisions along that trajectory. In terms of handling deformable models, we check for collision between two different objects (i.e. inter-object collision) as well as between the primitives of the same object (i.e. intra-object or self-collision). If there is a collision, the resulting algorithm also computes the first time of contact along the trajectory.

The problem of CCD computation arises in many applications, including local planning in sample-based planners, deformable simulations, haptics, grasping, etc. Most of the prior work in these areas has focused on fast collision checking between rigid or articulated models. There is increasing use of deformable models in motion planning [1], [2] and soft finger grasping and manipulation [3]. In particular, for constraint dynamics and motion planning based on contact-space sampling, it is very important to find valid contact configurations (i.e. time of contact information) where non-penetration constraints are strictly enforced.

A key approach in terms of designing an efficient CCD algorithm for deformable models is to identify collision-free regions, including no self-contacts, in the models and cull them away. The culling methods tend to use bounding volume hierarchies (BVHs) to accelerate the computation. These methods work well for inter-object collision checking, but may not be very effective for self-collisions [4], [5], [6]. This is due to the fact that the bounding volumes of adjacent or neighboring primitives tend to overlap and the resulting algorithms need to perform exact CCD tests between these primitives that boil down to solving a cubic equation for each feature pair, i.e. vertex-face (VF) or edge-edge (EE) of the primitives. As a result, prior CCD algorithms for deformable models tend to be rather conservative and may result in a high number of false positive collision checks between the primitives.

Main Contribution: In this paper, we propose a new algorithm to perform efficient primitive-level CCD tests, which can be used to find a time of contact for VF and EE contacts. Our local advancement (LA) algorithm is a generalization of conservative advancement for rigid models [7] to deforming triangles and relies on a simple and tight bound computation on the motion of vertex (V), edge (E), and face (F) primitives. Thus, our approach is orthogonal to the existing hierarchical or other culling methods and can be easily combined with these methods. Moreover, our formulation can also be extended to bounding volume hierarchies (i.e. dynamic bounding volume). In practice, our method is compact and provides tighter bounds on the motion as compared to prior approaches. We can easily combine our algorithm with prior CCD algorithms and improve the performance of collision queries by a factor of 1.3∼4 on different benchmarks. Moreover, our primitive-level CCD tests do not require any high-order polynomial solvers, and thus are more robust and non susceptible to numerical problems. In practice, our algorithm can compute the time of contact for deformable models consisting of hundreds of thousands of triangles in a few hundreds milli-seconds on a PC equipped with an Intel Core Q9450 2.66GHz CPU.

Organization: The rest of this paper is organized as follows. In Sec.II, we briefly review the related work, and give an overview of our algorithm in Sec.III. In Sec.IV and Sec.V, we present our methods to apply LA to bounding volumes and the primitives, respectively. We present experimental results in Sec.VI.

M. Tang and Y. J. Kim are with the Department of Computer Science and Engineering at Ewha Womans University in Seoul, Korea. {tangmin|kimy}@ewha.ac.kr
D. Manocha is with the Department of Computer Science at the University of North Carolina at Chapel Hill, U.S.A. dm@cs.unc.edu

II. Previous Work

In this section, we give a brief survey on continuous collision detection algorithms for deformable models.

A. Bounding Volume Hierarchies

In order to efficiently compute potentially-colliding primitive, bounding volume hierarchies (BVHs) have been used for both rigid and non-rigid models. Unlike rigid models, the BVHs of deformable models needs to be refit or rebuilt as the underlying models deform. In order to lower the update cost, relatively simple bounding volumes such as axis-aligned bounding boxes (AABBs) [8], [9], [10], [11], spheres[12] and k-DOPs [13], [4] have been used. When the deformation is relatively small, refitting BVH while keeping its topological structure is often sufficient to maintain the tightness of the hierarchy [8], [12]. However, when the deformation is large or the model undergoes topological changes, these algorithms reconstruct the BVH to obtain good culling efficiency. This reconstruction is often performed in a lazy top-down manner since the reconstruction cost is generally higher than the refitting cost [14]. Other methods use selective reconstruction to reduce the reconstruction cost [5], [6].

B. Self-collision Checking

The main distinction between rigid and non-rigid CCD computation is that non-rigid CCD algorithms need to check for self-collisions within a model. The self-collisions can be classified into two types: self-collision between adjacent and between non-adjacent triangles or the primitives. In practice, the cost of self-collisions can account for 50-90% of the total running time of the algorithm [9]. The normal cone technique was proposed to cull non-self-colliding triangles by Provot [15] for discrete collision detection, and was extended to CCD by Tang *et al.* [4]. The chromatic decomposition technique can also be used for self-collisions [9].

