



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학박사학위논문

Convergence Guaranteed Untangling for Clothing Simulation

의상 시뮬레이션을 위한 수렴 보장 풀림 방법

2022년 2월

서울대학교 대학원

전기·정보 공학부

차 익 훈

Convergence Guaranteed Untangling for Clothing Simulation

의상 시뮬레이션을 위한 수렴 보장 풀림 방법

지도교수 : 고 형 석

이 논문을 공학박사 학위논문으로 제출함

2021년 11월

서울대학교 대학원

전기·정보 공학부

차 익 훈

차익훈의 공학박사 학위논문을 인준함

2022년 2월

위원장	최진영 (인)
부위원장	고형석 (인)
위원	조남익 (인)
위원	김영민 (인)
위원	채영호 (인)

Abstract

Convergence Guaranteed Untangling for Clothing Simulation

Ick-Hoon Cha

Department of Electrical and Computer Engineering

The Graduate School

Seoul National University

For decades, methods have been proposed to solve the failure of self-collision (intersection) that occurs during clothing simulation. But when applied in reality, they report failure in various cases. In this paper, we divide these intersections into two groups and propose a new discrete collision handling (DCH) method for each to solve them properly in real situations.

The first method, **Edge-Shortening / Epsilon-Finessing (ESEF)**, is a method that guarantees convergence of six among seven intersection classifications except for BLI. It 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 tangles in an out-to-in manner by applying the proposed operations, **triangle shrinkage** and **vertex pull**. The operations reinterpret the traditional use of tolerance value in continuous collision handling (CCH) methods, which were normally used for defending round-off errors. It gives a second thought to that tolerance

value, and proposes a new DCH method that uses the tolerance value for the resolution purpose. Under certain conditions, ESEF turns out to guarantee the resolution of the tangles in a finite number of time steps.

The second method, **BLI-Resolver**, specifically targets BLI only. We analyze how BLIs occur, and realize that the desired form of resolution (i.e., resolution style) can vary depending on the type or particular region of the garment. Therefore, we identify the need for three resolution algorithms for BLI, namely, **Mesh-Tearing**, **Regional-Flip**, **Crease-Flip**, in order to cover the resolution styles. BLI-Resolver is the first to (1) identify the need for the resolution styles for the case of BLI, (2) propose the actual algorithms to cover each resolution style, and (3) demonstrate that the proposed resolution styles and algorithms work stably for BLIs.

With the two methods, we can now cover the full spectrum of intersections. Intersections are guaranteed to resolve, in a design-appropriate direction when sufficient information of the clothing is given. Experiments report success in various and practical clothing where previous methods failed to resolve.

Keywords: Clothing Simulation, Discrete Collision Handling, Intersection, Tanglement, Untangling

Student Number: 2014-21691

Contents

Abstract	i
Contents	iii
List of Figures	vii
List of Tables	xvii
1 Introduction	1
2 Related Works	5
2.1 Cloth Untangling: General	5
2.2 Cloth Untangling: Multi-Garment	7
2.3 Summary and Limitations	8
2.4 Contribution of Proposed Work	12
3 Preliminary	17
3.1 Edge-Shortening / Epsilon-finessing	17
3.2 Boundary-Loop-Interior Resolver	20

3.2.1	Repulsive-ICM on BLI	20
3.2.2	New Approach for BLI	23
4	Edge Shortening / Epsilon Finessing	25
4.1	Overview	25
4.2	Modifications to Conventional Simulator	28
4.2.1	UV-Space Mesh Update	28
4.2.2	CCH vs. m-CCH	33
4.2.3	Resolution of Elementary Tanglements over Simulation Loop	35
4.2.4	Working of ϵ_{CCD} -Finesses in a Cloth Mesh	36
4.2.5	Possible Scenario of Edge Shortening Hindrance	39
4.3	Scheduling the Operations	40
4.3.1	Possible Scenarios when No Fan is TIT-Passable	44
4.4	Soundness in Intrinsically Planar Cases	44
4.5	Extensions to Process Clothing	48
5	Boundary-Loop-Interior Resolver	51
5.1	Overview	51
5.2	Modifications to Conventional Simulator	52
5.3	Mesh-Tearing	52
5.3.1	L-to-B Propagation	55
5.3.2	Revived Triangles	56
5.4	Regional-Flip	59

