Radial View Based Culling for Continuous Self-Collision Detection of Skeletal Models

Sai-Keung Wong Wen-Chieh Lin Chun-Hung Hung Yi-Jheng Huang Shing-Yeu Lii
National Chiao Tung University, Taiwan

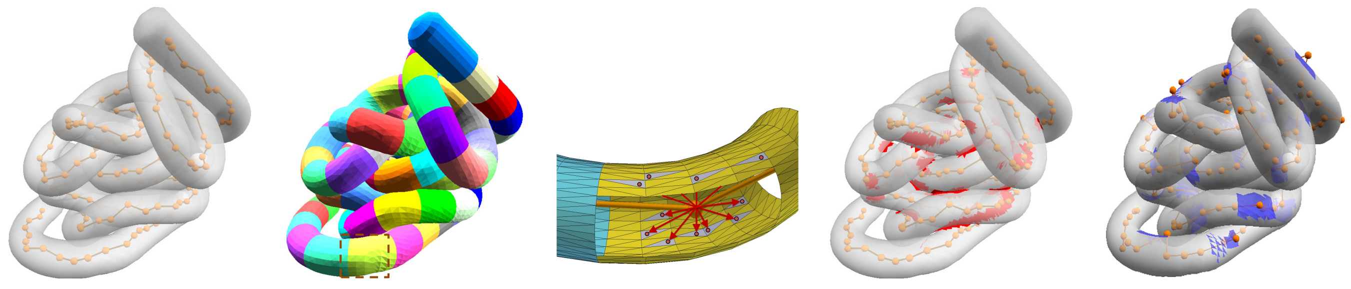

Figure 1: *The major procedures of our method: (a) Input of a deforming mesh with a skeleton; (b) Clustering triangles based on the skeleton; (c) A radial view test from an observer primitive (an enlarged view of the dashed region in (b)); (d) Culling results. Potentially colliding triangles are colored as red; (e) Our method works better for observer points lying inside the mesh, but it can tolerate some observer points lying outside the mesh. Negatively oriented triangles are colored as blue.*

Abstract

We present a novel radial-view-based culling method for continuous self-collision detection (CSCD) of skeletal models. Our method targets closed triangular meshes used to represent the surface of a model. It can be easily integrated with bounding volume hierarchies (BVHs) and used as the first stage for culling non-colliding triangle pairs. A mesh is decomposed into clusters with respect to a set of observer primitives (i.e., observer points and line segments) on the skeleton of the mesh so that each cluster is associated with an observer primitive. One BVH is then built for each cluster. At the runtime stage, a radial view test is performed from the observer primitive of each cluster to check its collision state. Every pair of clusters is also checked for collisions. We evaluated our method on various models and compared its performance with prior methods. Experimental results show that our method reduces the number of the bounding volume overlapping tests and the number of potentially colliding triangle pairs, thereby improving the overall process of CSCD.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality;

Keywords: continuous self-collision detection, deformable model, view test, skeleton

Links: DL PDF

ACM Reference Format
Wong, S., Lin, W., Hung, C., Huang, Y., Lii, S. 2013. Radial View Based Culling for Continuous Self-Collision Detection of Skeletal Models. ACM Trans. Graph. 32, 4, Article 114 (July 2013), 10 pages.
DOI = 10.1145/2461912.2461951 http://doi.acm.org/10.1145/2461912.2461951.

1 Introduction

To realistically simulate deformable models, it is critical to handle the self-collision detection and response problems. Collision detection for deformable models has thus received much research interest and many elegant methods have been proposed [Teschner et al. 2005]. Most of these methods rely on bounding volume hierarchies (BVHs) for accelerating the collision detection process. However, they rarely utilize higher-level geometrical information when constructing the BVHs and hence may spend too much time on checking adjacent polygons of which a large portion do not collide. This problem can be illustrated by considering a simple sphere. The surface of the sphere is represented by a closed triangular mesh which does not collide with itself. However, each triangle is checked with its neighboring triangles in the BVH.

It is apparent that building a BVH according to the topological structure of deformable models would greatly improve the culling efficiency of self-collision detection since there is movement correlation between a deformable model's structure and its surface polygons. A good choice of such topological structure for deformable models is a skeleton, which is widely used as a representation of the structure of a geometrical model in computer graphics, image processing, character animation and model retrieval [Sundar et al. 2003].

In this paper, we propose an efficient method that accelerates the continuous self-collision detection (CSCD) process for deformable closed models embedded with skeletons. We focus on skeleton-driven animation, such as man-made and simulated animation. For example, our method can be used to detect self-collision events in the skinning process by hinting animators when to adjust bone weights for skinning.

The intuition of our method is based on a key observation: Given a closed 2-manifold mesh, if there exists a point such that all polygons of the mesh are fully visible from the point, then there is no self-collision. We develop a "view" test based on the Jordan Surface Theorem [Kopperman et al. 1991] for efficiently determining the collision state of a deformable model driven by a skeleton. Our method performs better when the entire skeleton or most of its parts are inside the model during deformation. To check for potential collisions based on the view tests, points are placed on the skeleton. We call these points *observer points* and the view test *radial view test*. The radial view test is based on evaluating the facing directions of the polygons w.r.t. the observer points. If the radial view test passes, the mesh does not collide with itself. In conventional BVH-based methods, the deformable model is usually decomposed into several parts of which the bounding volumes overlap with each other. This usually increases the number of bounding volume overlapping tests and other computational costs. On the other hand, the radial view test fails if there are back facing polygons w.r.t. the observer points. In this case, further processing is required to evaluate the collision status of the mesh in our method.

To efficiently perform the radial view test, the mesh is decomposed into clusters based on the skeleton such that each cluster is associated with an observer primitive (i.e., point or line segment). Intra-cluster check (based on the radial view test) and inter-cluster check (using a hierarchical-based method) are then employed to collect the potentially colliding triangle pairs. We evaluated our method on various deformable models with skeletons. Our experimental results show that our method culls away non-colliding triangle pairs efficiently and reduces the number of bounding volume overlapping tests.

The major contributions of our work are: 1) A novel skeleton-aware technique is presented for decomposing a closed 2-manifold triangular mesh into clusters, which are used for culling non-colliding feature pairs. 2) A novel radial view test is proposed for efficiently determining the collision state of the mesh. 3) A method to compute inter-cluster orphan sets is proposed which further culls adjacent triangle pairs during the inter-cluster check.

2 Related Work

Self-collision techniques are widely applied in simulating breakable models [Larsson and Akenine-Möller 2006], articulated rigid bodies [Redon et al. 2002], cloth [Bridson et al. 2002], character animation [Capell et al. 2002; Baran and Popović 2007], and deformable tetrahedron models [Tang et al. 2011c]. In [Zheng and Doug 2012], an affine-invariant Laplacian based energy model is proposed for precomputing a set of certificates. At the runtime stage, the deformation energy is computed for determining whether or not the sub-meshes are free of collision.

Bounding Volume Hierarchies. Bounding volume hierarchies are employed extensively for rigid and deformable models. BVHs with several different bounding volume types have been proposed, such as K-DOPs, axis-aligned bounding box and sphere [Teschner et al. 2005]. For models that may only deform partially, lazy update methods are proposed such that the BVHs are updated only when necessary, e.g., kinetic bounding volumes [Zachmann and Weller 2006]. On the other hand, for models that may severely deform, the BVHs are restructured so as to improve culling effectiveness, such as using the dynamic bounding volume hierarchy [Larsson and Akenine-Möller 2006]. These techniques can efficiently cull away distant triangle pairs, but may spend too much time on checking non-colliding triangle pairs that are adjacent.