C. Reduction in Pairwise Primitive Tests

In order to perform a collision test between the triangle primitives, numerical root-finding methods for polynomials have been used [15], [16]. In this case, for a pair of triangles, fifteen elementary feature pairs of VF and EE contacts need to be tested. In order to avoid redundant elementary tests, the connectivity and adjacency information of a deformable mesh can be used [9], [4], [17] or feature-based hierarchies [10] can be also employed.

D. Parallel and Hybrid Methods

By exploiting the abundant parallelism residing in self-collision detection, some recent works have been proposed to utilize multi-core CPUs [18] [19], GPUs [20] and their hybrid combinations [11]. These methods also rely on the similar culling methods based on BVH-based culling and elementary test reductions.

III. Overview

In this section, we provide some preliminary concepts relevant to our algorithm and give an overview of our approach.

A. Preliminaries

Conservative Advancement: The goal of continuous collision detection is to compute the first time of contact (ToC), τ between two moving objects. The original idea of conservative advancement (CA) was developed for rigid, convex models [7], and later extended to non-convex [21], articulated models, and polygon-soup models [22]. Compared to other prior CCD algorithms, the CA-based algorithms are conceptually simple, easy to implement and exhibit better performance.

Given one movable object $\mathscr{A}(t)$ and fixed object $\mathscr{B}$ (both of them are convex) with a constant linear motion for $\mathscr{A}(t)$ in the configuration space, CA is a simple technique to calculate the lower bound on τ between $\mathscr{A}(t)$ and $\mathscr{B}$ by repeatedly advancing one object $\mathscr{A}(t)$ toward another $\mathscr{B}$ by dt_i without any collision by using the following bound:

$$dt_i \leq \frac{d(\mathscr{A}(t),\mathscr{B})}{\mu} \tag{1}$$

where $d(\mathscr{A}(t),\mathscr{B})$ is the closest distance between $\mathscr{A}(t)$ and $\mathscr{B}$, μ is an upper bound of the motion of $\mathscr{A}(t)$ projected onto $d(\mathscr{A}(t),\mathscr{B})$. The CA is iterated until $d(\mathscr{A}(t),\mathscr{B}) < \varepsilon$. Then, the ToC is $\tau = \sum dt_i$. We generalize this idea to the bounding volumes of the deformable primitives in Sec.IV, and deforming triangles and feature pairs in Sec.V, and call it local advancement (LA).

Motion Formulation: CCD algorithms require an explicit representation of the underlying trajectory of a moving object. If an explicit trajectory is not available, an interpolating motion is computed based on the given positions of the object. In case of CCD computation between deformable models, the interpolating motion is often calculated by simply linearly interpolating the position of the vertices between successive object configurations in the workspace [15], [14]. We also use the linearly interpolating motion for deformable models and use it to compute the motion bound μ in Eq.1 (more details in Sec.V).

B. Our Approach

At a broad level, existing CCD algorithms proceed in two stages: BVH-level collision culling and elementary-level feature pair tests. Our CCD algorithm also uses this framework, but the actual computation based on LA is quite different from prior methods. Other algorithms used to improve the overall performance, such as BVH restructuring or self-collision culling are complementary to our approach and can be easily combined.

For BVH-level collision culling, we use the feature-based hierarchy [10], as opposed to the object-based hierarchy, because of its culling efficiency. More specifically, we build a hierarchy for each feature in the model such as V, E, and F, and compute the extent of the motion of V/E/F under deformation using CA (explained in more detail in Sec.IV).

For an elementary test between the VF and EE feature pairs, many existing algorithms rely on a coplanarity test to find the ToC (time-of-contact) between the VF or EE pairs, which can be represented as solving a cubic equation based on the linear interpolating motion assumption. However, in our algorithm, we use the LA technique to find the ToC between the feature pairs, as explained in Sec.V. We also describe a method to use LA to find the ToC between the triangle pairs, when the object-based hierarchies are used for collision-culling.

IV. Local Advancement for Bounding Volume

As described in Sec.II, many existing algorithms rely on bounding volumes (BVs) to perform collision culling. In particular, axis-aligned bounding boxes (AABBs) and its generalization to k discrete orientation polytopes k-DOP [23] are commonly used because of their simplicity and efficiency. In this section, we explain how one can apply LA to two types of bounding volumes (AABB and k-DOP) when they undergo deformable motion. The result of this computation is the first time of contact (ToC) between two overlapping deformable BVs, and is used as a lower bound for ToC between the primitives contained in the BVs.

