Tanglement Resolution in Clothing Simulation With Explicit Convergence

Ick-Hoon Cha and Hyeong-Seok Ko

Abstract—This article proposes a new discrete collision handling method (DCH method) that can be used for resolving tanglements in clothing simulation, based on the existing continuous collision handling methods (CCH methods). The proposed method performs intersection analysis of the clothing mesh at every time step, and stores the result in the form of coloring the vertices, edges, and triangles. Referring to the coloring, the method resolves the tanglements in the out-to-in manner by applying the proposed operations, the triangle shrinkage and vertex pull, to the triangles and vertices around the intersection path. We take note of the CCH methods that use some small tolerance value to defend the round-off errors for the purpose of preventing false negatives. This work gives a second thought to that tolerance value, and proposes a new DCH method which uses the tolerance value for the resolution purpose. Under certain conditions, the method turns out to guarantee resolution of the tanglements in a finite number of time steps.

Index Terms—Computer graphics, clothing simulation, self-collisions, tanglement resolution, re-meshing

1 Introduction

Handling self collisions in clothing simulation has been a challenging problem. For proper handling of them, continuous collision detection methods (CCD) and response generation methods based on CCD (CCR) have been adopted in many clothing simulation systems.

Continuous collision handling (CCH), i.e., CCD + CCR, is powerful to defend collisions, but it works under the following premise: when starting the current time step, the garment has to be in a collision-free state. If a time step starts with some self-collisions unresolved, we will say the mesh is *tangled* and call the problematic regions as *tanglements*. We will call the CCH that guarantees collision-free simulation when the above premise is met as the *full-proof CCH*. Unfortunately, when used in the real world situations (e.g., in the real-time virtual fitting systems or animation studios), even the simulators equipped with a full-proof CCH may produce anomalous results, due to the following violations of the premise:

- *Administrative quit*: Occasionally (for example, if the simulation is required to run at a certain fps), an administrative decision has to be made to move on to the next time step even if the CCH is not complete yet.
- *Onset tanglements*: When a garment is initially created, as the constituent panels are seamed, the panels can interpenetrate each other. Since the above has already happened when the garment is born, the first frame of the simulation has to start with tanglements.
- *User-generated tanglements*: The premise can be violated when the user applies interactive manipulations to the garment while the simulation is being performed.
- *Pinching*: If cloth vertices are pinched [1] by the body, it may be impossible to perform a valid CCR without creating cloth-body collision, calling for an administrative quit.

Once tanglements are introduced (via the above sources), we cannot expect them to disappear by the operation of CCH. Being history based, CCH tries to *preserve* the existing self-collisions instead of resolving it. The preceding discussions imply that the simulator needs a discrete collision handling method (DCH) regardless of how concrete the CCH is.

This paper proposes a new DCH method such that, for certain categories of tanglements (as detailed in Section 2), the method resolves them within a finite number of steps. The method should operate under the following conditions: (1) there should be no pinched cloth vertices when the method is initiated, and (2) until the resolution is complete, administrative quit, user-generated tanglements, or pinching should not occur.[1] We make the simulator have the DCH capability by making simple modifications to the conventional simulator that has a full-proof CCH. The merit of the proposed method does not lie in the resolution speed or physical correctness, but in explicit convergence.

2 Related Work

Several DCH methods have been proposed to escape from the tangled state. Baraff and Witkin [1] proposed global

• *The authors are with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea*
E-mail: {picklepie, hsko}@graphics.snu.ac.kr.

Manuscript received 25 Dec. 2019; revised 21 Oct. 2020; accepted 16 Nov. 2020. Date of publication 23 Nov. 2020; date of current version 27 May 2022. (Corresponding author: Ick-Hoon Cha.)
Recommended for acceptance by S. Schaefer.
Digital Object Identifier no. 10.1109/TVCG.2020.3039566

1. We need this condition for technical discussion. Although the proposed DCH method does not introduce new tanglements by itself, if new tanglements are generated continuously for extraneous reasons (e.g., the user moves the cloth vertices sinusoidal way to generate tanglements artificially, or the character lowers the arms and pinches the clothing at the armpit), it would be impossible to technically discuss the completion of the resolution. We will assume this condition (i.e., *no new extraneous tanglements during DCH*) throughout this paper.

intersection analysis (GIA), which colors the vertices based on the intersection analysis. The coloring could guide to which direction each vertex should move to resolve tanglement. Wicke *et al.* [2] investigated possible cases of intersection paths, then classified them into seven categories, and proposed methods to handle those categories. Volino and Magnenat-Thalmann [3] proposed intersection contour minimization (ICM), which is based on the observation that the tangled situation can be resolved by minimizing the intersection contour. Ye and Zhao [4] suggested some improvements to the formulation of ICM, and proposed a modified version of ICM such that collision can be resolved with a less amount of iterations when additional information is given. Ye *et al.* [5] proposed unified intersection resolver (UIR), which is a new pipeline of collision handling by harmoniously combining some of the previous methods. The UIR calculated the minimal displacements for resolving the penetrations with the stencil-based formulation introduced in [6]. Recently, Buffet *et al.* [7] proposed a solution to untangle intersecting garment layers when each garment is independently prepared to a single body. Each garment was approximated to a closed implicit surface, and with the user's input of layering order and rigidness, each surface was deformed to remove intersections.

In the prior works (including [1], [2], [3], [5]), the convergence/coverage was often not explicit. Therefore, with those methods, when a tanglement is persistent, it was difficult to judge whether the method needs more iteration or if the method cannot cover the case. This paper attempts to develop a DCH method that is more explicit in the convergence/coverage compared to the existing DCH methods.

3 Preliminary

In this paper, we consider only triangular meshes. We denote an edge which is composed of vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ as $E(\mathbf{x}_i, \mathbf{x}_j)$, and a triangle composed of vertices $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$ as $T(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$. We assume that a garment is comprised by seaming a number of (sewing) panels, each existing separately in its own *uv*-space.

For resolving tanglements, the first step should be identifying the tangled regions,[2] which can be achieved by analyzing the intersections the clothing mesh is currently making. We store the result of the analysis in the form of coloring the vertices, edges, and triangles. Note that our coloring, which is detailed below, is different from the previous intersection analysis coloring ([1], [2]).

- *Red vertices*
  If an intersection path divides the mesh into two separate regions, we classify the vertices in the smaller region (in terms of the number of vertices) as red vertices.
- *Red triangles*
  If any of the three vertices of the triangle is red, that triangle is a red triangle. Additionally, the triangle T_i that intersects with another triangle T_j is colored red regardless of whether the vertices of T_i are red or not. (Of course, the above is mutual, thus T_j is also red in this case.) For example, in Fig. 2a, even if all three vertices $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ are green, $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ intersects with $T(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5)$, so it is a red triangle.
- *Red edges*
  If any of the two vertices making up the edge is red, it is a red edge. Additionally, the edge E_i that penetrates a triangle is colored red regardless of whether the vertices of E_i are red or not. For example, even if all four vertices $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ are green in Fig. 2b, both $E(\mathbf{x}_1, \mathbf{x}_2)$ and $E(\mathbf{x}_3, \mathbf{x}_4)$ are red edges since they both penetrate a triangle.
- *Green vertices, edges, triangles*
  All vertices, edges, and triangles which are not colored as red are green.

Under the above definitions, we note a special subsets of the red vertices and red triangles by giving more definitive names, since they play important roles in the proposed method.

- Among the red vertices, the ones which are adjacent to at least one green vertex are called *out-most red vertices* (e.g., $\mathbf{x}_1$ in Fig. 3a).
- Among the red triangles, the ones which consist of three green vertices, but in terms of edge coloring, which consist of two green edges and one red edge are called *red* triangles* (e.g., $T(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_5)$ in Fig. 3e).

Throughout the paper, we will use Wicke *et al.*'s classification [2] of tanglements. Seven categories (namely, BLI, BBII, BIBI, LL, CLOSED, EIGHT, and CROSS) were named based on the property of the intersection path in the *uv*-space. In BLI, BBII, BIBI, and LL, **B** indicates that the intersection path meets the boundary of the mesh, **I** indicates the intersection path ends in the interior of the mesh, and **L** indicates there is a loop vertex. The other categories (CLOSED, EIGHT, CROSS) were named after their shape.

If the coloring done as introduced in this section, the coloring itself contains the essential information to which direction the resolution should be performed to all cases but BLI.

Tanglements can be classified into two elementary tanglements, i.e., *vertex-triangle* and *edge-edge* tanglements, which are shown in Figs. 2a and 2b, respectively.

A segment of cloth mesh is said *intrinsically planar* if (1) it is comprised of a single panel without any seam, or (2) in the case when it is comprised by stitching multiple panels, the panels can be positioned in the *uv*-space such that there is no gap or overlap.

