Adaptive Continuous Collision Detection for Cloth Models using a Skipping Frame Session

Sai-Keung Wong
Department of Computer Science, National Chiao Tung University, Taiwan, ROC
Email: cswingo@cs.nctu.edu.tw

***Abstract*—We propose a novel adaptive pipeline for continuous collision detection (APCCD) in simulating cloth models. The proposed pipeline consists of four components: bounding volume hierarchy (BVH) update, BVH traversal, a skipping frame session, and elementary test processing. It supports both inter- and self-collision detection. A skipping frame session is activated adaptively for skipping both BVH update and BVH traversal. Our method tracks all interacting primitive pairs. Experimental results show that the proposed method significantly improves the performance of collision detection in simulating cloth models.**

***Index Terms*—Computer graphics, continuous collision detection, self-collision detection, cloth**

I. Introduction

Continuous collision detection (CCD) attracts much attention due to that the accurate contact information can be computed. CCD is widely applied in simulating deformable models, such as cloth models. The deformable models are discretized into triangular meshes at the preprocessing phase. 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 deformable models between two discrete time frames. By employing CCD, the first time of contact of the interacting objects can be computed. In the conventional collision detection pipeline, both stages BVH update and BVH traversal are invoked in order to collect potentially colliding pairs. We observe that when two models are interacting with each other, collision events happen in some localized regions and these regions may interact with each other over several frames. A full BVH update and a full BVH traversal are not necessary. In this paper, we propose an approach to handle continuous collision detection for cloth models adaptively by skipping BVH update and BVH traversal.

Summary of results: We propose a novel adaptive pipeline for continuous collision detection (APCCD) in the simulation of cloth models (Figure 1). The results are listed as follows:

1) The framework of APCCD: It 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. In order to employ the skipping frame session, the bounding volumes of BVH nodes are inflated based on both the local and global information of the cloth models.
2) The partial traversal scheme: As the inflated bounding volumes are not tight and they may be kept for several frames, there will be many redundant potentially colliding pairs. Instead of performing a full BVH traversal to eliminate the redundant pairs, a partial traversal scheme is proposed to handle them. The partial traversal scheme is conservative. Thus, we will not miss any colliding primitive pairs.
3) The distance heuristic: Based on the primitive assignment, we use the assigned primitives to compute the shortest distance of two potentially colliding triangles. After that, we update the estimated distance between the triangle pairs each time step to determine whether a full CCD check should be invoked for them.
4) Robustness: We adopt a history-based approach to keep track of the relative orientation of all the primitive pairs in close proximity. These primitive pairs are maintained in a hash

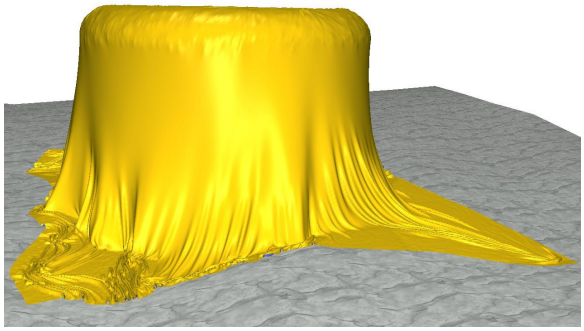

Fig. 1. The cloth model consists of 320k triangles. Our CCD method took 660 ms to detect all the colliding pairs and kept track all of them throughout the simulation.

table throughout the simulation.

The remainder of the paper is organized as follows: Section II presents the related work. Section III presents the framework of APCCD. Section IV presents BVH update with inflation distance and BVH traversal. The elementary test processing and distance heuristic are discussed in Section VI. Self-collision detection is presented in V. Section VII presents the skipping frame session and Section VIII analyzes the proposed method. Section IX presents the experimental results. Finally, the conclusion and future work are presented in Section X.

II. Related Work

Collision detection is widely applied in the simulation of deformable models, for example, cloth simulation [23], [5], [16], [17], [1], [3], [4], [18]. During the simulation, the topology of the deformable models does not change. The contact information, such as contact points and collision normal vectors should be computed. Some approaches [23], [17], [24], [20] exploited the regularity properties of cloth models to perform self-collision detection. A comprehensive survey on collision detection for deformable models can be found in [22].