Normal Cone-Based Culling Methods. Volino and Magnenat-Thalmann [1994] proposed a culling method for discrete self-collision detection based on mesh regularity. The method computes the normal cones of triangles and perform contour tests for clusters of the mesh. Later on, the method was extended for continuous collision detection in [Mezger et al. 2003; Wong and Baciu 2005; Tang et al. 2009a]. The idea was further extended in [Schvartzman et al. 2009] for handling reduced deformable models. Schvartzman et al. [2010] proposed an efficient data structure based on star contours for performing contour tests in discrete self-collision detection. In practice, the contour tests are usually ignored due to the expensive projection and intersection operations [Schvartzman et al. 2010].

Elementary Test Processing. In the elementary test processing, the feature pairs (vertex-triangle and edge-edge) of a potentially colliding triangle pair are processed for computing the first time of contact. Some techniques assign each vertex and each edge of the mesh to one of its adjacent triangles so as to eliminate the duplicate tests for feature pairs [Wong and Baciu 2006; Curtis et al. 2008]. A filtering technique based on deforming non-penetration filters [Tang et al. 2010], is employed for culling the non-coplanar feature pairs. There is a variety of culling techniques that are proposed in [Tang et al. 2011a; Zhang and Kim 2012].

GPU-Based and Parallel Techniques. Early GPU-based methods rasterize models onto colour/stencil/depth buffers for collision checks [Knott and Pai 2003]. Govindaraju et al. [2005] improved the performance of collision detection by applying visibility-based culling techniques to the chromatic decomposition of triangle meshes. Readers are also referred to the work on employing multicore CPUs and GPUs for collision detection, such as [Kim et al. 2009; Tang et al. 2011b].

Specific Model Types. Methods for better handling collision detection of specific types of models have also been suggested. Barbič and James [2010] make use of vertex movement constraints in reduced deformable models to precompute sets of potentially colliding parts, called *certificates*. Their results show that the cost spent on inter-collision and self-collision are comparable. For collision detection applied to the skin of characters, Capell et al. [2005] proposed to keep track of active nodes. This method particularly benefits collisions around creases. It imposes the assumption that collisions occurring in the near future will be in the neighborhood of the collision seeds. A collision detection method targeting spherical blend skinning can be found in [Kavan et al. 2006] in which bounding spheres are adopted for bounding the models.

3D Guarding Problem. Our method was inspired by the 3D guarding problem. In solving a 3D guarding problem, a minimal set of points is computed so that the entire 3D mesh is visible from this set of points. Yu and Li [2011] proposed the progressive integer linear programming algorithm to tackle the 3D guarding problem. Our method proposes the use of skeleton features. We first decompose the mesh into clusters and then perform a radial view check on each cluster. Yu and Li's method cannot be directly applied to our problem because a deformable model may change shape and the computed 3D guarding points may not cover the entire surface of the model during deformation.

3 Theoretical Basis and Algorithm Overview

Our major idea is to decompose a closed 2-manifold triangular mesh into clusters based on its skeleton where each cluster is associated with an observer primitive (point or line segment). According to the Jordan Surface Theorem [Kopperman et al. 1991], we develop a radial view test for checking the collision state of each cluster separately. In the following, we describe the notations, definitions, intuitive concepts and overview of our method.

Notations and Definitions: A deformable closed 2-manifold triangular mesh is denoted as the triplet $M = (V, E, F)$, where V is a set of vertices, E is a set of edges, and F is a set of triangles. Assume that within a time interval $[0, \Delta t]$, the mesh M deforms and the velocity of each vertex of M is a constant, where Δt is the time step size. Thus, the position of a vertex (or a point) i within the time interval can be expressed as $\mathbf{p}_i(t) = \mathbf{p}_i(0) + \tilde{\mathbf{v}}_i t$, where $\mathbf{p}_i(0)$ is the position of the vertex at the beginning of the time interval, $\tilde{\mathbf{v}}_i$

is the velocity of i and $t \in [0, \Delta t]$. The normal vector $\mathbf{n}_T(t)$ of a triangle $T(\mathbf{p}_0(t), \mathbf{p}_1(t), \mathbf{p}_2(t))$ points to the exterior of the mesh M within the time interval. The normal vector $\mathbf{n}_T(t)$ is defined as $\overrightarrow{\mathbf{p}_1(t)\mathbf{p}_0(t)} \times \overrightarrow{\mathbf{p}_2(t)\mathbf{p}_0(t)}$. The shorthand $\overrightarrow{\mathbf{p}_i\mathbf{p}_k}$ is used to denote $\mathbf{p}_i - \mathbf{p}_k$ in this paper.

A feature is a vertex, an edge or a triangle. Two triangles are adjacent if they share at least one vertex and non-adjacent otherwise. Similarly, a feature pair is adjacent if they share at least a common vertex and non-adjacent otherwise. A cluster of triangles (short: cluster) is a collection of triangles and the collection of vertices and edges incident to the triangles. Two clusters C_k and C_l are adjacent if they share at least one vertex. For a deformable model represented by a triangular mesh, self-collision occurs if two non-adjacent features collide with each other. In continuous collision detection, there exist two kinds of collision events: vertex-triangle and edge-edge. For two non-adjacent triangles, there are fifteen feature pairs: six vertex-triangle and nine edge-edge pairs. A skeleton is denoted $S = (S_v, S_e)$, where S_v is the set of skeleton joints and S_e is the set of skeleton bones.

Classification of Triangles. Let q be an observer point defined on the skeleton of the deformable mesh. The position of q is a function of time in the reference frame of the deformable mesh. We define a time-dependent characteristic function $\phi(T, q, t)$ for a triangle $T(\mathbf{p}_0(t), \mathbf{p}_1(t), \mathbf{p}_2(t))$ with respect to q. We classify the triangle into one of the three categories based on the sign of the characteristic function: (1) *positively oriented* if $\phi(T, q, t) > 0$ for all $t \in [0, \Delta t]$; (2) *negatively oriented* if $\phi(T, q, t) < 0$ for all $t \in [0, \Delta t]$; (3) *uncertain* otherwise. For an arbitrary observer point q, we have

$$\phi(T, q, t) := \mathbf{n}_T(t) \cdot (\mathbf{p}_0(t) - \mathbf{q}(t)). \tag{1}$$

The characteristic function is a cubic function in general (see the supplemental materials for details). Note that the signs of $\mathbf{n}_T(t) \cdot (\mathbf{p}_i(t) - \mathbf{q}(t))$ are the same for $i = 0, 1, 2$.

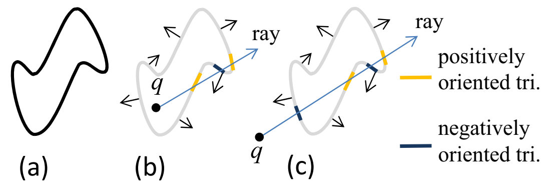

Figure 2: *A ray intersects the triangles of a closed mesh. The ray leaves the mesh when intersecting with a positively oriented triangle, and enters the mesh when intersecting with a negatively oriented triangle. Leaving and entering are always interleaved, i.e. any intersection after leaving must be entering, and vice versa. The black arrows represent the outer normals of the triangles. (a) a planar slice of the closed mesh. (b) a ray is cast from a point lying inside the mesh. (c) a ray is cast from a point lying outside the mesh.*

Collision Conditions. Assume that the mesh M does not have any self-intersection initially. Let $\mathbf{d}_r$ denote the direction of a ray r. If the ray r hits a positively oriented triangle, i.e., $\mathbf{d}_r \cdot \mathbf{n}_T(t) > 0$, then the ray *leaves* the mesh. If a ray hits a negatively oriented triangle, i.e., $\mathbf{d}_r \cdot \mathbf{n}_T(t) < 0$, then the ray *enters* the mesh.

According to the Jordan Surface Theorem [Kopperman et al. 1991], a closed surface separates the three-dimensional space into two connected components. Hence, a ray cast from an interior (exterior) point of a mesh intersects the mesh M an odd (even) number of times if the ray is not parallel to any triangles. Furthermore, the ray intersects the mesh in an interleaving *leave/enter* pattern. Fig. 2 shows two examples.

