Pergamon

Comput. & Graphics, Vol. 22, No. 1, pp. 117–128, 1998

PII: S0097-8493(97)00087-3
Printed in Great Britain
0097-8493/98 $19.00 + 0.00

Technical Section

A SIMPLE SELF-COLLISION AVOIDANCE FOR CLOTH ANIMATION

JEN-DUO LIU[1], MING-TAT KO[2]† and RUEI-CHUAN CHANG[2]

[1]Institute of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China
[2]Institute of Information Science, Academia Sinica, Taipei, Taiwan
e-mail: mtko@iis.sinica.edu.tw

Abstract—This paper discusses the self-collision avoidance problem in simulating the dynamic behavior of deformable cloth. It is not an easy task to simulate realistically the response of cloth in various types of complicated self-collisions. In this paper, the bounding box technique in addition to ordinary pre-checking for collision detection is used to avoid self-collision. Instead of avoiding self-collision of cloth triangles, we relax to avoid the collision of the bounding boxes of cloth triangles. Some constraints are enforced on the vertices of the cloth triangles to prevent their bounding boxes from penetrating in the direction of collision. The experimental results show that the relaxed self-collision avoidance method can create realistic cloth behavior and wrinkling formation processes. Since the types of interaction between bounding boxes are simple, our method is simple but robust in avoiding self-collisions. © 1998 Elsevier Science Ltd. All rights reserved

Key words: cloth animation, self-collision, bounding box, iteration method.

1. INTRODUCTION

Many researches have focused on representing the natural motion of deformable cloth [3–8, 11–13, 15, 16, 18, 19]. In these works, many realistic cloth animations, such as a flag moving in the wind, a tablecloth falling on a table and a garment on a human body, have been presented. Interested readers are referred to [10] for a survey of the current state of cloth modeling and animation. However, it is still difficult to animate a cloth with clinging deep wrinkles. The method used to respond realistically to collisions of clinging deep wrinkles is a primary issue in producing such a cloth animation.

There are two kinds of collisions which occur in cloth animation on physically based models: collisions of cloth with other rigid objects and self-collisions. Several techniques have been developed for detecting and avoiding collisions of cloth with rigid objects [2, 3, 5, 7–9, 15, 16, 19].

It is relatively difficult to deal with self-collisions. Previous works have focused on designing efficient self-collision detection algorithms and appropriate self-collision response methods. A realistic cloth model is usually composed of a huge number of cloth triangles. An *ad hoc* self-collision detection algorithm which checks the intersections of all the pairs of cloth triangles would be time consuming. The bounding box technique and hierarchical data structure have been widely used to reduce the number of pairs of cloth triangles which must be checked in self-collision detection [9, 15–17, 19]. In [15], Volino *et al*. avoided "false" collisions between adjacent cloth triangles, thus improving the efficiency of self-collision detection.

After detection of self-collisions, colliding cloth triangles must be responded to in appropriate ways. In normal cloth animation there are various geometric types of self-collisions, each of which can be caused by several different motion situations. It is difficult to respond appropriately to all types of self-collision situations, especially those involving multiple collisions where a cloth triangle intersects more than one triangle. Previously, self-collisions were directly dealt with according to their geometric situations, such as vertex-to-triangle, edge-to-triangle, edge-to-edge, edge-to-vertex and vertex-to-vertex collisions [5, 7, 16]. In those previous methods auxiliary response forces were used to avoid self-collisions. However, for complex wrinkling situations, such as clinging deep wrinkles which appear in the animation of cloth falling to the ground, it would be very difficult to calculate an appropriate response force to avoid all collisions. In fact, a response force which enables a cloth triangle to avoid collision may cause a new collision with another cloth triangle. On the other hand, high response forces may alter the stability of numerical simulation. In a previous article [8] we proposed a

† To whom correspondence should be addressed at: 128 Yen-Chiu-Yuan Road, Sec. II, Institute of Information Science, Academia Sinica, Taipei, 11529 Taiwan..

self-collision avoidance method without auxiliary response forces. Once self-collisions are detected the vertices of the colliding cloth triangles are fixed by setting constraints on the current simulation time step. This heuristic can avoid simple self-collisions, such as a cloth falling on a table. However, it cannot produce realistic cloth animation in complex collision situations, such as a cloth falling to the ground. For example, in the case where two cloth triangles collide in two consecutive simulation steps, the vertices of the cloth triangles are constrained to the current positions in the following simulation steps, and the movement of the cloth will be unrealistic.

In this paper we propose a self-collision avoidance scheme for cloth animation in a physically elastic surface model [11]. Our avoidance scheme is designed to handle general self-collision situations, such as cling deep wrinkles which appear in cloth falling to the ground. To reduce the number of intersection tests, similar to the structure in [16], we construct a hierarchical structure of bounding boxes to skip a large region during self-collision detection.

To avoid responding to various complex self-collision situations of triangles, our avoidance scheme avoids self-collisions by preventing intersections of their iso-oriented bounding boxes. For each pair of bounding boxes about to collide, some constraints are set on the vertices of the included triangles to prevent bounding boxes from colliding. By calculating a "collision plane" perpendicular to one of the three coordinate axes, the constraints are defined so as to prevent the bounding boxes from moving across this plane. The colliding vertices can move freely in the other two dimensions. To avoid multiple self-collisions the avoidance scheme is applied to all the bounding box pairs involved in multiple self-collisions.

Our avoidance scheme can robustly and realistically respond to complex self-collision situations. Since self-collisions are handled at the bounding box level, time consuming geometric intersection checking between triangles is avoided and implementation of the avoidance scheme is easier. The cloth animations produced by our method are comparable in quality with that obtained by Volino *et al.* [16].

The following sections of this paper are organized as follows. In Section 2 the dynamic cloth model is introduced. In Section 3 we describe our hierarchical data structure for collision detection. The self-collision avoidance method is proposed in Section 4. Experimental results and demonstration images are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. CLOTH MODEL AND SIMULATION PROCESS

Our work is based on the deformable model proposed by Terzopoulos *et al.* [11], in which elasticity theory is employed. The dynamic motion of a point in the material coordinates of a deformable model is governed by the following Lagrange equation [11]:

$$\frac{\partial}{\partial t}\left(\rho(a)\frac{\partial r(a,t)}{\partial t}\right)+\gamma(a)\frac{\partial r(a,t)}{\partial t}+\frac{\delta \varepsilon(r)}{\delta r}(a,t)=f(r(a,t),t), \quad (1)$$

where $r(a,t)$ is the position of point a at time t. The first term of the Lagrange equation is the force due to the mass point's intrinsic motion, the second term is the damping force due to dissipation, and the third term is the elastic force due to deformation of the body. The external force dynamically keeps in balance with these internal forces.