A. The AABB Case

Given a time-dependent, deformable object $\mathscr{A}(t)$ at the initial configuration $\mathscr{A}(0)$ and final configuration $\mathscr{A}(1)$, many existing algorithms attempt to bound the motion of $\mathscr{A}(t)$ by computing the BVs $\mathbf{BV}_{\mathscr{A}(0)}$ and $\mathbf{BV}_{\mathscr{A}(1)}$ at the initial and final configurations, respectively, and compute another BV that encloses $\mathbf{BV}_{\mathscr{A}(0)}$ and $\mathbf{BV}_{\mathscr{A}(1)}$, as illustrated in Fig.1-(a). In this case, the choice of BV is an AABB. This is rather simple, but can be overly conservative when the underlying motion has large deformation. In our case, as illustrated in Fig.1-(b), we consider the swept volume of $\mathbf{BV}_{\mathscr{A}(t)}$ and compute its ToC against other BVs.

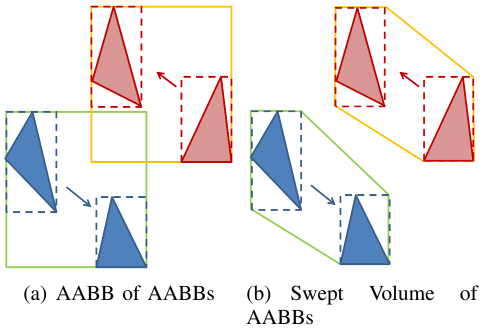


Fig. 1. **Bounding Volume Comparisons.** (a) The AABB that enclosed the AABBs at the initial and final configurations is computed. (b) The swept volume from the initial to final configurations is shown. In general, (b) is tighter than (a).

The main idea behind computing the ToC between two deformable AABBs is as follows. If $\mathbf{BV}_{\mathscr{A}(t)}$ and $\mathbf{BV}_{\mathscr{B}(t)}$ collide at time t, the separation distance along each principal axis between them will be zero. This computation is fairly easy and simple, especially when the underlying deformation is linearly interpolating motion.

We first consider the separation distance between AABBs $\mathbf{BV}_{\mathscr{A}(0)}, \mathbf{BV}_{\mathscr{B}(0)}$ at the initial configurations along each axis. If the distance is zero, the two AABBs overlap. Otherwise, we perform LA on these AABBs to find their ToC, τ. We explain this idea in 2D and it can be easily extended to higher dimensions. From the initial $\mathbf{BV}_{\mathscr{A}(0)}, \mathbf{BV}_{\mathscr{B}(0)}$ and final configurations $\mathbf{BV}_{\mathscr{A}(1)}, \mathbf{BV}_{\mathscr{B}(1)}$ of $\mathbf{BV}_{\mathscr{A}(t)}, \mathbf{BV}_{\mathscr{B}(t)}$, we compute the velocity of each boundary edge (i.e. face in 3D, respectively) of the AABB along X and Y axes. Without loss of generality, let us assume that $\mathbf{BV}_{\mathscr{A}(t)}$ is initially placed above and to the left of $\mathbf{BV}_{\mathscr{B}(t)}$ as illustrated in Fig. 2. Further, let us assume that $\mathbf{BV}_{\mathscr{A}(t)}, \mathbf{BV}_{\mathscr{B}(t)}$ move diagonally downwards and upwards, respectively. Let us define $\mathbf{v}^{+x}$ as the velocity[1] of the rightmost boundary edge of $\mathbf{BV}_{\mathscr{A}(0)}$ between $[0,1]$, and define $\mathbf{v}^{-x}$ as the velocity of the leftmost boundary edge of $\mathbf{BV}_{\mathscr{B}(0)}$ between $[0,1]$. Similarly, one can compute $\mathbf{v}^{+y}$, $\mathbf{v}^{-y}$ for the top and bottom boundary edges of $\mathbf{BV}_{\mathscr{B}(0)}$ and $\mathbf{BV}_{\mathscr{A}(0)}$ respectively. Then, we have:

$$dt_x = \frac{dx}{\mathbf{v}^{+x} - \mathbf{v}^{-x}}, \; dt_y = \frac{dy}{\mathbf{v}^{+y} - \mathbf{v}^{-y}} \tag{2}$$

where dx and dy represent the displacement between $\mathbf{BV}_{\mathscr{A}(0)}$ and $\mathbf{BV}_{\mathscr{B}(0)}$ along x and y axes, respectively. If $0 < dt_x < 1$ and $0 < dt_y < 1$, there is a contact between $\mathbf{BV}_{\mathscr{A}(t)}$ and $\mathbf{BV}_{\mathscr{B}(t)}$ during the time step of $[0,1]$, and the time of contact $\tau = \max(dt_x, dt_y)$; otherwise, there is no contact.