A brute force approach for collision detection is to perform tests for all primitive pairs. The running time complexity is $O(n^2)$, where n is the number of primitives. The worst case happens when all the triangles are packed closely. However, this hardly happens in practice. In reality, the triangles will not be packed together. There are spatial partitioning schemes and bounding volume hierarchies (BVHs) that have been developed in [19], [7], [9], [2], [14]. The techniques are adopted to narrow down the number of potentially colliding pairs. The axis-aligned bounding box hierarchy (AABB) [2] and k-DOPs [9] are widely applied in simulating deformable models. Larsson and Akenine-Moller [11] proposed a technique to perform BVH refitting for models deformed by morphing and a lazy evaluation method [12] to perform BVH update for breakable objects. Some methods employ extra BVs to bound the vertices and edges of each triangle [8], [6]. Thus, during BVH update, the extra BVs should be updated.

Mezger et al. [14] suggested that the bounding volume could be inflated by a predefined distance. If the enclosed primitives do not move farther than a predefined distance, BVH update is not needed. 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 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. In this paper, we develop an adaptive scheme to effectively compute the inflated distance. Moreover, our method also skips BVH traversal.

Continuous collision detection is dominated for simulating deformable models as accurate time of contact can be obtained. Continuous collision detection was studied by Moore and Wilhelms [15]. Liu et al. [13] proposed a technique so that a cubic equation is required to solve for each primitive pair. The coplanar time is computed and then the shortest distance of the primitive pair is checked for collision. For each triangle pair, there are six point-triangle pairs and nine edge-edge pairs. Similar techniques are employed in [17], [3].

Wong and Baciu [25] proposed a primitive assignment scheme to assign edges and vertices to incident triangles. The method significantly reduces the number of potentially colliding triangle pairs and the number of potentially colliding primitive pairs. The technique [6] integrated both the tech-