The cloth model is discretized as regular grids of $\mathbf{M}\times\mathbf{N}$ nodes spread on a unit square. Applying the finite difference approximation method to Equation (1) [1], we obtain a linear system:

$$\mathscr{A}_t\mathbf{r}_{t+\Delta t}=\mathbf{g}_t, \quad (2)$$

where the $\mathbf{MN}\times\mathbf{MN}$ matrix $\mathscr{A}_t$ is known as the stiffness matrix and the $\mathbf{MN}\times 1$ matrix $\mathbf{g}_t$ is the effective force vector. Notice that Equation (2) actually consists of three equations, one for each of the x, y and z coordinates:

$$\mathcal{A}_t\mathbf{r}_{x,t+\Delta t}=\mathbf{g}_{x,t}, \quad (3)$$

$$\mathcal{A}_t\mathbf{r}_{y,t+\Delta t}=\mathbf{g}_{y,t}, \quad (4)$$

$$\mathcal{A}_t\mathbf{r}_{z,t+\Delta t}=\mathbf{g}_{z,t}. \quad (5)$$

The dynamic motion of cloth can be obtained by solving the linear Equations (3)–(5) along the times. Any efficient iteration method, such as the Gauss–Seidel method, can be used to solve the linear system. To avoid self-collisions, in each simulation iteration, some constraints are enforced on the vertices of colliding triangles to prevent their bounding boxes from intersecting. The constraints on the vertices are translated into constraints on their corresponding variables in the linear system. How the constraints are set is described later in Section 4.

Conceptually, a simulation iteration is composed of several runs of the response phase and the detection phase, which are performed iteratively until no collisions exist. Assume the simulation time is t:

•*Response phase*: In the response phase we add a constraint on each colliding cloth point detected in the preceding detection phase by means of the avoidance method (at the initial point of the simulation iteration there is no constraint). The constraints set in the previous response phase are reserved and updated. With these constraints on the colliding cloth points we solve the linear system at time t to obtain intermediate positions of the cloth points and then go to the next detection phase.

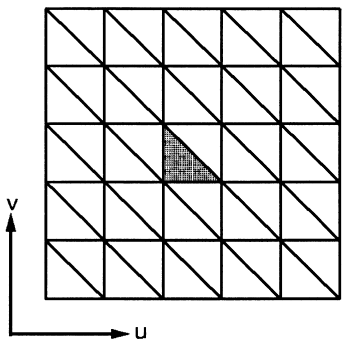

Fig. 1. Triangles in the material coordinates.

•*Detection phase*: In the detection phase we detect all the collisions and self-collisions from the intermediate grid point positions. If any collisions are detected we go to the response phase for further simulation. If no collisions are found the intermediate positions are set to be the positions of grid points at time $t + \Delta t$, and then we simulate the cloth behaviors in the next time interval.

3. SELF-COLLISION DETECTION

The adopted cloth model is represented by a large number of small triangles, and each triangle is defined by three adjacent grid points. The triangles in the material coordinates are illustrated in Fig. 1. To deal with the self-collisions, triangles which may collide have to be extracted from the whole set of triangles in the cloth. Detecting self-collisions is very time consuming since it often involves a huge number of geometrical tests needed to determine which elements are colliding.

In our method the bounding boxes of triangles are used to both detect and avoid self-collisions. The bounding box of a triangle is the minimum iso-oriented box including the triangle. Let the vertices of triangle T_i at time t be $V_{ij}(t)$, $j = 1, 2, 3$. The bounding box of T_i can be described by its two corner vertices,

$$V_{i,min}(t) = (x_{i,l}(t), y_{i,l}(t), z_{i,l}(t))$$
$$\text{and} \qquad V_{i,max}(t) = (x_{i,h}(t), y_{i,h}(t), z_{i,h}(t)), \quad (6)$$

where the x, y and z coordinates of $V_{i,\,min}(t)$($V_{i,}$ $max(t)$ resp.) are the minimum (maximum resp.) of those of $V_{ij}(t)$s, $j = 1, 2, 3$. Since the bounding box is iso-oriented, projecting it onto the x, y and z axes results in three intervals which can also represent the bounding box itself. Two bounding boxes have an intersection if and only if their corresponding intervals overlap in all three dimensions. In our method two triangles are assumed to be colliding if their bounding boxes intersect or penetrate.

Though it is easy to check whether or not a pair of bounding boxes intersect, it is still time consuming to check all the $O(n^2)$ pairs of bounding boxes. To reduce the number of tests a hierarchical structure is used. The hierarchical structure is a binary tree constructed by recursively partitioning the cloth evenly on the material coordinates until each node contains only a single triangle (for example see Fig. 2). Each node of the tree records the bounding box which is the minimum box enclosing the triangles in its sub-trees.

To detect collisions with a bounding box, say B, we search the binary tree from the root to the leaves. Let N be a searched node. For node N, we check if B intersects with the bounding box of N. If it does, we further search the two children of node N. Otherwise there is no intersection between B and the bounding boxes of the descendant leaves of node N, and no intersection checking is needed for the descendant nodes of node N. If the detected colliding bounding boxes correspond to adjacent triangles, no constraint is set.

The hierarchical data structure is constructed at the beginning, and the bounding box of each node is updated with each animation frame. It takes $O(n)$ time to update the bounding boxes from the leaves up to the root. In the worst case the detection algorithm may take $O(n^2)$ time for n triangles. However, in general, the detection algorithm efficiently skips most of the non-intersecting parts of the cloth and concentrates on the parts which are likely to collide. According to the experimental results shown in Section 5, the detection algorithm really saves much time in self-collision detection.

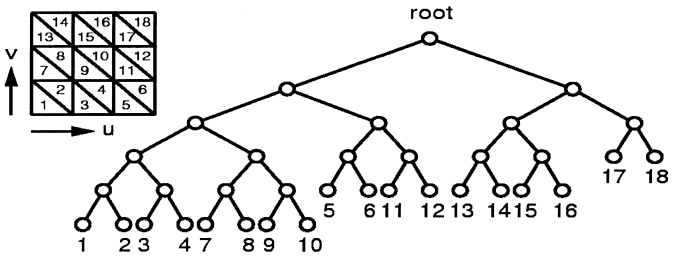

Fig. 2. Hierarchical structure of a cloth.

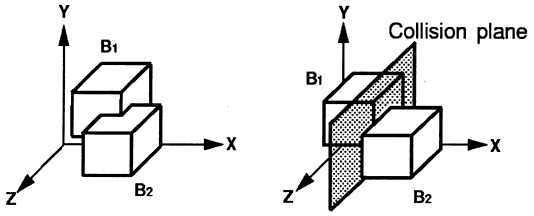

Fig. 3. Collision plane of colliding bounding boxes.

4. DYNAMIC MOTION WITH SELF-COLLISION AVOIDANCE

In order to simulate realistic cloth behaviors, it is necessary to respond properly to collisions of a cloth with rigid objects and with different parts of the cloth itself. In the following we will present our new avoidance method for dealing with self-collisions. As to the cloth collisions with rigid objects, the method proposed in [8] is adopted.

