Dynamic Radial View Based Culling for Continuous Self-Collision Detection

Sai-Keung Wong Wen-Chieh Lin Yu-Shuen Wang Chun-Hung Hung Yi-Jheng Huang
National Chiao Tung University, Taiwan

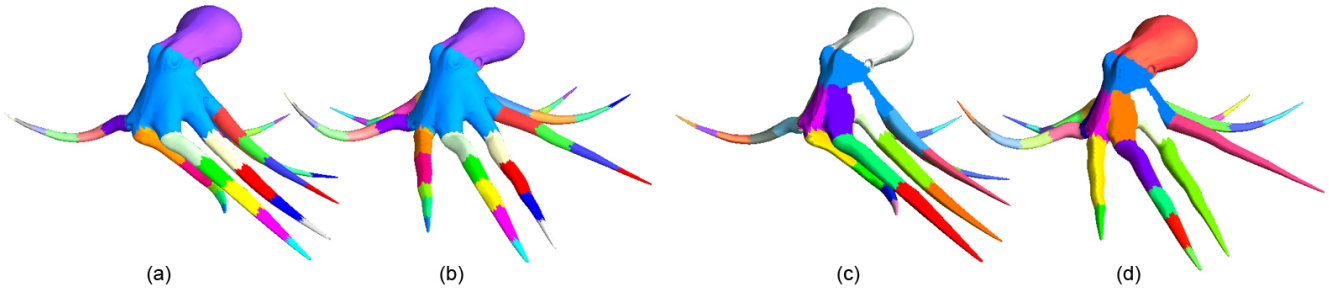

Figure 1: *(a) (b) The clusters are fixed in the radial view based culling (RVBC) method at runtime. (c) (d) Our method dynamically merges the atomic clusters to improve the overall performance of continuous self-collision detection over RVBC. The speedup of our method compared to RVBC is 5.2× in this octopus example.*

Abstract

The radial view-based culling (RVBC) method has been presented for continuous self-collision detection to efficiently cull away non-colliding regions. While this technique mainly relies on the segmented clusters of the reference pose and the associated fixed observer points, it has several drawbacks during the animation and the reduced cost of executing collision detection is limited. We thus present a modified framework to improve the culling efficiency of RVBC. At the preprocessing stage, we segment the closed deformable mesh according to not only the attached skeleton but also the triangle orientations, in order to minimize the collision checks of triangles in a cluster. At the runtime stage, we dynamically merge adjacent clusters and update the positions of observer points if the merged shape is nearly convex. This strategy minimizes the number of triangles in different clusters that required collision check. Our framework can be easily integrated with bounding volume hierarchies to boost the culling efficiency. Experimental results show that our framework achieves up to 5.2 times speedup over the original RVBC method and even more times over the recent techniques.

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

Keywords: continuous self-collision detection, deformable model, culling technique, dynamic clustering

1 Introduction

Continuous self-collision detection (CSCD) has been developed to compute the colliding regions of a deformable mesh within a time interval. While the traditional discrete methods focus on the intersection state at an instance of time, CSCD determines whether or not collision events happen within a period of time and enables simulating deformable surfaces realistically [Bridson et al. 2002]. Given that the cost of continuous self-collision detection is expensive, a variety of techniques have been presented to accelerate the CSCD process. The goal is to improve the culling efficiency by avoiding the check of triangle pairs that are close to each other but collision is not possible. Among the CSCD techniques, bounding volume hierarchies (BVHs) are widely used. If the bounding volumes do not overlap, the triangles lying inside the respective bounding volumes are not required to check.

Recently, a radial view-based culling (RVBC) method is presented to improve the culling efficiency of CSCD [Wong et al. 2013]. Compared to the general BVHs, which strives to prevent collision checks of triangles from different clusters (i.e., inter-cluster check), the goal of RVBC is to reduce collision checks of triangles in the same cluster (i.e., intra-cluster check). This technique is particularly designed for skeletal meshes because character animations are mostly driven by skeletons or can be represented by using skinning meshes [James and Twigg 2005]. Specifically, given a closed deformable mesh, RVBC segments a mesh into clusters based on the attached skeleton. For each cluster, an observer point is placed to check whether or not the viewing directions are similar to the normals of the corresponding triangles. If this condition is satisfied, the cluster is free of self-collision and the collision check of triangles in the cluster can be omitted. However, the segmentation only relies on the distance from a vertex to a bone and the associated observer point is fixed. This strategy easily results in many triangles whose normals are dissimilar to the viewing directions, and the collision check of these triangles are still demanded at the runtime stage. The problem becomes even worse when the deformed mesh has a different shape compared to its original shape. Besides, the RVBC method may over-segments the mesh due to highly deformable animations and thus the reduced cost of executing collision detection is limited.

To prevent the problems, at the preprocessing stage, we refine the clusters once the deformable mesh is segmented based on the attached skeleton. Specifically, each triangle belonging to a cluster may be reassigned to one of its adjacent clusters in order to allow the triangle having a view direction similar to its normal. At the runtime stage, we merge adjacent clusters whenever their merged shape is nearly convex so as to reduce the total number of clusters. Then, we update the position of the observer point during the animation. This improvement is very effective because the merged clusters are close to each other. A self-collision detection approach using the bounding volume technique has to traverse the structure to a deep level and in many scenarios the triangle checks cannot be omitted. Apparently, our method would greatly reduce the cost of inter-cluster check, although that of intra-cluster check may increase a little bit. Our experimental results show that our method achieves up to 5.2 times speedup compared to the original RVBC method even though the dynamic cluster merging has to be computed in each time step.

Our key contributions that improve continuous self-collision detection are 1) quality improvement of each cluster and 2) dynamic cluster merging and observer point updating. As our method is designed for high-level culling, it can be easily integrated with other existing collision detection methods.

2 Related Work

Continuous self-collision detection plays an important role to simulating deformable surfaces realistically [Brochu et al. 2012]. To achieve the goal, vertex-triangle and edge-edge pairs are checked to determine whether or not the mesh collides with itself. Considering the heavy computational cost raised by the complex mesh structure, one way is to leverage the multiple-core CPU/GPU computing architectures to achieve parallel computing [Kim et al. 2009; Tang et al. 2009b; Pabst et al. 2010; Tang et al. 2011b]. Another is to prevent tests of triangle pairs that are not possible to collide. Our work falls into this category. Below we describe the details.

