JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 1545-1559 (2011)

Adaptive Continuous Collision Detection for Cloth Models Using a Skipping Frame Session*

SAI-KEUNG WONG
Department of Computer Science
National Chiao Tung University
Hsinchu, 300 Taiwan

We propose a novel adaptive pipeline for continuous collision detection in cloth simulation. The pipeline consists of four components: bounding volume hierarchy (BVH) update, BVH traversal, a skipping frame session, and elementary test processing. The skipping frame session is activated adaptively for skipping both BVH update and BVH traversal during the process of collision detection. The proposed method improves the performance of collision detection in simulating cloth models comparing to some existing feature-based collision detection techniques, as indicated from experimental results.

Keywords: computer graphics, continuous collision detection, cloth models, deformable objects, feature-based technique

1. INTRODUCTION

Continuous collision detection computes accurate contact information, such as the time of contact, of colliding objects. It is widely applied in simulating deformable models, such as cloth models. The deformable models are discretized into triangular meshes at the preprocessing phase and each model is bounded by some kind of bounding volume hierarchies (BVHs). During the simulation, an interpolation approach is employed to interpolate the motion of models between two discrete time frames. For example, a cubic equation is solved for each feature pair if linear interpolation is adopted. There are fifteen feature pairs for each triangle pair, including six vertex-triangle pairs and nine edge-edge pairs. Both stages BVH update and BVH traversal are invoked in order to collect potentially colliding pairs in the conventional pipeline of collision detection. However, collision events happen in some localized regions for interacting models as pointed out by some researchers. By exploiting this coherent property, we develop an adaptive pipeline for continuous collision detection in the simulation of cloth models.

Summary of results: We propose a novel adaptive pipeline for continuous collision detection in cloth simulation. We summarize the major results as follows:

1. The pipeline consists of four components: BVH update, BVH traversal, a skipping frame session and elementary test processing. By employing the skipping frame session, both BVH update and BVH traversal stages can be skipped. The bounding volumes of BVH nodes are inflated based on both the local and global information of the

Received January 19, 2010; revised April 27 & June 22, 2010; accepted July 5, 2010.
Communicated by Tong-Yee Lee.
* This paper was partially supported by the National Science Council of Taiwan (No. NSC 97-2218-E-009-040). The content of the paper was presented in National Computer Symposium, Workshop on Image Processing, Computer Graphics, and Multimedia Technologies, Taiwan, 2009.

cloth models in order to employ the skipping frame session.
2. The partial traversal scheme: As the inflated bounding volumes are not tight and they may be kept for several frames, there would be redundant potentially colliding pairs. A partial traversal scheme is proposed to handle them. The partial traversal scheme is conservative, thereby detecting all the colliding pairs.
3. Adaptive self-collision detection: Adaptively perform self-collision detection based on the partitioning of low curvatured sub-surfaces.
4. Robustness: We adopt a history-based approach to keep track of all the feature pairs in proximity. The sidedness of the colliding pairs can be determined according to their history instead of relying on the unreliable geometry information.

The remainder of the paper is organized as follows. The related work and the overview of our algorithm are presented in sections 2 and 3, respectively. BVH update and BVH traversal of the proposed pipeline are discussed in section 4. The elementary test processing and self-collision detection are discussed in sections 5 and 6, respectively. Section 7 presents the skipping frame session and section 8 analyzes the proposed method. Experimental results are given in section 9. Finally, conclusion and future work are presented in section 10.

2. RELATED WORK

Collision detection was widely applied in cloth simulation [1-8]. Some approaches [1, 4, 9, 10] exploited the regularity properties of cloth models to perform self-collision detection. A comprehensive survey is described on collision detection for cloth and deformable models in [11]. There are spatial partitioning schemes and bounding volume hierarchies (BVHs) that have been developed in [12-16] to search for possible colliding pairs. Many non-colliding pairs are eliminated by using BVHs. Hence, the techniques are adopted to narrow down the number of potentially colliding pairs. The axis-aligned bounding box hierarchy (AABB) [15] and *k*-DOPs [12] are widely adopted. Larsson and Akenine-Moller [17] proposed a technique to perform BVH refitting for models deformed by morphing and a lazy evaluation method [18] to perform BVH update for breakable objects. Some methods employ extra bounding volumes to bound the vertices and edges of each triangle [19, 20].

Mezger *et al.* [16] suggested that the bounding volume of each node could be inflated by a predefined distance. If the enclosed primitives (*e.g.* triangles) do not move farther than a predefined distance, BVH update is not required for the current frame. However, it is crucial to compute the distance of inflation in order to apply their method but they did not specify a way to compute the distance of inflation automatically. It is well known that BVH update and BVH traversal are not necessary for a brute force approach. However, the running time complexity of the brute force approach is too high due to that there are many elementary tests. Similarly, if the inflation distance is not computed appropriately, the running time complexity of their method would be high.

Moore and Wilhelms [21] studied continuous collision detection for rigid body simulation. Later on, an efficient method for computing the time of contact in the simulation of cloth models was developed in [22]. Similar techniques were employed in [4, 6, 10]. Tang *et al.* [23] proposed a method for reducing the number of elementary tests

based on deforming non-penetration filters.