Our avoidance method avoids self-collisions by constraining the bounding boxes of triangles. Once the bounding boxes of triangles which may collide are located, a constraint-setting process is applied to prevent intersections. For a pair of bounding boxes about to collide, the collision plane on which they first collide is calculated. The collision plane is perpendicular to the x, y or z coordinate axis and is defined as $d = k$, where $d = x$, y or z and k is a calculated value. For the pair, constraints are set on the vertices of the corresponding triangles in the simulation iteration such that their bounding boxes can not move across the collision plane (see Fig. 3); thus, the triangles are free from self-intersection at time $t + \Delta t$.

In general, multiple self-collisions of a bounding box which intersects with more than one bounding box may occur in a simulation iteration. Multiple self-collisions can occur when a set of bounding boxes have a common intersection, when there is a chain of bounding boxes, each of which collides with the next box in the chain, and so on. In our avoidance method we take general collision situations as the set of all the colliding bounding box pairs. A vertex may be contained in several colliding bounding box pairs. The constraints arising

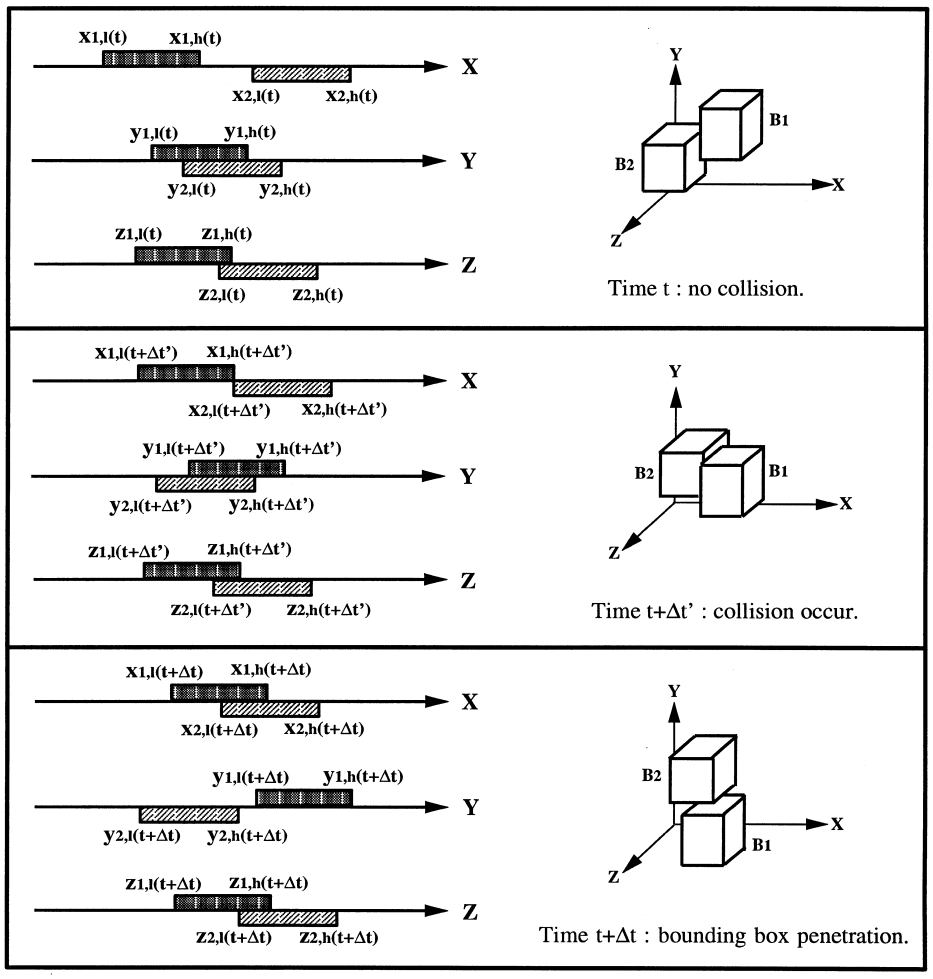

Fig. 4. Bounding boxes projected onto three coordinate axes.

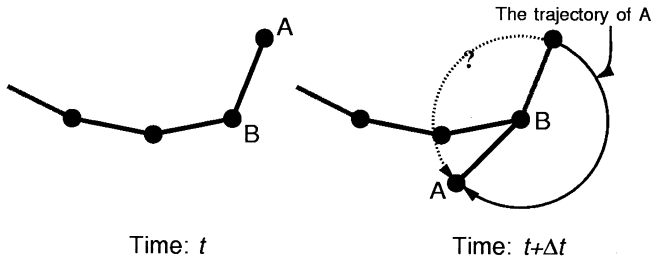

Fig. 5. The trajectory of a vertex on the boundary patches.

from the colliding bounding box pairs containing the vertex are summarized to provide the constraints of the vertex.

Two problems need to be discussed and dealt with to guarantee that our method will work. One problem is on the constraint-setting conflict arising from vertices that are involved in more than one colliding bounding box pair. The other problem is self-collision avoidance of adjacent triangles since our detection method cannot detect the penetration of adjacent triangles.

In the following a constraint setting for a colliding bounding box pair and constraint setting conflict resolution will be given in Section 4.1. The problem of the self-collision of adjacent triangles will be discussed in Section 4.2.

4.1. *Avoiding penetration of bounding box pair*

Intuitively, the collision plane of two bounding boxes about to collide is the first plane in which they touch each other. This can be determined by calculating the collision time of the bounding box pair between time t and $t + \Delta t$. Let bounding boxes B_1 and B_2 be defined as in Equation (6). They do not collide at time t but collide with each other at time $t + \Delta t'(0 \leqslant \Delta t' \leqslant \Delta t)$. By projecting the two iso-oriented bounding boxes at time t and time $t + \Delta t$ onto the three coordinate axes, we obtain three pairs of bounding intervals (as shown in Fig. 4). In the time interval t to $t + \Delta t$ the bounding intervals can be assumed to move linearly. For each pair of bounding intervals, on say coordinate d, we can calculate the time interval $T_d = [\Delta t_{d,\,min},\ \Delta t_{d,\,max}] \subseteq [0,\ \Delta t]$ in which they intersect with or penetrate each other. If the bounding intervals intersect at time t, then $\Delta t_{d,\,min}$ is set to be zero. If the bounding intervals intersect at time $t + \Delta t$, then $\Delta t_{d,\,max}$ is set to be Δt. For a pair of bounding boxes the intersection time interval T is calculated by $T = T_x \cap T_y \cap T_z$. The bounding boxes collide if $T \neq \emptyset$. The collision time $\Delta t'$ is the minimum value of T.