4 Overview

In contrast to the previous works for untangling that applied external forces, displacements ([1], [2], [3]), or constraints ([5]) in the *world space*, this work resorts to some algorithmic modification of the mesh in the *uv-space*, then let the simulator do the rest: due to the *internal* forces caused by the *uv*-space mesh modification, the world-space mesh

2. Determining the tangled regions from the partitioning (via the intersection analysis) of the given clothing configuration cannot be done *strictly*. As most non-history based DCH methods do, the proposed method assumes that majority of the clothing is in the right side (i.e., not tangled). Of course, a case can be fabricated such that majority is tangled. Often, it is not matter of right or wrong, but of the users intention. Therefore, throughout the paper, for the sake of technicality of the discussions, we will assume that the clothing mesh is not tangled to the extent the majority is tangled.

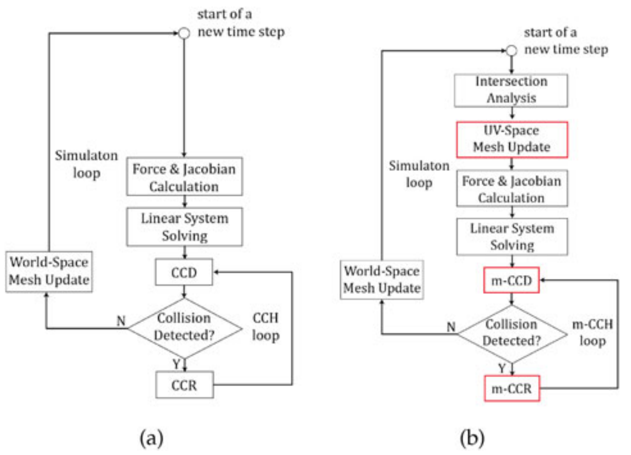

Fig. 1. Simulation flowcharts. (a) shows the conventional simulator, (b) shows how the proposed DCH method (red boxes) fits into the conventional simulator to form the new simulator.

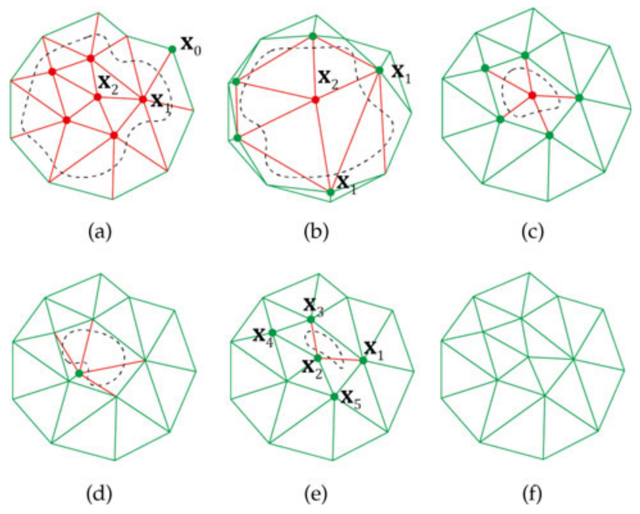

Fig. 3. Operation of the proposed method in a cloth mesh fragment. (a) to (f) chronologically show various stages of the resolution.

untangles by itself after it goes through finite iterations of the simulation loop.

This section will refer to Fig. 1b which shows how we obtain the DCH capability from the conventional simulator of Fig. 1a with full-proof CCH capability. It first performs the Intersection Analysis. Then, it performs the UV-Space Mesh Update such that the simulation based on the resultant mesh has a tendency to resolve the tanglements. As detailed in Algorithm 1, the UV-Space Mesh Update colors the mesh according to the Intersection Analysis, then updates the *uv*-space mesh with the operations *triangle shrinkage*, *vertex pull*, and *revoking* of the shrinkage and pull (Lines 3, 5 and 6). The details of those operations will be presented in Section 5.1.

Algorithm 1. UV-Space Mesh Update

```
1: Color clothing mesh based on intersection analysis
2: for all red* triangles do
3:     Apply red* triangle shrinkage by $\gamma$
4: for all out-most red vertices do
5:     Apply out-most red vertex pull by $\gamma$
6: Revoke shrinkage and pull by $1/\gamma$ where needed
```

When the UV-Space Mesh Update is done, (1) Force and Jacobian Calculation and (2) Linear System Solving (i.e., time-integration of physics by one time step) are performed. Note that the results of the Force and Jacobian Calculation are affected by the updates made in the *uv*-space mesh; the resultant world-space mesh after the Linear System Solving will tend to reflect the *uv*-space updates.

The result of Linear System Solving goes through the modified CCD (m-CCD), and if any collision is detected, the proposed simulator generates the responses with the modified CCR (m-CCR). We will call the loop consisting of CCD, branch based on whether any collision is detected, and CCR in Fig. 1a as the *CCH loop*, and the counterpart in Fig. 1b as *m-CCH loop*. We will call the main loop (that includes all the steps described above including the CCH/m-CCH) as the *simulation loop*.

Fig. 3 shows the evolution of a *uv*-space mesh fragment as the flowchart of Fig. 1b operates. In Fig. 3a, $\mathbf{x}_1$ is an out-most red vertex, while $\mathbf{x}_2$ is not. Pulling $\mathbf{x}_1$ out of the intersection path (Line 5 of of Algorithm 1) is achieved by applying the vertex pull operation (Section 5.1), which moves $\mathbf{x}_1$ towards $\mathbf{x}_0$.

If an out-most red vertex has escaped the intersection path thus colored green, the above *uv*-space mesh update must be revoked in the neighborhood of the *was-red-now-green* vertex, otherwise the resultant clothing would not be identical to the original one. Fig. 3c shows the situation when the restoration is complete for all five was-red-now-green vertices.

The above process is continued to the remaining out-most red vertices (in this case $\mathbf{x}_2$ only), which produces the situation shown in Fig. 3d, and is revoked as shown in (Fig. 3e). Then, through the intersection analysis, $T(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_5)$ and $T(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ in Fig. 3e are identified as red* triangles. For red* triangles, triangle shrinkage is applied. The resultant fully untangled mesh as shown in Fig. 3f.

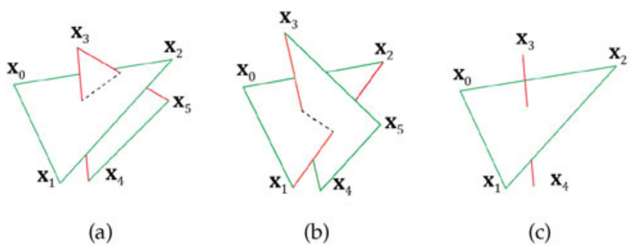

Fig. 2. Two elementary tanglements. The intersection path is shown with black dashed lines. The edges are colored according to the proposed coloring scheme. (a) vertex-triangle tanglement, (b) edge-edge tanglement. (c) is a simplified drawing of (a), which is used when the discussion focuses on the behavior of $E(\mathbf{x}_3, \mathbf{x}_4)$ with respect to the triangle $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$.

5 Modifications to Conventional Simulator

Sections 5.1 and 5.2 describe more details of the UV-Space Mesh Update (Algorithm 1) and m-CCH, respectively. Sections 5.3 and 5.4 explain how the above two components work over the the simulation loop to resolve tanglements in elementary and cloth meshes, respectively.

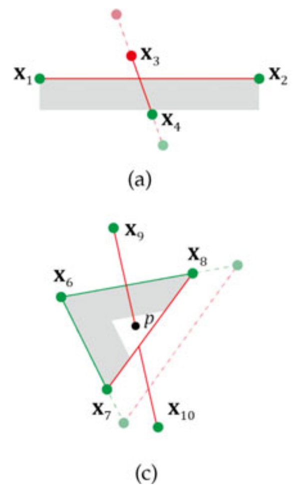

Fig. 4. (a) and (c) show the outcome in the world-space mesh (after the World-Space Mesh Update) when the vertex pull and triangle shrinkage is applied to the elementary vertex-triangle tanglement (Fig. 2c) and edge-edge tanglement (Fig. 2b), respectively. Note that (a) is shown with the cross-sectional view. (b) and (d) show the outcome when the above operations are repeatedly applied. The details of those processes will be explained in the subsequent sections.

5.1 UV-Space Mesh Update

Section 5.1.1 presents how we define the four operations (i.e., the vertex pull, triangle shrinkage, and revoke of the two), postponing the details of their implementation to Section 5.1.2. Throughout this paper, $\gamma < 1$ is a user controlled scale factor.

5.1.1 Definition of the Operations

- *Vertex Pull*: This operation can be executed to the out-most red vertex. It moves the red vertex toward the green vertex; their distance after the operation is scaled by γ. Therefore the vertex pull will be equivalently called as *edge shortening*. In the vertex pull, we call the green vertex as the *target vertex*, and the red vertex as the *subject vertex* and the edge between the two as the *subject edge*. Fig. 4a shows the edge $E(\mathbf{x}_3, \mathbf{x}_4)$ before (shown in dashed) and after (shown in solid) the vertex pull of $\mathbf{x}_3$.
- *Triangle Shrinkage*: This operation can be executed to the red* triangle as shown in Fig. 4c. It shortens the two green edges; their lengths after the operation are scaled by γ. Therefore, the triangle shrinkage operation can also be considered as edge shortening; in this case, two edges are shortened. In the triangle shrinkage, we will call the two pulled green vertices as the *subject vertices*, the third vertex as the *target vertex*, and the two edges that connect the subject and target vertices as the *subject edges*. Fig. 4c shows the triangle $T(\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8)$ before (shown in dashed) and after (shown in solid) the shrinkage.
- *Revoke of the two operations*: The revoking of the vertex pull and triangle shrinkage can be executed to an edge if that edge is not subject to the vertex pull or triangle shrinkage but its length has been changed from the original uv-length. The revoking scales the edge(s) by $1/\gamma$ (thus extends).