Given the mesh M, it deforms within the time interval $[0, \Delta t]$ and q is any observer point of the mesh. Assume that the first collision event occurs between two non-adjacent features f_1 and f_2 (vertex-triangle or edge-edge). Furthermore, they collide at a point $\mathbf{p}_c$ at time t_c. There exist two different triangles T_1 and T_2 incident to f_1 and f_2, respectively, such that either of the following two collision conditions holds within the time interval: (1) one triangle is positively oriented w.r.t. q and the other is negatively oriented w.r.t. q; (2) at least one of the triangles is uncertain w.r.t. q. This can be realized by casting a ray passing $\mathbf{q}(t_c)$ and $\mathbf{p}_c$ (see Fig. 3), i.e., $\mathbf{d}_r = \mathbf{p}_c - \mathbf{q}(t_c)$. Let $\Pi(f_1)$ denote the set of triangles incident to f_1 and $\Pi(f_2)$ denote the set of triangles incident to f_2. The ray intersects at least one triangle of $\Pi(f_1)$ and at least one triangle of $\Pi(f_2)$. If all of the triangles of $\Pi(f_1) \cup \Pi(f_2)$ were positively (negatively) oriented w.r.t. q within the time interval, the ray would leave (enter) the mesh at $\mathbf{p}_c$ twice consecutively. This is not possible as the mesh M is a closed 2-manifold mesh. The proof can be found in the supplemental materials.

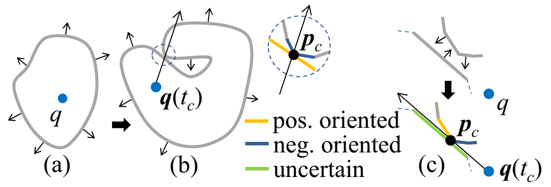

Figure 3: *Two collision conditions for a closed deformable mesh. The mesh is viewed from an observer point q. (a) The mesh is free of collision. In (b) and (c), the mesh collides with itself at a point* $\mathbf{p}_c$*. In (b), at least one triangle is positively oriented w.r.t. q and at least one triangle is negatively oriented w.r.t. q. In (c), at least one triangle is uncertain w.r.t. q.*

Collision Culling. We can exploit the above two collision conditions to collect the potentially colliding triangle pairs. Given an observer point q, we can classify the triangles of the mesh within the time interval into three sets: $H^+(q)$, $H^-(q)$, and $H^u(q)$, which denote the set of triangles that are positively oriented, negatively oriented, and uncertain w.r.t. q, respectively. Based on the collision conditions, we only need to check the triangles of the two set pairs $(H^+(q), H^-(q))$, $(H^+(q) \cup H^-(q) \cup H^u(q), H^u(q)) = (F, H^u(q))$ for collision. In other words, we check the triangles between $H^+(q)$ and $H^-(q)$ as well as the triangles between F and $H^u(q)$ in each time step.

To efficiently cull potentially colliding triangles, we would like to reduce the number of triangles that are negatively oriented or uncertain. It is obvious that there may be many uncertain and negatively oriented triangles if only one observer point is used since the mesh surface may have many concave parts. This problem can be alleviated by decomposing the mesh into clusters and each cluster is associated with an observer point. A good place to put observer points is the skeleton of the mesh since the skeleton usually lies inside the mesh when the model deforms. It also captures the main structure of the mesh. We therefore exploit the skeleton for placing the observer points. By appropriately computing multiple observer points on the skeleton of the mesh, we can reduce the number of negatively oriented and uncertain triangles in each cluster. For example, an observer point can be placed at a joint or in a bone of the skeleton. The mesh is then decomposed into clusters with respect to the observer points. Besides checking $(H^+(q), H^-(q))$ and $(F, H^u(q))$ of each cluster with an observer point q for collision, we should also check the triangles between every pair of clusters. Fig. 4 illustrates how our method detects the first colliding feature pairs for two examples. In Case (1), there is one cluster. The model deforms within the time interval and the two triangles T_1 and T_2 collide with each other. Before the model collides with itself, all the triangles are positively oriented w.r.t. the observer point of the cluster within the time interval. At the first time of contact, the two triangles are uncertain. Hence, they form a potentially colliding triangle pair and this pair is handled in the elementary test processing. In Case (2), there are two clusters C_1 and C_2. The model deforms and then the vertex p_1 collides with T_4. All the triangles are positively oriented w.r.t. their observer point within the time interval. Our method can still collect the colliding vertex-triangle pair (p_1, T_4). This is because T_1 and T_4 belong to different clusters and they form a potentially colliding pair in the inter-cluster check.

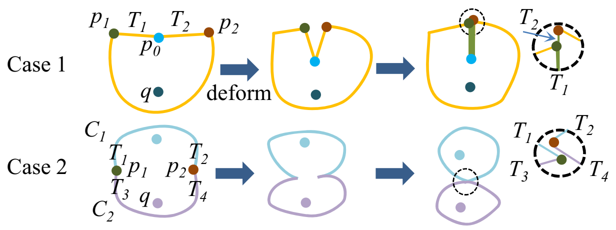

Figure 4: *Two examples for illustrating how our method detects the first colliding feature pairs of two deforming meshes. The vertices and the observer points are shown in different colors. In Case (1), there is one cluster. The triangle T_1 collides with another triangle T_2. In Case (2), there are two clusters. The vertex p_1 collides with triangle T_4.*

Algorithm Overview. We summarize our method in Algorithm 1. At the preprocessing stage, we perform cluster decomposition, cluster boundary refinement, cluster reduction, BVH construction, and finally compute inter-cluster orphan sets. At the runtime stage, we refit the bounding volumes of BVH (BVH refit) and classify the triangles of each cluster and progressively update the cluster BVHs (Radial-View update). During the intra-cluster check, a radial view test is performed to check collisions within each cluster. If there are uncertain triangles or negatively oriented triangles, further processing is required to collect the potentially colliding triangle pairs (PCTPs). During the inter-cluster check, a hierarchical-based method is performed to collect the PCTPs between every pair of clusters. Finally, elementary tests are performed.

4 Preprocessing Stage

In the preprocessing stage, we decompose a given mesh $M(V, E, F)$ into a disjoint set of clusters $\{C_1, C_2, \cdots, C_N\}$ such that $F = \cup_{k=1}^{N} C_k$ (Fig. 5) and construct a BVH of these clusters. Each cluster C_k is associated with an observer point denoted as q_k. The desirable goals for clustering are: 1) The number of the positively oriented triangles should be the majority within a cluster. 2) The triangles of a cluster C_k should be visible and close to q_k. 3) The number of clusters is small so as to reduce the cost of the inter-collision check. For the first and second goals, we adopt a heat diffusion process in conjunction with the distance field to decompose the mesh. For the third goal, we perform cluster reduction to merge the adjacent clusters.