Wong and Baciu [24] proposed a feature assignment scheme to assign edges and vertices to incident triangles. The method reduces the number of potentially colliding feature pairs. The technique [20] integrated both the techniques [19, 24] to reduce the number of potentially colliding feature pairs further.

Selle *et al.* [8] proposed a method to handle complex cloth models. In some of their animations, the number of triangles of cloth models is more than one million. They suggested that the history based approach should be adopted so as to keep track of the relative orientation of interacting pairs in proximity. Moreover, it is important to control the strain ratio [3] in order to prevent cloth models from overstretching.

3. ALGORITHM OVERVIEW

Each cloth model is discretized into a triangular mesh. The topology of the meshes does not change. Furthermore, two features collide if their shortest distance is smaller than or equal to a predefined threshold δ_d which is larger than or equal to the thickness of cloth. If the bounding volumes of two triangles overlap, the two triangles form a potentially colliding pair. The simulation time step is Δt.

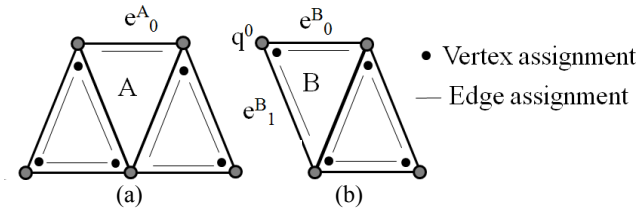


Fig. 1. The feature assignment for two triangles. The small dot (line) inside a triangle at a vertex (edge) indicates that the vertex (edge) is assigned to the triangle.

We employ the feature assignment scheme [25] to assign each feature of meshes to its incident triangle: a triangle assigned to itself and a vertex or edge assigned to one of its incident triangles. The information of assignment is stored in each triangle as a feature assignment mask. The mask of a triangle indicates the vertices or edges assigned to the triangle. For example, the feature assignment of two meshes is illustrated in Fig. 1. The assigned feature pairs of two potentially colliding triangles are computed based on the mask as follows. Let $F_{vertex}(T)$ and $F_{edge}(T)$ denote the two sets of assigned vertices and assigned edges to a triangle T, respectively. Consider two triangles A and B. Then the set of vertex-triangle pairs is $\{F_{vertex}(A) \times \{B\}\} \cup \{F_{vertex}(B) \times \{A\}\}$. Similarly, the set of edge-edge pairs is $\{F_{edge}(A) \times F_{edge}(B)\}$. For example, assume that the edge e^A_0 is assigned to A, and two edges e^B_0, e^B_1, and the vertex q_0 are assigned to B. Hence, the set of the assigned feature pairs contains q_0A, $e^A_0\, e^B_0$, and $e^A_0\, e^B_1$ for the triangles A and B. The assigned feature pairs are checked instead of the fifteen feature pairs.

There is a skipping frame session at the runtime phase of the adaptive pipeline (see Algorithm 1). During the skipping frame session, both BVH update and BVH traversal are skipped. The number of frames that the skipping frame session lasts is n_f. In the first frame of the skipping frame session, *flagSkippingFrame* is set as false. Otherwise, it is set as true.

Algorithm 1 Algorithm
1: **if flagSkippingFrame then**
2: collect dangling vertices
3: collect dangling triangles
4: perform traversal for dangling triangles
5: **else**
6: perform BVH update
7: perform BVH traversal
8: **end if**
9: perform front-end filtering for potentially colliding pairs
10: perform back-end filtering for potentially colliding pairs

If *flagSkippingFrame* is false, a full BVH update and a full BVH traversal are performed. The size of the bounding volume is extended adaptively at the stage of BVH update according to the motion state of the objects. BVH traversal is then performed for collecting potentially colliding triangle pairs and these pairs are stored in a pending list. If *flagSkippingFrame* is true, we would collect the vertices and triangles moving farther from their estimated movement distance. We call the vertices *dangling vertices* and the triangles *dangling triangles*. A partial BVH traversal scheme is applied for processing the dangling triangles.

We proceed to perform the elementary test processing for the potentially colliding triangle pairs.

4. BVH UPDATE AND BVH TRAVERSAL

In our approach, each object has one BVH. In BVH update, the task is to refit the bounding volume of each node so that the bounding volume bounds the swept volume of the triangles assigned to the node. The bounding volumes of leaf nodes are updated first and they are extended by the thickness of cloth. The bounding volume of each internal node is then computed by merging the bounding volumes of its children. The process is performed recursively until the root node is updated.

4.1 BVH Update with Inflation Distance

In the following we assume that each leaf node contains one triangle. The bounding volume of each leaf node is inflated with a certain inflation distance for performing the skipping frame session. Assume that the leaf node is associated with a triangle T. Initially the bounding volume of each node is computed for bounding the swept volume of its triangle and then the bounding volume is inflated with an inflation distance. The inflation distance is the estimated movement distance $d_e(T)$ of the triangle which is equal to the maximum estimated movement distance of its three vertices. Assume that $v(P)$ is the speed of P. The estimated movement distance of P is $v(P)c_t(P)$ in the current simulation time interval, where $c_t(P)$ is the contact time of P. However, $c_t(P)$ is unknown before performing collision detection. Since $c_t(P) \le \Delta t$, it implies that $v(P)c_t(P) \le v\Delta t$. The estimated movement distance could be therefore computed as $v(P)\Delta t$ for one time step. However, we attempt to skip more than one frame in the skipping frame session. Instead