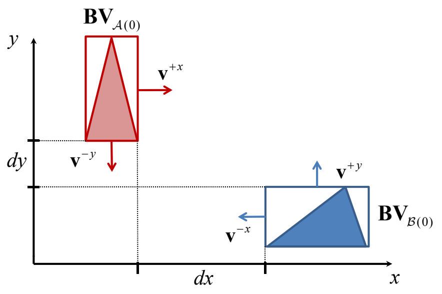


Fig. 2. **Time of Contact Computation between AABBs in 2D.**

B. The k-DOP Case

The LA algorithm used for AABBs can be extended to k-DOPs, which is a generalization of AABB with more projected intervals (an AABB is a k-DOP with $k = 6$). Similarly as the previous section, we define d_i as the separation distance along the ith orientation from $\mathbf{BV}_{\mathscr{A}}$ to $\mathbf{BV}_{\mathscr{B}}$. We also define the displacement of each boundary face in $\mathbf{BV}_{\mathscr{A}}$ and $\mathbf{BV}_{\mathscr{B}}$ as $\mathbf{v}^{+i}$ and $\mathbf{v}^{-i}$.

At the first time of contact, separation distance in some orientation will be equal to zero, so the ToC τ can be computed as $\tau = \max_{i \in [1, k/2]} t_i$, where $t_i = \frac{d_i}{\mathbf{v}^{+i} - \mathbf{v}^{-i}}$. If any of t_i is less than zero or greater than one, then $\mathbf{BV}_{\mathscr{A}}$ and $\mathbf{BV}_{\mathscr{B}}$ can not have a collision during the time step.

V. Local Advancement for Triangles and Feature Pairs

In this section, we show how to apply LA to deformable triangles and feature pairs such as VF and EE.

[1]It is equivalent to a displacement between the time interval of $[0,1]$.

A. The Triangle Case

Given two time-dependent, deformable triangles $\triangle_{\mathscr{A}}$ and $\triangle_{\mathscr{B}}$, the LA algorithm can be directly used to compute a safe time step size dt that can advance $\triangle_{\mathscr{A}}$ and $\triangle_{\mathscr{B}}$ without collisions (also shown in Fig.3):

$$dt = \frac{d(\triangle_{\mathscr{A}}, \triangle_{\mathscr{B}})}{\mu} \tag{3}$$

where the motion bound is $\mu = \max_{t\in[0,1]} ((\mathbf{v}_i - \mathbf{v}_j)\cdot \mathbf{n})$, and $\mathbf{v_i}$ and $\mathbf{v_j}$ are the velocities of arbitrary points $\mathbf{p}_i$ and $\mathbf{p}_j$ on $\triangle_{\mathscr{A}}$ and $\triangle_{\mathscr{B}}$, respectively. Here, $dt \geq 1$ means that the two triangles do not collide during the motion. Eq.3 is valid since the triangles $\triangle_{\mathscr{A}}$ and $\triangle_{\mathscr{B}}$ maintain the convexity during the deformation and the underlying deformation is also linear. Computing the Euclidean distance between two triangles $d(\triangle_{\mathscr{A}}, \triangle_{\mathscr{B}})$ is well-known [24]; thus our LA algorithm boils down to computing the tight motion bound μ efficiently and robustly.

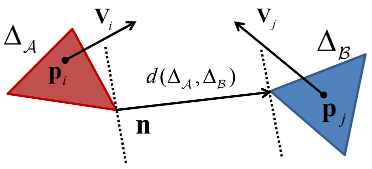


Fig. 3. **Local Advancement for Deformable Triangles.**

Let us denote the vertices of $\triangle_{\mathscr{A}}$ at the initial ($t = 0$) and final configurations ($t = 1$) as $V_i(0)$ and $V_i(1)$, $i = 1, \cdots, 3$, respectively. Similarly for the vertices of $\triangle_{\mathscr{B}}$ as $V'_j(0)$ and $V'_j(1)$, $j = 1, \cdots, 3$. Let us denote arbitrary points on $\triangle_{\mathscr{A}}$ and $\triangle_{\mathscr{B}}$ as $\mathbf{p}_i$ and $\mathbf{p'}_j$ with their barycentric coordinates as (α, β, γ) and $(\alpha', \beta', \gamma')$, respectively. Then, their velocities are $\mathbf{v}_i = \alpha(V_1(1) - V_1(0)) + \beta(V_2(1) - V_2(0)) + \gamma(V_3(1) - V_3(0))$ and $\mathbf{v'}_i = \alpha'(V'_1(1) - V'_1(0)) + \beta'(V'_2(1) - V'_2(0)) + \gamma'(V'_3(1) - V'_3(0))$. Then, we can find the motion bound μ as follows.