Algorithm 1 APCCD 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 PCP record filtering
10: perform back-end PCP record filtering
```

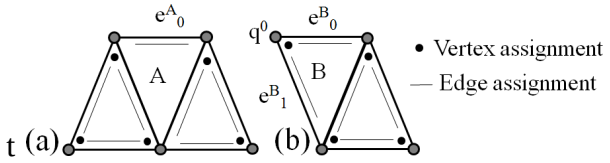

Fig. 2. Primitive assignment scheme and shortest distance between two triangles. Primitive assignment scheme: The small dot (line) inside a triangle at a vertex (edge) indicates that the vertex (edge) is assigned to the triangle. The shortest distance $d(A,B)$ between the two triangles A and B is the smallest value of $d(e_0^A, e_0^B)$, $d(e_0^A, e_1^B)$, and $d(A, q_0)$. where $d(\cdot,\cdot)$ is the shortest distance between two primitives. The assigned primitives are considered to compute $d(A,B)$.

niques [8], [25] to reduce the number of potentially colliding primitive pairs further.

Selle et al. [18] 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 to keep track the relative orientation of interacting pairs in proximity. Relying solely on the voting mechanism is not reliable [5] to determine collision orientation of colliding primitive pairs. Moreover, it is important to control the strain ratio [16] when simulating cloth models.

III. Algorithm overview

We assume that the topology of the cloth models does not change and the simulation time step is Δt. Two primitives 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 (PCP). Before the simulation is performed, we employ the primitive assignment scheme [26] to assign each primitive to its incident triangle: a triangle assigned to itself and a vertex or edge assigned to one of its incident triangles. Each triangle record stores a six-bit assignment mask for the primitive assignment. The assignment mask of the triangle indicates the vertices or edges assigned to the triangle. Figure 2 shows the primitive assignment of two meshes. Based on the assignment mask, the potentially colliding primitive pairs of two triangles can be computed. For example, the potentially colliding primitive pairs are $e_0^A e_0^B$, $e_0^A e_1^B$, and $q_0 A$ for the triangles A and B.

Algorithm 1 shows the runtime phase of APCCD. The flow chart and the related data structures are shown in Fig. 3. At each simulation time step, the speed of each vertex and the speed of each triangle are computed. The speed of a triangle is the maximum speed of its three vertices.

At the runtime phase, there is a skipping frame session. 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 and the value n_f is determined adaptively. In the first frame of the skipping frame session, $flagSkippingFrame$ is set as false. Otherwise, it is set as true.

If $flagSkippingFrame$ is false, we perform a full BVH update and a full BVH traversal. In BVH update, the size of the bounding volume is extended adaptively according to the moving state of the objects. After that, BVH traversal is performed to gather potentially colliding triangle pairs (PCTPs) and the PCTPs are stored in the PCP pending list.

If $flagSkippingFrame$ is true, we will collect the vertices and triangles which move 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 elementary test processing which consists of two sub-phases: front-end PCP filtering and back-end PCP filtering. In the front-end PCP filtering phase, non-colliding pairs will be further eliminated based on the distance heuristic. The PCTPs in the PCP pending list are inserted into the admissible PCP list if they pass the check of the distance heuristic. The PCTPs in the

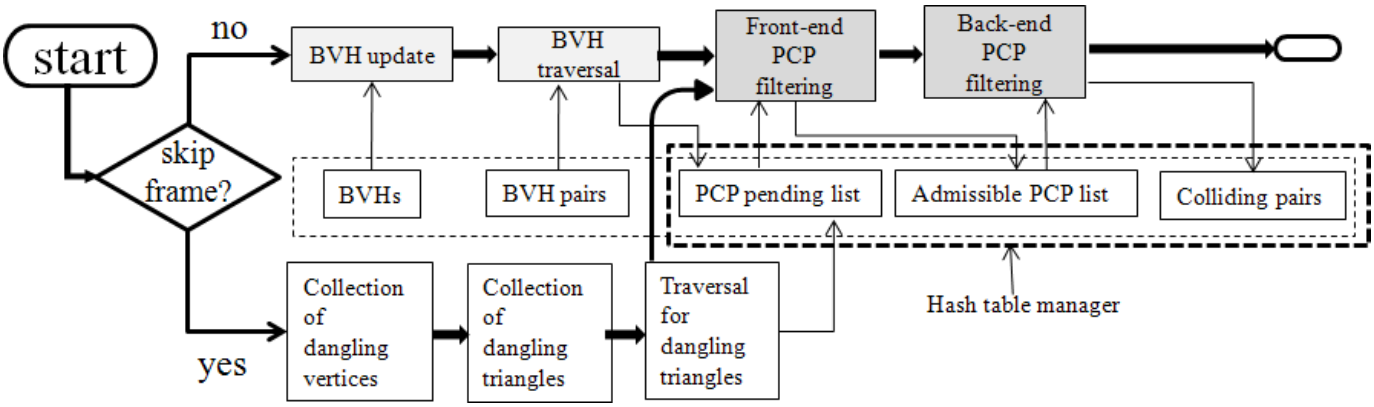

Fig. 3. Collision detection pipeline.

admissible PCP list will be passed to the back-end PCP filtering phase and their contact information is computed. Our method guarantees to detect all PCTPs whose shortest distance is smaller than or equal to δ_d.

IV. BVH UPDATE AND BVH TRAVERSAL

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

A. BVH update with inflation distance

In the following we assume that each leaf node contains one triangle. It is straightforward to handle a leaf node that contains more than one triangle. In our method, we will inflate the BV of each leaf node in order to perform the skipping frame session. Assume that the leaf node is associated with a triangle T. The inflation distance is the estimated movement distance $d_e(T)$ of the triangle which is the maximum estimated movement distance of its three vertices. We compute the estimated movement distance $d_e(P)$ of a vertex P as follows.

Assume that $v(P)$ is the speed of P. Then 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 collision detection is performed. As $c_t(P) \leq \Delta t$, it implies that $v(P)c_t(P) \leq v\Delta t$. Thus the estimated movement distance could be computed as $v(P)\Delta t$ for one time step. The inflation distance $v(P)\Delta t$ is adopted in the conventional BVH refitting methods. However, in our approach, we attempt to skip more than one frame in the skipping frame session. We observe that the movement of a vertex is greatly affected by its neighborhood as it is a part of a continuum material (e.g. cloth model). In simulating cloth models, the strain ratio should be smaller than certain percentages (less than 15%) [17], [3]. Thus, we propose to adaptively estimate $d_e(P)$ as $d_e(P) = (\alpha v(P) + \beta\bar{v})\Delta t + \gamma l$, where $\bar{v}$ 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. 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 $\bar{v}$, we have to know the connectivity of the cloth model and compute the average for its neighborhood. In this way, the cost is expensive. In order to reduce the computation cost, $\bar{v}$ is approximated as the average speed of all the vertices of the cloth model. The crucial part is to compute α, β and γ. The detailed information is presented in Section VII.

B. BVH traversal with PCP registration

The task of BVH traversal is to collect potentially colliding triangle pairs. Algorithm 2 shows the pseudocode of BVH traversal. The root nodes of the two BVHs are explored. If they overlap, then their children are checked. This is done recursively

Algorithm 2 traverse($node_0$, $node_1$)