Let the collision plane be perpendicular to coordinate axis d, where $\Delta t' = \Delta t_{d,min}$. Thus, the collision plane is that defined by $d = k$, where

$$k = \begin{cases} d_{1,h}(t) + \dfrac{\Delta t'}{\Delta t}\left(d_{1,h}(t+\Delta t) - d_{1,h}(t)\right) \\ \qquad \text{if } d_{1,h}(t) < d_{2,l}(t) \\ d_{2,h}(t) + \dfrac{\Delta t'}{\Delta t}\left(d_{2,h}(t+\Delta t) - d_{2,h}(t)\right) \\ \qquad \text{if } d_{2,h}(t) < d_{1,l}(t). \end{cases} \tag{7}$$

The above collision detection method might mistake the situation which occurs with the boundary patches shown in Fig. 5 as penetration, though it is not. However, since the movement of the cloth points in a single time quantum is small, we may assume that this situation does not occur.

To avoid collision of a bounding box pair, a constraint is set on coordinate d of each vertex, P, of the including triangles. The constraint is

$$P_d(t + \Delta t) < k \quad \text{if } P_d(t) < k \tag{8}$$

and

$$P_d(t + \Delta t) > k \quad \text{if } P_d(t) > k \tag{9}$$

The value k is called an upper constraint value (a lower constraint value resp.) of P on coordinate d

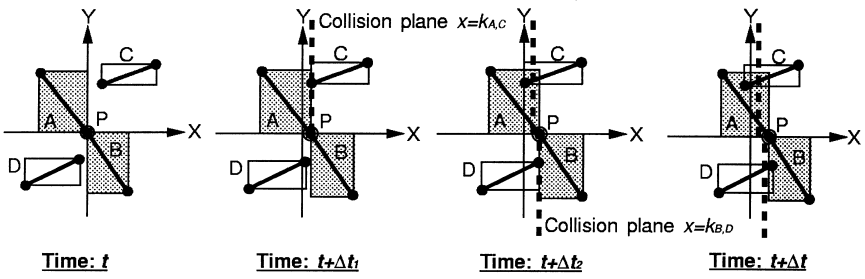

Fig. 6. Constraint conflict at vertex P.

with respect to the bounding box pair if constraint in (8) (constraint in (9) resp.) holds. In general, a vertex P may be involved in several colliding bounding box pairs, each of which gives an upper constraint value or a lower constraint value to some coordinate of P. For each coordinate d of P, let upper limit be the minimum of all its upper constraint values and lower limit be the maximum of all its lower constraint values. If the simulation result of P_d is less than its upper limit and greater than its lower limit, the vertex P will not penetrate any bounding box (not containing P) along the direction of the d-axis.

Ordinarily, the upper limit is greater than or equal to the lower limit. In this case, the value of $P_d(t + \Delta t)$ is constrained between the upper limit and the lower limit. However, the case that the upper limit is less than the lower limit, called a constraint conflict, may occur. For a coordinate of a vertex, P_d has a constraint conflict, which means that P is involved in two bounding boxes, where one has a dynamic obstacle from the upper side and the other has a dynamic obstacle from the lower side along the direction of the d-axis. The constraint conflict situation is illustrated in Fig. 6. In Fig. 6, bounding boxes A and B share a vertex P. At time $t + \Delta t$, self-collisions occur between A and C as well as between B and D; thus, the collision planes $x = k_{A,C}$ and $x = k_{B,D}$ are calculated. Consider the constraint on the x coordinate of vertex P. The upper limit is $k_{A,C}$ according to the collision of A and C, and the lower limit is $k_{B,D}$ according to the collision of B and D. This is a constraint conflict. Intuitively, the forces, due to the collisions, exerted on a vertex with constraint conflict will balance at a certain point such that the vertex is between the upper limit and the lower limit at time $t + \Delta t$. In other words, P_d at time $t + \Delta t$ should be between the upper limit and the lower limit. By this intuition, we set the constraint interval on the coordinate d of P as follows:

$$\begin{aligned}&\text{Constraint}[P]_d = [\text{lower limit, upper limit}]\\ &\quad\text{if there is no constraint conflict,}\\ &\text{Constraint}[P]_d = [\text{upper limit, lower limit}]\\ &\quad\text{if there is a constraint conflict.}\end{aligned} \tag{10}$$

To guarantee that the simulation satisfies the constraints set above, we go back to time t and solve the linear systems (3), (4) and (5) using the constrained Gauss–Seidel method, in which the iteration steps are as follows:

$$d_i^{(m+1)} = \begin{cases} \text{Constraint}[i]_d.\text{upper} \\ \text{if Constraint}[i]_d.\text{upper} \neq \text{null} \\ \text{and } d_i^{(m+1)} > \text{Constraint}[i]_d.\text{upper}, \\ \text{Constraint}[i]_d.\text{lower} \\ \text{if Constraint}[i]_d.\text{lower} \neq \text{null} \\ \text{and } d_i^{(m+1)} < \text{Constraint}[i]_d.\text{lower}, \\ \frac{1}{a_{ii}}\left\{ g_{d_i} - \sum_{j=1}^{i-1} a_{ij} d_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij} d_j^{(m)} \right\} \\ \text{otherwise}, \end{cases} \tag{11}$$

where $d = x$, y or z.

Solving the linear system using the constrained Gauss–Seidel method, each coordinate of any vertex will be in its constrained interval. For every cloth point, the constraint defines an interval of motion, and the cloth point can move freely within this interval. Notice that the vertex can move as usual in the unconstrained directions (see Fig. 7).

After a run of the constrained simulation phase, collisions of bounding boxes which include vertices without constraint conflicts will be prevented. Colliding bounding boxes with constraint conflicts may continue to collide. However, the constrained interval of the vertex having a constraint conflict shrinks with each run of the response and detection phases. Consider the constraint conflict depicted in Fig. 6. Let P_C(P_D resp.) be the vertex of bounding box C (D resp.) with the minimum (maximum resp.) x coordinate at time t, $P_C(t) > P_D(t)$. In the first simulation iteration, the upper limit P_U^1 and the lower limit P_L^1 of P can be calculated by (7):

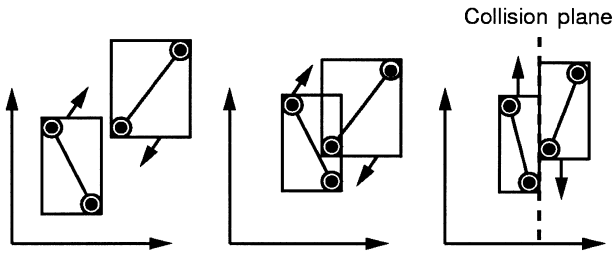

Fig. 7. Bounding box sliding over a collision plane.