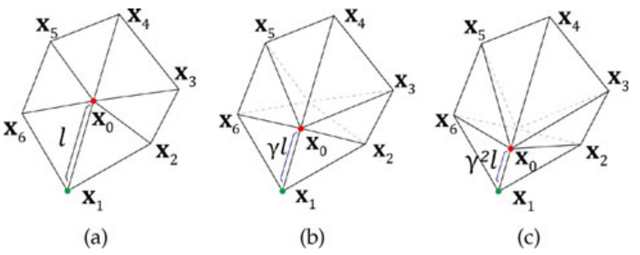

Fig. 5. Shortening of the edge $E(\mathbf{x}_0, \mathbf{x}_1)$, which has the original length *l*, toward the target vertex $\mathbf{x}_1$ in the *uv*-space. (a) the mesh before the shortening. (b) the mesh after the edge is shortened by γ-**scaling**. Note that it results in modification of the whole fan($\mathbf{x}_0$). (c) the mesh after another round of edge shortening.

5.1.2 Implementation of the Operations

For more detailed discussion on the above four operations, we define the *fan*. For a given mesh vertex $\mathbf{x}_i$, the fan of it (denoted as fan($\mathbf{x}_i$)) is defined as the set of all the vertices adjacent to $\mathbf{x}_i$ in the uv-space. For example, in Fig. 5, fan($\mathbf{x}_0$) = $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_6\}$.[3] In the context of pulling $\mathbf{x}_i$, we will call fan($\mathbf{x}_i$) as the *subject fan*. We extend the definition to an edge, thus, the fan of an edge $E(\mathbf{x}_i, \mathbf{x}_j)$ (denoted as fan ($E(\mathbf{x}_i, \mathbf{x}_j)$)) is the set of all the vertices adjacent to either of the edge vertices. For example, in Fig. 9a, fan($E(\mathbf{x}_0, \mathbf{x}_1)$) consists of all the contour vertices.

We note that, to achieve the edge shortening in the uv-space mesh, the operation should be in fact implemented as the *subject fan re-meshing*[4] rather than shortening the target edge alone. As shown in Fig. 5, while the fan contour remains the same, interior edges have to be extended or shortened to conform to the shortening of the target edge (in this case $E(\mathbf{x}_0, \mathbf{x}_1)$).

An out-most red vertex can have multiple candidate green vertices for the pull as shown in Fig. 6a. When the fan is convex, any green vertex can be chosen for the subject vertex. However, when the fan is concave, the target vertex needs to be selected after some checking, since an improperly chosen target vertex (e.g., $\mathbf{x}_2$ in Fig. 6c) can produce an *inverted triangle*.

To avoid this, we perform the *triangle inversion test* (TIT) for each candidate green vertex. TIT checks if any triangle is inverted as the subject vertex is pulled to the extremity (i.e., when the subject vertex comes to the target vertex). If so, the candidate green vertex did not pass the TIT. The test for the triangle shrinkage is done in a similar way. If a fan has at least one TIT-passable vertex, we will say the fan is *TIT-passable*.

5.2 CCH Versus m-CCH

Before explaining how we modify CCH to obtain m-CCH, we briefly review the properties of CCH. CCD examines two consecutive frames and finds out (1) the penetration time and (2) barycentric coordinate of the penetration point if there was a collision. Both of the above require arithmetic operations which are exposed to round-off errors, so [8] and [9] used some small tolerance value ε_{CCD} to prevent false negatives.

3. Note that the set does not include itself, i.e., $\mathbf{x}_0$.
4. The phrase re-meshing normally includes the change of topology as well as the change of vertex position. In this work, however, we use the phrase to mean only the change of vertex position leaving the topology the same.

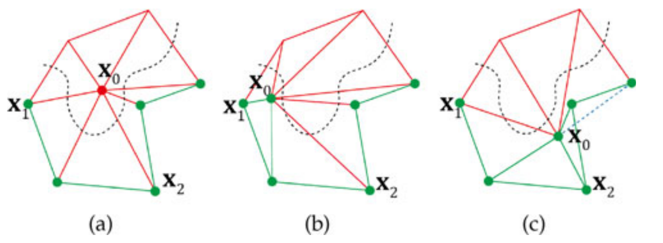

Fig. 6. Determining the target vertex for edge shortening. (a) before shortening. (b) the shortened outcome when $\mathbf{x}_1$ is chosen as the target vertex. (c) the shortened outcome when $\mathbf{x}_2$ is chosen as the target vertex, which produces an inverted triangle.

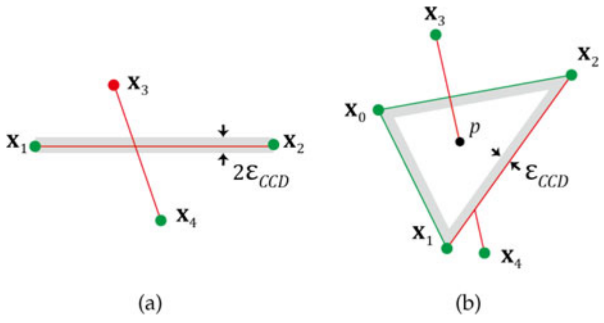

Fig. 7. The effect of ε_{CCD} after the execution of full-proof CCH. (a) and (b) show the effect of ε_{CCD} acting as the thickness that prohibits elementary pairs from approaching too close. Note that the thickness in (a) exists also in (b), and vice versa.

Some researchers used exact geometric computation [10] or Bernstein sign classification [11] that are free from floating-point errors. This paper will not consider such approaches, but will assume a conventional CCD that employs ε_{CCD}.

Until now, ε_{CCD} did not receive much attention and has been considered only as a value that needs to be small but just large enough to be distinguished from the round-off errors. In this work, we give a second thought to ε_{CCD}.

Assuming that there is no DCH in play (thus referring to Fig. 1a), if the CCH loop normally ends (i.e., not by the administrative quit) and the control exits to the World-Space Mesh Update, we note the following two: (1) The penetrating edge $E(\mathbf{x}_3, \mathbf{x}_4)$ in Fig. 7 cannot escape the triangle in any means, by crossing the triangle edges or by either of its vertices crossing the triangle $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$, and (2) ε_{CCD}, which is used to defend the round-off errors, further confines where the penetrating edge can exist. The vertices of the penetrating edge are displaced at least ε_{CCD} from the triangle, i.e., neither of $\mathbf{x}_3$ or $\mathbf{x}_4$ can come into the grey-shaded space in Fig. 7a. Also, the penetrating edge is displaced at least ε_{CCD} from the triangle contour, i.e., the penetration point p cannot come to the grey-shaded region in Fig. 7b.

We make the following two modifications to CCD and CCR to obtain m-CCD and m-CCR:

- *Allowing red-red elementary pairs to cross*: Intersection path acts as the border that distinguish red and green regions. The idea behind allowing only red-red elementary pairs to cross is to convert red vertices/edges to green by making them cross the intersection path while prohibiting the green vertices/edges becoming red. (Allowing certain elementary pairs to cross in CCH has been implicitly practiced in the previous DCH methods. For clarity, this work explicitly points out for which elementary pairs the crossing is granted.) This modification applies to CCD.
- *Controlling the value of ε_{CCD} for resolution purpose*: If it helps resolution, we propose to use a larger value than the conventional round-off defending value. We explain why using a larger ε_{CCD} can facilitate the resolution in Section 5.3. This modification (of ε_{CCD} value) applies to both CCD and CCR.

5.3 Resolution of Elementary Tanglements Over Simulation Loop

A single application of vertex pull or triangle shrinkage may not resolve the tanglement, but repeated application of them (over the simulation loop) can. The present section explains how the looping works for the elementary tanglements.

When edge shortening is applied to $E(\mathbf{x}_3, \mathbf{x}_4)$ as shown in Fig. 4a, whereas $\mathbf{x}_4$ is defended by ε_{CCD}, the defense for the red vertex $\mathbf{x}_3$ is off. Therefore, as we shorten the edge further (by repeating the simulation loop), $\mathbf{x}_3$ *has to cross* the intersecting triangle as shown in Fig. 4b thus becomes green. Note that ε_{CCD} (i.e., the grey shaded region beneath $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$) does its role for the resolution. We will call repeated application of vertex pull until the subject vertex crosses the intersecting triangle as the *VT-type ε_{CCD}-finesse*.