```
1: if (BV(node0) ∩ BV(node1) = nil ) return
2: if node0 != leaf || node1 != leaf then
3:     if num(node0) > num(node1) then
4:         traverse(c0, node1) ∀ child c0 ∈ node0
5:     else
6:         traverse(node0, c1) ∀ child c1 ∈ node1
7:     end if
8: else
9:     compute pcpMask for (tri(node0), tri(node1))
10:    if pcpMask not nil then
11:        registerPCP(tri(node0), tri(node1))
12:    end if
13: end if
```

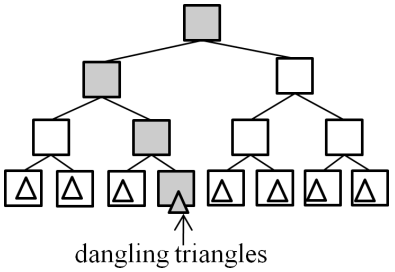

Fig. 4. Partial BVH update. Update the BVH starting from leaf nodes of the dangling triangles, and then merge BVs while climb up the BVH until reaching the root node.

until the two leaf nodes are detected. Then the two corresponding triangles are checked further.

If the two triangles are assigned vertices or edges, the potentially colliding primitive mask, namely, *pcpMask*, is computed. The mask *pcpMask* indicates which primitive pairs are required for further processing in the back-end filtering phase. The mask *pcpMask* is a fifteen-bit string and each bit of it corresponds to one of the fifteen primitive pairs of the two triangles. The primitive pairs include six point-triangle pairs and nine edge-edge pairs. The bit of *pcpMask* associated with two primitives (m_0, m_1) is set if the primitives m_0 and m_1 are assigned to the first triangle and the second triangle, respectively. If none of the bits are set in *pcpMask*, then the two triangles are ignored. Otherwise, the triangle pair is stored as a PCP entry in a hash table [10] via a PCP registration process. A PCP manager is used to manage the hash table and PCP entries. In order to quickly retrieve the PCP entries, their entry pointers are stored in a PCP pending list.

We observe that by enlarging the size of the AABBs, there are many potentially colliding pairs but they are too far to collide within the simulation time interval. In order to eliminate these pairs, we employ the distance heuristic (Sec. VI). In order to employ the distance heuristic, the shortest distance between two triangles should be computed. Conventionally all primitive pairs of two triangles are required to compute in order to obtain the shortest distance of two triangles. However, in our method, we only consider the assigned primitives. After a triangle pair (T_0, T_1) is collected during BVH traversal, the shortest distance $d(T_0, T_1)$ is computed between the two triangles based on *pcpMask* (see Fig. 2). If $d(T_0, T_1) \leq d_e(T_0) + d_e(T_1) + \delta_d$, the triangle pair is registered. In this way, the triangle pair which is far away will be eliminated.

C. Processing dangling triangles

As the skipping frame session may last for several frames, there are dangling triangles that move beyond their inflated bounding volumes. We have to handle them in order to avoid penetration.

At the beginning of a skipping frame session, the maximum distance of each vertex P is computed as $d_e(P) = (\alpha v(P) + \beta \bar{v})\Delta t + \gamma \bar{l}$. 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 [12]. 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 (see Figure 4). Finally, we perform BVH traversal for the marked nodes to collect PCTPs. We should avoid storing duplicated copies of the PCTPs in the hash table. In our simulation, we would like to keep the number of dangling triangles as small as possible. If it is not, the cost in traversal for dangling triangles will be expensive.

V. Self-collision detection

In order to handle self-collision events for a cloth model, we partition it into a set of low curvature sub-surfaces. Each triangle will be assigned to a low curvature sub-surface. A triangle belonging to a low curvature 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.