Table 1. Computation times of various avoidance methods (S1, geometrical collision avoidance method; S2, bounding box avoidance method without hierarchy; S3, bounding box avoidance method with hierarchy)

Cloth falling on the table	21×21			51×51		
	S1	S2	S3	S1	S2	S3
Average number of iterations	2	2	2	2	2	2
Maximum number of iterations	4	4	4	4	4	4
Average time for self-collision detection in one simulation step	3.97	1.33	0.05	116.53	40.14	0.23
Average total time in one simulation step	12.96	2.79	0.39	272.76	73.26	2.02

$$\mathrm{P}_{\mathrm{U}}^{1} = \mathrm{P}_{\mathrm{C}}(t) + t_{\mathrm{A,C}}^{1}\big(\mathrm{P}_{\mathrm{C}}(t+\Delta t) - \mathrm{P}_{\mathrm{C}}(t)\big),$$
$$\mathrm{P}_{\mathrm{L}}^{1} = \mathrm{P}_{\mathrm{D}}(t) + t_{\mathrm{B,D}}^{1}\big(\mathrm{P}_{\mathrm{D}}(t+\Delta t) - \mathrm{P}_{\mathrm{D}}(t)\big), \quad (12)$$

where $0 < t_{\mathrm{A,\,C}}^{1}, t_{\mathrm{B,\,D}}^{1} < 1$. The points P, $\mathrm{P_C}$ and $\mathrm{P_D}$ are constrained by the values $\mathrm{P}_{\mathrm{U}}^{1}$ and $\mathrm{P}_{\mathrm{L}}^{1}$ in the second iteration. After n iterations, the upper and lower limits then become:

$$\begin{aligned}\mathrm{P}_{\mathrm{U}}^{n} &= \mathrm{P}_{\mathrm{C}}(t) + t_{\mathrm{A,C}}^{n}\big(\mathrm{P}_{\mathrm{U}}^{n-1} - \mathrm{P}_{\mathrm{C}}(t)\big)\\ &= \mathrm{P}_{\mathrm{C}}(t) + t_{\mathrm{A,C}}^{n}\, t_{\mathrm{A,C}}^{n-1} \cdots t_{\mathrm{A,C}}^{1}\big(\mathrm{P}_{\mathrm{C}}(t+\Delta t) - \mathrm{P}_{\mathrm{C}}(t)\big),\\ \mathrm{P}_{\mathrm{L}}^{n} &= \mathrm{P}_{\mathrm{D}}(t) + t_{\mathrm{B,D}}^{n}\big(\mathrm{P}_{\mathrm{L}}^{n-1} - \mathrm{P}_{\mathrm{D}}(t)\big)\\ &= \mathrm{P}_{\mathrm{D}}(t) + t_{\mathrm{B,D}}^{n}\, t_{\mathrm{B,D}}^{n-1} \cdots t_{\mathrm{B,D}}^{1}\big(\mathrm{P}_{\mathrm{D}}(t+\Delta t) - \mathrm{P}_{\mathrm{D}}(t)\big),\end{aligned} \quad (13)$$

where $0 < t^{i}_{\mathrm{A,\,C}}, t^{i}_{\mathrm{B,\,D}} < 1$ and $i = 1, 2, \ldots, n$. Since the collision times $t^{i}_{\mathrm{A,\,C}}$ and $t^{i}_{\mathrm{B,\,D}}$ are smaller than 1, the constraint interval shrinks as n increases. Our experiments show that the upper limit and lower limit will converge to a specific value to resolve the constraint conflict.

The number of iterations needed to avoid constraint conflict depends on the complexity of the cloth collision situation. As illustrated in Tables 1 and 2, our experiments show that when a 21×21 point cloth model falling to the floor vertically is simulated, 1 to 8 runs are needed to avoid all self-collisions and constraint conflict. To simulate some simple cloth behaviors, for example a cloth falling over a table, only 1 to 4 runs are needed.

4.2. *Avoiding the collision of adjacent triangles*

In this subsection we will discuss the avoidance of collisions of adjacent triangles. Since the bounding boxes of two adjacent triangles always intersect, we will not spend time on detecting this kind of collision. In fact, the avoidance method for nonadjacent triangles cannot be applied to adjacent ones to avoid collisions because the collision plane and, thus, the constraints can not be well defined.

However, the collision avoidance of non-adjacent triangles in our method can automatically prevent the collision of adjacent triangles. A patch has two main different kinds of adjacent triangles; one shares a vertex and the other shares an edge (see Fig. 8). Let us take the adjacency condition depicted in Figs 9 and 10 as an example to explain why the collision of adjacent triangles can be avoided by our method. The reason for the other conditions is similar. In Fig. 9 the adjacent triangles A and B share an edge. Although we do not detect the intersection of bounding boxes A and B, however, by avoiding the collision of bounding boxes $\mathrm{A_1}$–$\mathrm{A_3}$ with B and $\mathrm{B_1}$–$\mathrm{B_3}$ with A, the vertex P_{A} will not penetrate the bounding box of B and the vertex P_{B} will not penetrate the bounding box of A. Thus the triangles A and B do not move across each other and the self-collision is prevented. In Fig. 10, the adjacent triangles A and B share a vertex. By avoiding the collision of bounding boxes of $\mathrm{B_1}$–$\mathrm{B_7}$ with A and $\mathrm{A_1}$–$\mathrm{A_7}$ with B, the penetration of triangles A and B is prevented since the vertices P_{A1} and P_{A2} do not penetrate the bounding box of B and the vertices P_{B1} and P_{B2} do not penetrate the bounding box of A. That is, the collision of adjacent triangles can be avoided by our avoidance method.

Table 2. Computation times of various avoidance methods (S1, geometrical collision avoidance method; S2, bounding box avoidance method without hierarchy; S3, bounding box avoidance method with hierarchy)

Cloth falling on the table	21×21			51×51		
	S1	S2	S3	S1	S2	S3
Average number of iterations	3	3	3	4	4	4
Maximum number of iterations	4	5	5	5	8	8
Average time for self-collision detection in one simulation step	4.07	1.32	0.03	118.43	40.09	0.21
Average total time in one simulation step	10.42	4.04	0.40	312.87	149.54	3.56

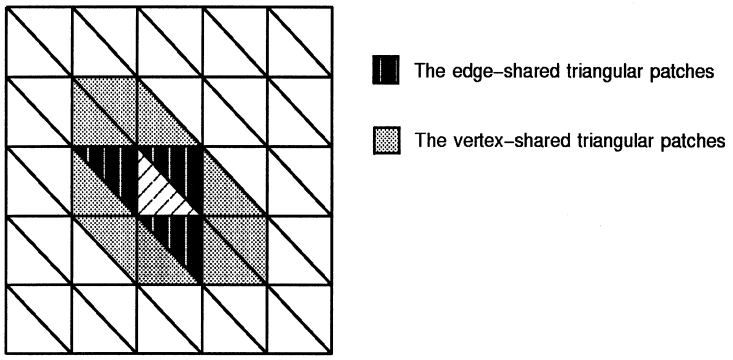

Fig. 8. Edge-shared and vertex-shared triangles.