Bounding volume hierarchy. Bounding volume hierarchies (BVHs) are widely used to eliminate non-proximal triangle pairs. Many applications such as cloth simulation [Bridson et al. 2002], character animation [Capell et al. 2002; Baran and Popović 2007] and tetrahedral models [Tang et al. 2011c] were benefited from this data structure. The idea of using BVHs is that, triangles from different bounding volumes must be collision free if their bounding volumes are collision free. However, for the triangles in the same bounding volume, the BVH traversal is required to the lowest level, even though the entire mesh is collision free.

Normal cone-based techniques. Observing the smooth meshes having a less chance to self-collide in local regions, Volino and Magnenat-Thalmann [1994] presented the first idea to reduce the cost of discrete self-collision detection. They divided the deformable mesh into sub-meshes and constructed a hierarchical structure to represent their relations. At runtime, the method computes a normal cone for each sub-mesh and determines whether or not a sub-mesh intersects itself by first examining the respective cone and the collision state of the contour of the sub-mesh. The method was then extended by Schvartzman et al. [2009] to handle reduced deformable models. They also presented an efficient data structure based on star-contours to improve the process of contour tests in discrete self-collision detection [2010]. Later, the discrete normal cone-based method was used to continuous self-collision detection [Wong and Baciu 2005; Tang et al. 2009a]. Tang et al. [2009a] also improved the process of continuous contour tests by projecting contour edges onto a plane and using axis-aligned bounding boxes to bound the edges. They applied overlapping tests for boxes to prevent the unnecessary tests of edge-edge pairs. In [Heo et al. 2010], a dual-cone culling method and a selective restructuring method were proposed for fracturing objects. Our method aims for closed deformable objects.

Energy based approach. Zheng and Doug [2012] assumed a deformable mesh is collision free if the bending is gentle. They presented an energy-based method to determine if the mesh bending exceeds a certain degree based on an affine invariant Laplacian. Specifically, a set of certificates are determined at the preprocessing and deformation energy of each local region is computed at runtime. To achieve culling efficiency, they considered the region to be collision free if the energy is less than a precomputed value.

Other techniques. In [Barbič and James 2010], subspace self-collision culling certificates are computed for evaluating whether or not reduced deformable objects have self-collisions. In [Kavan and Zara 2005], a collision detection method is proposed for skeletal deformable objects with linear blending. An efficient sphere-tree structure is proposed for bounding the objects. Our method also aims for skeletal deformable objects but we do not make assumption on the deformation of the objects.

Radial view based culling. Most of the above high level culling techniques strived to prevent the collision checks of triangles from different clusters. Recently, a radial view-based culling [Wong et al. 2013] method was presented to prevent the checks of triangles in the same cluster. The main idea of this algorithm is to assign an observer point for a segmented cluster. Then, it checks whether or not the viewing directions from the point to the triangles that have similar directions to the respective triangle normals. The cluster is considered collision free if this condition is satisfied. However, the method may over-segments the mesh and results in unnecessary computation when checking collisions of different clusters. In contrast, our framework allows clusters merging dynamically so as to minimize the inter-cluster check and boost the performance up to a 5.2 times speedup compared to the original RVBC method.

Elementary tests. While vertex-triangle and edge-edge pairs cannot be culled away, elementary tests are inevitable. Considering a vertex shared by triangles, the vertex-triangle pairs involved with that vertex could be checked for multiple times. This problem also happens at the checks of edge-edge pairs. Therefore, Curtis et al. [2008] assigned vertices and edges to one of their incident triangles, and checked only the assigned triangles when detecting collision. In addition, given that computing the contact time of the two features requires solving the equation using Newton's interval, which is expensive. Low cost filters such as deforming non-penetrating filters [Tang et al. 2010] and parallel filters [Tang et al. 2011a] are presented. We refer the readers to [Zhang and Kim 2012; Du et al. 2012] for more details of the elementary tests.

3 Theoretical Background and Overview

We first review the radial based view culling (RVBC) and present the overview of our method in this section. Given that our algorithm is designed for continuous self-collision detection, we assume the velocity of a vertex is linear within the time interval $[0, \Delta t]$, where $\Delta t = 1$ in our implementation. Therefore, the position of vertex $\mathbf{p}$ can be expressed as $\mathbf{p}(t) = \mathbf{p}(0) + \mathbf{v}t$, with $\mathbf{p}(0)$ and $\mathbf{v}$ the start position and the velocity, respectively. The normal vector $\mathbf{n}_T(t)$ of each triangle $T(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ points outside of the deformable mesh and it is defined as: $\mathbf{n}_T(t) = (\mathbf{p}_2(t) - \mathbf{p}_1(t)) \times (\mathbf{p}_3(t) - \mathbf{p}_1(t))$.

3.1 Radial View Based Culling

Based on the condition of collision free stated in RVBC [Wong et al. 2013], *the mesh is collision free if it is free of self-collision initially and the mesh triangles are all positively oriented with respect to the observer point.* Specifically, consider a vertex $\mathbf{p}$ of triangle T and an observer point $\mathbf{q}$. T is positively oriented with respect to $\mathbf{q}$ if $(\mathbf{p}(t) - \mathbf{q}(t)) \cdot \mathbf{n}_T(t) > 0, \forall t \in [0, \Delta t]$. The triangle is considered *negatively oriented* or *uncertainly oriented* if the dot product of the two vectors is negative or zero within the time interval, respectively (illustrated in Fig. 2). We show the triangles of a sphere that are all positively oriented with respect to an observer point $\mathbf{q}$ in Fig. 3(a). The transformations of the sphere such as isotropic scaling, rotation, and translation do not change the state of triangles from positively oriented to another (Fig. 3(b) and (c)). The sphere remains self-collision free after the deformation.