There are two stages to partition a cloth model into a set of low curvature sub-surfaces. In the first stage, we compute the continuous canonical cone [24] for each triangle. If the continuous canonical cone exists, then the triangle is itself a low curvature sub-surface. In the second stage, we traverse up the BVH to check for each internal node to merge canonical cones of its children. If the merging process is successful, then the triangles at the leaf nodes rooted at that node form a low curvature sub-surface. Then, we keep traversing up the BVH and repeat the process. 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 curvature sub-surfaces associated with the child nodes.

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. There is no collision check for the triangles belonging to the same sub-surfaces. Collision events may happen at the boundary of a low curvature surface. In this case, the method in [20] should be adopted for checking triangles at the boundary.

By employing the method to partition a cloth model, it is required that the sub-surfaces of two sibling nodes should be connected. To avoid the problem that the sub-surfaces not connected, we can perform region glowing to identify the connected components of a cloth model in the preprocessing stage. After that the BVH of the cloth model is constructed so that the sub-surfaces of two sibling nodes are connected.

VI. Elementary test processing and distance heuristic

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

Now in our case for handling a PCTP in the PCP pending list, the two objects are two triangles (T_0, T_1). The maximum displacement of a triangle T is $v(T)\Delta t$, where $v(T)$ is the speed of T. If $d_e^i(T_0, T_1) \leq d_e^{i+1}(T_0) + d_e^{i+1}(T_1) + \delta_d$, the PCTP is added to the admissible PCP list. As $d_e^i(T_0, T_1)$ is always less than or equal to the actual distance $d(T_0, T_1)$, the proposed method is conservative. After we have checked all the PCTP in the PCP pending list, we proceed to the phase back-end PCP filtering.

In the phase of back-end PCP filtering, we perform continuous collision detection for each pair in the admissible PCP list. We process the point-triangle pairs and the edge-edges pairs that are encoded in the *pcpMask*. The relative orientation of all pairs is also updated [18]. After the phase of back-end PCP filtering, all the colliding point-triangle and edge-edge pairs are detected.

VII. 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. The idea is given as follows.

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 and at the same time change α, β and γ when 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. So, finally the expected movement distance of a vertex P in the remaining frames is adjusted to $d_e(P) = ((n_f-1)v(P)+\beta\bar{v})\Delta t + \gamma\bar{l}$. Thus, α is set as (n_f-1). 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. 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 based on the simulation time step, they are assigned constant values. Experimental results showed that the performance of the skipping frame session is reasonable when $\beta = 0.2$ and $\gamma = 0.1$.

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 will 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 know the CPU time spent on collision detection without the skipping frame session. In this way, we will 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 will 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.

VIII. Analysis and Discussion

In our framework of APCCD, we rely on the speed of vertices of the cloth models to estimate expected movement distance of vertices. In this section, 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. 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.