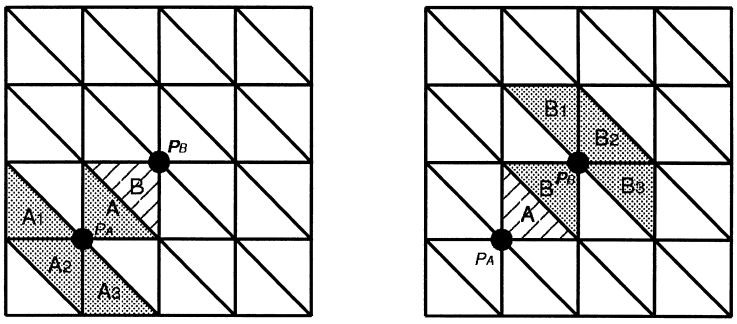

Fig. 9. Adjacent triangles share an edge.

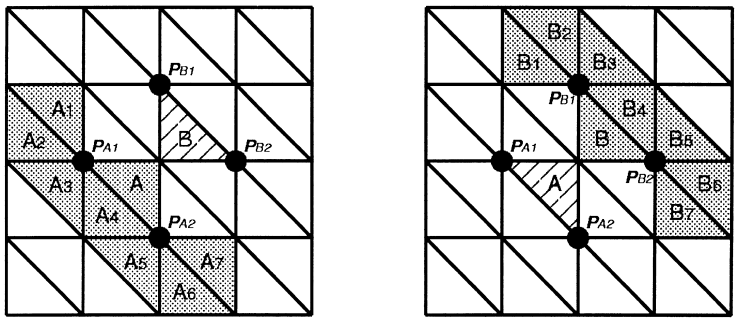

Fig. 10. Adjacent triangles share a vertex.

5. EXPERIMENTAL RESULTS

In our experiments we simulated 21×21 and 51×51 grid points of cloth falling on a square table and falling freely to the ground. The experimental environment was a DEC 5000/260 workstation with an R4400 CPU and 64 Mbytes of memory. Each cloth animation was obtained by letting the simulation run for 1000 time steps.

The efficiency of the proposed bounding box avoidance was compared with the geometrical collision avoidance method given in [8]. In the geometrical collision avoidance method, no hierarchical data structure is used in the detection procedure. The performance of the bounding box methods using and not using the hierarchical data structure described in Section 3 in collision detection was also compared. Tables 1 and 2 show the average number of iterations, the maxi-

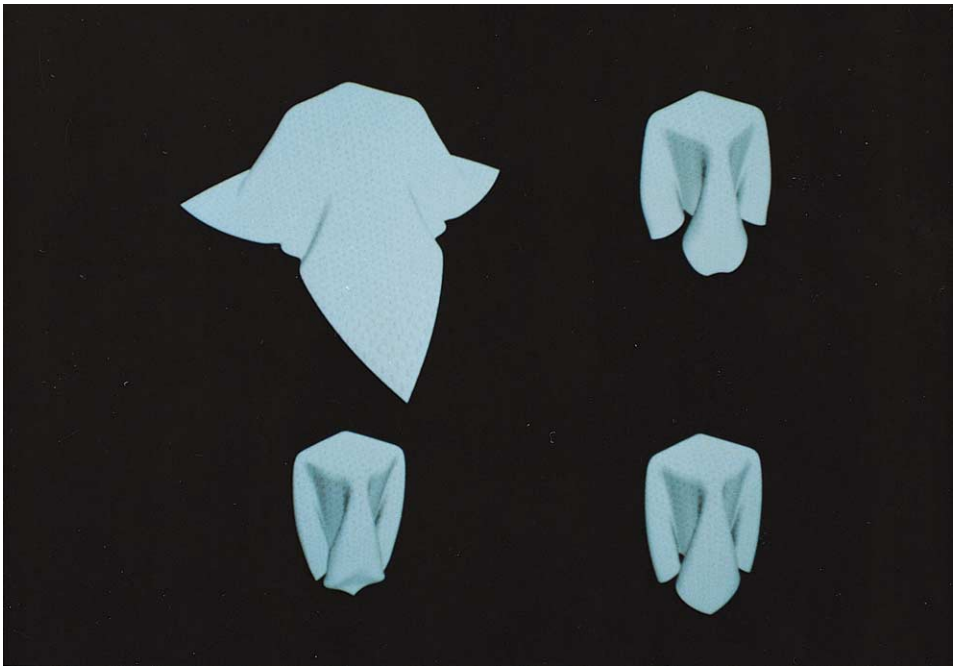

Fig. 11. 21×21 grid points of cloth falling on a table.

mum number of iterations, the average time for self-collision detection and the average total time in one time step. The average total time includes the time needed for cloth–object collision detection and the time required to evaluate the linear system.

As shown in Tables 1 and 2, the geometrical collision avoidance method takes much more time than the bounding box avoidance method in self-collision detection. The performance is drastically improved by applying the hierarchical algorithm. Because additional self-collision detection and reiterating procedures are needed to

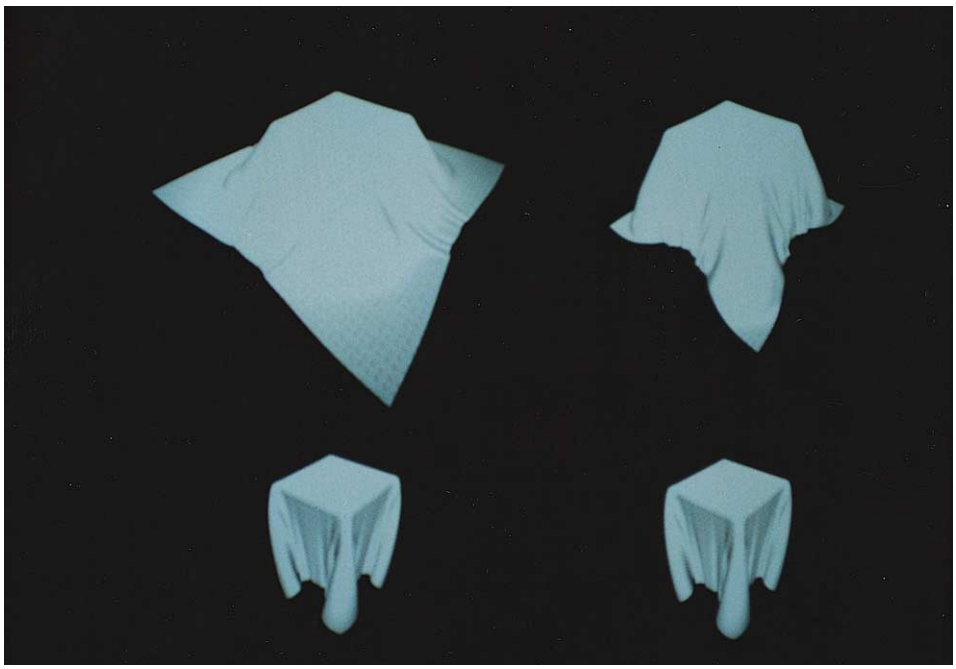

Fig. 12. 51×51 grid points of cloth falling on a table.