of estimating the movement distance of triangles for one time step, we need to estimate their movement distance for several frames. The movement of a vertex is affected by its neighborhood as it is a part of a continuum material (*e.g.* cloth model). Moreover, the strain ratio should be smaller than certain percentages (*e.g.* less than 15%) in simulating cloth models, as reported previously in [4, 6]. Thus, we propose to adaptively estimate $d_e(P)$ as $d_e(P) = (\alpha v(P) + \beta v')\Delta t + \gamma l'$, where v' is the average speed of the vertices in the neighborhood of P, and l' is the average edge length of the cloth model. The three values α, β and γ are adaptively adjusted. The value of α depends on the number of skipping frames. There are other terms that can be included in computing $d_e(P)$, such as acceleration. In this paper, our focus is on the linear terms. To compute v', we have to know the connectivity of the cloth model and compute the average for its neighborhood. But the cost is expensive by doing so. Moreover, it would be unreliable to estimate the speed of a vertex based on its neighborhood over a number of frames due to uncertainty in a simulation environment. Hence v' is approximated as the average speed of all the vertices of the cloth model. The information is presented in section 7 for computing α, β and γ.

4.2 BVH Traversal

Potentially colliding triangle pairs are collected during BVH traversal and they are stored in a hash table. In BVH traversal, the root nodes of the two BVHs are checked first. If they overlap, then their children are checked. This is done recursively until the leaf nodes are reached. Then the corresponding two triangles of the two leaf nodes are checked further. After two triangles are detected, they are determined whether or not they should be registered. The set of the assigned feature pairs are computed if the two triangles are assigned vertices or edges. If the set of assigned feature pairs is empty, then the two triangles are ignored. Otherwise, the triangle pair is registered as an entry in a hash table [26]. In order to quickly retrieve the entries from the hash table, the entry pointers are stored in a pending list.

We maintain a list L_{prev} for all the active slots of the hash table in the previous frame and another list L_{cur} for all the active slots in the current frame. Each element in the L_{prev} and L_{cur} is an index of a slot. In the current frame, if a slot is visited, the slot index is removed from the L_{prev} and it is added to another list L_{cur}. This can be implemented efficiently by using a doubly linked list.

After the potentially colliding pairs are collected, we remove all the hash entries for each slot of the hash table in L_{prev}. We then visit all the hash entries for each element in L_{cur} and remove all the old hash entries. After that we set L_{prev} as L_{cur} and then clear L_{cur}.

There would be many potentially colliding pairs but they are too far to collide within the simulation time interval due to the inflation of bounding volumes. The distance heuristic (section 6) is employed for eliminating these pairs. In order to employ the distance heuristic, the shortest distance between two triangles should be computed. It is not necessary to use all the fifteen feature pairs of two triangles for computing the shortest distance as some feature pairs would be checked for multiple times. Thus, we consider only the assigned feature pairs. By doing so, each feature pair is checked once. The shortest distance $d(T_0,T_1)$ is computed based on the assigned feature pairs. The triangle pair is registered if $d(T_0, T_1) \leq d_e(T_0) + d_e(T_1) + \delta_d$. The triangle pair which is far away would be therefore eliminated.

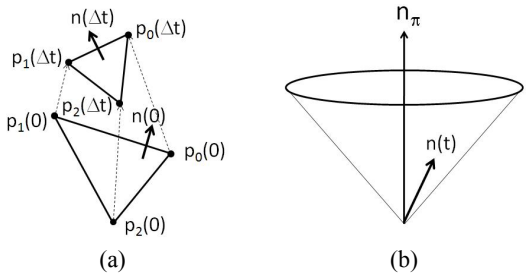
4.3 Processing Dangling Triangles

The bounding volumes are updated at the beginning of the skipping frame session. They are kept unchanged until the end of the skipping frame session. There are dangling triangles that probably move beyond their inflated bounding volumes. However, they are not registered in the hash table during the BVH traversal. These dangling triangles should be handled as they would intersect with each other.

At the beginning of a skipping frame session, the maximum distance of each vertex P is computed as $d_e(P) = (\alpha v(P) + \beta v')\Delta t + \gamma'$. In the remaining frames of the session, $d_e(P)$ is updated as $d_e(P) - v(P)\Delta t$. If $d_e(P) \leq \delta_d$, then the vertex is marked as a dangling vertex. After that we have to collect all the dangling vertices and the dangling triangles. If the number of dangling triangles is small, we perform BVH traversal for each dangling triangle individually. On the other hand, if there are many dangling triangles, we partially update the BVH in a bottom-up manner [18]. Firstly update the BVs of the dangling triangles and secondly climb up the BVH to the root and keep merging the BVs of internal nodes and mark the nodes. Finally, we perform BVH traversal for the marked nodes for collecting potentially colliding triangle pairs.

5. SELF-COLLISION DETECTION

Collecting the potentially colliding pairs of a mesh itself can be performed [1, 4] by partitioning a cloth model into a set of low curvatured sub-surfaces and then performing the hierarchy traversal for each pair of low curvatured sub-surfaces. A triangle belonging to a low curvatured sub-surface satisfies the condition $n(t) \cdot n_\pi > 0$ within the time interval $[0, \Delta t]$, where $n(t)$ is the normal of the triangle at time t and n_π is the representative normal of the sub-surface. In the following discussion, we assume that the sub-surfaces of two sibling nodes are connected.