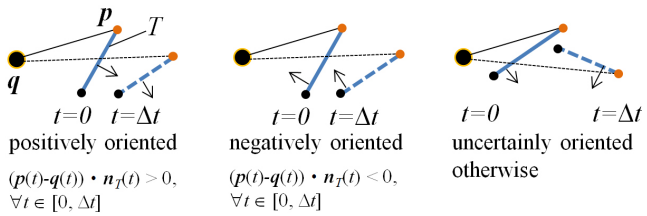

Figure 2: *The types of a triangle T with respect to an observer point $\mathbf{q}$ within a time interval $[0, \Delta t]$. $\mathbf{p}$ is a vertex of T. The arrows indicate the normal vector $\mathbf{n}_T(t)$ of T at different time instances. Note that we determine the type of a triangle within the time interval (not at two discrete time instances).*

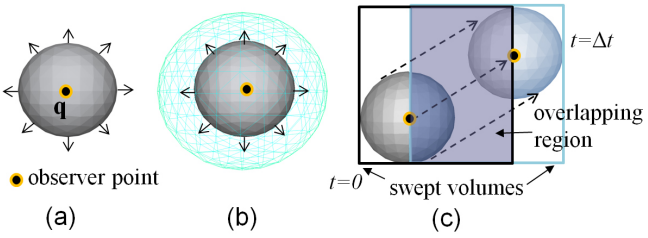

Figure 3: *A deforming sphere and an observer point $\mathbf{q}$. The short arrows indicate the normal vectors. All the triangles are positively oriented and they are collision free within the time interval $[0, \Delta t]$. (a) Initial state of a sphere; (b) The sphere expands; (c) The sphere translates. BVH based methods require collision checks for triangles lying inside the overlapping region of the swept volumes (marked in gray).*

Intuitively, the key to achieve high culling efficiency is increasing the number of positively oriented triangles with respect to the observer point. Hence, RVBC divides a mesh $\mathbf{M}$ into a set of clusters $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_n\}$ *s.t.* $\triangle\mathbf{C}_i \cap \triangle\mathbf{C}_j = \emptyset$ if $i \neq j$, where $\triangle\mathbf{C}$ denotes the set of triangles in $\mathbf{C}$. In addition, each cluster $\mathbf{C}_j$ is associated with an observer primitive $o(\mathbf{C}_j)$. In the method, $\mathbf{H}^+(\mathbf{C}, o(\mathbf{C}))$, $\mathbf{H}^-(\mathbf{C}, o(\mathbf{C}))$, and $\mathbf{H}^u(\mathbf{C}, o(\mathbf{C}))$ are denoted as the sets of triangles that are positively, negatively, and uncertainly oriented in cluster $\mathbf{C}$ with respect to the observer point $o(\mathbf{C})$, respectively. Note that the three sets are updated within each time interval because of the dynamic vertex positions.

Given that the mesh is segmented, collisions should be detected using intra- and inter- cluster checks. The former procedure detects whether or not the triangles collide with each other in the same cluster while the latter one handles the triangles from different clusters.

To execute an intra-cluster check, the method only needs to consider triangle sets under the two conditions: 1) $\mathbf{H}^-(\mathbf{C}, o(\mathbf{C}))$ to $\mathbf{H}^+(\mathbf{C}, o(\mathbf{C}))$ and 2) $\mathbf{H}^u(\mathbf{C}, o(\mathbf{C}))$ to $\triangle\mathbf{C}$, because the triangles that are all positively oriented or all negatively oriented do not collide. On the other hand, to execute an inter-cluster check, one can simply utilize existing methods such as bounding volume overlapping tests [Curtis et al. 2008; Tang et al. 2009a].

3.2 Algorithm Overview

Given a 3D mesh embedded with a skeleton, our system segments the mesh into atomic clusters according to the attached bones and the triangle orientations. To reduce the cost of intra-cluster check, each cluster is associated with an observer primitive on the bone with the goal of maximizing the number of positively oriented triangles. Specifically, an observer primitive could be a point or a line. The former implies a fixed viewing point while the latter allows the viewing position sliding along the bone segment. The positions of the observer primitives are updated as the bones are transformed during a time interval at the runtime stage. These atomic clusters and the associated observer primitives are precomputed at the preprocessing stage.

Although the use of atomic clusters can efficiently reduces the cost of intra-cluster check, more atomic clusters results in a more expensive inter-cluster check. An extreme example is considering each triangle a cluster. In such case the performance of RVBC method is poor. We thus strive to merge neighboring atomic clusters to a mega cluster at runtime in order to reduce the cost of inter-cluster checks while increasing little or even no cost to intra-cluster checks. To achieve this goal, we update the observer primitives when atomic clusters are merged and count the number of non-positively oriented triangles. In our implementation, neighboring clusters will be merged only when less than 5% of the triangles change the state from positively oriented to another. The experiment shows that our dynamic merging scheme significantly cuts down the number of bounding volume overlapping tests and the number of potentially colliding triangle pairs. Fig. 4 illustrates the cluster hierarchy. We show the details of atomic cluster precomputation and dynamic mega cluster merging in Sections 4 and 5, respectively.

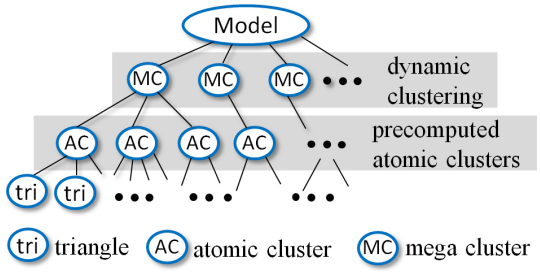

Figure 4: *The hierarchical structure maintained in our method.*

4 Atomic Clusters

We segment the mesh into atomic clusters based on the bone segments because most character animations are driven by a skeleton. Each cluster is associated with an observer primitive (i.e., a point or a line). Under this circumstance, triangles in a cluster usually have similar transformations except those regions that are close to the cluster boundary. Most positively oriented triangles can remain positively oriented during the motion if the associated observer primitive is updated appropriately.