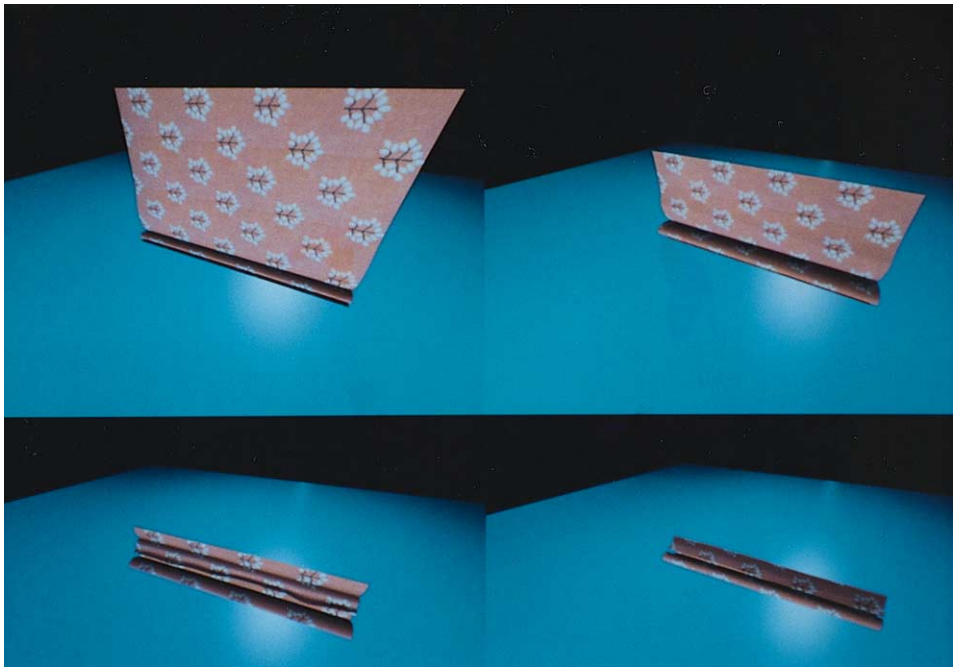

Fig. 13. 21×21 grid points of cloth falling to the ground.

avoid complex self-collision situations, the simulation of a cloth falling to the ground took much more time than did the simulation of a cloth falling on a table.

Figures 11 and 12 show the 21×21 and 51×51 grid points of cloth falling on a square table. In this cloth animation, self-collisions only occurred at the four corners. Our avoidance method could produce realistic cloth behavior. Figures 13–16 illustrate a complex self-collision situation which involved simulation of 21×21 and 51×51 grid points of cloth falling to the ground. Figures 13 and 14 show that cloth with low resolution could cause some inaccuracy. However, when the cloth was represented with high resolution, for example, 51×51 grid points of cloth as shown in Figs 15 and 16, this kind of inaccuracy would be reduced.

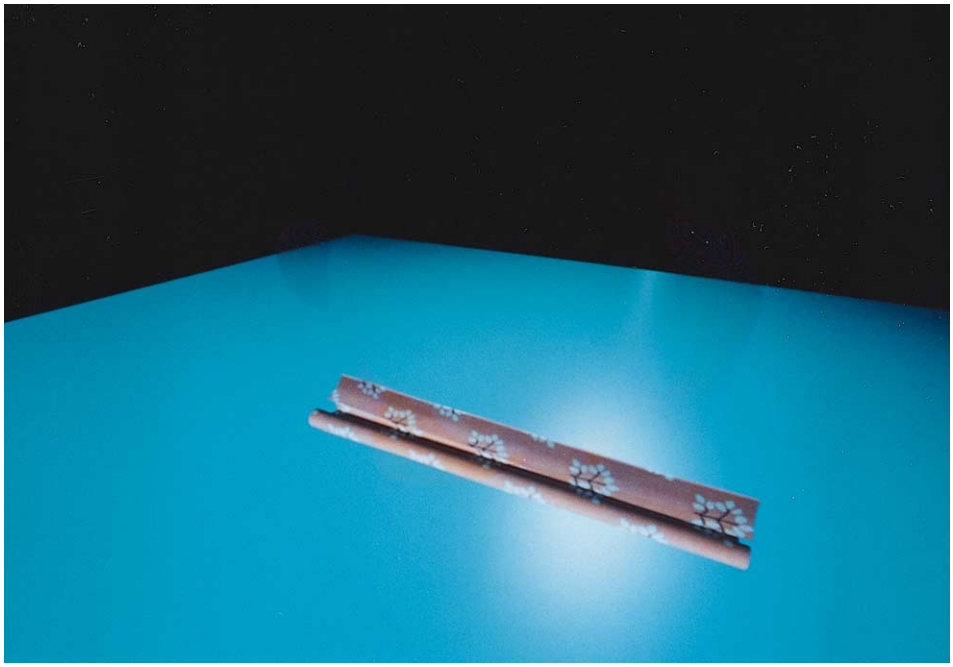

Fig. 14. Final static appearance of 21×21 grid points of cloth falling to the ground.

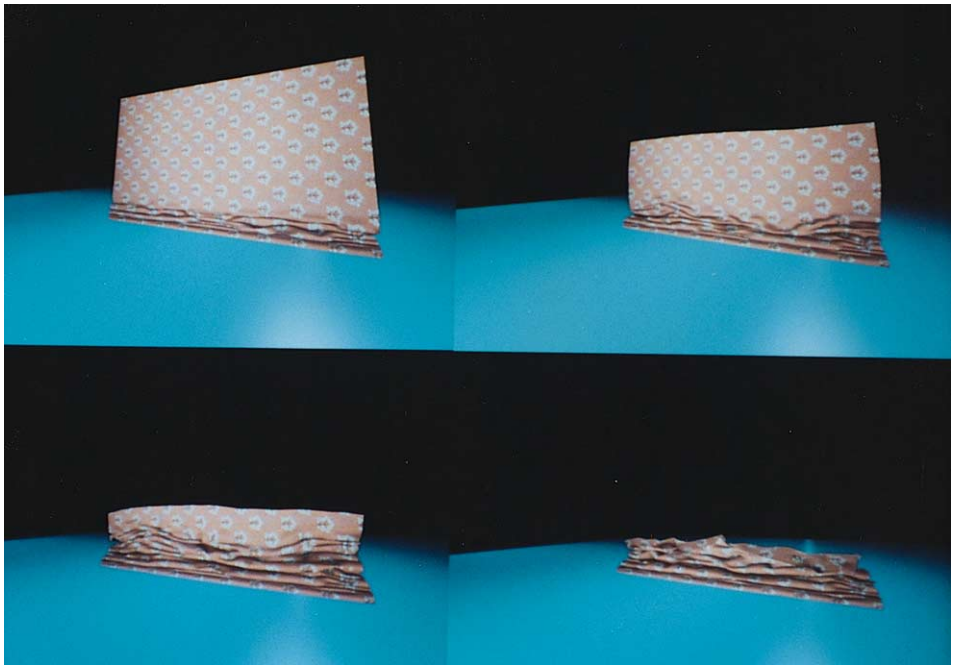

Fig. 15. 51×51 grid points of cloth falling to the ground.