For the case of edge-edge tanglement, as Fig. 4c shows, since the ε_{CCD} defense is off for the red edge pair, as we shorten the two green edges further (by repeating the simulation loop), $E(\mathbf{x}_9, \mathbf{x}_{10})$ *has to cross* $E(\mathbf{x}_7, \mathbf{x}_8)$ thus both become green as shown in Fig. 4d. Note that ε_{CCD} (this time, the grey shaded area inside $T(\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8)$) does its role for the resolution. We will call repeated application of the triangle shrinkage until the subject edge $E(\mathbf{x}_7, \mathbf{x}_8)$ crosses the penetrating edge $E(\mathbf{x}_9, \mathbf{x}_{10})$ as the *EE-type ε_{CCD}-finesse*. We will call the VT-type ε_{CCD}-finesse and EE-type ε_{CCD}-finesse collectively as the *ε_{CCD}-finesse*.

5.3.1 Discussion on the Value of ε_{CCD}

In the VT-type ε_{CCD}-finesse shown in Figs. 4a and 4b, using a larger ε_{CCD} may expedite the finesse. The strategy regarding ε_{CCD} we find effective is to use different ε_{CCD} values for different elementary pairs. For the green-green elementary pairs, for normal round-off error defending, we use the original ε_{CCD}, 10^{-6}. But for red-green elementary pairs, we use a value (hereafter referred as ε_{RG}) that is larger than ε_{CCD}. In the experiments reported in this paper, for ε_{RG}, we used $0.1 * \mathrm{avg}(l)$, where $\mathrm{avg}(l)$ is the average length of the mesh edges. (Further discussion on the value of ε_{CCD} is given in Appendix D, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2020.3039566.)

Although the above discussion was made in regard to VT-type ε_{CCD}-finesse, we note that the same applies to EE-type ε_{CCD}-finesse.

5.4 Working of ε_{CCD}-Finesses in a Cloth Mesh

In this section, we extend the explanation in Section 5.3 to the cloth mesh, i.e., we explain how repeated execution of the simulation loop resolves tanglements in the cloth mesh. The following description is done referring to Fig. 8.

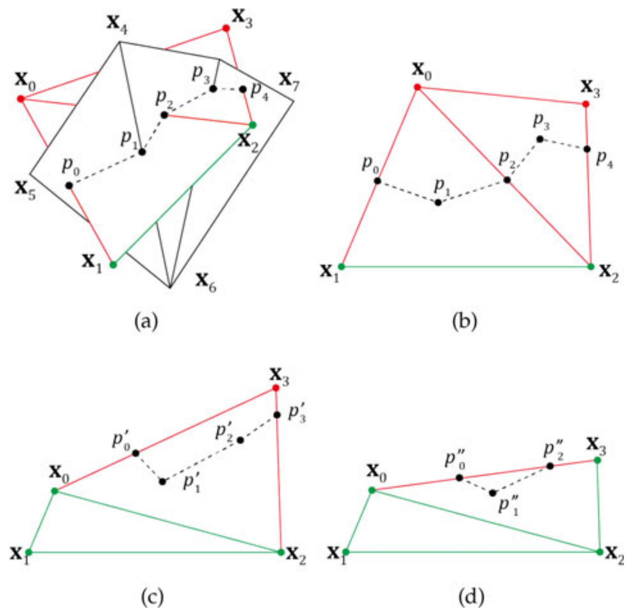

Fig. 8. Applying ε_{CCD}-finesse to a cloth mesh. Black points represent the penetration points, and black dashed lines represent the intersection paths. Coloring of the vertices and edges is done based on the scheme described in Section 3. For visual comfort, the color is not shown or shown in black for some vertices/edges. (a) shows a typical intersection between 2- and 3-triangle fans. (b) shows the same situation without visualizing the 3-triangle fan. (c) and (d) show the results of applying the VT-type ε_{CCD}-finesse to x_0 and x_3, respectively.

Although the situation shown in Fig. 8 for a cloth mesh may look complex (compared to Fig. 2), it is just concatenation of a number of elementary tanglements in a sequence. Therefore, we can use the two finesses defined for the elementary tanglements.

Three possible VT-type ε_{CCD}-finesses (i.e., pulling $\mathbf{x}_0$ to $\mathbf{x}_1$, $\mathbf{x}_0$ to $\mathbf{x}_2$, and $\mathbf{x}_3$ to $\mathbf{x}_2$, but no EE-type ε_{CCD}-finesse) can be considered in Figs. 8a and 8b. In this situation, the finesses need to be *scheduled* (which will be introduced in Section 6). For now, let's suppose that the scheduling algorithm selected $\mathbf{x}_0$-to-$\mathbf{x}_1$ finesse. The VT-type ε_{CCD}-finesse makes $\mathbf{x}_0$ cross the *intersecting triangle*$T(\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6)$, thus the situation is more clearly visualized by Fig. 8a. (We will call this as the *elementary view*.) But $\mathbf{x}_0$ crossing the intersecting triangle is equivalent to $\mathbf{x}_0$ crossing the *intersection path*, $(p_0, p_1, p_2, p_3, p_4)$, which is more clearly visualized by Fig. 8b. The latter, which we will call as the *abstract view*, is more convenient when considering the resolution in a cloth mesh. It is because the finesse can be described in terms of only the intersection path (without referring to the intersecting triangles). Note that the abstract view was already introduced in explaining Fig. 3.

By executing $\mathbf{x}_0$-to-$\mathbf{x}_1$ finesse to the configuration shown in Fig. 8b, $\mathbf{x}_0$ crosses the intersection path, the result of which is shown in Fig. 8c. (The restoration step will relocate $\mathbf{x}_0$ to its original position, but that part is omitted in the figure.) Now, in Fig. 8c, two possible VT-type ε_{CCD}-finesses (pulling $\mathbf{x}_3$ to $\mathbf{x}_2$ and $\mathbf{x}_3$ to $\mathbf{x}_0$) can be considered. Let's suppose that the scheduling algorithm selected $\mathbf{x}_3$-to-$\mathbf{x}_2$ finesse. That finesse results in Fig. 8d. Finally, all vertices are green and there is only one EE-type-finesse left. Performing that finesse will fully untangle the two- and three- triangle fans.

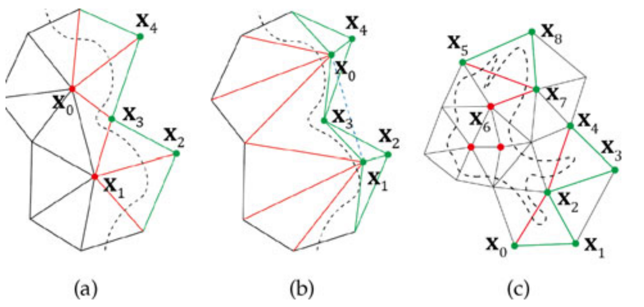

Fig. 9. Cases that call for scheduling. (a) shows an example of Case 1. (b) when the two red vertices $\mathbf{x}_0$ and $\mathbf{x}_1$ of (a) are simultaneously pulled toward $\mathbf{x}_4$ and $\mathbf{x}_2$, respectively. (c) shows examples of Cases 2 and 3.

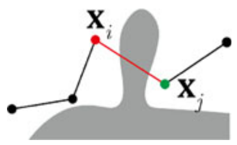

Fig. 10. An example when edge shortening is hindered by sharp body.

5.5 Possible Scenario of Edge Shortening Hindrance

One possible concern is that, even if an edge is shortened in the *uv*-space mesh, that shortening in the world-space mesh may be hindered for some reason. Then, the finesse may not proceed normally.

We note that such hindrances rarely occur in practice. UV-Space Mesh Update is a local operation; only neighboring two or three vertices are involved. The only case we can imagine in which such hindrance can occur is when a sharp body part intervenes two cloth vertices as shown in Fig. 10 the drawing on the right. We can setup the body such that the simulation does not need to consider sharp features such as fingers/toes (e.g., by wrapping caps to hands/feet). Section 10 will give an experimental report on this issue.

6 Scheduling the Operations

Note that (1) a large number of pull, shrink, and revoke operations have to be performed, and (2) the observation of their effect in the world-space is possible only after the World-Space Mesh Update. Sequential processing of them (i.e., processing only one operation per simulation loop) will be inefficient, which is why Algorithm 1 performs multiple operations at once.

Unfortunately, if we perform pulls and shrinks to all outmost red vertices and red* triangles, respectively, it can introduce inverted triangles.[5] This paper finds that we should not perform simultaneous pulls/shrinks in the following three cases, and proposes how to schedule the operations in those cases. The significance of the classification below is that we can perform multiple pulls/shrinks simultaneously as long as they do not belong to those three cases.