4.1 Mesh segmentation

Similar to the RVBC method [Wong et al. 2013], one can simply apply the heat diffusion function to compute bone weights for each vertex and assign a triangle to the nearest bone. However, this function considers only the distance from the bone to the surface and results in many non-positively oriented triangles no matter where the observer primitive is placed. Fig. 5 (a) and (b) show an example, in which the triangles in the red box are assigned to the upper arm cluster. Apparently, these triangles are negatively oriented because the observer primitive has to locate at the upper arm region. To solve this problem, we refine the atomic clusters by re-assigning these triangles to appropriate neighboring clusters. Specifically, we first remove the non-positively oriented triangles from each atomic cluster. We then grow each cluster by merging the triangles adjacent to the cluster itself, with the constraint that these triangles should be positively oriented to the observer primitive associated with the cluster. While there may be some triangles remaining unassigned due to the representation of fine details or a bumpy surface, we omit the positively oriented constraint and grows each cluster again to ensure that each triangle belongs to a cluster. Fig. 5 (c) and (d) show the results after our cluster refinement. As can be seen, the total number of non-positively oriented triangles is reduced.

A discrete distance field is required to compute the heat diffusion. The resolution of this field could be very high resolution due to fine details of a 3D mesh, resulting in heavy computational cost. We observe that the heat values are only used to assign vertices to bones. Thus, the heat values need not to be precise in our framework. We first smooth the mesh and apply a coarse discrete distance field to reduce the cost. Our experiment results show that this strategy brings a significant speedup in cluster segmentation compared to the RVBC method.

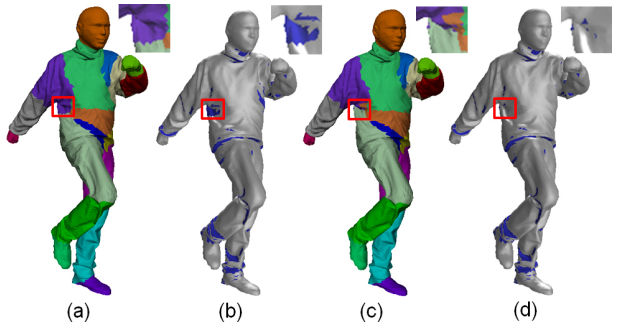

Figure 5: *Reassignment of non-positively oriented triangles to neighbouring atomic clusters. (a)(b) → (c)(d) shows the difference from before reassignment to after reassignment. (a)(c) The atomic clusters. (b)(d) Non-positively oriented triangles are marked in blue.*

4.2 Observer primitives

Given that an observer point is fixed at a time instance while an observer line allows the viewing position to slide along the bone segment, intuitively the cluster will contain more positively oriented triangles if a line structure is used. However, computing the viewing positions of all triangles in a cluster is expensive. We thus simply associate the cluster with an observer point if the surface of the cluster is smooth. This observer point locates at the bone center. In contrast, we associate the cluster with an observer line if the cluster has sharp geometrical features. In this way, we can reduce as many non-positively oriented triangles as possible. Specifically, an arbitrary vertex of the triangle is projected onto the bone segment to obtain the viewing position, followed by determining whether the triangle is positively oriented or negatively oriented (see Fig. 6).

Some triangles change orientation easily with respect to their observer primitives within a time interval because these triangles are almost coplanar with their observer primitives. Thus, we need to determine the suitable observer primitives for the atomic clusters. The observer points are preferred over the observer line segments because the projection of triangle vertices to the line segment is not necessary. Furthermore, if the relative velocity of two end points of an observer line segment is not zero, the rational polynomial function has a degree of five (see [Wong et al. 2013]), which is very expensive. We thus assume that the relative velocities of the two end points of observer line segments are zero when we determine the suitable observer primitives.

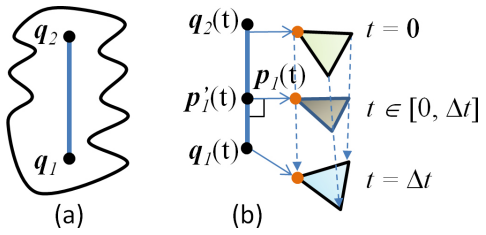

Figure 6: *(a) An observer line segment $\mathbf{q}_1\mathbf{q}_2$ and an atomic cluster; (b) A triangle vertex is projected onto $\mathbf{q}_1\mathbf{q}_2$ to determine the triangle orientation with respect to the observer primitive.*

Given an atomic cluster $\mathbf{C}^A$ with bone $\mathbf{q}_1\mathbf{q}_2$, we denote by $\mathbf{q} = \frac{1}{2}(\mathbf{q}_1 + \mathbf{q}_2)$ the observer point and by $\mathbf{q}_1\mathbf{q}_2$ the observer line. An observer primitive is determined for $\mathbf{C}^A$ as follows. We collect the triangles that have different orientations $\mathbf{H}^+(\mathbf{C}^A, o(\mathbf{C}^A))$, $\mathbf{H}^-(\mathbf{C}^A, o(\mathbf{C}^A))$, and $\mathbf{H}^u(\mathbf{C}, o(\mathbf{C}^A))$, where $o(\mathbf{C}^A) \in \{\mathbf{q}, \mathbf{q}_1\mathbf{q}_2\}$. Since cluster $\mathbf{C}^A$ may rotate when the mesh is deformed, we check whether or not the positively oriented triangles are stable. Therefore, we also consider another three conditions for $\mathbf{q}_1\mathbf{q}_2$. That is, we fix the joint $\mathbf{q}_1$ at its original position and rotate cluster $\mathbf{C}^A$ along x, y, and z axes by θ degrees, as illustrated in Fig. 7. The average numbers of positively oriented triangles $|\bar{\mathbf{H}}^+(\mathbf{C}^A, \mathbf{q}_1\mathbf{q}_2)|$ and $|\bar{\mathbf{H}}^+(\mathbf{C}^A, \mathbf{q})|$ under the mentioned four conditions are then computed (i.e., original, rotated by x, y, and z axes). Finally, we set the observer primitive to $\mathbf{q}$ if $|\bar{\mathbf{H}}^+(\mathbf{C}^A, \mathbf{q})| > |\bar{\mathbf{H}}^+(\mathbf{C}^A, \mathbf{q}_1\mathbf{q}_2)|$, and to $\mathbf{q}_1\mathbf{q}_2$ otherwise. In our animation examples, we found that it is adequate to set $\theta = 10$ for determining the type of observer primitives.