5.4.1	Crease-Flip	60
5.5	Selecting Resolution Style/Algorithm	62
6	Experiment Results	65
6.1	Overview	65
6.2	Rudimentary Cases	69
6.3	Exploded Handkerchief	69
6.4	Clothes	73
6.5	Round Folds	78
6.6	Sharp Folds	78
6.7	User Interactions	78
6.8	Exploded Handkerchief	80
7	Conclusion	85
A	Edge Shortening When Intersection Path Exists Across Multiple Panels	89
B	Edge Shortening When Intersection Path Exists Across the Dart Opening	95
C	Convexification	97
D	Discussion on the Values of ϵ_{RG} and γ	99
E	Details of BLI Coupling for Regional-Flip	103

Bibliography	105
초 록	119

List of Figures

2.1	Coverage of previous methods in terms of categorization made in Wicke et al. [71]. The table is prepared based on the coverage explicitly stated in each paper or the theoretical limit of their methods. In the table, a circle means a complete (or theoretical) solution, a triangle means an experimentally shown partial solution, and a minus means that it is out of scope.	13
2.2	Simulation flowcharts. (a) shows the conventional simulator, (b) shows how the two proposed DCH methods (red boxes) fits into the conventional simulator to form the new simulator. . . .	15
3.1	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 $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$	20

3.2	A BLI intersection. (a) shows it in 3D. (b) shows intersection analysis of (a) in 2D. Note that \mathbf{B} and \mathbf{I} coincide in 3D. . . .	21
3.3	Diagrams taken from [75]. Two red triangles that share \mathbf{E} are intersecting the green plane. (b) is the horizontal view of (a).	21
3.4	The behavior of the Repulsive-ICM for a cone-shaped BLI. (b) is the side view of (a).	23
4.1	Operation of the proposed method in a cloth mesh fragment. (a) to (f) chronologically show various stages of the resolution.	26
4.2	(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 (Figure 3.1(c)) and edge-edge tanglement (Figure 3.1(b)) , 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.	29
4.3	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. .	31

4.4	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. . . .	32
4.5	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.	34
4.6	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.	38
4.7	An example when edge shortening is caught between a sharp feature of a body.	39
4.8	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.	41

4.9	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. . .	46
4.10	An example in which the intersection path exists across multiple panels.	48
5.1	Operation of Mesh-Tearing. (a)~(d) chronologically shows how it works. (e) is a diagrammatic depiction of (a) in 2D, where the black dashed line is the intersection path, and the sky blue triangles are the omitted triangles, which enclose the intersection path from L to B. The green region in (f) shows where the bending stiffness is increased, which facilitates the loop vertex to slide out to B. In fact, (a) is identical to the third case in Figure 9 of [64].	53
5.2	Operation of Mesh-Tearing in 2D. (a)~(c) are the 2D version of Figure 5.1(a) ~ 5.1(c). The color indicates the winding direction, and the color thickness indicates the bending curvature. In (e), the revived triangle is shown in orange. Take a note of where the new loop vertex comes.	55

5.3	A simplified flowchart of the simulator. From the start of a new time step, the Intersection Analysis module reports intersections, and applies necessary measures to resolve them. Then, the simulator advances by Δt , i.e., calculates the mesh accounting for the time flow. The History-Based Collision Handling module verifies if the resultant mesh is valid, and if needed, resolves collisions. The Intersection Analysis module works with un-omitted mesh, then based on the BLI intersections present, it generates the omitted mesh for the History-Based Collision Handling module.	57
5.4	BLIs being resolved by Regional-Flip. (a) shows two BLIs on a round fold. Assuming that the grey-side cloth should be above the red-side cloth, (b) shows the situation in 2D, where purple and blue colored triangles are those that need to be switched by the proposed method. (c) shows possible resultant intersections as the BLIs are approaching each other. (d) shows another possible resultant intersection.	58