(a) (b)

Fig. 2. (a) The changes of the normal vector $n(t)$ of a triangle within the time interval; (b) A canonical cone of a triangle.

There are two stages to partition a cloth model into a set of low curvatured sub-surfaces. We compute the canonical cone by using the method in [9, 10] for each triangle in the first stage. A canonical cone with a representative normal vector n_π bounds the continuous normal vector $n(t)$ of a triangle within the time interval $[0, \Delta t]$, as illustrated in Fig. 2. The triangle is itself a low curvatured subsurface if the canonical cone exists. We

traverse up the BVH for checking each internal node to merge canonical cones of its children in the second stage. If the children are connected and the merging process [4] is successful, then the triangles at the leaf nodes rooted at that internal node form a low curvature sub-surface. Merging two cones is performed by computing a larger cone enclosing them. If the canonical cone of a node does not exist, the merging process is stopped for the node. In this case, we obtain some low curvatured sub-surfaces associated with the child nodes. The maximum cone angle should be less than 90 degrees. We set it to 70 degrees for avoiding drastic deformation within one sub-surface.

Collision detection check is performed between each pair of the sub-surfaces. The collision check between two sub-surfaces is performed as an inter-collision check. A low curvatured sub-surface could not intersect itself unless the contour of the surface has self-intersection. To check whether the contour has collisions, the method in [10] should be adopted to perform collision detection for the line segments of the contour.

6. ELEMENTARY TEST PROCESSING AND DISTANCE HEURISTIC

This section presents elementary test processing which consists of two sub-phases: front-end filtering and back-end filtering phase. In the front-end filtering phase, we employ the distance heuristic to eliminate non-colliding pairs in the pending list. The idea of the distance heuristic is presented as follows. Let $d^j{}_e(O_A, O_B)$ be the estimated distance between two objects O_A and O_B at frame j. The estimated maximum displacement of the two objects are $d^{j+1}{}_e(O_A)$ and $d^{j+1}{}_e(O_B)$ in the next frame, respectively. If $d^j{}_e(O_A, O_B) > d^{j+1}{}_e(O_A) + d^{j+1}{}_e(O_B) + \delta_d$, the two objects cannot collide at the current frame. The estimated distance $d^{j+1}{}_e(O_A, O_B)$ is then updated as $d^j{}_e(O_A, O_B) - (d^{j+1}{}_e(O_A) + d^{j+1}{}_e(O_B))$.

In our case, the two objects are two triangles (T_0, T_1) for every pair in the pending list. The maximum displacement of a triangle T is $v(T)\Delta t$, where $v(T)$ is the speed of T. If $d^j{}_e(T_0, T_1) \le d^{j+1}{}_e(T_0) + d^{j+1}{}_e(T_1) + \delta_d$, the potentially colliding pair is added to the admissible list. As $d^j{}_e(T_0, T_1)$ is always less than or equal to the actual distance $d(T_0, T_1)$, the proposed method is conservative. After all pairs in the pending list are filtered, we proceed to the phase of back-end filtering for handling pairs in the admissible list.

Continuous collision detection [4, 6] is performed for each triangle pair in the admissible list in the phase of back-end filtering. The assigned feature pairs of each triangle pair are considered. Linear interpolation is adopted for computing the motion path of each vertex in continuous collision detection. Higher order interpolation schemes are possible but higher cost of computation is required. After the phase of back-end filtering, all the colliding point-triangle and edge-edge pairs are detected.

7. THE SKIPPING FRAME SESSION

A skipping frame session consists of n_f frames. At the first frame of the skipping frame session, both BVH update and BVH traversal are performed. For the remaining $(n_f - 1)$ frames, both BVH update and BVH traversal are skipped. In order to improve the performance, n_f should be computed adaptively.

Recall that the maximum movement distance $d_e(P)$ of a vertex P is estimated as $(\alpha v(P) + \beta v')\Delta t + \gamma l'$ (see section 4.1). During the simulation, we keep track of the CPU time

spent on collision detection and adaptively adjust α, β and γ. If the average time is getting better, we increase n_f by one and at the same time change α, β and γ if necessary. In the remaining $(n_f - 1)$ frames, the expected movement distance of P is $(n_f - 1)v(P)\Delta t$. However, P is affected by its neighboring vertices and the length of edges also affects the speed of a vertex due to the strain rate of a cloth model. The expected movement distance of a vertex P is therefore adjusted to $d_e(P) = ((n_f - 1)v(P) + \beta v')\Delta t + \gamma l'$ in the remaining frames. The value α is set as $(n_f - 1)$ consequently. The two values β and γ are the weights for the average velocity of vertices and length of edges of the cloth model in computing $d_e(P)$ of the vertex P, respectively. In our experiments, there is wind drag affecting the motion of cloth models and the motion of cloth models is unpredictable. Thus, instead of computing these two values adaptively based on the simulation time step, they are assigned constant values, $\beta = 0.2$ and $\gamma = 0.1$, in our experiments. On the other hand, if the simulation is known beforehand, a better way for estimating the speed of a vertex could be possible, such as considering the local region of the vertex.