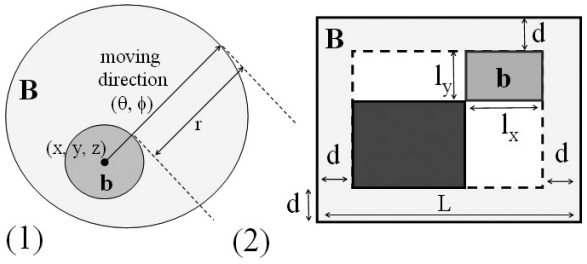

Fig. 5. Movement without collision. (1): A bounding volume **b** moves inside a region **B**. The region is not necessarily to be a disk. It can be other shapes. (2): An AABB **b** moves inside another AABB **B** with a distance of d in an arbitrary direction. **b** will not collide with the boundary of **B** if the vertex of **b** at the lower left corner lying inside the dark region.

A. 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:

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

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**.

Consider the case in the one-dimensional space. Then, 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**. Then the expected free movement distance is computed as:

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

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.

B. 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:

$$\int \cdots \int_{\{D,\Omega\}} p(\theta,\phi,x,y,z)c(\theta,\phi,x,y,z)dxdydzd\theta d\phi, \quad (3)$$

where c is a characteristic function which is given by:

$$\begin{aligned} c &= 0, \; if \quad r(\theta,\phi,x,y,z) \leq d+\delta_d \\ c &= 1, \; otherwise \end{aligned}$$

Consider a simple example with a weak condition that **b** moves with distance d in all directions. 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 $\prod_j \frac{L_j - l_j - 2d - 2\delta_d}{L_j}$, where j indicates a coordinate axis. The smaller the size of **b**, the higher the probability is. 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 will be higher if the time step Δt is smaller. A higher resolution of cloth models requires smaller Δt so as to satisfy the Courant condition [16]. The analysis is applicable to inter-collision detection.

IX. Experiments

In order to understand the performance characteristic of our methods, we performed two set of experiments. In each set, there were four animations. Table I shows the model complexities. In Experiment Set One, the complexity of cloth models is up to tens of thousands of triangles while in Experiment Set Two, the complexity of cloth models is up to hundreds of thousands of triangles. 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. We compare our methods with two methods proposed by [25] and [6], and other methods at the end of this section. We denote the method by [25] as NoDup and the method by [6] as R-TRI. Our methods are labeled as nSwD and SwD. In nSwD, there is no skipping frame session but in SwD, the skipping frame session is enabled. NoDup relies on the primitive assignment scheme to perform CCD for the primitive pairs. R-TRI employs the improved primitive assignment scheme and bounds each primitive (vertex or edge) with an extra BV.

TABLE I
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

A. Experiment Set One

In Experiment Set One, the cloth models were affected by a wind drag model in Animation Two, Three and Four. Fig. 6 shows the snapshots. In Animation One, a cloth mode interacted with a spinning bumpy ball and there were many self-collision events. In Animation Two, a cloth model interacted with a ball. The potentially colliding pairs changed drastically. 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. Finally, in Animation Four, a garment interacted with a mannequin.

Table II shows the performance statistics of Experiment Set One. Compared with NoDup, nSwD

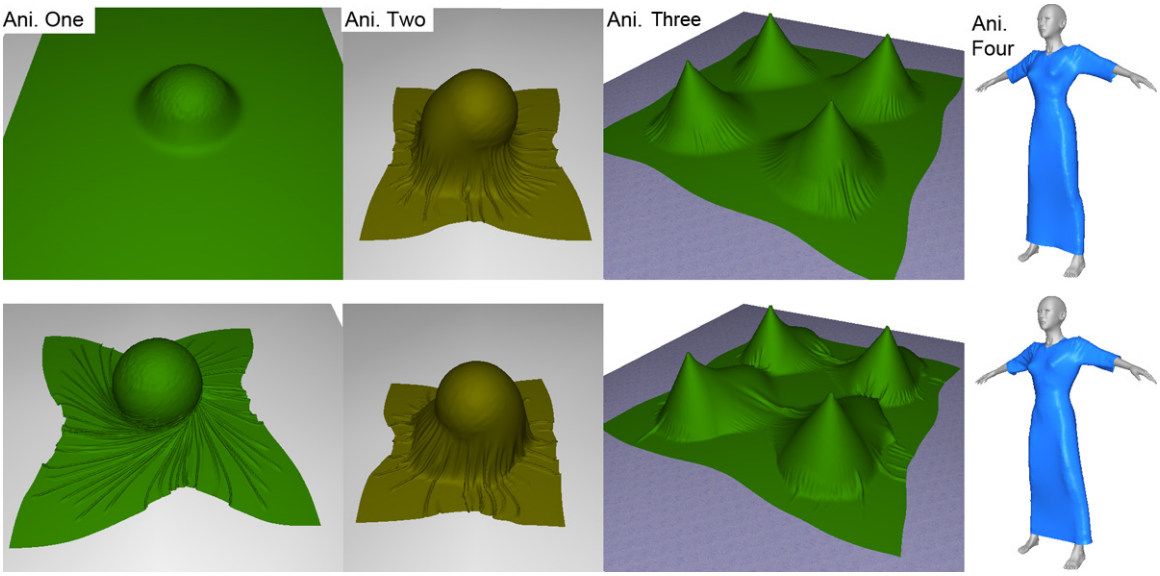

Fig. 6. Experiment Set One: Snapshots.

TABLE II
EXPERIMENT SET ONE: 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

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.

B. Experiment Set Two