5.5	BLI being resolved by Crease-Flip. (a) shows the situation in 3D. (b) shows the same situation in 2D; notice that compared to Regional-Flip, the fold is significantly sharper. The horizontal lines in (b)~(d) represent the fold; the lines are colored in green if the current angle is close to the rest angle, and in red if not. (c) and (d) show a possible progression after (b), in which the BLI is converted into BIBI.	61
5.6	Shirts with a collar on a transparent avatar. (a) shows the initial situation in which multiple BLIs exist, (b) BLIs are all resolved with Regional-Flip, (c) BLIs are resolved with Regional-Flip but the collar is folded inward, (d) BLIs are resolved with Mesh-Tearing but the collar is raised upward. All three resolutions are intersection-free, but (b) is in general considered most appropriate.	62
5.7	Flowchart for determining the resolution algorithm.	64
6.1	CLOSED & LL. In 3D, every edge that penetrates a triangle is shown in red as in Figure 3.1(a) and Figure 3.1(b), which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.	66

6.2	EIGHT & CROSS. In 3D, every edge that penetrates a triangle is shown in red as in Figure 3.1(a) and Figure 3.1(b), which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.	67
6.3	BBII & BIBI. In 3D, every edge that penetrates a triangle is shown in red as in Figure 3.1(a) and Figure 3.1(b), which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.	68
6.4	Exploded handkerchief. (a) initial tanglement, (b) after running the proposed method 75 time steps, (c) after running SIM-PREV additional 2,000 time steps after the first 75 time steps.	70
6.5	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.	71
6.6	Experiments on clothing cases. (a)~(d) shows the initial tanglements. (e) is a capture during a session of interactive tanglement generation and resolution.	73
6.7	Experiments on clothing cases. (a)~(d) show the results after executing SIM-ESEF to Figure 6.6(a) ~ 6.6(d), respectively. (e) shows the results after 500 times steps of simulation with the proposed method.	74

6.8	Experiments on clothing cases. (a)~(d) show the results after executing SIM-PREV to Figure 6.6(a) ~ 6.6(d), respectively. (e) shows the results after 500 times steps of simulation with SIM-PREV.	75
6.9	Crease-Flip applied to sharp-folded lapel and collar. (a) is the given setup which contains multiple intersections including BLIs. (b) shows the resolution with the SIM-BLIR, and (c) shows the resolution with the SIM-PREV.	79
6.10	Mesh-Tearing applied to sleeve. Panels in blue indicate that they remain fixed during the test. (a) shows the initial state in which the sleeve is dragged up. (b) shows the result with SIM-BLIR, which left no intersection. (c) shows the result of another trial with SIM-BLIR, in which no intersection is left but the sleeve is left folded up. (d) shows the result with SIM-PREV, in which some BLIs are left unresolved.	82
6.11	Mesh-Tearing applied to the skirt. (a) shows the initial state in which the bottom of the skirt is pulled up. (b) shows the result with SIM-BLIR. (c) shows the result with SIM-PREV.	83
6.12	Exploded handkerchief. (a) shows the initial state, (b) with SIM-BLIR (c) with SIM-PREV.	84