4.1 Cluster Decomposition

A desirable property of the clusters is that their triangles should move along with the observer points. In this way, the orientations

Algorithm 1 Collision Detection Process

```
Begin Preprocessing Stage
Perform cluster decomposition
Perform cluster boundary refinement
Perform cluster reduction
Build BVHs of clusters
Compute inter-cluster orphan sets
End Preprocessing Stage

Begin Runtime Stage
for each time step do
    Refit BVHs
    Radial-view update: classify triangles; update cluster BVHs
    for each cluster $C$ do
        if radial view test for $C$ is not passed then
            Perform intra-cluster check for $C$
        end if
    end for
    Perform inter-cluster check
    Perform elementary tests
end for
End Runtime Stage
```

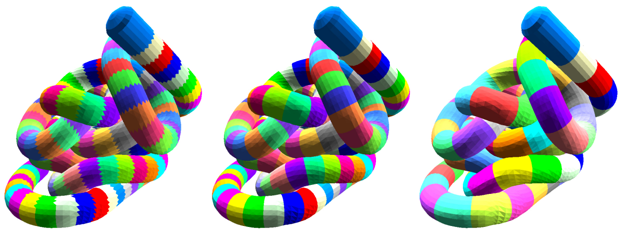

Figure 5: *Cluster decomposition process. From left to right: clusters obtained after heat diffusion, cluster boundary refinement, and cluster reduction.*

of the triangles of a cluster w.r.t. the corresponding observer point do not change often. This desirable property is in fact similar to the requirements for computing the bone weights in skin deformation [Baran and Popović 2007], where the bone weights of each vertex indicate how much the movement of the bones affect the movement of the vertex. Hence, we adopt the heat diffusion scheme [Baran and Popović 2007] to compute bone weights of vertices. Then we decompose the mesh into clusters based on the bone weights and compute observer points.

The heat diffusion scheme works as follows. Let $\mathbf{w}^j$ be the vector of all weights of bone j. $\mathbf{w}^j$ is obtained by solving the equilibrium equation for the heat diffusion process: $\frac{\partial \mathbf{w}^j}{\partial t} = \Delta \mathbf{w}^j + \mathbf{H}(\mathbf{r}^j - \mathbf{w}^j)$, where Δ is the discrete surface Laplacian, $\mathbf{r}^j$ is a vector with $r_i^j = 1$ if the nearest bone to vertex i is j; otherwise, $r_i^j = 0$. $\mathbf{H}$ is a diagonal matrix with diagonal elements $\mathbf{H}_{ii} = c/(d(i))^2$ if vertex i is visible from the bone and $\mathbf{H}_{ii} = 0$ otherwise, where $d(i)$ is the distance from a vertex i to the nearest bone, computed from the distance field. c is a free parameter. A vertex i is visible from the bone if the shortest line segment from the bone to the vertex lies inside the volume of the mesh. In [Baran and Popović 2007], an adaptive depth field [Frisken et al. 2000] is employed for checking the visibility status between a bone and a vertex. The method for solving the equilibrium equation can be found in [Baran and Popović 2007]. We performed preliminary experiments and found that similar results were obtained for c between 1.0 and 3.0.

After computing the bones weights of each vertex, we assign the vertex to the bone with the highest bone weight. Then, we assign each triangle to a bone as follows: (1) If there are at least two vertices of a triangle assigned to the same bone, the triangle is assigned to the bone. (2) If the three vertices of a triangle are assigned to three different bones, a candidate set is formed by those bones that the triangle is positively oriented to. The triangle is then assigned to the bone with the highest weight in the candidate set. If a triangle T is not assigned to any bone, we repeatedly check its adjacent triangles and assign T to the bone of any of its adjacent triangles until all the triangles are assigned. Finally, an observer point is put on each bone and the triangles assigned to the bone are assigned to the observer point. In our experiment, an observer point is placed at one of the joints of a bone.

4.2 Cluster Refinement

The boundary of two adjacent clusters may be a zigzag shape as shown in Fig. 6. We want to reduce the number of edges shared by every pair of adjacent clusters so as to make the boundaries smooth. Let $\tilde{\Gamma}$ be the set of all the boundary edges between two clusters C_k and C_l before refinement. We reassign the triangles (in C_k and C_l) incident to any edges in $\tilde{\Gamma}$. The cluster boundary refinement is performed locally as we do not want the new boundary to drift away from the original boundary. To do so, we sort the triangles incident to the edges in $\tilde{\Gamma}$ and inspect these triangles one by one. If a triangle T is adjacent to two triangles belonging to the same cluster, T is assigned to that cluster. An example is shown in Fig. 6 and Γ is the boundary edges of C_k and C_l after refinement.

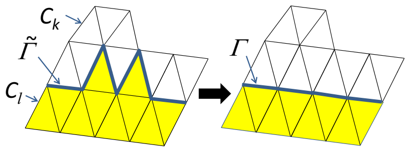

Figure 6: *Boundary refinement for reducing the number of shared edges of the adjacent clusters.*

4.3 Cluster Reduction

The idea of our cluster reduction method is illustrated in Fig. 7. Two adjacent clusters can be merged into a single cluster to reduce the number of clusters. However, the computational cost of the intra-cluster check for the merged cluster may increase if the number of negatively and uncertain triangles increases. A simple method is that two adjacent clusters C_k and C_l are merged if most of the triangles of $\Pi(C_k) \cup \Pi(C_l)$ are positively oriented w.r.t. q, where $\Pi(C_k)$ and $\Pi(C_l)$ are the set of triangles in C_k and C_l, respectively; and q is one of the observer points of C_k and C_l. This simple method works well if the cluster does not deform w.r.t. q. This is because the orientation of each triangle of the cluster does not change. However, if the model deforms, the incident angle for the triangle should also be considered for merging. The incident angle is defined for measuring the deviation of the normal vector of the triangle from the viewing direction with respect to q. Specifically, the incident angle θ of a triangle[1] $T(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ is defined as the angle between the normal vector of the triangle and the vector $\mathbf{g} - \mathbf{q}$, where $\mathbf{g} = \frac{\mathbf{p}_0+\mathbf{p}_1+\mathbf{p}_2}{3}$.

The larger the incident angle for a triangle T, the more likely T will change orientation. With the consideration of the incident angle, the orientations of the triangles do not change much before and after merging. A triangle is admissible w.r.t. an observer point q if it is positively oriented w.r.t. q and its incident angle is less than a given threshold ϵ. Let the number of the admissible triangles be N^+. If $\frac{N^+}{|\Pi(C_k) \cup \Pi(C_l)|} \geq \beta$ for q, then the two clusters C_k and C_l are merged, where the ratio β is a user defined value. The observer point of the new cluster is q. If the condition is satisfied by both observer points, we select the observer point with higher number of branches[2] as the new observer point. In this way, we can reduce the number of clusters. In our experiments, we found that our method performed well for $\epsilon = 80^o$ and $\beta = 0.9$.

[1]We omit t for the time-varying variables within the same time interval hereafter unless it is needed to specify the meaning.

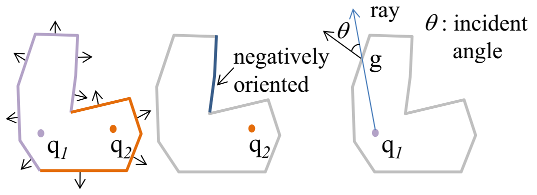

Figure 7: *Cluster reduction for two clusters. The arrows indicate the directions of the normal vectors of the triangles. Left: q_1 and q_2 are the observer points of two clusters. Middle: If q_2 is selected as the new observer point, there are negatively oriented triangles after merging two clusters; Right: If q_1 is selected as the new observer point, then all the triangles are positively oriented w.r.t. q_1.*

4.4 BVH Construction

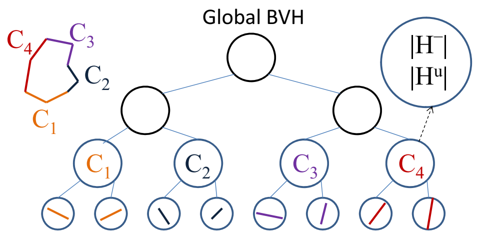

Figure 8: *The global BVH of a model. The model is divided into four clusters C_1, C_2, C_3 and C_4. Each node has two counters for recording the numbers of elements of H^- and H^u, respectively.*

After the clusters are refined and reduced, we construct the BVH of each cluster. We use K-DOP as the bounding volume for its tightness, but other kinds of bounding volumes can also be used. The BVH of a cluster is constructed by adopting the median splitting scheme in a top-down manner until the leaf nodes are reached. Each leaf node of the BVH of a cluster contains a triangle. Then we construct the *global* BVH of the mesh. We employ again the median splitting scheme in a top-down manner to divide the clusters into two halves recursively until each node of the global BVH contains one cluster. Fig. 8 shows the structure of the global BVH.