In Experiment Set Two, there were four animations. Fig. 8 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 $axfy$ with the skipping frame session enabled, where x is the value of n_f and y is the value of α. 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 [1]. On average, our method took 660 ms and NoDup took 830 ms to detect all the colliding pairs including self-collision events near the end of the simulation.

In Animation Two, Three and Four, each cloth model consisted of a half million triangles. The ridges of the underneath objects are clearly shown. In Animation Two, there was a deformable volumetric model. The timing information is shown in Figure 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 session, the speedup factor of our method is in the range from two to five.

C. Comparison with other methods

The spinning ball benchmark was performed in several papers. On average, our method took 130 ms to detect both inter- and self-collision events for the cloth model consisting of 97 k triangles. There were many folds and wrinkles on the cloth model in our animation. In [21], it took 246 ms for the cloth model consisting of around 92 k triangles on a 2.66 GHz Intel Pentium machine with 2GB RAM using a single thread. In [20], it took 290 ms on a

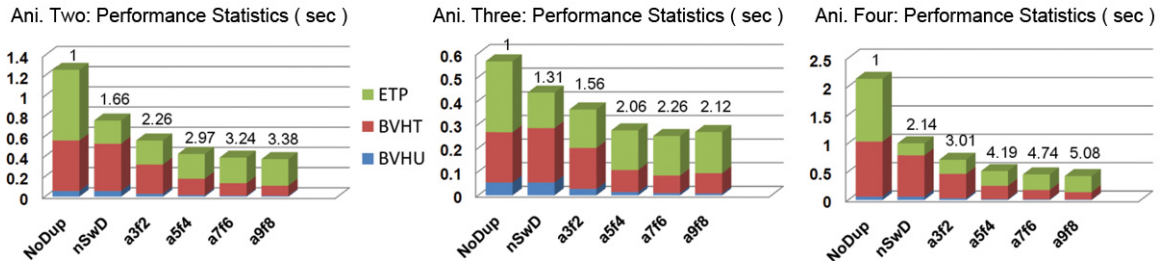

Fig. 7. Experiment Set Two: Performance Statistics of Animation Two, Three and Four. BVHU: BVH update. BVHT: BVH Traversal. ETP: Elementary test processing. The numbers at the top of each bar indicate the speedup factors.

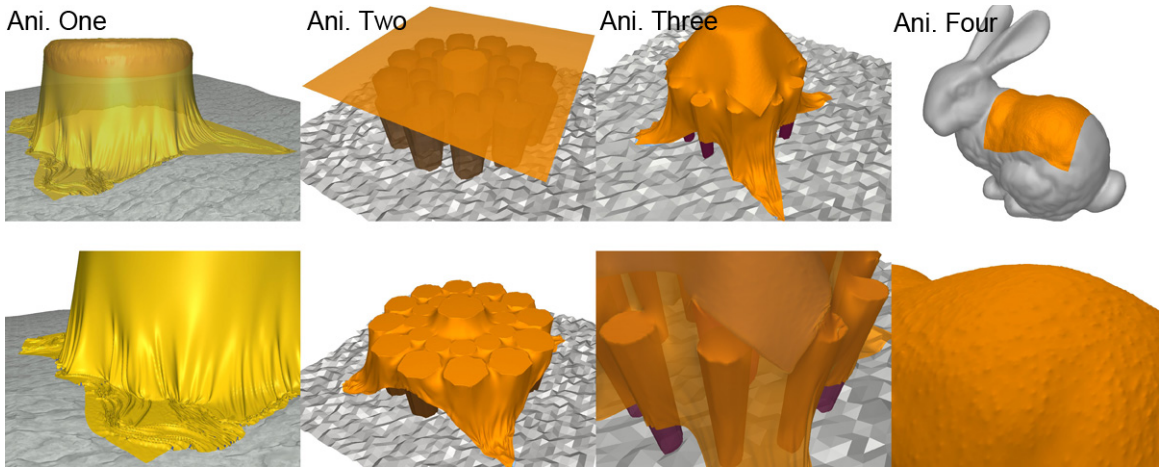

Fig. 8. Experiment Set Two: Snapshots.

machine with similar settings.

As our method is a history-based method, the required memory is mainly used for storing the potentially colliding triangle pairs. The memory size is propotional to the number of potentially colliding triangle pairs. For example, in the spinning ball benchmark, the average number of potentially colliding triangle pairs was 142 k and the memory size was 48 M.

X. Conclusion and Future work

We proposed a novel history-based approach to perform continuous collision detection for cloth models using a skipping frame session. Our approach combines the primitive 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 techniques.

There are two limitations in our method. First, in order to employ the skipping frame session, the movement distance of the vertices of the cloth models should be small compared to the size of bounding volumes of other objects. 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 can be disabled. By employing the distance heuristic alone, our method also performs efficiently. Second, as our method is a history-based method, all colliding pairs and potentially colliding pairs in close proximity are hashed, the memory space is quite demanding. On the other hand, as suggested in [18], tracking pairs in close proximity is necessary in order to reliably compute the relative orientation of colliding pairs.

Our future work is to adaptively estimate the speed of vertices and apply the skipping frame session to multilayered garments incrementally. As our method stores all the potentially colliding pairs,

the storage size will be large for models of high complexity. We are investigating methods to minimize the storage size.

Acknowledgements: We would like to thank the reviewers for their constructive comments. This research is supported by the National Science Council of Taiwan (No. NSC 97-2218-E-009-040).

REFERENCES

[1] D.-E. Baraff and A. Witkin, "Large Steps in Cloth Simulation", *Computer Graphics (SIGGRAPH'98)*, pp. 43–54, 1998.

[2] G. van den Bergen, "Efficient Collision Detection of Complex Deformable Models using AABB Trees", *Journal of Graphics Tools*, vol. 2, no. 4, pp. 1–14, 1999.