Lemma 1 *For deformable triangles $\triangle_{\mathscr{A}}$ and $\triangle_{\mathscr{B}}$, the motion bound μ for the triangle pair is:*

$$\mu \leq \max_{i=1,2,3} (\mathbf{L}_i \cdot \mathbf{n}) + \max_{j=1,2,3} (\mathbf{K}_j \cdot \mathbf{n}) \tag{4}$$

where

$$\mathbf{L}_i = (V_i(1) - V_i(0)),\ i = 1,2,3$$
$$\mathbf{K}_j = (V_j(0) - V_j(1)),\ j = 1,2,3$$

Proof:

$$\begin{aligned} \mu &= \max_{i,j} ((V_i - V_j)\cdot \mathbf{n}) \\ &\leq \max_{\alpha,\beta,\gamma} (\alpha \mathbf{L}_1 \cdot \mathbf{n} + \beta \mathbf{L}_2 \cdot \mathbf{n} + \gamma \mathbf{L}_3 \cdot \mathbf{n}) \\ &\quad + \max_{\alpha',\beta',\gamma'} (\alpha' \mathbf{K}_1 \cdot \mathbf{n} + \beta' \mathbf{K}_2 \cdot \mathbf{n} + \gamma' \mathbf{K}_3 \cdot \mathbf{n}) \end{aligned} \tag{5}$$

Let us analyze the first item of the inequality. As shown in Fig. 4, $\mathbf{L}'_k$ is the projection of $\mathbf{L}_k$ in the direction of $\mathbf{n}$, so any $\mathbf{L}'_k$ will be parallel with each other. It is obvious that the length of $\mathbf{L}'_k$ must satisfy $\|\mathbf{L}'_k\| \leq \max_{i=1,2,3} \|\mathbf{L}'_i\|$, as $\|\mathbf{L}'_i\| = \mathbf{L}_i \cdot \mathbf{n}$, so we can obtain

$$\max_{\alpha,\beta,\gamma} (\alpha \mathbf{L}_1 \cdot \mathbf{n} + \beta \mathbf{L}_2 \cdot \mathbf{n} + \gamma \mathbf{L}_3 \cdot \mathbf{n}) \leq \max_{i=1,2,3} (\mathbf{L}_i \cdot \mathbf{n}). \tag{6}$$

Similarly we can get the same result for the second item. ■

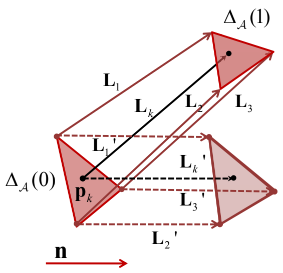


Fig. 4. **Motion Bound Computation for Deformable Triangles.**

B. The Feature Pair Case

We consider two types of feature pairs including VF and EE and apply the LA to find the ToC between the pairs. We start with the VF case first. In order to utilize the result from Lemma 1, we consider a triangle $\triangle_{\mathscr{A}}$ that corresponds to the face F and a triangle $\triangle_{\mathscr{B}}$ that degenerates to a vertex V. Furthermore, similar to Lemma 1, we compute the velocity of each vertex in F as $\mathbf{L}_i, i = 1,2,3$ and the velocity of V as $\mathbf{L}'_1$. Then, the corresponding motion bound can be computed as follows.

Lemma 2 *For a deformable triangle F and a vertex V, the motion bound μ can be given as:*

$$\mu \leq \max_{i=1,2,3} (\mathbf{L}_i \cdot \mathbf{n}) - \mathbf{L}'_1 \cdot \mathbf{n} \tag{7}$$

Proof: The lemma can be easily deduced from Lemma 1, as $\mathbf{K}_j = -\mathbf{L}'_1, j = 1,2,3.$ ■

Now we deal with the EE case. Similar to the VF case, we consider each edge in the EE feature pair as a degenerate triangle to an edge (by merging two vertices) and utilize the result from Lemma 1. Let us call the velocities for the end-points of one edge as $\mathbf{L}_i$ where $i = 1,2$, and those for the other edge as $\mathbf{L}'_j$ where $j = 1,2$. Then, the corresponding motion bound can be computed as follows.