5 Runtime Stage: Collision Culling

In the runtime stage, we first perform BVH refitting for the global BVH of the mesh. The global BVH of the mesh is processed in a bottom-up manner so that the bounding volume of each node tightly encloses the triangles associated with the node. Notice that after performing BVH refitting, the BVH of each cluster is also updated since it is a part of the global BVH. Then intra- and inter-cluster checks are performed for collecting potentially colliding triangle pairs.

[2]Note that an observer point is a joint on a skeleton.

5.1 Intra-Cluster Check

Radial View Test and Triangle Classification for a Cluster C. In each time step, we classify each triangle of C into two sets $H^+(C)$ and $H^-(C)$ using the characteristic function (Eq. 1). $H^u(C)$ is then computed as $\Pi(C) \setminus (H^+(C) \cup H^-(C))$, where $\Pi(C)$ is the set of all triangles in C. Based on the discussion in Section 3, if all the triangles of C belong to $H^+(C)$, we can skip the cluster during the collision check; Otherwise, two triangles of C may collide with each other and we need to check pairs of triangles for the two set pairs ($H^+(C)$, $H^-(C)$) and ($\Pi(C)$, $H^u(C)$).

BVH Update for Each Cluster. To efficiently collect the potentially colliding triangle pairs for each cluster C, we adopt a hierarchical-based method. We maintain two counters in each node of the global BVH to record the numbers of triangles belonging to $H^-(C)$ and $H^u(C)$, respectively (Fig. 8). Changes in the two counters are propagated all the way to the root. For example, if a triangle in the leaf node is negatively oriented, the counter for $H^-(C)$ is *one* and the counter for $H^u(C)$ is zero. If, at a later point in time, the triangle changes to be uncertain, the counter for $H^-(C)$ becomes zero and the counter for $H^u(C)$ becomes one. In that case, the counters for $H^u(C)$ and $H^-(C)$ in all the ascendant nodes of the leaf node are also updated correspondingly.

To perform the intra-cluster check for the pair ($H^+(C)$, $H^-(C)$), we traverse down the hierarchy from the root node of the BVH of C only if the counter of $H^-(C)$ is larger than zero. Similarly, to check for the pair $\Pi(C)$ and $H^u(C)$, we traverse down from the root of the BVH of C only if the counter of $H^u(C)$ is larger than zero. When there are two leaf nodes that are reached, the corresponding two triangles form a potentially colliding triangle pair.

5.2 Inter-Cluster Check

We perform the hierarchical method for checking collision between clusters [Klosowski et al. 1998]. Inspired by the *orphan set* concept [Tang et al. 2009a], we developed a method that allows us to ignore adjacent triangle pairs during the inter-cluster check.

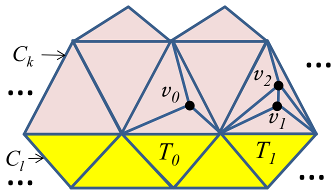

Figure 9: *Inter-cluster orphan set (ICOS). Collision events may be missed if the adjacent triangle pairs are ignored between two adjacent clusters C_k and C_l. Consider that the feature pair v_0 and T_0 collides. Ignoring the adjacent triangle pairs may fail to detect the collision events. Hence, (v_0, T_0) is a member of the ICOS. Similarly, (v_1, T_1) and (v_2, T_1) are also members of the ICOS.*

Inter-Cluster Orphan Set (ICOS). To ensure that we can detect all the colliding feature pairs while skipping the adjacent triangle pairs during the inter-cluster check, we need to handle some special cases, as illustrated in Fig 9. Consider a vertex-triangle case (v, T). The vertex v is shared by a set of triangles which are adjacent to a triangle T of another cluster. If we ignore the adjacent triangle pairs involving T, we may miss the collision occurring between v and T. It is therefore necessary to precompute these kinds of feature pairs and store them in a set, called the inter-cluster orphan set (ICOS). At the runtime stage, we need to perform tests for the feature pairs of ICOS.

Denote $\Pi(f, C)$ as the set of triangles incident to a feature f and belonging to a cluster C. Consider two features f_1 and f_2 such that f_1 and f_2 belong to two adjacent clusters C_1 and C_2, respectively. A feature pair (f_1, f_2) is assigned to ICOS if the following two conditions are both satisfied: (1) C_1 and C_2 are different; (2) any triangle in $\Pi(f_1, C_1)$ is adjacent to a triangle in $\Pi(f_2, C_2)$. To compute ICOS for two adjacent clusters C_1 and C_2, we check all the adjacent triangle pairs (T_1, T_2) along the boundary of the two clusters, where $T_1 \in C_1$ and $T_2 \in C_2$. Then we collect the feature pairs of all these adjacent triangle pairs that satisfy the two conditions. Assume that the valence of the vertices is less than a given threshold. Then the running time complexity of computing ICOS for two adjacent clusters is proportional to the number of triangles on their boundary.

5.3 Elementary Test Processing

We adopt the method presented in [Tang et al. 2010] for computing the times of contacts for the potentially colliding triangle pairs. The feature pairs of ICOS are also processed. To eliminate the duplicate tests for the feature pairs in the inter-cluster check for non-adjacent cluster pairs, we adopt the representative-triangle scheme [Curtis et al. 2008] for its simplicity.

6 Extension to Observer Line Segments

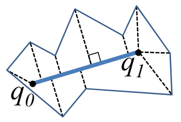

Figure 10: *An observer line segment $\overline{q_0q_1}$ is useful for viewing a model with sharp geometric features. Each vertex is projected to the closest point on $\overline{q_0q_1}$.*

Observer points are more suitable as the observer primitives for the radial-view test on the clusters with round shape. To handle clusters with sharp geometric features, we can employ observer line segments as their observer primitives (Fig. 10). Given an observer line segment $\overline{q_0q_1}$, the characteristic function for a triangle $T(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$ w.r.t. $\overline{q_0q_1}$ is defined as

$$\phi(T, \overline{q_0q_1}, t) := \begin{cases} 1 & \text{if } \mathbf{n}_T(t) \cdot \overrightarrow{\mathbf{p}_i(t)\mathbf{p}'_i(t)} > 0, \forall i \in \{0,1,2\} \\ -1 & \text{if } \mathbf{n}_T(t) \cdot \overrightarrow{\mathbf{p}_i(t)\mathbf{p}'_i(t)} < 0, \forall i \in \{0,1,2\} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $\mathbf{p}'_i(t)$ is a point on the line segment $\overline{q_0q_1}$ and it is closest to $\mathbf{p}_i(t)$. The characteristic functions are either cubic functions or rational polynomial functions. Notice that if $\overline{q_0q_1}$ is known to lie inside the mesh, only one vertex of the triangle is required to be evaluated for classifying the triangle. In this case, the signs of $\mathbf{n}_T(t) \cdot \overrightarrow{\mathbf{p}_i(t)\mathbf{p}'_i(t)}$ are the same for $i = 0, 1,$ and 2. Please refer to the supplemental materials for computing $\phi(T, \overline{q_0q_1}, t)$ and $\mathbf{p}'_i(t)$.