[3] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation", *ACM ToG*, vol. 21, no. 3, pp. 594–603, 2002.

[4] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth", in *ACM SIGGRAPH*, 2004, pp. 604–611.

[5] M. Courchesne, P. Volino, and N. Magnenat-Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects", in *SIGGRAPH*, 1995, pp. 137–144.

[6] S. Curtis, R. Tamstorf, and D. Manocha, "Fast collision detection for deformable models using representative-triangles", in *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, 2008, pp. 61–69.

[7] S. Gottschalk, M.-C. Lin, and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection", in *ACM SIGGRAPH*, 1996, pp. 171–180.

[8] M. Hutter and A. Fuhrmann, "Optimizied continuous collision detection for deformable triangle meshes", in *WSCG*, 2007, pp. 25–32.

[9] J.-T. Klosowski, M. Held, S.-B.-J. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-DOPs", *IEEE Trans. on Vis. and Comp. Graphics*, vol. 4, no. 1, pp. 21–36, Jan. 1998.

[10] D.-E. Knuth, *Sorting and Searching: The Art of Computer Programming*, 2nd ed. Addison-Wesley, 1997, vol. 3.

[11] T. Larsson and T. Akenine-Moller, "Efficient collision detection for models deformed by morphing", *Visual computer*, vol. 19, no. 2, pp. 164–174, May 2003.

[12] T. Larsson and T. Akenine-Moller, "A dynamic bounding volume hierarchy for generalized collision detection", *Computer and Graphics*, vol. 30, pp. 451–460, 2006.

[13] J.-D. Liu, M.-T. Ko, and R.-C. Chang, "Collision Avoidance in Cloth Animation", *Visual Computer*, vol. 12, no. 5, pp. 234–243, 1996.

[14] J. Mezger, S. Kimmerle, and O. Etzmuss, "Hierarchical techniques in collision detection for cloth animation", *Journal of WSCG*, vol. 11, no. 1, pp. 322–329, 2003.

[15] M. Moore and J.-P. Wilhelms, "Collision detection and response for computer animation", *Computer Graph.*, vol. 22, no. 4, pp. 289–298, 1988.

[16] X. Provot, "Deformation constraints in a mass-spring model to describe rigid cloth behaviour", in *Graph. Interface*, 1995, pp. 147–154.

[17] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments", in *Computer Animation and Simulation*, 1997, pp. 177–189.

[18] A. Selle, J. Su, G. Irving, and R. Fedkiw, "Robust High-Resolution Cloth Using Parallelism History-Based Collisions and Accurate Friction", *IEEE TVCG*, 2008.

[19] 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 *Virtual Reality Annual International Symposium*, 1995, pp. 136–145.

[20] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "ICCD: Interactive Continuous Collision Detection between Deformable Models Using Connectivity-Based Culling", *IEEE TVCG*, vol. 15, no. 4, pp. 544–557, 2009.

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

[22] M. Teschner, S. Kimmerle, G. Zachmann, B. Hei-delberger, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnetat-Thalmann, and W. Strasser, "Collision detection for deformable objects", in *Eurographics State-of-the-Art Report*, 2005.

[23] P. Volino and N. Magnenat-Thalmann, "Efficient self-collision detection on smoothly discretised surface animation using geometrical shape regularity", *Computer Graph. Forum*, vol. 13, no. 3, pp. 155–166, 1994.

[24] S.-K. Wong and G. Baciu, "Dynamic interaction between deformable surfaces and nonsmooth objects", *IEEE TVCG*, vol. 11, no. 3, pp. 329–340, 2005.

[25] S.-K. Wong and G. Baciu, "A Randomized Marking Scheme for Continuous Collision Detection in Simulation of Deformable Surfaces", in *ACM International Conference on Virtual Reality Continuum and Its Applications*, 2006, pp. 181–188.

[26] S.-K. Wong and G. Baciu, "Robust Continuous Collision Detection for Interactive Deformable Surfaces", *Computer Animation and Virtual Worlds*, vol. 18, no. 3, p. 179, 2007.