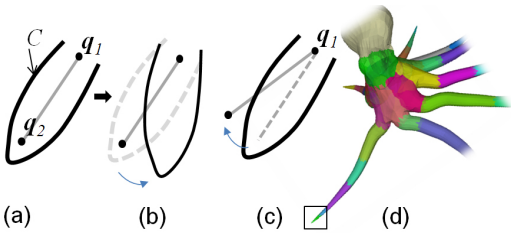

Figure 7: *Determining the quality of an observer line segment $\mathbf{q}_1\mathbf{q}_2$. (a) atomic cluster $\mathbf{C}$ for the end part of a tentacle of an octopus; (b) new position of $\mathbf{C}$ after rotation; (c) new position of $\mathbf{q}_1\mathbf{q}_2$ after rotation. (d) An octopus model.*

5 Dynamic Cluster Assembling

Two neighboring clusters can be merged whenever the merged shape of the two clusters is nearly convex because the non-

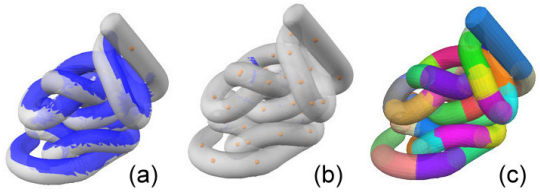

Figure 8: *One observer point vs. multiple observer points of a deformable mesh. (a) and (b) have the same shape but have different numbers of clusters. The deformable mesh is rendered transparently, negatively triangles are marked in blue, and observer points are visualized using small spheres. One can see that potentially colliding triangle pairs in (b) are much fewer than those in (a) due to more clusters. (c) shows the clusters of (b).*

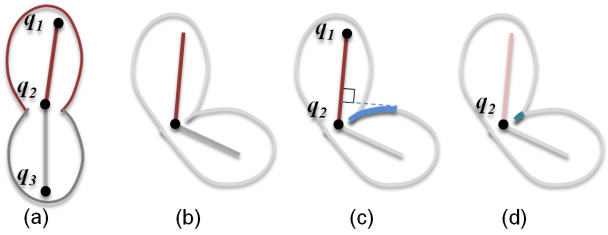

Figure 9: *Merging two atomic clusters. Non-positively oriented triangles are marked as blue. (a) Two atomic clusters before the deformation. (b) The two atomic clusters are merged into a mega cluster after deformation. (c) The bone $\mathbf{q}_1\mathbf{q}_2$ is chosen to be the observer primitive of the mega cluster. (d) The point $\mathbf{q}_2$ is chosen to be the observer primitive of the mega cluster, resulting fewer number of non-positively oriented triangles than that of using $\mathbf{q}_1\mathbf{q}_2$.*

positively oriented triangles are still few. The increased cost may be small or even no newly added triangle pairs that require further collision check. Hence, the cost of inter-cluster check can be greatly reduced in this scenario and the culling efficiency is improved. Figure 8 shows an example.

For a detailed explanation, from now on we denote by $\mathbf{C}^M$ the mega cluster. A mega cluster can be formally expressed as $\mathbf{C}^M = \{\mathbf{C}_1^A, \mathbf{C}_2^A, ..., \mathbf{C}_m^A\}$; and two mega clusters are disjoint. At each time interval, we merge atomic clusters to a mega cluster. We randomly select an atomic cluster as the seed $\hat{\mathbf{C}}^A$ and iteratively merge the adjacent atomic clusters until the cost is larger than a threshold. Our system will convert $\hat{\mathbf{C}}^A$ to a mega cluster before merging. We repeat the process that performs a random selection and a region growing process until all the atomic clusters are merged. We point out that our system resets mega clusters at each time interval and merges adjacent atomic clusters iteratively to prevent topological structures from becoming worse for radial view based culling. This achieves similar function of 'splitting' mega clusters.

To determine the merging cost, the number of non-positively oriented triangles are counted in our system. Let $\tilde{\mathbf{C}}^A$ be an atomic cluster adjacent to the mega cluster $\mathbf{C}^M$. Also remind that $\mathbf{H}^-(\tilde{\mathbf{C}}^A, o(\mathbf{C}^M))$ and $\mathbf{H}^u(\tilde{\mathbf{C}}^A, o(\mathbf{C}^M))$ are the negatively and the uncertainly oriented triangles of cluster $\tilde{\mathbf{C}}^A$ associated with an observer primitive $o(\mathbf{C}^M)$, respectively, and $|\triangle\tilde{\mathbf{C}}^A|$ is the total number of triangles in $\tilde{\mathbf{C}}^A$. We formulate the cost

$$D(\mathbf{C}^M, \tilde{\mathbf{C}}^A) = \frac{|\mathbf{H}^-(\tilde{\mathbf{C}}^A, o(\mathbf{C}^M))| + |\mathbf{H}^u(\tilde{\mathbf{C}}^A, o(\mathbf{C}^M))|}{|\triangle\tilde{\mathbf{C}}^A|}$$

and merge $\tilde{\mathbf{C}}^A$ to $\mathbf{C}^M$ if $D(\mathbf{C}^M, \tilde{\mathbf{C}}^A) < \gamma$. A larger γ allows more atomic clusters to be merged as a mega cluster. However, the cost of intra-cluster check may increase due to the increasing number of non-positively oriented triangles. In order to balance the computation cost between inter- and intra-cluster checks, we set $\gamma = 0.05$ (due to preliminary evaluation).

The observer primitive $o(\mathbf{C}^M)$ should be updated once an adjacent atomic cluster is merged. As many non-positively oriented triangles may occur when a mega cluster is associated with an observer line obtained from one of its consisted atomic clusters (see Fig. 9), we simply associate the mega cluster $\mathbf{C}^M$ with an observer point. That is, given that the bones of adjacent clusters must share a skeleton joint, we set the observer point to the joint position. Note that $o(\mathbf{C}^M)$ could be a line segment if $\mathbf{C}^M$ consists of only an atomic cluster. Also, the observer primitive is updated only when $\mathbf{C}^M$ consists of two atomic clusters and this primitive will be fixed when new atomic clusters are merged. This strategy prevents counting the number of non-positively oriented triangles frequently and thus reduces the expensive cost.