- *Case 1 – When two out-most red vertices are adjacent*: For example, if $E(\mathbf{x}_1, \mathbf{x}_2)$ and $E(\mathbf{x}_0, \mathbf{x}_4)$ of Fig. 9a are

5. Note that, whereas the TIT discussed in Section 5.1.2 checks the inversion within the subject fan, the inversions that are considered here are the ones when two neighboring fans are re-meshed simultaneously.

shortened simultaneously, the two fans can *interfere*, which can result in an inverted triangle as shown in Fig. 9b.[6] The above interference can be avoided if we shorten them *sequentially*, i.e., we propose that the pull should be performed to only one of them (say $E(\mathbf{x}_1, \mathbf{x}_2)$) and the other ($E(\mathbf{x}_0, \mathbf{x}_4)$) should wait for its turn, until $\mathbf{x}_1$ escapes the intersection path and restores to its original *uv*-position.

- *Case 2 – When two red* triangles are adjacent*: In Fig. 9c, the two red* triangles $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ and $T(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ share $\mathbf{x}_2$ thus adjacent. If we shrink $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$, $\mathbf{x}_2$ will move toward $\mathbf{x}_1$. But if we shrink $T(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$, $\mathbf{x}_2$ will move toward $\mathbf{x}_3$. In such a conflicting case, they need to be *sequentialized*, i.e., either $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ or $T(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ has to wait until processing of the other is complete.
- *Case 3 – When an out-most red vertex and a red* triangle are neighboring*: An example case is the out-most red vertex $\mathbf{x}_6$ and the red* triangle $T(\mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8)$ in Fig. 9c. Note that the *neighboredness* here does not mean adjacency, but means that the subject vertex of one operation (pull or shrinkage) is included in the fan of the other operation. In the current example, fan($\mathbf{x}_6$) includes $\mathbf{x}_5$ and $\mathbf{x}_7$, and fan($E(\mathbf{x}_5, \mathbf{x}_7)$) includes $\mathbf{x}_6$. Since the pull of $\mathbf{x}_6$ and shrinkage of $T(\mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8)$ can interfere each other, we propose to *sequentialize* their processing. We give more priority to red* triangles, i.e., the red* triangles ahead of the out-most red vertices.

Algorithm 2. *uv*-space Mesh Update With Scheduling

```
Input: original uv-mesh and current uv-mesh
Output: updated uv-mesh
 1: Color cloth vertices based on intersection analysis
 2: for all red* triangles $T(\mathbf{x}_i^*, \mathbf{x}_j^*, \mathbf{x}_k^*)$ do
 3:   if fan($E(\mathbf{x}_j^*, \mathbf{x}_k^*)$) is in its original uv-position then
 4:     if fan($E(\mathbf{x}_j^*, \mathbf{x}_k^*)$) passes the TIT then
 5:       Perform triangle shrinkage
 6: for all out-most red vertices $\mathbf{x}_r$ do
 7:   if fan($\mathbf{x}_r$) is in its original uv-position then
 8:     if fan($\mathbf{x}_r$) passes TIT then
 9:       Perform vertex pull
10: for all green vertices $\mathbf{x}_g$ do
11:   if $\mathbf{x}_g$ is not in its original uv-position then
12:     if $\mathbf{x}_g$ is not a subject vertex then
13:       Revoke any operations applied
```

Algorithm 2 is one possible implementation of the above sequentializations and priority. If it is replaced to Algorithm 1, the pulls/shrinks can be performed as simultaneously as possible without producing any anomalies. We find the listing lacks readability, thus we give some explanations below.

By having Lines 2∼5 at the beginning, red* triangles are processed ahead of out-most red vertices, observing the scheduling rule of Case 3. Note that $\mathbf{x}_i^*$ is the target vertex, $\mathbf{x}_j^*$ and $\mathbf{x}_k^*$ are the subject vertices. Line 3 checks if its fan ($E(\mathbf{x}_j^*, \mathbf{x}_k^*)$) is not interfered by other triangle shrinkage. By Line 4, if the fan is not TIT-passable, it yields the turn to the next red* triangle. The shrinkage in Line 5 is made to the current *uv*-mesh *immediately* so that the check made in Line 3 is valid. We note that the for-loop in Line 2 of Algorithm 2 should not be parallelized, so that, if the shrinkage of a red* triangle already started (by Line 5), for an adjacent red* triangle, its fan is not in the original *uv*-position. So the shrinkage of that triangle is skipped by Line 3, observing the scheduling rule of Case 2.

Similarly, the for-loops in Lines 6 and 10 should not be parallelized. Therefore Line 7 ensures that an out-most red vertex is not processed if it is adjacent to another out-most red vertex or is neighboring to a red* triangle, observing the scheduling rule of Case 1 and Case 3. Line 8 is to yield to the next TIT-passable out-most red vertex. Finally, Lines 10∼13 are for restoring the green vertices that used to be red.

6.1 Possible Scenarios When No Fan is TIT-Passable

We consider the exceptional cases in which no fan is TIT-passable.

- Coloring is performed by examining *all* existing intersection paths. A vertex that was green after examining an intersection path can become red after examining another intersection path. In an extreme case, the entire mesh may turn out red, leaving no green vertex to role as the target vertex. But it corresponds to the case in which the majority is tangled, which we assume does not occur (see Section 1).
- In running Algorithm 2, we can imagine a case in which no TIT-passable fan exists. In that case, the proposed DCH has to halt. A heuristic procedure to evade such a case is described in Appendix C, available in the online supplemental material. However, this case is very rare in clothing meshes. For this case to occur, the original *uv*-space triangulation should be very irregular (e.g., the star-shaped triangulation diagrammed in Appendix C, available in the online supplemental material). We have never met such a case so far.

7 Soundness in Intrinsically Planar Cases

In this section, we show that the proposed DCH method is capable of resolving tanglements in finite time steps in intrinsically planar cases. The scheduling algorithm described in Section 6 enables multiple finesses to run without interfering each other. Proving that the algorithm is interference-free can be tedious, thus this paper will take it without proof. Then, we can show the convergence of the proposed DCH method.

Theorem. *The following is in regard to the given intrinsically planar cloth mesh. For any given tanglement except for BLI, if it is colored according to Section 3, the proposed DCH method resolves the tanglement within a finite number of time steps under the following two conditions: throughout the DCH resolution steps, (i) the edge shortening is not hindered (Section 5.5), and* (ii) *there is at least one TIT-passable fan (Section 6.1).*

6. Note that if two subject vertices are not adjacent, the corresponding subject fans do not overlap, thus do not interfere each other.

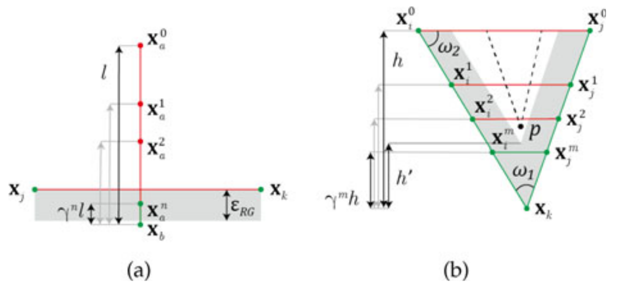

Fig. 11. ε_{CCD}-finesse to vertex-triangle tanglement and edge-edge tanglement. (a) VT-type ε_{CCD}-finesse. (b) EE-type ε_{CCD}-finesse, in which the black dashed lines represent the intersection path.

Proof. As discussed in Sections 5.5 and 6.1, it is difficult to encounter a case in which the two conditions of this theorem are not met. In order to prove the theorem, it suffices to show that (1) during the resolution, red elements do not increase, and (2) each finesse terminates in finite time steps.

We prove the first part by showing that the red vertices, edges, triangles do not increase. With the proposed modifications to CCH, m-CCH encourages *exiting* red vertices/edges, but it does not necessarily prevent *entering* red vertices/edges. To show the entering red vertices do not increase in the total number, in Fig. 4a, imagine an additional red vertex x_{new} which was originally below $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$, after Linear System Solving, comes above $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$. According to its definition, m-CCH does not prevent that penetration, since both x_{new} and $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ are red. But note that the total number of red vertices does not increase, since that vertex was red before the penetration.[7] A similar argument can be made for the red edges using Fig. 4c to show that the total number does not increase. By definition of the red triangle, since a green vertex/triangle can never become red (the property of the full-proof CCH), the number of red triangles does not increase.

Now we prove the second part. Let's suppose that the VT-type ε_{CCD}-finesse does not successfully pull an outmost red vertex out of the intersection path in finite time steps. Fig. 11a shows the situation, in which $\mathbf{x}_a$ is the outmost red vertex and $\mathbf{x}_b$ is the target vertex, and $E(\mathbf{x}_a, \mathbf{x}_b)$ with the original length l is penetrating $T(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$. The superscript z in $\mathbf{x}_a^z$ represents the number of vertex pull operations applied.

Since m-CCH prevents $\mathbf{x}_b$ from entering the grey shaded area, by repeated application of pull by $\gamma < 1$, $\mathbf{x}_a$ has to move toward $\mathbf{x}_b$ due to the shortened edge length $\gamma^n l$. Since ε_{RG} is a constant, there exists an integer n such that after n time steps,

$$\gamma^n l < \varepsilon_{RG}, \tag{1}$$

holds. At this point, it is certain that $\mathbf{x}_a$ has crossed the triangle (thus the intersection path), which is a contradiction to the original assumption.