We compute observer points and observer line segments as follows. We adopt the method describe in Section 4.1 for assigning triangles to bones and then determine whether an observer point or line segment should be created for each bone. Let the two endpoints (i.e., joints) of the bone be q_j and q_k. An observer line segment $\overline{q_jq_k}$ is created if the ratio of the negatively oriented triangles at either observer point is greater than a threshold α and that ratio at the observer line segment is less than α, then we choose the observer point. The idea is that we want to reduce the number of

Table 1: *Model complexities and timing results for preprocessing.* $|H^+|$, $|H^-|$ *and* $|H^u|$ *are the average number of positively-oriented, negatively-oriented, and uncertain triangles, respectively.* $X \rightarrow Y$*:* X *and* Y *are the numbers of clusters before and after cluster reduction, respectively.*

	#Tri.	#Vert.	$\|H^+\|$	$\|H^-\|$	$\|H^u\|$	#Clusters	Depth Field Discretization (sec)	Cluster Decomposition (msec)	Boundary Refinement (msec)	Cluster Reduction (msec)	ICOS (msec)
Rope	48440	24222	48354	3	83	199→100	9.0	20	4	6	17
Octopus	47976	23990	46413	572	991	71→40	10.3	10	4	7	14
Urchin	2840	1422	2755	73	12	4→4	23.3	3	0.2	1	8
Man	20000	10002	18342	1051	607	21→16	6.2	3	1	4	9
Box	25088	12546	24758	317	13	33→17	10.7	4	2	4	10
Hand	11208	5606	11000	195	13	20→11	4.19	2	1	3	7

the negatively oriented triangles if $\overline{q_j q_k}$ is used as an observer. If the two conditions are not satisfied, we select the endpoint q with the highest number of $H^+(q)$ as the observer point. We repeatedly check each of the remaining bones for creating either an observer line segment or an observer point. α is set to 0.1 by preliminary experiments.

7 Experimental Results

We performed our experiments on an Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz with 4 GB RAM (one thread was used). Double floating point precision was used in solving the cubic equations. The type of the bounding volume is 16-DOPs.

Six animations were used as the testing benchmarks in our experiments: **Rope:** A rope is dropped onto a hard surface; **Octopus:** An octopus moves its legs; **Urchin:** A spiky surface bends; **Man:** A man walks in a circle; **Box:** A box is twisted; **Hand:** A hand motion. In **Urchin**, the observer line segments were effective due to its sharp geometric features. The snapshots of the animations are shown in Fig. 11. All animations but Man were generated by skin deformation driven by skeletons, which are used as the skeletons in our approach. **Man** is obtained from `http://people.csail.mit.edu/drdaniel/mesh_animation/`. We only used the captured mesh data while the skeleton of the character is manually constructed for each frame. In this case, the deformation of cloth was not driven by the skeleton.

The complexities of the deformable models in these examples ranged from 2.8K triangles to 48K triangles as shown in Table 1. The average number of H^+, H^-, and H^u triangles across different frames in the six animations are listed in Table 1. As predicted, most of the triangles are in H^+. Table 1 also lists the computational time for each preprocessing task including cluster decomposition, cluster boundary refinement, and cluster reduction. All the preprocessing tasks can be complete within less than 0.02 seconds except for cluster decomposition. Cluster decomposition is a lot more expensive because it performs depth field discretization [Frisken et al. 2000] which takes between 4.19 and 23.3 seconds. In **Urchin**, the model consists of many spiky geometric features which slows down depth field discretization since they cause a lot more cells to be created. Computing the ICOS takes less than 0.02 seconds for all the animations.

Comparison. We compared our method with a K-DOP method denoted as Base and a modified ICCD method denoted as MCD. MCD is a modified version of ICCD ([Tang et al. 2009a]), which is one of the representative techniques in continuous self-collision detection for general deformable models. The major features of MCD include: (1) Continuous normal cones for grouping triangles; (2) Procedural representative triangles for eliminating duplicate tests of feature pairs; (3) Orphan set for handling feature pairs of adjacent triangles; (4) Deforming non-penetration filters (DNPFs) [Tang et al. 2010]. The difference between MCD and ICCD is that MCD employs DNPFs which makes it faster than ICCD. MCD computes continuous normal cones but ignores contour tests. In Base, the culling method is completely based on the bounding volume tests. DNPFs are also adopted in Base.

All the methods (ours, Base, MCD) used 16-DOP as their bounding volume and utilized the median splitting scheme for constructing the BVH of the models [Heo et al. 2010]. We did not apply any lazy update schemes or reconstruction techniques for BVHs at the runtime stage [Larsson and Akenine-Möller 2006]. All the methods only detect collisions for non-adjacent feature pairs.

We first compare the average number of bounding volume (BV) tests. Table 2 shows the comparison result. Our method only generates 18% to 39% of BV tests of Base. When compared to MCD, we only get 25% to 68%. As our method decomposes deformable models into clusters, the number of BV overlapping tests is reduced. Fig. 12(a) shows the number of the BV tests in **Rope** during the simulation. Our method performs much fewer BV tests than Base and MCD because our method ignores clusters with only positively oriented triangles. The fewer BV tests are also due to cluster decomposition which makes the triangles of each cluster move coherently such that their orientations do not change often during deformation.

Table 2: *The average number of bounding volume overlapping tests and the ratios.*

	Ours			$\frac{\text{Ours}}{\text{Base}}$	$\frac{\text{Ours}}{\text{MCD}}$
	Inter-Cluster	Intra-Cluster	Total		
Rope	252K	8K	260K	0.29	0.68
Octopus	98K	255K	353K	0.30	0.42
Urchin	6.9K	5.4K	12.3K	0.22	0.25
Man	64.8K	130.0K	194.8K	0.39	0.53
Box	64.3K	10.2K	74.5K	0.22	0.46
Hand	19.3K	12.1K	31.4K	0.18	0.28

Table 3: *The average query time of the collision detection in our method, speedup factor over Base and MCD, and the culling ratio over Base in PCTPs.*

	Query time (msec)	$\frac{\text{Base}}{\text{Ours}}$	$\frac{\text{MCD}}{\text{Ours}}$	$\frac{\text{Our \#PCTP}}{\text{Base \#PCTP}}$
Rope	74.1	6.92x	2.59x	0.08
Octopus	107.2	4.37x	2.12x	0.13
Urchin	3.33	7.57x	3.15x	0.07
Man	59.8	4.31x	2.54x	0.19
Box	16.6	8.03x	1.78x	0.02
Hand	9.9	7.42x	2.35x	0.04

Table 3 shows the average query time of collision detection of our method and the culling ratio over Base. Our method is up to 8.03 and 3.15 faster than Base and MCD, respectively. The culling ratio, which is defined as the number of potentially colliding triangle pairs (PCTPs) of our method divided by that of Base, is less than 10% on average. These comparisons show that our method effectively culls away non-colliding triangle pairs. Fig. 12(b) shows the number of PCTPs in **Rope**. When a long **Rope** falls to the ground, the displacement of the clusters near the ends is larger than the length of the clusters. This kind of motion induces a large number of PCTPs in Base. Our method processes the fewest number of PCTPs as the triangles are grouped into clusters. We do not need to collect the non-adjacent triangle pairs of a cluster if all the triangles of the cluster are positively oriented w.r.t. the observer primitive of the cluster within the time interval.

Table 4 shows the comparison of the computational time at different stages and the average number of PCTPs. The time spent on BVH updates is almost the same for all the methods since the BVH node counts are about equal. As for the BVH traversal time (BVH-TRA), our method outperforms Base and MCD in all the six benchmarks. For example, the BVH-TRA of our method is reduced by 63% (**Man**) to 81% (**Hand**) of that of Base, and reduced by 43% (**Rope**) to 77% (**Urchin**) of MCD's. In the elementary test processing time (ETP), our method also outperforms the other two methods in all benchmarks. For example, compared to Base, our method reduces ETP by 82% (**Man**) to 98% (**Box**). Compared to MCD, ETP is reduced by 58% (**Octopus**) to 79% (**Urchin**). Our method for performing radial-view update (RV-Update) is less than 6.8 $msec$ in all the benchmarks.