Implementation details. We store the indices of the triangles that are non-positively oriented with respect to the observer point $o(\mathbf{C}^M)$ at each time interval. These triangles are likely to be non-positively oriented at the next time interval based on the assumption that the overall shape change of each cluster is gentle. Therefore, at the next time interval, we first examine the orientations of these triangles and check if the merging cost exceeds the threshold γ. If so, the merging process could be terminated and the orientation examinations of the remaining triangles are not necessary.

6 Intra-Cluster Check, Inter-Cluster Check and Relative Axis-Aligned Bounding Box

We execute intra- and inter- cluster checks to collect the potentially colliding triangle pairs before the elementary test. Since there could be many triangle pairs not yet culled away, the BVH is used to cooperate with our framework. Specifically, we adopt the median splitting scheme to build a local BVH of each atomic cluster [Bridson et al. 2002]. We then gather the atomic clusters to build a global BVH such that the root of each local BVH is the leaf of a global BVH [Wong et al. 2013]. The global BVH is built at the processing stage. Below we describe the details.

Intra-Cluster Check. Within a cluster, collision occurs in two cases: 1) a positively oriented triangle collides with a negatively oriented triangle or 2) an uncertainly oriented triangle collides with an arbitrary triangle [Wong et al. 2013]. Therefore, for each cluster $\mathbf{C}^M$, we check 1) $\mathbf{H}^+(\mathbf{C}^M, o(\mathbf{C}^M))$ with $\mathbf{H}^-(\mathbf{C}^M, o(\mathbf{C}^M))$, and 2) $\mathbf{H}^u(\mathbf{C}^M, o(\mathbf{C}^M))$ with $\triangle\mathbf{C}^M$. We further improve the culling efficiency by computing respective swept bounding volumes to prevent unnecessary collision checks of triangles from different atomic clusters. BVH traversal is the final step that we use to detect continuous self-collision.

Inter-Cluster Check. For a mega cluster pair $\mathbf{C}_i^M$ and $\mathbf{C}_j^M$, BVH traversal is required for collecting the potentially colliding triangle pairs of the cluster pair. Given that BVH has to be built at preprocess but mega clusters are determined at run time, we check all the atomic cluster pairs from $\mathbf{C}_i^M$ and $\mathbf{C}_j^M$. Although the complexity of this comparison is $O(N^2)$, we found the cost is acceptable because the total number of atomic clusters is less than 200 in all our examples.

Relative Axis-Aligned Bounding Box. The feature pairs of each triangle pair collected from intra- and inter-cluster checks require elementary tests. Although the swept bounding volumes are used to prevent unnecessary checks, this approach considers absolute tri-

angle positions during the time interval. There are potentially colliding triangle pairs that do not collide because their relative velocities are small. We thus adopt relative axis aligned bounding boxes for quickly culling away such non-colliding triangle pairs. Let two triangles be $\mathbf{T}_1(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ and $\mathbf{T}_2(\mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6)$, and the vertex velocities are $\mathbf{v}_i$, $i = 1, 2, ..., 6$. We compute the average velocity $\bar{\mathbf{v}}$ and determine the vertex positions $\mathbf{p}'_i(t) = \mathbf{p}_i(0) + (\mathbf{v}_i - \bar{\mathbf{v}})t$, $\forall t \in [0, \Delta t]$. We then compute the swept axis aligned bounding boxes for triangles $\mathbf{T}'_1(\mathbf{p}'_1, \mathbf{p}'_2, \mathbf{p}'_3)$ and $\mathbf{T}'_2(\mathbf{p}'_4, \mathbf{p}'_5, \mathbf{p}'_6)$, and the triangles are free of collision if the boxes do not overlap. This strategy is applied to non-adjacent triangle pairs.

7 Experiments and Results

We have compared our algorithm to the state-of-the-art methods on several examples to demonstrate its effectiveness. All the reported statistics were measured on an Intel(R) Core(TM) i7 CPU 870 @ 2.93GHz with 4 GB RAM and one thread was used. Fig. 10 shows the snapshots of the examples. Our accompanying video shows the animations with continuous self-collision detection. Table 1 shows the model complexities, the timings of pre-computation, the number of atomic clusters, and the average size of non-positively triangles (i.e., $|H^-|$ and $|H^u|$) at runtime.

7.1 Culling Efficiency Comparison

We compare our dynamic clustering framework to the methods of bounding volume hierarchy [Klosowski et al. 1998], normal cone culling [Tang et al. 2009a], and radial view based culling [Wong et al. 2013]. For simplicity, these methods are denoted by *Base*, *MCD*, and *RVBC*, respectively. *Base* is a traditional BVH structure. Triangle pairs are prevented from elementary tests if their bounding volumes do not overlap. *MCD* is the integration of a normal cone-based method [Tang et al. 2009a] and deforming non-penetration filter [Tang et al. 2011b]. Triangle pairs are prevented from elementary tests using normal cone culling and BVH culling. *RVBC* is a radial view based method [Wong et al. 2013] that segments the deformable surface into clusters. Triangles in the same cluster and from different clusters are prevented from elementary tests by considering their orientations and by BVH culling.

To achieve a fair comparison, the type of bounding volume was 16 KDOPs ([Klosowski et al. 1998]) for all the methods. In the elementary test, the deforming non-penetration filters [Tang et al. 2011b] were employed to eliminate non-colliding feature pairs. All the computation was carried out in double precision. Since the type of bounding volume was the same in all the methods, the costs of BVH refit were similar. Specifically, the timings of BVH refit in some examples are listed below: rope (20.0ms), hand (5.0ms), octopus (21.1ms) and man (9.4ms). We also implemented BVH refit using the Intel Streaming SIMD Extensions (SSE) to further reduce the cost by around one-half.

Fig. 11 and Fig. 12 show that RVBC and our method reduce the number of bounding volume overlapping tests as well as the number of potentially colliding triangle pairs. This result is not surprising because only the two methods cull away triangles in the same cluster. Below we show the detailed comparison.