Let's suppose that the EE-type ε_{CCD}-finesse does not successfully push the intersection path out of a red* triangle in finite time steps. Fig. 11b shows a red* triangle $T(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ with the original height h, in which $E(\mathbf{x}_i, \mathbf{x}_j)$ is the red edge. The superscript z in $\mathbf{x}_i^z$ and $\mathbf{x}_j^z$ represents the number of red* triangle shrinks applied.

By repeated application of triangle shrinkage by $\gamma < 1$, $\mathbf{x}_i$ and $\mathbf{x}_j$ have to move toward $\mathbf{x}_k$ due to the shortened height $\gamma^m h$. Since ε_{RG} is a constant, there exists an integer m such that after m time steps

$$\gamma^m h < h'\left(= \frac{sin(0.5\omega_1 + \omega_2)}{sin(0.5\omega_1)}\varepsilon_{RG}\right), \tag{2}$$

holds, where the right hand side is the vertical distance to the bottom corner of the shaded region measured from $\mathbf{x}_k$ thus has a finite value. Since m-CCH prevents any intersection path point p from entering the grey shaded area, at this point, it is certain that the whole intersection path has crossed the red edge, which is a contradiction to the original assumption.

Combining the two parts, we can conclude that the proposed method monotonically resolves red elements, each in finite time steps, until all red elements become green thus the tanglements are completely resolved. □

7. In addition to the total number, we note that, by m-CCH, a red vertex can remain red or become green, but a green vertex can never become red.

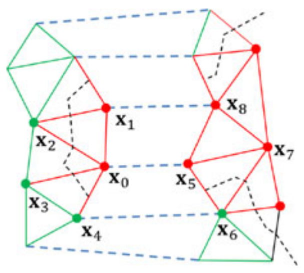

Fig. 12. An example in which the intersection path exists across multiple panels.

8 Extensions to Process Clothing

For a pull/shrinkage, when the enclosing fan exists within a single panel, the subject fan re-meshing is well-defined. As for intrinsically planar cases, even if the enclosing fan does not exist within a single panel, we can straightforwardly extend the subject fan re-meshing to work across multiple panels. However, the fan re-meshing can be problematic for intrinsically non-planar cases.

Since a garment is constructed by seaming multiple panels, the resultant mesh is in general intrinsically non-planar and the intersection path can lie over multiple panels as shown with the black dashed lines in Fig. 12, in which the blue dashed lines label the vertices being stitched, and the coloring of the vertices and edges reflects the result of intersection analysis performed in world-space.

In pulling out $\mathbf{x}_0$ in Fig. 12, suppose that $\mathbf{x}_2$ has been chosen as the target vertex. Note that $\mathbf{x}_0$ and $\mathbf{x}_5$ are identical in the world space mesh (and so are $\mathbf{x}_1$ and $\mathbf{x}_8$, $\mathbf{x}_4$ and $\mathbf{x}_6$). For shortening $E(\mathbf{x}_0, \mathbf{x}_2)$, performing the fan re-meshing only in the left panel will leave $\mathbf{x}_5$ at its original position in the right panel, thus $T(\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7)$ and $T(\mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8)$ can hinder the shortening of $E(\mathbf{x}_0, \mathbf{x}_2)$ in the world space simulation. This is violation of the condition (*i*) of the theorem. To be able to use the proposed DCH method to clothing simulation, therefore, the subject fan re-meshing operation has to be extended to cover non-planar cases.

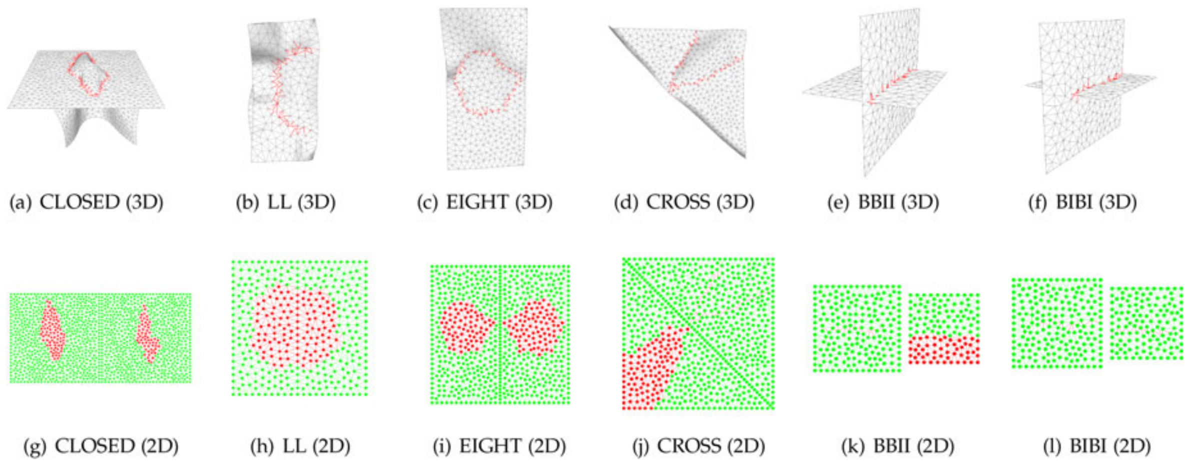

Fig. 13. Six rudimentary cases. In 3D, every edge that penetrates a triangle is shown in red as in Figs. 2a and 2b, which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.

For fan($\mathbf{x}_0$) = $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ and fan($\mathbf{x}_5$) = $\{\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\}$ in Fig. 12, it is clear that not only fan($\mathbf{x}_0$) but fan($\mathbf{x}_5$) also needs to be considered in shortening $E(\mathbf{x}_0, \mathbf{x}_2)$. The difficulty here is that, although the two panels are seamed in world-space, they exist as two separate *uv*-space entities; The coordinate systems associated with the left and right panels can have different origins and orientations. For example, we cannot expect the *uv*-coordinates of $\mathbf{x}_0$ and $\mathbf{x}_5$ to be the same.

In the above circumstance, we propose that (1) the fan re-meshing should be performed separately for each panel (i.e., the re-meshing should be performed to fan($\mathbf{x}_0$) and fan($\mathbf{x}_5$) separately), but (2) the two fan re-meshings should be done in coordination such that shortening of $E(\mathbf{x}_0, \mathbf{x}_2)$ is not hindered by fan($\mathbf{x}_5$) during the simulation. We put the detailed procedure in Appendix A, available in the online supplemental material.[8]

The procedures presented in Appendices A and B are developed such that they resemble the handling of intrinsically planar cases as much as possible. For example, if the procedures are applied to an intrinsically planar mesh, they will produce identical result with the one produced with the algorithms presented up to Section 7. Here, we note that the extensions in Appendices A and B, available in the online supplemental material, are *heuristic* methods, for which this paper does not attempt to show theoretical convergence. Experiments in the results section report that the proposed heuristic methods work quite well.

9 Results

We implemented the proposed DCH method on the conventional simulator that was built based on the combination of the prior simulation techniques ([8], [12], [13]) and the CCD acceleration methods ([14], [15], [16], [17], [18]).

In terms of the classification of tanglements in [2], the proposed method is supposed to handle six out of seven cases (i.e., all the cases except for BLI) if the required conditions of the theorem are met. Among the six covered cases, the first five cases (CLOSED, LL, EIGHT, CROSS, BBII) divide the mesh such that the red vertices are identifiable, and the remaining case (BIBI) produces only red* triangles.

9.1 Rudimentary Cases

We setup a handkerchief (or two handkerchiefs) to produce the above six cases as shown in Fig. 13. In the simulation, to facilitate the observation, the gravity was set to zero and a few vertices were constrained to stay at their initial position. As the accompanying video shows, the proposed method resolved the six cases successfully.

9.2 Exploded Handkerchief

In this experiment, a handkerchief consisting of 388 triangles was set up to be in a tangled state. After positioning the handkerchief such that its center comes to the origin, each vertex of the handkerchief was re-positioned to $\mathbf{z} = (r, \phi, \psi)$ in spherical coordinates, where $\mathbf{z}$ was randomly chosen from $[0, r_0] \times [0, 2\pi) \times [-\pi, \pi]$ for some constant r_0. We confined re-positioning to the central area to avoid the BLI cases occurring.

Intersection analysis performed at the first frame resulted in 7,801 intersection path fragments (intersection between two red triangles) across 163 red triangles. In average, one red triangle intersected with 47 other red triangles. The proposed method resolved tanglements without any failure (Fig. 14b) in 75 time steps.

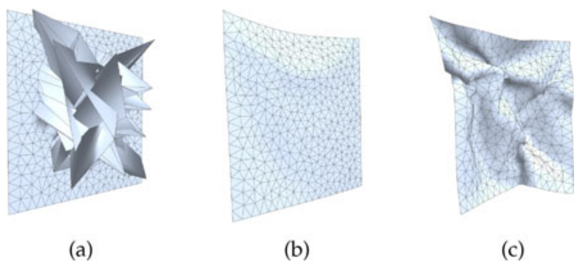

Fig. 14. Exploded handkerchief. (a) initial tanglement, (b) after running the proposed method 75 time steps, (c) after running PREV-DCHs additional 2,000 time steps after the first 75 time steps.