Lemma 3 *For the deformable edges E and E′, the motion bound μ is:*

$$\mu \leq \max_{i=1,2} (\mathbf{L}_i \cdot \mathbf{n}) + \max_{j=1,2} \left(-\mathbf{L}'_j \cdot \mathbf{n}\right) \tag{8}$$

Proof: The lemma can be easily deduced from Lemma 1, as $\mathbf{K}_j = -\mathbf{L}'_j, j = 1,2$ and $\mathbf{K}_3 = -\mathbf{L}'_1$ ■

VI. Implementation and Result

In this section, we present the implementation results of our CCD algorithm and discuss its performance. We have implemented our CCD algorithm using C++ on a PC running Windows XP, equipped with an Intel Core Q9450 2.66GHz CPU and 3GB main memory. All the timings reported in this paper are generated using a single thread.

Since our approach is orthogonal to the existing BVH refitting and reconstruction step, we use an existing method such as [5] for our implementation, which is based on a lazy reconstruction method.

For each object, we construct a hybrid BVH with two layers, as illustrated in Fig.5. The first layer is built recursively in a bottom-up fashion and each node in the BVH includes a pair of k-DOP BVs where each of these k-DOP bounds their children BVs at the initial and final configurations, respectively. This hierarchy is used to perform the LA step explained in Sec.IV.B and to find the time of contact. We stop growing the first layer when the height of the layer exceeds a pre-defined threshold, and start building the second layer where each BV node bounds the extent of the motion of the children BV nodes using k-DOP. This hierarchy is used to find potentially overlapping BV nodes, which have been used by other existing CCD algorithms [5], [10], [9]. The reasons why we adopt a hybrid, layered BVH are as follows:

- We use the first layer to find the time of contact between the models.
- If we occupy the entire BVH only with the first layer, the construction cost and memory overhead will be high since we need to maintain a pair of k-DOPs.
- As we reach the root level in the BVH, the pair of k-DOPs from the first layer are likely to overlap. Thus, constructing the second layer (k-DOP of the extent of k-DOP) is more efficient for collision culling.

In our implementation, we set the maximum height of the first layer as three and it works well in practice.

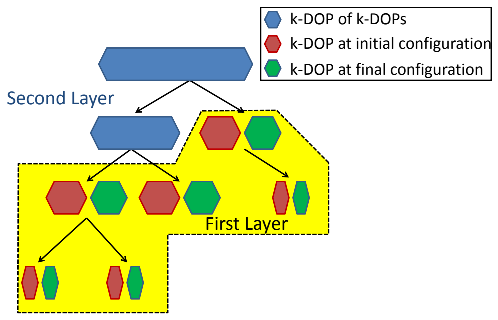
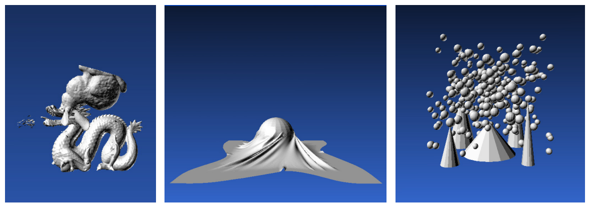


Fig. 5. **Hybrid BVH with Two Layers.** *The first layer is used to find the time of contact between the models, and the second layer is used to find potentially overlapping BV nodes.*

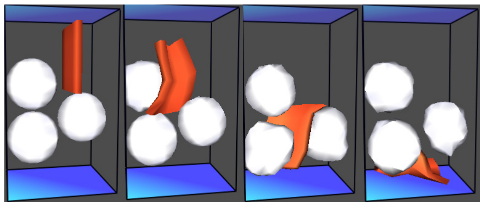
To test the performance of our algorithm, as shown in Fig.6, we use the UNC dynamic scene benchmarks[2] including the exploding dragon, cloth simulation with self-collisions and N-body simulation, which are used by many other CCD algorithms[10], [4], [11]. The benchmarking statistics are summarized in Table I. In this table, we show the timing of finding the global, single ToC for the entire model as well as that of finding per-triangle ToC. Our algorithm becomes more efficient when computing only the global ToC, as other techniques like temporal culling [25] can be used. Note that the global ToC is often sufficient for application that strictly impose non-penetration constraints such as the motion planning technique based on contact-space sampling [26]. We also compare the performance of our algorithm against that of the representative-triangle method [10], and our algorithm shows 1.3∼1.6 times performance improvement over [10]. Moreover, unlike [10], our algorithm does not use high-order polynomial solvers (i.e. cubic). For both the representative-triangle and our method, we use the same distance threshold 10^{-6} to find the ToC.