Comparison with RVBC. Table 1 shows the statistics of our method and RVBC. The pre-computational cost of our method is lower than that of RVBC due to the use of a lower resolution distance field when computing the heat diffusion. Besides, fewer non-positively triangles appear in our method because we select observer primitives based on the movement of the deformable objects. In the rope example, the slightly more negatively oriented triangles in our method is due to dynamic cluster merging. Overall, our performance is still much better than that of RVBC.

Comparison with Base and MCD. In the following, we denote by Z the average query timing and by Z_r the average timing for BVH refit. Table 2 shows the statistics of the bounding volume overlapping tests (BVOTs), the potentially colliding triangle pairs (PCTPs), as well as the speedup factors. Clearly, our method processes the fewest number of BVOTs. The numbers of our BVOTs compared to those of Base and MCD methods are at the ratios ranged among [0.07, 0.19] and [0.11, 0.30], respectively. In addition, our method also proceeds the fewest number of PCTPs. The numbers of our PCTPs compared to those of Base and MCD methods are at the ratios ranged among [0.02, 0.12] and [0.10, 0.36], respectively. As a result, our method outperforms Base and MCD with the speedup factors ranged among [15.5, 30.8] and [4.3, 14.0], respectively.

Table 3 shows the statistics collected at runtime. Our method significantly reduces the number of BVOTs and PCTPs. This table also shows the total cost of triangle classification (RV-Update), BVH traversal (BVH-TRA) and elementary test (ET). In our method, the cost of dynamic clustering is included in RV-Update. In the table, we list the average query timings ($Z - Z_r$) at the Total column (cost of BVH refit is not counted). As can be seen, the speedup factor of our method compared to RVBC is up to 5.2×. In the octopus example, the displacements of the tentacles are large in the animation. Because the observer primitives of RVBC are determined based on the reference pose, a large portion of triangles at the end parts are classified as uncertainly or negatively oriented in many time intervals. In contrast, our method dynamically updates the clusters and observer primitives (Fig. 9), and greatly reduces the requirements of elementary tests to boost the performance.

We also compared our method to RVBC using the objects with sharp geometry features: box (25088 triangles) and urchin (2840 triangles). In these two examples, the dynamic clustering scheme does not have advantages over RVBC because the atomic clusters are hardly merged. However, as relative axis-align bounding boxes (Section 6) are adopted for culling triangle pairs in our method, our method still outperforms RVBC $1.2\times$ and $1.3\times$ with the box and the urchin models, respectively. Compared with MCD, the speedups are $2.1\times$ in box and $4.0\times$ in urchin.

Comparison to axis-align bounding box and SelfCCD. We implemented the axis-align bounding box (AABB) method [van den Bergen 1999] and integrate the method with deforming non-penetration filters (DNPFs) [Tang et al. 2011b] to improve the process of elementary test. The advantage of AABB is fast at BVH refit but it usually collects a lot more PCTPs than that of KDOPs based methods. The ratios of our PCTPs compared to AABB are: rope (0.027), hand (0.023), octopus (0.017), and man (0.071). The speedup factors of our method over AABB for the total query time are: rope (11.5×), hand (9.4×), octopus (18.3×), and man (17.5×).

SelfCCD [UNC Gamma Group] is a famous continuous self-collision detection software that can handle general deformable surfaces. The speedup factors of our method over SelfCCD for the total query time are: rope (8.5×), hand (5.9×), octopus (10.9×), and man (13.5×).

7.2 Limitations and Future Work

Generally, continuous self-collision detection methods should be able to handle 3D meshes with boundaries. However, RVBC and our method are designed for closed deformable surfaces. In addition, because the mesh is segmented with each cluster approximately rigid, our framework requires the mesh to be attached with

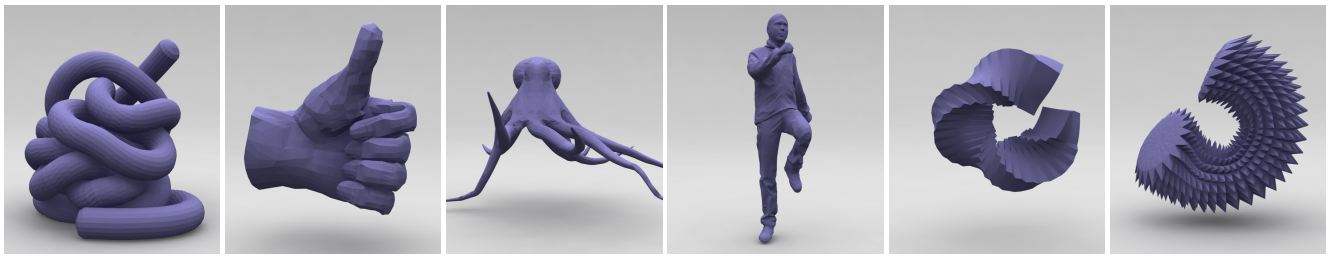

Figure 10: *The snapshots of the testing animations. From left to right are rope, hand, octopus, man, box and urchin.*

Table 1: *Model complexity and statistics of RVBC and our method. $|H^-|$ and $|H^u|$ are the average numbers of negatively and uncertainly oriented triangles, respectively. DfHd: Distance Field and Heat Diffusion; Decomp.: Cluster decomposition. The range of the number of mega clusters is stated in brackets.*

Example	Complexity		RVBC					Our Method					
	Tri	Vtx	DfHd (sec)	Decomp. (msec)	#Atomic Clusters	$\|H^-\|$	$\|H^u\|$	DfHd (sec)	Decomp. (msec)	#Atomic Clusters	$\|H^-\|$	$\|H^u\|$	#Avg. Mega Clusters
rope	48440	24222	21.5	30.0	100	3	83	6.8	28.4	199	95	53	38 [2, 76]
hand	11208	5606	4.3	6.0	11	195	13	1.1	5.6	20	193	13	12 [9, 15]
octopus	47976	23990	10.3	21.0	40	572	991	4.7	25.0	71	379	11	37 [32, 42]
man	20000	10002	6.2	8.0	16	1051	607	1.6	9.1	21	292	278	16 [11, 17]