8. The two seamed edges can share a common vertex in the *uv*-space (e.g., in a dart), in which case, the whole panel is represented in a single *uv*-coordinate system. This case has to be handled differently from the method described in Appendix A, available in the online supplemental material, and is presented in Appendix B, available in the online supplemental material

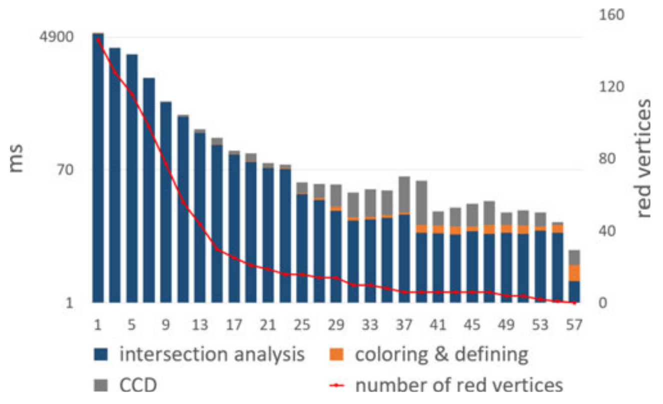

Fig. 15. Plotting of the resolution time and red vertex count. The bar graph represents the time taken for the resolution in log scale, and the red curve represents the number of remaining red vertices.

The same handkerchief was processed by a DCH module which was obtained by combining GIA ([1]), UIR ([5]), and ICM ([3], [4]) with necessary modifications to fit to force-based simulation. We will hereafter call that module as *PREV-DCHs*. With PREV-DCHs, three LL cases were left unresolved even after 2000 time steps as shown in Fig. 14c and the accompanying video. We can attribute the above failures to (1) GIA and UIR do not take any actions for the LL cases, and (2) ICM may not produce correct/consistent vertex movement direction for the resolution.

Fig. 15 shows how the proposed method performed in resolving the scene shown in Fig. 14. The x-axis represents the frame number. Fig. 15 shows two measurements at the same time, i.e., (1) the bar graph represents the time taken in milliseconds for the collision resolution at each odd frame, and (2) the red curve represents the number of remaining red vertices at that frame. It shows that the proposed method monotonically reduces the number of red vertices, which can be expected throughout this paper.

In all experiments reported in this paper, we used $\Delta t = 0.02s$ and $\gamma = 0.8$. Obviously, a smaller γ will expedite the finesse. However, when γ is too small, the force-based simulator can become unstable. More detailed discussion about the choice of the γ value is placed in Appendix D, available in the online supplemental material.

The exploded handkerchief is a manipulated experiment to fit to the theoretical range of the proposed method. Whereas the prior works may or may not be able to handle the exploded handkerchief (depending on the given handkerchief mesh configuration), the proposed method guarantees to fully untangle the handkerchief no matter how

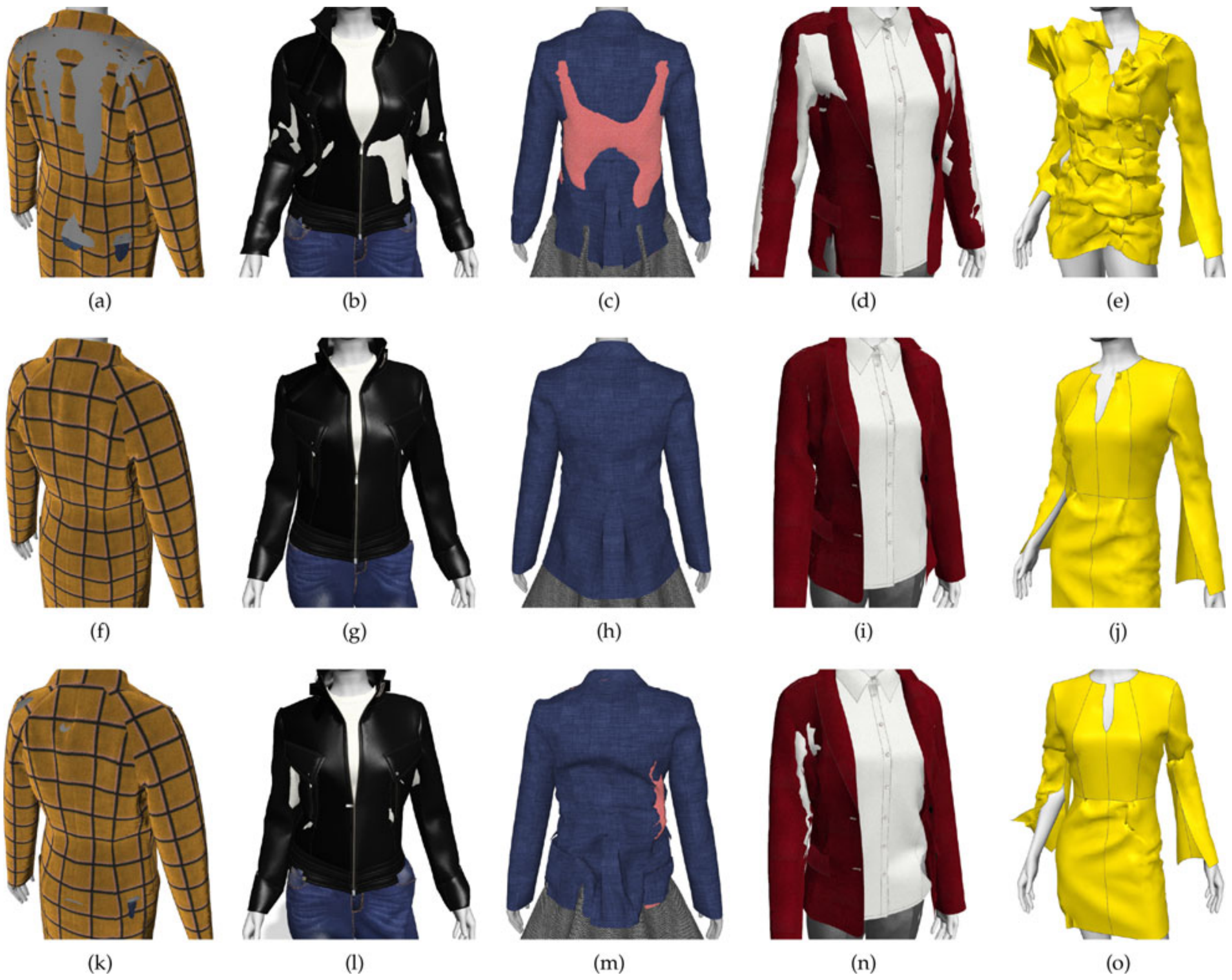

Fig. 16. Experiments on clothing cases. (a)~(d) show the initial tanglements. (f)~(i) show the results after executing the proposed method to (a)~(d), respectively. (k)~(n) show the results after executing PREV-DCHs to (a)~(d), respectively. (e) Is a capture during a session of interactive tanglement generation and resolution. (j) and (o) show the results after 500 times steps of simulation with the proposed method and PREV-DCHs, respectively.

TABLE 1
#(*), %(*) and t(*) Represent the Occurrences of Each Case, Percentage, and Average Number of Time Steps Required, Respectively

Scene	vertex type	#(*)	%(*)	t(*)
Fig. 15a	OffSeam	571	92.24	6.10
	OnSeam-A	42	6.79	7.86
	OnSeam-B	6	0.97	6.30
Fig. 15b	OffSeam	1390	91.27	9.88
	OnSeam-A	121	7.94	10.57
	OnSeam-B	12	0.79	11.01
Fig. 15c	OffSeam	898	90.25	7.53
	OnSeam-A	75	7.54	7.10
	OnSeam-B	22	2.21	6.90
Fig. 15d	OffSeam	1,451	91.09	6.84
	OnSeam-A	68	4.27	7.02
	OnSeam-B	74	5.64	6.59
Fig. 15e	OffSeam	2,192	82.81	5.76
	OnSeam-A	357	13.49	5.90
	OnSeam-B	98	3.70	5.11

complicated the tanglements are as long as it contains no BLI case. We experimented tens of other handkerchief explosions, and the proposed method resolved them without failure. By observing the proposed method resolving this case, which is the straight implementation of the theorem, we reassure that the theory works in practice.

9.3 Clothes

Tanglements in four ensembles shown in Figs. 16a, 16b, 16c, and 16d were created by deliberately disabling self-collision handling for a few time steps. BLIs could be introduced, which were manually removed before running the proposed method and PREV-DCHs. As shown in Figs. 16f, 16g, 16h, and 16i, the proposed method resolved all cases successfully. With PREV-DCHs, however, there were resolution failures in all four cases, as shown in Figs. 16k, 16l, 16m, and 16n, even after we ran 1,000 additional time steps compared to those required for Figs. 16f, 16g, 16h, and 16i, respectively.