If the CPU time spent on collision detection is getting worse, n_f should be decreased and the current skipping frame session should be terminated. If n_f is changed to one, the skipping frame session would be disabled for a while before a new skipping frame session begins. Sometimes, it is necessary to set n_f as one in order to obtain the CPU time spent on collision detection without the skipping frame session. We would know whether or not the performance of the skipping frame session is reasonable at the moment.

When self-collision detection is performed, each cloth model is partitioned in the first frame of the skipping frame session. In the remaining frames, the continuous canonical cone is computed for each triangle per time step. We identify the triangles that violate the low curvature property of their current assigned low curvature sub-surfaces. These triangles may lead to self-collision events. They are handled individually to check whether or not they collide with the other parts of the cloth model. There would be a few such kind of triangles if the objects do not deform rapidly.

8. ANALYSIS AND DISCUSSION

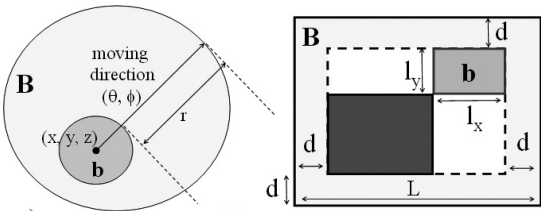
We rely on the speed of vertices of the cloth models to estimate the expected movement distance of vertices. The expected movement distance would affect the number of skipping frames. Increasing the probability of penetration free movement for the models is crucial for achieving high performance. We show that the higher the resolution of cloth models the higher the probability of penetration free movement is. Let **b** be a small bounding region and **B** a large bounding region (see Fig. 3). Both of them are convex. Assume that **b** is randomly allocated inside **B** without overlapping the boundary of **B**. We want to compute: (1) the expected free movement distance of **b** and (2) the probability that the boundaries of **b** and **B** do not overlap if **b** moves in an arbitrary direction.

8.1 Expected Free Movement Distance

The expected free movement distance of **b** inside **B** is the average free movement distance of **b** without overlapping the boundary of **B**. It is given by:

$$\int \cdots \int_{\{D,\Omega\}} p(\theta,\phi,x,y,z) r(\theta,\phi,x,y,z) dx dy dz d\theta d\phi \tag{1}$$

where (θ, ϕ) is the movement direction of **b** in the spherical coordinate system, (x, y, z) is the location of the reference point of **b**, D is the spatial domain, Ω is the set of possible movement directions, $p(\theta, \phi, x, y, z)$ is the probability density function that **b** moving in direction (θ, ϕ), and $r(\theta, \phi, x, y, z)$ is the maximum distance that **b** moving in the direction (θ, ϕ) at (x, y, z) without overlapping the boundary of **B**.



(a) (b)

Fig. 3. Movement without collision; (a) A bounding volume **b** moves inside a region **B**. The region is not necessarily to be a disk. It can be other shapes; (b) An AABB **b** moves inside another AABB **B** with a distance of d in an arbitrary direction. **b** would not collide with the boundary of **B** if the vertex of **b** at the lower left corner lying inside the dark region.

Consider the case in the one-dimensional space. Both **b** and **B** are bounding intervals which are lying horizontally. Let L be the length of **B** and l ($\leq L$) the length of **b**. In this case, **b** can only move horizontally either to the left side or right side. Let x be the location of the right hand side of **b**. The expected free movement distance is computed as:

$$\int_l^L \frac{1}{L-l}\frac{1}{2}(L-x)dx + \int_l^L \frac{1}{L-l}\frac{1}{2}(x-l)dx = \frac{L-l}{2}. \tag{2}$$

The smaller l, the longer the expected free movement distance is. Similarly in the three-dimension space, the smaller the size of **b**, the longer the expected free movement distance is.

8.2 Probability of Penetration Free Movement

Assume that **b** moves with a distance d in an arbitrary direction. The probability that the boundaries of **b** and **B** do not overlap is given by:

$$\int \cdots \int_{\{D,\Omega\}} p(\theta,\phi,x,y,z) c(\theta,\phi,x,y,z) dx dy dz d\theta d\phi \tag{3}$$

where c is a characteristic function which is given by:

$c = 0$, if $r(\theta, \phi, x, y, z) \leq d + \delta_d$,
$c = 1$, otherwise.

Consider a simple example with a weak condition that **b** moves with distance d in an arbitrary direction. In the three-dimensional space, assume that both **B** and **b** are AABBs. In the next frame, the probability that **b** still lies inside **B** is $\Pi_j(L_j - l_j - 2d - 2\delta_d)/L_j$, where j indicates a coordinate axis. The smaller the size of **b**, the higher the probability is. In other words, the probability for a triangle not becoming a dangling triangle is higher if the size of **b** is smaller. Assume that the speed of **b** is v in each direction and the time step is Δt. Then $d = v\Delta t$. It implies that the probability of penetration free movement is higher if the time step Δt is smaller. A higher resolution of cloth models requires smaller Δt so as to satisfy the Courant condition [3].