6. CONCLUSION

We have proposed a method for avoiding self-collision in cloth animation. The main idea is to use bounding boxes not only to detect self-collisions, but also to prevent self-collisions. Using the bounding box avoidance method, self-collisions can be detected efficiently by using a hierarchical data structure. We avoid self-collisions by enforcing some constraints on the vertices of triangles such that after simulation using the constrained Gauss–Seidel method, their bounding boxes will not collide and self-collisions are avoided.

Our avoidance method is simple, easy to implement and robust in preventing self-collisions. Furthermore, it can handle complicated self-collision situations which have been considered difficult to respond to appropriately. The experimental

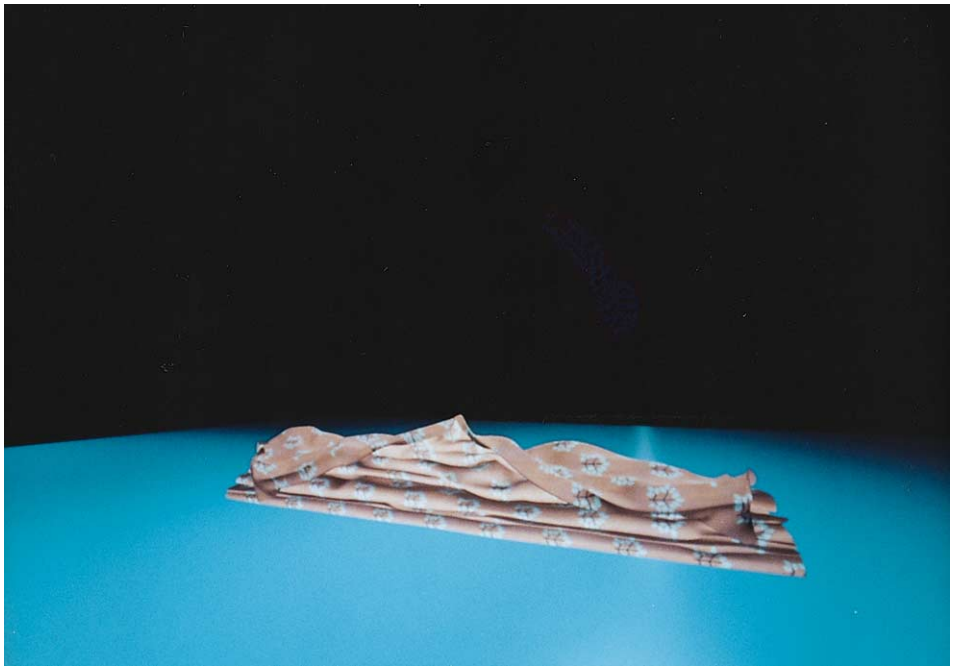

Fig. 16. Final static appearance of 51×51 grid points of cloth falling to the ground.

results show that our avoidance method can simulate realistic cloth behaviors while avoiding self-collisions.

REFERENCES

1. Atkinson, K. E., *An Introduction to Numerical Analysis*. John Wiley & Sons, New York, 1988.
2. Barzel, R. and Barr, A. H., A modeling system based on dynamic constraints. *Computer Graphics*, 1988, **22,** 179–188.
3. Breen, D. E., House, D. H. and Getto, P. H., A physically-based particle model of woven cloth. *Visual Computer*, 1992, **8,** 264–277.
4. Breen, D. E., House, D. H. and Wozny, M. J., Predicting the drape of woven cloth using interacting particles. *Computer Graphics*, 1994, **28,** 365–372.
5. Carignan, M., Yang, Y., Thalmann, N. M. and Thalmann, D., Dressing animated synthesis actors with complex deformable clothes. *Computer Graphics*, 1992, **26,** 99–104.
6. Kunii, T. L. and Gotoda, H., Singularity theoretical modeling and animation of garment wrinkle formation processes. *Visual Computer*, 1990, **6,** 326–336.
7. Lafleur, B., Thalmann N. M. and Thalmann, D., Cloth animation with self-collision detection. In *Modeling in Computer Graphics*, ed. T. L. Kunnii. Springer, Berlin, 1991, pp. 179–187.
8. Liu, J. D., Ko, M. T. and Chang, R. C., Collision avoidance of cloth animation. *Visual Computer*, 1996, **12,** 234–243.
9. Moore, M. and Wilhelms, J., Collision detection and response for computer animation. *Computer Graphics*, 1988, **22,** 289–298.
10. Ng, N. and Grimsdale, R. L., Computer graphics techniques for modeling cloth. *IEEE Computer Graphics & Applications*, 1996, **16,** 28–41.
11. Terzopoulos, D., Platt, J., Fleischer, K. and Barr, A. H., Elastically deformable models. *Computer Graphics*, 1987, **21,** 205–214.
12. Terzopoulos, D. and Fleischer, K., Modeling inelastic deformation: viscoelasticity, plasticity, fracture. *Computer Graphics*, 1988, **22,** 269–278.
13. Terzopoulos, D. and Fleischer, K., Deformable models. *Visual Computer*, 1988, **4,** 306–331.
14. Terzopoulos, D. and Witkin, A., Physically based models with rigid and deformable components. *IEEE Computer Graphics & Application*, 1988, **8,** 41–51.
15. Volino, P. and Thalmann, N. M., Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum*, 1994, **13,** 155–166.
16. Volino, P., Courchesne, M. and Thalmann, N. M., Versatile and efficient techniques for simulating cloth and other deformable objects. *Computer Graphics*, 1995, **29,** 137–144.
17. Webb, R. C. and Gigante, M. A., Using dynamic bounding volume hierarchies to improve efficiency of rigid body simulations. In *Computer Graphics International Proc.*. Springer-Verlag, 1992, pp. 825–841.
18. Weil, J., The synthesis of cloth objects. *Computer Graphics*, 1986, **20,** 49–54.
19. Yang Y. and Thalmann, N. M., An improved algorithm for collision detection in cloth animation with human body. In *Proceedings of Pacific Graphics*. World Scientific Press, Singapore, 1993, pp. 237–251.