[2]`http://www.cs.unc.edu/~geom/DynamicB/`

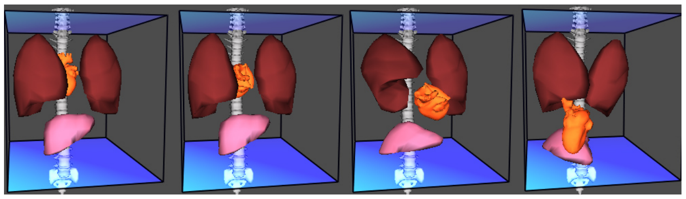


(a) Exploding Dragon (b) Cloth Simulation (c) N-body Simulation

Fig. 6. **UNC Dynamic Scene Benchmark.** (a) A bunny models drops on top of a dragon model and the dragon model breaks into smaller pieces. (b) A cloth model drapes over a ball model and as the ball rotates, the cloth is deformed and wrinkled. (c) Multiple balls undergo rigid or deformable motion and the balls collide with each other and with other obstacles.



(a) Bar and Spheres



(b) Human Organs

Fig. 7. **Deformable Motion Planning Benchmarks.** (a) A bar-like, deformable robot in orange falls down to the ground while avoiding deformable spherical objects in white. (b) A heart-shaped robot in orange is taken out of lung-shaped obstacles in dark red while avoiding a liver obstacle in pink and a spine obstacle in white. These models are all deformable.

Benchmark	# of Tri	R-Tri	Ours	
			All ToCs	Global ToC
Exploding dragon	252K	938ms	792ms	717ms
Cloth simulation	92K	350ms	286ms	274ms
N-body simulation	146K	1131ms	720ms	669ms

TABLE I

UNC Dynamics Benchmark Statistics. EACH COLUMN REPRESENTS, FROM LEFT TO RIGHT, THE BENCHMARK TYPE, THE NUMBER OF TRIANGLES IN THE MODELS, THE PERFORMANCE OF REPRESENTATIVE-TRIANGLE METHOD [10], AND AVERAGE QUERY TIME WITH OUR CA METHOD IN FINDING ALL THE TOCS AND THE GLOBAL TOC.

We also used a different set of benchmarks based on motion planing in an environment composed of deformable objects [1], in which both robots and obstacles can have non-rigid motions and the robots try to reach the goal configuration under such constraints such as shape preservation, collision response, manipulation and gravity forces. Specifically, as shown in Fig. 7-(a), there are three spheres, used as obstacles in the environment and a bar-like (shown

Benchmark	# of Tri	R-Tri		Ours	
		Boolean	ToC	Boolean	ToC
Bar/Spheres	636	7.2ms	14.9ms	3.4ms	10.6ms
Human Organs	14K	27.5ms	2986.3ms	22.5ms	727.5ms

TABLE II

Motion Planning Benchmark Statistics. Each column represents, from left to right, the benchmark type, the number of triangles, the performance of Representative-triangle method [10] to check whether the path is collision-free or not (Boolean answer) and to find the global ToC, and the performance of our algorithm.

in orange), deformable robot that needs to move from the start configuration to the goal configuration. In Fig. 7-(b), the benchmark environment simulates the internal structure of human body including spine, lung, and liver, and a heart-like robot (shown in orange) attempts to move from the start configuration to the goal configuration. The motion paths generated by the above motion planning method were used to test the performance of our algorithm, and we also compare the performance to that of the representative-triangle algorithm [10]. Our algorithm outperforms [10] by a factor of 1.4∼4. The benchmark statistics are summarized in Table II.

VII. Conclusion and Future Work

We have presented an algorithm to perform CCD between deformable models using local advancement. We observe up to four times speedup over prior CCD algorithms and our approach can be easily combined with other culling methods used to accelerate CCD computations. In terms of future work, we would like to integrate our algorithm into deformable dynamics simulation systems, in particular constraint-based dynamics. We are also interested in designing a special version of CCD algorithm that reports only Boolean answer very rapidly, which might be better suitable for deformable motion planning. Finally, we would like to apply our technique to grasp planning.

Acknowledgements

This research work was supported in part by the NRF grant funded by the Korean government(MEST) (No. 2009-0086684) and the IT R&D program of MKE/KEIT (2008-F-033-02, Development of Real-time Physics Simulation Engine for e-Entertainment). Dinesh Manocha was supported in part by ARO Contract W911NF-04-1-0088, NSF awards 0636208, 0917040 and 0904990, DARPA/RDECOM Contract WR91CRB-08-C-0137, and Intel. We would also like to thank Jyh-Ming Lien and Duk-Su Kim for their help and discussion.