9. EXPERIMENTS

We performed a Monte Carlo simulation to collect the expected free movement distance and probability of penetration free movement. The small bounding volume is randomly generated inside the large bounding volume. The number of tests is one million. Both bounding volumes are cubes. The ratio of side length k equals to the side length of the small bounding volume divided by the side length of the large bounding volume. The movement distance ratio of a vertex per time step is defined as the maximum movement distance of the vertex divided by the side length of the small bounding volume. The expected free movement distance is almost linear to the ratio of side length as shown in Fig. 4. Moreover, the probability of penetration free movement is almost one when the movement distance ratio is small. That is the small bounding volume has a small chance to move beyond the large bounding volume for small movement distance ratio. The higher the ratio of side length the higher the probability is. Consider that the small bounding volume is a unit cube. Assume that the time step is 0.01 (sec) and the movement distance of a vertex is 0.15. Then the movement distance per time step is 0.0015 which is much smaller than 0.05. The probability of penetration free movement is almost one in this case.

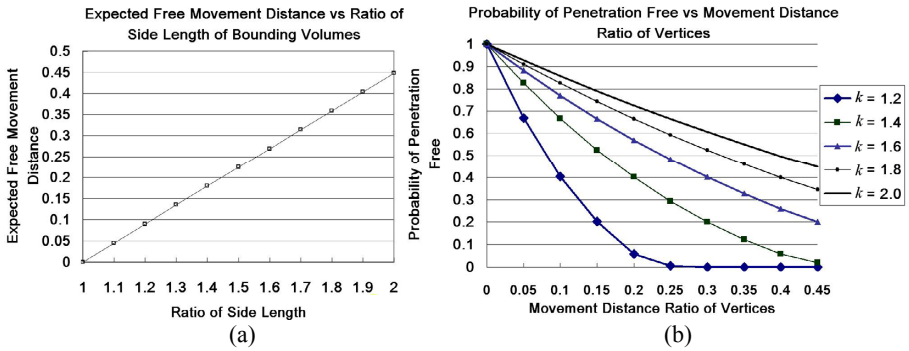


Fig. 4. (a) Expected free movement distance; (b) Probability of penetration free movement.

Furthermore, we performed two sets of experiments for collecting the performance characteristics of the proposed method. In each set, there were four animations. Table 1

shows the model complexities. The complexity of cloth models is up to tens of thousands of triangles in Experiment Set One while the complexity of cloth models is up to hundreds of thousands of triangles in Experiment Set Two. The experiments were all performed on an Intel(R) Core(TM2) Quadcore CPU machine with 2.4GHz of 2GB memory and one thread was employed to perform the computation. The predefined threshold δ_d was set as 0.015 for all the experiments. The hash table had 200031 slots for all experiments. We compare our methods with two methods proposed by [24] and [20], and other methods at the end of this section. We denote the method by [24] as NoDup and the method by [20] as R-TRI. Our methods are labeled as nSwD and SwD. In both nSwD and SwD, distance heuristic is employed but there is no skipping frame session in nSwD. NoDup relies on the feature assignment scheme to perform continuous collision detection for the feature pairs. R-TRI employs the improved feature assignment scheme and bounds each feature (vertex or edge) with an extra bounding volume.

Table 1. The model complexities.

Experiment Set One: Model Complexities		
	Rigid Objects (#Tri)	Cloth Models (#Tri)
Ani. One	5.2 k	97 k
Ani. Two	10 k	45 k
Ani. Three	0.5 k	97 k
Ani. Four	34 k	20 k
Experiment Set Two: Model Complexities		
Ani. One	11 k	320 k
Ani. Two	7 k	500 k
Ani. Three	1 k	502 k
Ani. Four	40 k	500 k

Table 2. Performance statistics.

Experiment Set One: Average Collision Detection Time Per Time Step (sec) (including self-collision detection)				
	NoDup	R-TRI	nSwD	SwD
Ani. One Spinning ball	0.23	0.18	0.15	0.13
Ani. Two Ball-cloth	0.12	0.10	0.082	0.074
Ani. Three Four cones	0.31	0.27	0.23	0.16
Ani. Four Garment	0.092	0.076	0.062	0.052

9.1 Experiment Set One

In Experiment Set One, the cloth models were affected by a wind drag model in Animations Two, Three and Four. Fig. 5 shows the snapshots. In Animation One, a cloth model interacted with a spinning bumpy ball. In Animation Two, a cloth model interacted with a ball. In Animation Three, a cloth model interacted with four rigid cones. There were many collision events due to the large bounding volumes of the cones. In Animation Four, a garment interacted with a mannequin. Table 2 shows the performance statistics of Experiment Set One. Compared with NoDup, nSwD and SwD outperform it by up to around 50% and 100%, respectively. Compared with R-TRI, nSwD and SwD outperform it by up to around 20% and 70%, respectively.

9.2 Experiment Set Two

In Experiment Set Two, there were four animations. Fig. 6 shows the snapshots of Experiment Set Two. The motion of cloth models was not changing drastically. The timing information was collected for different number of frames in the skipping frame session. We denote αxfy with the skipping frame session enabled, where x is the value of α and y is the value of n_f. We compare our methods with NoDup. In Animation One, the

cloth model consisted of 320k triangles. The initial time step was 5 ms and it was dynamically adjusted during the simulation [5]. On average, our method took 660 ms and NoDup took 830 ms to detect all the colliding pairs including self-collision events.