Our method for classifying triangles and propagating triangle set counts to ascendants in the BVH is efficient. In MCD (or other CNC-based methods, such as [Tang et al. 2009a]), it is required to compute the continuous normal cone (i.e., an axis and an opening angle) of each deforming triangle. Then the continuous normal cones of the internal nodes of the BVH are updated in a bottom-up manner. In contrast to MCD, our method only needs to determine the signs of the characteristic functions of the triangles. Nevertheless, our method is catered for deformable models with skeletons while MCD can be applied to general deformable models.

We compared our method with SelfCCD [UNC Gamma Group] which can be applied for self-collision detection of general deformable models. The same set of benchmarks were performed on the same machine. The speedup factor of our method over SelfCCD is up to five.

Robustness. Our method does not require very precise skeleton information nor very smooth skeleton movements. It is preferred but not required that the most of the skeleton lies inside the model during the deformation. In fact, the movement of the skeleton in **Man** contains many noisy disturbances since they are manually constructed. To further verify the robustness of our method, we added random displacements to the joint positions of the skeleton in each frame. Let r be the radius of the inscribed ball centered at each joint. A random displacement vector $sr\mathbf{u}$ was computed for the joint, where $s = 0.1$, 0.2 and 0.3 is the magnitude ratio of the displacement and $\mathbf{u}$ is a unit-length random vector. The performance of our method was not affected much by the random displacement. The average difference is less than 5% except **Urchin**. In **Urchin**, the average difference is around 10% due to its spiky geometric features. We also conducted experiments for observer points lying outside the mesh by increasing the displacement ratio s, as shown in Fig. 1(e). Let L be the percentages of observer points subjected to noisy displacement. For $L = 20\%$, the total time of collision detection increases by 15% and 20% for $s = 1.2$ and $s = 1.4$, respectively. Fig. 1(e) shows the skeleton of the rope when $L = 50\%$ and $s = 1.4$ at a frame.

Limitations. Our method has four limitations. First, the input model should be a closed 2-manifold mesh. Our method may not be applied to cloth simulation, fracture simulation, or models with severe deformation since it is difficult to compute a good skeleton for cluster decomposition. Furthermore, models with a lot of geometric detail, causes performance to deteriorate if many observer primitives are generated during clustering. Second, the culling efficiency of our method may degrade if the observer primitives move outside the model. In this situation, there may be many negatively oriented or uncertain triangles. Third, our method may fail to detect new collisions within a cluster if the cluster intersects itself initially. This limitation may be resolved by applying the contour test. Nevertheless, our method is still able to detect all the new collisions between a pair of clusters. Fourth, currently our method utilizes the skeleton motion for updating the positions of the observer primitives. If the deformation is not driven by a skeleton, we need to update the observer primitives based on other efficient methods.

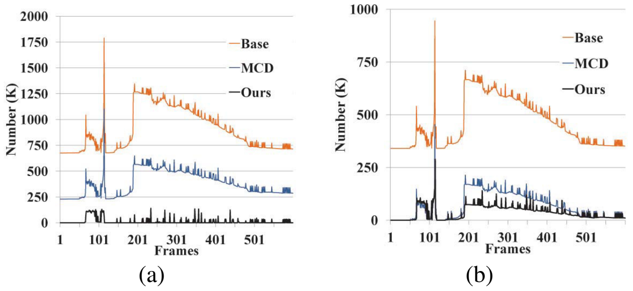

Figure 12: *The number of (a) the bounding volume tests and (b) potentially colliding triangle pairs in each frame in* **Rope**.

Discussion. The computational performance of our method largely depends on the number of clusters. If the number of clusters increases, time spent on inter-cluster checks usually also increases but the time spent on intra-cluster checks decreases. We are considering applying optimization methods to figure out an optimal number of clusters. We also compared the computational time of our method with and without performing cluster refinement. We found that the overall computational time is similar in both cases.

Our method can be easily integrated with other existing techniques aimed at improving BVH updates, BVH reconstruction, or elementary intersection tests. For instance, the lazy BVH update schemes and dynamic bounding volume hierarchies [Zachmann and Weller 2006; Larsson and Akenine-Möller 2006; Schvartzman et al. 2010] can be employed.

Currently, most of the precomputation time is spent on the distance field discretization [Baran and Popović 2007] for clustering. It would be interesting to explore the techniques for fast computation of distance fields of deformable models, e.g. [Fisher and Ming 2001]. On the other hand, the depth field computation can be skipped. To speed up the preprocessing for clustering, we can compute a seed vertex that is visible w.r.t. an observer primitive and then perform region growing to compute a cluster. It is a challenging problem to develop an efficient method for computing the clusters at real time rates. Note that our method can be combined with parallel processing techniques. For example, the BVH traversal and elementary test processing can be accelerated with existing techniques such as [Kim et al. 2009; Tang et al. 2009b]. In particular, parallel culling techniques are reported to be faster than the technique based on the deforming non-penetration filters [Tang et al. 2011a]. Finally, skeletonization methods, such as that of [Bradshaw and O'Sullivan 2004], can be used to automatize skeleton generation of a deformable model.

8 Conclusion

We present a novel radial-view-based culling method for closed deformable models that are embedded with skeletons. The heat diffusion process is applied to decompose the deformable models into

Table 4: *Average timings and the average number of potentially colliding triangle pairs (#PCTP). Timing items include: BVHU: BVH update; BVH-TRA: traversal for bounding volume hierarchy; ETP: the elementary test processing time; CNC: computation for continuous normal cones; and RV-Update: Radial-view update.*

	Base				MCD					Ours				
	BVHU (ms)	BVH-TRA (ms)	ETP (ms)	#PCTP (K)	BVHU (ms)	CNC (ms)	BVH-TRA (ms)	ETP (ms)	#PCTP (K)	BVHU (ms)	RV-Update (ms)	BVH-TRA (ms)	ETP (ms)	#PCTP (K)
Rope	19.2	33.9	459.7	457	19.3	12.8	18.0	142	67.4	19.6	5.9	10.3	38.3	35.4
Octopus	20.6	44.7	402.9	467	20.5	12.5	34.2	160	131	20.2	6.8	12.5	67.7	60.0
Urchin	1.2	2.3	21.7	26.1	1.2	0.7	2.3	6.3	4.9	1.1	0.4	0.53	1.3	1.7
Man	9.4	19.6	228.5	237	9.4	5.4	16.0	121	94.4	9.2	3.0	7.3	40.3	45.8
Box	8.7	14.4	110.2	156	8.8	6.4	8.4	5.9	4.6	8.7	3.0	3.0	1.9	2.9
hand	5.0	7.3	61.2	78.1	5.0	2.9	5.2	10.2	7.8	4.9	1.3	1.4	2.3	3.2