References

[1] S. Rodriguez, J.-M. Lien, and N. M. Amato, "Planning motion in completely deformable environments," *Proc. of IEEE Conference on Robotics and Automation*, 2006.

[2] R. Gayle, W. Segars, M. C. Lin, and D. Manocha, "Path planning for deformable robots in complex environments," *Proceedings of Robotics: Systems and Science*, 2005.

[3] M. Ciocarlie, C. Lackner, and P. Allen, "Soft finger model with adaptive contact geometry for grasping and manipulation tasks," in *WHC '07: Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 219–224.

[4] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "Interactive continuous collision detection between deformable models using connectivity-based culling," *ACM Symposium on Solid and Physical Modeling*, pp. 25–36, 2008.

[5] T. Larsson and T. Akenine-Möller, "A dynamic bounding volume hierarchy for generalized collision detection," *Computers & Graphics*, vol. 3, pp. 450–459, 2006.

[6] S. Yoon, S. Curtis, and D. Manocha, "Ray tracing dynamic scenes using selective restructuring," *Proc. of Eurographics Symposium on Rendering*, pp. 73–84, 2007.

[7] B. V. Mirtich, "Impulse-based dynamic simulation of rigid body systems," Ph.D. dissertation, University of California, Berkeley, 1996.

[8] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, no. 4, pp. 1–13, 1997.

[9] N. K. Govindaraju, D. Knott, N. Jain, I. Kabul, R. T. R. Gayle, M. C. Lin, and D. Manocha, "Interactive collision detection between deformable models using chromatic decomposition," *ACM Trans. on Graphics (Proc. of ACM SIGGRAPH)*, vol. 24, no. 3, pp. 991–999, 2005.

[10] S. Curtis, R. Tamstorf, and D. Manocha, "Fast collision detection for deformable models using representative-triangles," *ACM Symp. on Interactive 3D Graphics*, pp. 61–79, 2008.

[11] D. Kim, J.-P. Heo, J. Huh, J. Kim, and S.-E. Yoon, "HPCCD: Hybrid parallel continuous collision detection," *Computer Graphics Forum (Pacific Graphics)*, 2009.

[12] D. L. James and D. K. Pai, "BD-Tree: Output-sensitive collision detection for reduced deformable models," *ACM Transactions on Graphics (SIGGRAPH 2004)*, vol. 23, no. 3, Aug. 2004.

[13] J. Mezger, S. Kimmefie, and O. Etzmu, "Hierarchical techniques in collision detection for cloth animation," *Journal of WSCG*, vol. 11, no. 1, 2003.

[14] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnetat-Thalmann, and W. Strasser, "Collision detection for deformable objects," pp. 119–139, 2004.

[15] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garment," *Graphics Interface*, pp. 177–189, 1997.

[16] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *Proc. of ACM SIGGRAPH*, 2002.

[17] M. Tang, S.-E. Yoon, and D. Manocha, "Adjacency-based culling for continuous collision detection," *Visual Comput*, vol. 24, pp. 545–553, 2008.

[18] D. Kim, J.-P. Heo, and S.-E. Yoon, "PCCD: Parallel continuous collision detection," *Tech. rep., Dept. of CS, KAIST, Technical Report CS-TR-2008-298*, 2008.

[19] M. Tang, D. Manocha, and R. Tong, "MCCD: Multi-core collision detection between deformable models using front-based decomposition," *Graphical Models (accepted)*, 2010.

[20] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, , and D. Manocha, "Fast BVH construction on gpus," *Proc. Eurographics*, 2009.

[21] X. Zhang, M. Lee, and Y. J. Kim, "Interactive continuous collision detection for non-convex polyhedra," *The Visual Computer*, pp. 749–760, 2006.

[22] M. Tang, Y. J. Kim, and D. Manocha, "C^2A: Controlled conservative advancement for continuous collision detection of polygonal models," *Proc. of IEEE Conference on Robotics and Automation*, 2009.

[23] J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-dops," *IEEE Transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.

[24] P. Schneider and D. H. Eberly, *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 2002.

[25] X. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using taylor models and temporal culling," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, vol. 26, no. 3, p. 15, 2007.

[26] S. Redon and M. Lin, "Practical local planning in the contact space," *Proc. of IEEE ICRA*, 2005.