In Animations Two, Three and Four, each cloth model consisted of a half million triangles. The ridges of the underneath objects are shown clearly. In Animation Two, there was a deformable volumetric model. The timing information is shown in Fig. 7 without including the timing in self-collision detection. The numbers at the top of each bar indicate the speedup factors. The results show that by employing the skipping frame

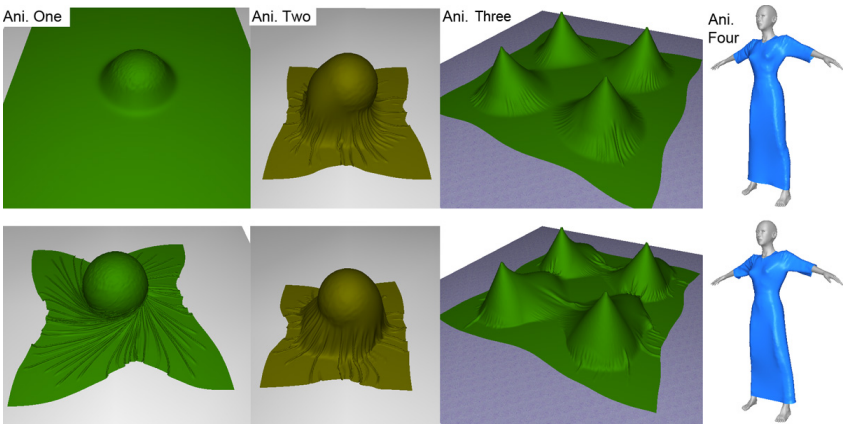


Fig. 5. Snapshots of experiment set one.

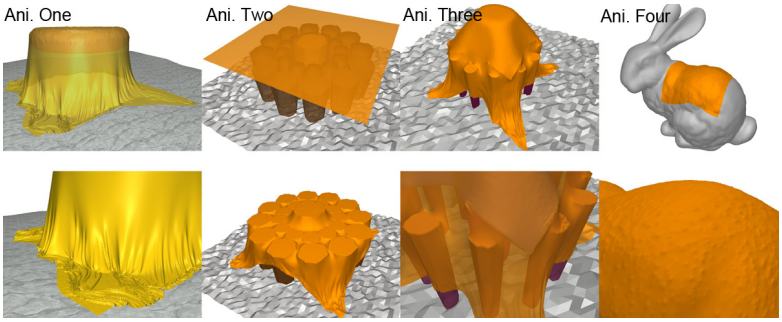


Fig. 6. Snapshots of experiment set two.

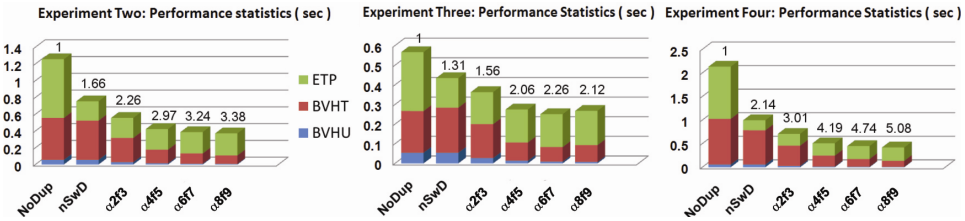


Fig. 7. Experiment set two: performance statistics of animation two, three and four. BVHU: BVH update. BVHT: BVH traversal. ETP: Elementary test processing. The numbers at the top of each bar indicate the speedup factors.

session, the speedup factor of our method is in the range from two to five.

9.3 Comparison with Other Methods

The spinning ball benchmark was performed in several papers. On average, our method took 130ms to detect both inter- and self-collision events for the cloth model consisting of 97k triangles. There were many folds and wrinkles on the cloth model in our animation. In [27], it took 246ms for the cloth model consisting of around 92k triangles on a 2.66 GHz Intel Pentium machine with 2GB RAM using a single thread. In [10], it took 290ms on a machine with similar settings.

Tang *et al.* [23] proposed a method for filtering non-colliding feature pairs so as to avoid solving high order polynomial equations. It took 144ms on a 2.4 GHz machine with 4GB RAM to perform the spinning ball benchmark. Its performance is quite similar to ours. However, our method aims at skipping BVH update and BVH traversal. Hence, their method and our method are complementary to each other.

The required memory is mainly used for storing the potentially colliding triangle pairs in our method. The memory size is proportional to the number of potentially colliding triangle pairs. For example, in the spinning ball benchmark, the average number of potentially colliding triangle pairs was 142k and the memory size was 48M. We not only kept information of each potentially colliding pair for performing collision detection but also information for performing dynamics computation.

10. CONCLUSION AND FUTURE WORK

We have proposed a novel adaptive approach to perform continuous collision detection for cloth models using a skipping frame session. Our approach combines the feature assignment scheme and the distance heuristic. Both inter- and self-collision detection are supported. The skipping frame session is activated adaptively to accelerate the process of collision detection. Even though there are external forces acting on the cloth models, our method still outperforms some existing efficient feature-based techniques. There are two limitations in our method. First, the movement distance of the vertices of the cloth models should be small compared to the size of bounding volumes of other objects in order to employ the skipping frame session. However, our experiment results show that when a wind drag model with moderate strength, the skipping frame session can still be employed. If the external forces are too strong, the skipping frame session could be disabled. The proposed method also performs efficiently by employing the distance heuristic alone. Second, all colliding pairs and potentially colliding pairs in close proximity are hashed as the proposed method is a history-based method. The memory space is quite demanding. On the other hand, as suggested in [8], tracking pairs in close proximity is necessary in order to reliably compute the relative orientation of colliding pairs. In the future, we will investigate methods to minimize the storage size.