Table 2: *Comparison with Base and MCD. Speedup factors and ratios for the numbers of (BVOTs) and PCTPs.*

Example	Ratios for BVOTs		Ratios for PCTPs		$Z - Z_r(msec)$	Speedup	
	$\frac{\text{our method}}{\text{Base}}$	$\frac{\text{our method}}{\text{MCD}}$	$\frac{\text{our method}}{\text{Base}}$	$\frac{\text{our method}}{\text{MCD}}$	our method	$\frac{\text{Base}}{\text{our method}}$	$\frac{\text{MCD}}{\text{our method}}$
rope	0.07	0.16	0.03	0.22	28.4	18.9×	6.0×
hand	0.10	0.19	0.02	0.20	3.77	18.0×	4.3×
octopus	0.07	0.11	0.02	0.09	17.2	30.8×	14.0×
man	0.17	0.27	0.10	0.31	16.2	15.5×	7.7×

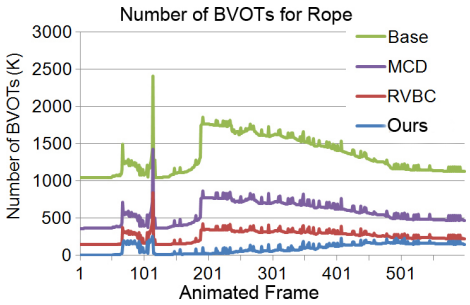

Figure 11: *Tracking the number of bounding volume overlapping tests for rope.*

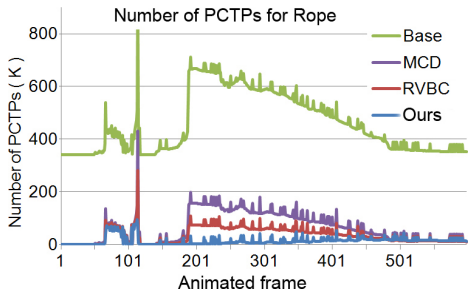

Figure 12: *Tracking the number of potentially colliding triangle pairs for rope.*

a skeleton. Fortunately, the latter problem could be easily solved because all types of the animation could be represented using a skinning mesh. In addition, the performance of our method may deteriorate for deformable meshes with folds, e.g. cloth. In such cases, an object has many high frequency features and many triangles may be negatively or uncertainly oriented, which results in collecting a large amount of potentially colliding triangle pairs. Given that the energy-based culling [Zheng and James 2012] can better handle this problem, we strive to combine the two approaches to detect collisions for models with high frequency features in the near future.

8 Conclusion

We present a dynamic clustering scheme to improve the radial view based culling method for continuous self-collision detection of closed deformable meshes. It segments the mesh into atomic clusters and dynamically merge them at runtime to reduce the number of potentially colliding triangle pairs. Compared to RVBC, our method effectively reduces the number of clusters and non-positively oriented triangles, thereby improving the overall culling efficiency. Our method also outperforms the BVH-based, normal cone-based, and AABB-based methods, and a famous software called SelfCCD significantly. Experiments and timing statistics demonstrate the effectiveness of our technique.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. The model of *man* was obtained from `http://people.csail.mit.edu/drdaniel/mesh_animation/`. This work was supported in part by the National Science Council Taiwan under contract number NSC 102-2221-E-009-103-MY2, 101-2628-E-009-021-MY3, 102-2221-E-009-082-MY3, 101-2628-E-009-020-MY3, and 102-2221-E-009-083-MY3

Table 3: *Statistics of BVOTs, PCTPs and timing. BVOTs: bounding volume overlapping tests; PCTPs: potentially colliding triangle pairs.*

Example	RVBC						Our Method						
	#BV (K)	#PCTP (K)	RV-Update (msec)	BVH-TRA (msec)	ET (msec)	Total (msec)	#BV (K)	#PCTP (K)	RV-Update (msec)	BVH-TRA (msec)	ET (msec)	Total (msec)	Speedup
rope	260.1	25.4	6.1	11.0	39.0	56.1	99.1	13.9	6.8	5.4	16.2	28.4	2.0×
hand	31.4	3.2	1.3	1.4	2.3	5.0	34.0	1.3	1.9	1.3	0.57	3.77	1.3×
octopus	319.6	63.0	6.6	12.5	70.8	89.9	137.2	11.1	6.8	6.9	3.5	17.2	5.2×
man	194.7	45.8	3.2	7.8	43.1	54.1	137.7	24.4	3.5	5.1	7.6	16.2	3.3×

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.

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.

BROCHU, T., EDWARDS, E., AND BRIDSON, R. 2012. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics 31*, 4, 96:1–96:7.

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.

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.

DU, P., TANG, M., AND TONG, R. 2012. Fast continuous collision culling with deforming noncollinear filters. *Computer Animation and Virtual Worlds 23*, 3-4, 375–383.

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.

JAMES, D., AND TWIGG, D. 2005. Skinning mesh animations. *ACM Transactions on Graphics 24*, 3, 399–407.

KAVAN, L., AND ZARA, J. 2005. Fast collision detection for skeletally deformable models. *Computer Graphics Forum 24*, 3, 363–372.

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.

PABST, S., KOCH, A., AND STRAßER, W. 2010. Fast and Scalable CPU/GPU Collision Detection for Rigid and Deformable Surfaces. *Computer Graphics Forum 29*, 5, 1605–1612.

SCHVARTZMAN, S., GASON, J., AND OTADUY, M. 2009. Bounded normal trees for reduced deformations of triangulated surfaces. In *Proceedings of the 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, 80:1–8.

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 Conf. 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 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, 111:1–15.

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

VAN DEN BERGEN, G. 1999. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics, GPU, and Game tools 2*, 4, 1–14.

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., LIN, W.-C., HUNG, C.-H., HUANG, Y.-J., AND LII, S.-Y. 2013. Radial view based culling for continuous self-collision detection of skeletal models. *ACM Transactions on Graphics 32*, 4, 114:1–114:10.

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 JAMES, D. 2012. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Transactions on Graphics 31*, 4, 98:1–12.