Fig. 16e is a single garment case that consists of 4,183 vertices, for which the user interactively generated tanglements by dragging the vertices while the resolution is running. As shown in Fig. 16j, except for a single BLI case at the corner of the front opening (which is out of scope of the proposed method), the resolution of the proposed method was successful, at the rate of 1.56 fps in average. The accompanying video contains the capture of the interactive session. With PREV-DCHs, however, some non-BLI cases were left unresolved as shown in Fig. 16o.[9]

In the clothing examples, the failure included CLOSED, BBII, and BIBI cases, as well as LL. Since the methods in PREV-DCHs have upper bounds in the force/displacement they can apply for the resolution, when there exist a tanglement that calls for a larger force/displacement, the resolution can fail. The above implies that the failure of PREV-DCHs is not necessarily limited to {LL, CLOSED, BBII, BIBI}.[10] Those failures do not occur in the proposed DCH method, since the method applies the pull and shrinkage limitlessly until they place the tangled vertex/edge to the other side of the intersection path.

Handling the clothing cases needed the heuristic methods presented in Section 8. Let *OffSeam* represent the intrinsically planar edge shortening type thus can be covered by the theorem, and *OnSeam-A* and *OnSeam-B* represent the edge shortening types in which the subject vertices lie on the seam thus should be processed by the procedures described in Appendices A and B, respectively. Table 1 summarizes the statistics for the three edge shortening types. In the table, #(*) and %(*) represent the total number and percentage, respectively, of the types, which were collected throughout the simulation of the above ensembles. There was no frame at which condition (*ii*) of the theorem was not met, i.e., at every time step, there existed at least one TIT-passable fan thus the convexification of Appendix C, available in the online supplemental material, was never needed.

A curiosity here is whether handling of the non-planar cases (i.e., OnSeam-A and OnSeam-B, which we will collectively denote as *OnSeam*) is as good as that of the intrinsically planar case OffSeam. One indicative measure would be the number of time steps required for completing the ε_{CCD}-finesse from the moment it is initiated. Let's denote the average number of time steps for completing the finesse of each type with t(*). As the last column of Table 1 shows, OnSeam took only about 1.76 additional time step at most compared to OffSeam. It experimentally indicates that the procedures proposed for OnSeam emulated that for OffSeam quite well.

10 Conclusion

In this paper, we proposed a new DCH method based on the existing CCH method. The proposed method performs intersection analysis of the clothing mesh in every time step, and stores the result in the form of coloring the vertices, edges, and triangles. Referring to the coloring, the method resolves tanglements in an out-to-in manner by applying the proposed operations, i.e., triangle shrinkage and vertex pull, to the triangles and vertices around the intersection path.

A key idea of this work was that ε_{CCD} can be utilized for the purpose of tanglement resolution, based on which we proposed a new DCH method which uses different ε_{CCD} values to red-red, red-green elementary pairs.

According to the categorization of tanglement cases by [2], the proposed method could cover six out of seven cases. BLI is the case in which we cannot determine the majority side. Although [2] proposed a heuristic method to identify the red vertices for that case, there can exist some tanglements the method cannot cover. A method to process the BLI case more stably needs to be developed in the future.

9. In this test, in which new tanglments were interactively generated while some were being resolved, we could not run the proposed method and PREV-DCHs in the strictly same condition. However, we note that, as we perform the test multiple times, if we exclude BLI, the proposed method was always successful while PREV-DCHs was not.

10. If experimented in different conditions, PREV-DCHs on the exploding handkerchief case may also have produced CLOSED, BBII, BIBI (and possibly other new) failures.

The proposed method can be considered as *peeling* the tangled region. Based on the shape of the tangled region, it may require several rounds of peeling. Compared with the previous DCH methods which tries to resolve the whole tanglements at once, our method can be slower. But we would like to note that the method does not require impractically large amount of computation. In typical clothing setup of Figs. 16a, 16b, 16c, and 16d, it took 161 time steps in average for the complete resolution.

The theorem holds for intrinsically planar meshes. Additionally, this paper developed two heuristic procedures to extend the proposed method to non-planar cases, which turned out to work quite well. We have never encountered a case in which condition (ii) of the theorem is not met. As for condition (i), we are not sure whether there was any temporary violation. But as we judge from the resolved results, even in a hazardously tangled case, there seemed to be no prolonged violation of it.

The proposed method can be obtained by applying the following very simple augmentation/modification to the conventional simulator.

- Re-meshing in the *uv*-space: The two operations (i.e., the pull and shrinkage) are achieved by re-meshing a region of the clothing mesh in the *uv*-space and leaving the rest to the simulator.
- CCH to m-CCH modification: The m-CCH is obtained by applying a very simple modification to the existing CCH.

In summary, the proposed method guarantees resolution of non-BLI tanglements for intrinsically planar cases in finite time steps, when shortening operation is not hindered and when there is at least one TIT-passable fan. As we experimented hundreds of other clothing cases, as long as the cases do not include BLI tanglements, we report that the proposed method always resolved the tanglements. Therefore, although the convergence has not been proven when the proposed heuristic methods are used, according to the experiments, we reveal that practically the proposed method converges even for the clothing cases.

Acknowledgments

This work was supported by R&D program for Advanced Integrated-intelligence for IDentification (AIID) through the National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT under Grant 2018M3E3A1057288, the Brain of Korea 21 Project in 2020, and ASRI (Automation and Systems Research Institute at Seoul National University).

References

[1] D. Baraff, A. Witkin, and M. Kass, "Untangling cloth," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 862–870, 2003.
[2] M. Wicke, H. Lanker, and M. Gross, "Untangling cloth with boundaries," in *Proc. 11th Int. Workshop Vis. Model. Vis.*, 2006, pp. 349–356.
[3] P. Volino and N. Magnenat-Thalmann, "Resolving surface collisions through intersection contour minimization," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1154–1159, 2006.
[4] J. Ye and J. Zhao, "The intersection contour minimization method for untangling oriented deformable surfaces," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2012, pp. 311–316.
[5] J. Ye *et al.*, "A unified cloth untangling framework through discrete collision detection," *Computer Graphics Forum*, vol. 36, pp. 217–228, 2017.
[6] J. Ye, "History-free collision response for deformable surfaces," 2014, *arXiv:1409.2081*.
[7] T. Buffet, D. Rohmer, L. Barthe, L. Boissieux, and M.-P. Cani, "Implicit untangling: A robust solution for modeling layered clothing," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, 2019.
[8] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graphi.*, vol. 21, no. 3, pp. 594–603, 2002.
[9] Z. Wang, M. Tang, R. Tong, and D. Manocha, "TightCCD: Efficient and robust continuous collision detection using tight error bounds," in *Computer Graphics Forum*, vol. 34, Hoboken, NJ, USA: Wiley, 2015, pp. 289–298.
[10] T. Brochu, E. Edwards, and R. Bridson, "Efficient geometrically exact continuous collision detection," *ACM Trans. Graph.*, vol. 31, no. 4, 2012, Art. no. 96.
[11] M. Tang, R. Tong, Z. Wang, and D. Manocha, "Fast and exact continuous collision detection with bernstein sign classification," *ACM Trans. Graph.*, vol. 33, no. 6, 2014, Art. no. 186.
[12] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Techn.*, 1998, pp. 43–54.
[13] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Trans. Graph.*, vol. 21, pp. 604–611, 2002.
[14] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garments," in *Computer Animation and Simulation*. Berlin, Germany: Springer, 1997, pp. 177–189.
[15] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "ICCD: Interactive continuous collision detection between deformable models using connectivity-based culling," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 4, pp. 544–557, Jul./Aug. 2009.
[16] S. Curtis, R. Tamstorf, and D. Manocha, "Fast collision detection for deformable models using representative-triangles," in *Proc. Symp. Interactive 3D Graph. Games*, 2008, pp. 61–69.
[17] M. Tang, D. Manocha, and R. Tong, "Fast continuous collision detection using deforming non-penetration filters," in *Proc. ACM SIGGRAPH Symp. Interactive 3D Graph. Games*, 2010, pp. 7–13.
[18] T. Wang, Z. Liu, M. Tang, R. Tong, and D. Manocha, "Efficient and reliable self-collision culling using unprojected normal cones," in *Computer Graphics Forum*. Hoboken, NJ, USA: Wiley, 2017.

Ick-Hoon Cha received the BS degree from the School of Electrical Engineering, Korea University. He is currently working toward the combined master's and doctorate degrees with the Department of Electrical and Computer Engineering, Seoul National University. His research interest includes computer graphics, clothing simulation and parallel processing.

Hyeong-Seok Ko received the BS and MS degrees in computer science from Seoul National University (SNU), South Korea, in 1985 and 1987, respectively, and the PhD degree in computer science from the University of Pennsylvania, in 1994. He is currently a professor with the Department of Electrical and Computer Engineering, SNU, where he has been working since 1996. His recent research interests include clothing simulation and rendering.