ACKNOWLEDGEMENTS

We thanked the reviewers for their helpful and insightful comments.

REFERENCES

1. P. Volino and N. Magnenat-Thalmann, "Efficient self-collision detection on smoothly discretised surface animation using geometrical shape regularity," *Computer Graphics Forum*, Vol. 13, 1994, pp. 155-166.
2. M. Courchesne, P. Volino, and N. Magnenat-Thalmann, "Versatile and efficient techniques for simulating cloth and other deformable objects," in *Proceedings of ACM SIGGRAPH*, 1995, pp. 137-144.
3. X. Provot, "Deformation constraints in a mass-spring model to describe rigid cloth behaviour," *Graphics Interface*, 1995, pp. 147-154.
4. X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments," *Computer Animation and Simulation*, 1997, pp. 177-189.
5. D. E. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of SIGGRAPH*, 1998, pp. 43-54.
6. R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Transactions on Graphics*, Vol. 21, 2002, pp. 594-603.
7. K. J. Choi and H. S. Ko, "Stable but responsive cloth," in *Proceedings of SIGGRAPH*, 2004, pp. 604-611.
8. A. Selle, J. Su, G. Irving, and R. Fedkiw, "Robust high-resolution cloth using parallelism history-based collisions and accurate friction," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, 2009, pp. 339-350.
9. S. K. Wong and G. Baciu, "Dynamic interaction between deformable surfaces and nonsmooth objects," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 11, 2005, pp. 329-340.
10. M. Tang, S. Curtis, S. E. Yoon, and D. Manocha, "ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, 2009, pp. 544-557.
11. M. Teschner, *et al.*, "Collision detection for deformable objects," in *Eurographics State-of-the-Art Report*, 2004, pp. 119-139.
12. A. Smith, Y. Kitamura, H. Takemura, and F. Kishino, "A simple and efficient method for accurate collision detection among deformable polyhedral objects in arbitrary motion," in *Proceedings of Virtual Reality Annual International Symposium*, 1995, pp. 136-145.
13. S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proceedings of SIGGRAPH*, 1996, pp. 171-180.
14. J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of *k*-DOPs," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, 1998, pp. 21-36.
15. G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *Journal of Graphics Tools*, Vol. 2, 1999, pp. 1-14.
16. J. Mezger, S. Kimmerle, and O. Etzmuss, "Hierarchical techniques in collision detection for cloth animation," *Journal of WSCG*, Vol. 11, 2003, pp. 322-329.
17. T. Larsson and T. Akenine-Moller, "Efficient collision detection for models deformed by morphing," *Visual Computer*, Vol. 19, 2003, pp. 164-174.
18. T. Larsson and T. Akenine-Moller, "A dynamic bounding volume hierarchy for gen-

eralized collision detection," *Computer and Graphics*, Vol. 30, 2006, pp. 451-460.
19. M. Hutter and A. Fuhrmann, "Optimizied continuous collision detection for deformable triangle meshes," *Journal of WSCG*, 2007, pp. 25-32.
20. S. Curtis, R. Tamstorf, and D. Manocha, "Fast collision detection for deformable models using representative-triangles," in *Proceedings of Symposium on Interactive 3D Graphics and Games*, 2008, pp. 61-69.
21. M. Moore and J. P. Wilhelms, "Collision detection and response for computer animation," *Computer Graphics*, Vol. 22, 1988, pp. 289-298.
22. J. D. Liu, M. T. Ko, and R. C. Chang, "Collision avoidance in cloth animation," *Visual Computer*, Vol. 12, 1996, pp. 234-243.
23. M. Tang, D. Manocha, and R. Tong, "Fast continuous collision detection using deforming non-penetration filters," in *Proceedings of SIGGRAPH*, 2010, pp. 7-13.
24. S. K. Wong and G. Baciu, "A randomized marking scheme for continuous collision detection in simulation of deformable surfaces," in *Proceedings of ACM International Conference on Virtual Reality Continuum and Its Applications*, 2006, pp. 181-188.
25. S. K. Wong and G. Baciu, "Robust continuous collision detection for interactive deformable surfaces," *Computer Animation and Virtual Worlds*, Vol. 18, 2007, pp. 179.
26. D. E. Knuth, *Sorting and Searching: The Art of Computer Programming*, 2nd ed., Vol. 3, Addison-Wesley, Massachusetts, 1997.
27. M. Tang, S. E. Yoon, and D. Manocha, "Adjacency-based culling for continuous collision detection," *Visual Computer*, Vol. 24, 2008, pp. 545-553.

Sai-Keung Wong (黃世強) received his Ph.D. and M.S. degrees in Computer Science from the Hong Kong University of Science and Technology (HKUST) in 2005 and 1999, respectively. From June 2005 to January 2008, he was a postdoctoral fellow of the Department of Computing of the Hong Kong Polytechnic University. He has been an Assistant Professor of the Department of Computer Science of the National Chiao Tung University, Taiwan, since 2008. His research interests include computer animation, collision detection, 3D game engines and visualization.