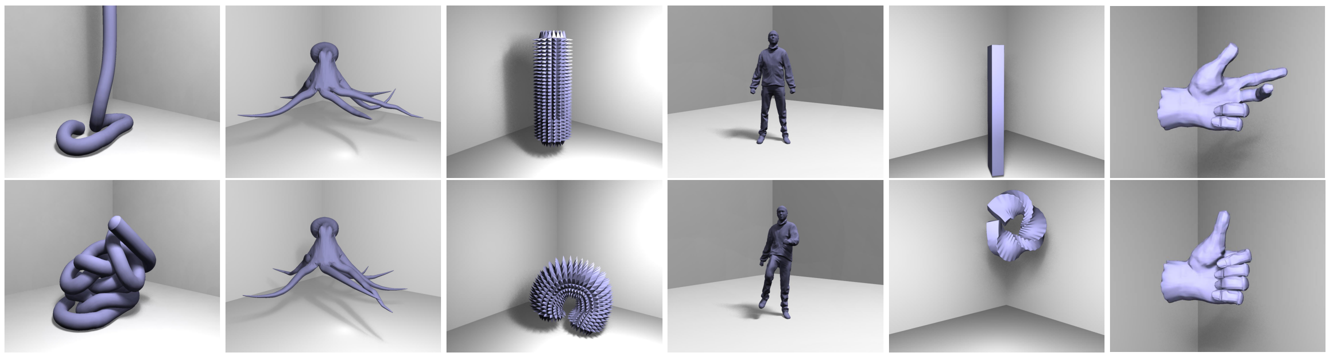

Figure 11: *The snapshots of the testing animations.*

clusters that are created by employing the observer points and line segments. Due to this decomposition, the radial-view test can be used for efficiently determining whether or not a cluster has self-collisions. The inter-cluster orphan set method helps us cull adjacent triangle pair tests between adjacent clusters. Our experimental results show that our method reduces the number of bounding volume overlapping tests significantly. Although our method is restricted to models with a provided skeleton, it still covers a pretty large class of interesting models. Besides skeleton-driven deformation, our method can also be applied to the characters reconstructed by motion capture techniques, where a predefined skeleton may exist but the information associating the mesh to the skeleton in a physical or geometric way may not be built. On the other hand, for mesh animation without skeleton, any skeleton extraction method can be utilized to extract skeletons from the mesh and our method is still applicable. Furthermore, our method may be applied to deformable models simulated by Finite Element Method in which the models consist of tetrahedra. The observer points can be computed based on the interior vertices of the models.

There are several avenues for future work. Firstly, the computation of the distance field is a bottleneck in the preprocessing stage. The distance field is only used for checking the visibility status of triangles w.r.t. bones of the skeleton. In our case, we only need to check for the orientations of triangles w.r.t. observer primitives. Thus, we would like to develop a fast method for performing cluster decomposition without using the distance field. Secondly, it would be interesting to extend our method to handle mesh surfaces with holes, which for example, could be filled with virtual triangles. Finally, an efficient method should be developed for updating the positions of the observer primitives if the deformation of the model is not skeleton-driven. We are considering to adopt the generalized barycentric coordinates. For example, a set of triangles can be precomputed for each observer primitive. In the runtime stage, the position of each observer primitive is updated based on the set of triangles associated with the observer primitive.

Acknowledgements

The authors would like to thank the referees for their insightful comments and suggestions. The idea of using the generalized barycentric coordinates for updating observer primitives was credited to an anonymous referee. We would like to thank I-Ting Fu for helping make the movie. This work was supported in part by the National Science Council of ROC (Taiwan) (NSC-101-2221-E-009-157, NSC-101-2628-E-009-021-MY3) and the UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant Number NSC-101-2911-I-009-101.

References

Baran, I., and Popović, J. 2007. Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics 30*, 7, 2087–2096.

Barbič, J., and James, D. L. 2010. Subspace self-collision culling. *ACM Transactions on Graphics 29*, 3, 81:1–81:9.

Bradshaw, G., and O'Sullivan, C. 2004. Adaptive medial-axis approximation for sphere tree construction. *ACM Transactions on Graphics 23*, 1, 1–26.

Bridson, R., Fedkiw, R., and Anderson, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics 21*, 3, 594–603.

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Transactions on Graphics 21*, 3, 586–593.

Capell, S., Burkhart, M., Curless, B., Duchamp, T., and Popović, Z. 2005. Physically based rigging for deformable characters. In *Symposium on Computer Animation*, 301–310.

CURTIS, S., TAMSTORF, R., AND MANOCHA, D. 2008. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, 61–69.

FISHER, S., AND MING, C.-L. 2001. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Eurographics Workshop on Computer Animation and Simulation*, 99–111.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of ACM SIGGRAPH*, 249–254.

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 Transactions on Graphics 24*, 3, 991–999.

HEO, J.-P., SEONG, J.-K., KIM, D.-S., OTADUY, M. A., HONG, J.-M., TANG, M., AND YOON, S.-E. 2010. FASTCD: Fracturing-aware stable collision detection. In *Symposium on Computer Animation*, 149–158.

KAVAN, L., O'SULLIVAN, C., AND ŽÁRA, J. 2006. Efficient collision detection for spherical blend skinning. In *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, 147–156.

KIM, D., HEO, J., HUH, J., KIM, J., AND YOON, S. 2009. HPCCD: Hybrid parallel continuous collision detection using CPUs and GPUs. *Computer Graphics Forum 28*, 7, 1791–1800.

KLOSOWSKI, J., HELD, M., MITCHELL, J., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics 4*, 1, 21–36.

KNOTT, D., AND PAI, K. 2003. Cinder: collision and interference detection in real-time using graphics hardware. In *Graphics Interface*, 73–80.

KOPPERMAN, R., MEYER, P., AND WILSON, R. 1991. A Jordan Surface Theorem for three-dimensional digital spaces. *Discrete & Computational Geometry 6*, 2, 155–161.

LARSSON, T., AND AKENINE-MÖLLER, T. 2006. A dynamic bounding volume hierarchy for generalized collision detection. *Computers & Graphics 30*, 3, 450–459.

MEZGER, J., KIMMERLE, S., AND ETZMUSS, O. 2003. Hierarchical techniques in collision detection for cloth animation. *Journal of WSCG 11*, 2, 322–329.

REDON, S., KHEDDAR, A., AND COQUILLART, S. 2002. Fast continuous collision detection between rigid bodies. In *Computer graphics forum*, vol. 21(3), 279–288.

SCHVARTZMAN, S., GASON, J., AND OTADUY, M. 2009. Bounded normal trees for reduced deformations of triangulated surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 75–82.

SCHVARTZMAN, S., PÉREZ, A. G., AND OTADUY, M. A. 2010. Star-contours for efficient hierarchical self-collision detection. *ACM Transactions on Graphics 29*, 3.

SUNDAR, H., SILVER, D., GAGVANI, N., AND DICKINSON, S. 2003. Skeleton based shape matching and retrieval. In *Proceedings of the Shape Modeling International*, 130–142.

TANG, M., CURTIS, S., YOON, S., 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., MANOCHA, D., AND TONG, R. 2009. Multi-core collision detection between deformable models. In *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, 355–360.

TANG, M., MANOCHA, D., AND TONG, R. 2010. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 7–13.

TANG, C., LI, S., AND WANG, G. 2011. Fast continuous collision detection using parallel filter in subspace. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 71–80.

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

TANG, M., MANOCHA, D., YOON, S.-E., DU, P., HEO, J.-P., AND TONG, R. 2011. VolCCD: Fast continuous collision culling between deforming volume meshes. *ACM Transactions on Graphics 30*, 5.

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. In *Computer Graphics Forum*, 61–81.

UNC GAMMA GROUP. SelfCCD: Continuous collision detection for deforming objects. `http://gamma.cs.unc.edu/SELFCD`.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1994. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Computer Graphics Forum*, 155–166.

WONG, S.-K., AND BACIU, G. 2005. Dynamic interaction between deformable surfaces and non-smooth objects. *IEEE Transactions on Visualization and Computer Graphics 11*, 3, 329–340.

WONG, S.-K., AND BACIU, G. 2006. A randomized marking scheme for continuous collision detection in simulation of deformable surfaces. In *Proceedings of the ACM Int'l Conf. on Virtual Reality Continuum and Its Applications*, 181–188.

YU, W., AND LI, X. 2011. Computing 3D shape guarding and star decomposition. *Computer Graphics Forum 26*, 3, 1–8.

ZACHMANN, G., AND WELLER, R. 2006. Kinetic bounding volume hierarchies for deformable objects. In *ACM Int'l Conf. on Virtual Reality Continuum and Its Applications*, 14–17.

ZHANG, X., AND KIM, Y. 2012. Simple culling methods for continuous collision detection of deforming triangles. *IEEE Transactions on Visualization and Computer Graphics 18*, 7, 1146 – 1155.

ZHENG, C., AND DOUG, J. L. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Transactions on Graphics 31*, 4.