A.1	Handling of the case when three panels are seamed at the subject vertex \mathbf{x}_0 . (a) the triangle fans that are directly adjacent to \mathbf{x}_0 . (b) the sector form version of (a).	90
B.1	\mathbf{x}_1 , \mathbf{x}_0 , and \mathbf{x}_2 form a dart, in which $E(\mathbf{x}_0, \mathbf{x}_1)$ is seamed with $E(\mathbf{x}_0, \mathbf{x}_2)$. We will call it the dart seam . (a) initial tanglement. (b) pulling \mathbf{x}_0 toward \mathbf{x}_1 without considering the dart seam. (c) pulling \mathbf{x}_0 downward while keeping the length of $E(\mathbf{x}_0, \mathbf{x}_1)$ and $E(\mathbf{x}_0, \mathbf{x}_2)$ the same. (d) pulling \mathbf{x}_1 and \mathbf{x}_2 toward \mathbf{x}_0	95
C.1	A case that needs the convexification heuristic. (a) $\text{fan}(\mathbf{x}_6)$ is not TIT-passable. (b) the result of convexification.	97
D.1	The scenario in which ϵ_{RG} can produce a sudden vertex displacement. (a) after the Intersection Analysis of time step n . (b) after World-Space Mesh Update of time step n , (c) after Intersection Analysis of time step $n + 1$, (d) after m-CCR of time step $n + 1$	101

E.1 Two different ways of coupling BLIs in Regional-Flip. Black dotted lines are BLIs, which define the cross-over regions. Both figures show the cut-away view, i.e., they do not show the middle parts to allow us to see the cross-section of the mesh. In (a) and (b), if the cross-over regions are flipped, it will result in folding-in and folding out, respectively. Note that, in (a) and (b), the surface normal along with the cross-section curls out and in, respectively. 104

List of Tables

2.1	Initial state and expected solution of CLOSED, BBII, and BIBI.	8
2.2	Initial state and expected solution of EIGHT and CROSS. . . .	9
2.3	Initial state and expected solution of LL and BLI.	10
2.4	GIA, Repulsive-ICM, and UIR on CLOSED, BBII, and BIBI. All tests were simulated up to 1000 time-steps, to show that the number of iterations is not the cause of failure.	11
2.5	GIA, Repulsive-ICM, and UIR on EIGHT, CROSS, LL, and BLI. All tests were simulated up to 1000 time-steps, to show that the number of iterations is not the cause of failure. . . .	12
6.1	#(*), %(*) and t(*) represent the occurrences of each case, per- centage, and average number of time steps required, respectively.	77

Chapter 1

Introduction

Clothing simulation has been a popular research topic in the graphics field for over decades. Starting from the early work of Terzopoulos et al. [60, 59, 58] which characterized cloth as a deformable surface, Carignan et al. [10] adopted the work and created clothing by seaming 2D polygonal panels in 3D and Provot [44] used a mass-spring model to lower the computation cost of elastic models used in [60, 59, 58]. Baraff et al. [1] first proposed an implicit numerical method to achieve stable and large time step simulation. After then, research in clothing simulation flourished, expanding its research area. Various time integration models were proposed to enhance the accuracy (IMEX [15, 4, 18], Runge-Kutta [21, 32]) or stability (BDF2 [11], FEM [26]), and at the same time, linear models [13, 24, 25] were also considered as a trade off in accuracy for speed. For more real-time application, studies stretched out from mass-spring / energy-based constraints to geometric constraints. Position based

approaches [37, 16, 38, 35, 33, 34] directly worked on positions, thus avoid overshooting problems in implicit methods. Projection based approaches [31, 3, 66, 39, 41, 5] were able to get a more global and physical behavior than position based approaches.

Side by side, collision handling has also developed its own path. Clothing is generally expressed as a deformable surface, it is vulnerable to self-collision, and a small artifact can disrupt the whole simulation. Usually being an un-oriented surface, its inside is unclear, thus a strict history-based approach is a reasonable choice. Also called continuous collision detection (CCD), which checks if there are any collisions between a time interval, was first studied in [27, 43]. Both vertex-triangle CCD and edge-edge CCD were expressed as solving a cubic equation that finds a contact time between the time interval, and second checks if the collision was actually inside the triangle or between the edges. The base work of CCD was later combined with exact and robust response methods which include friction and impact zones [6, 19, 46, 40], or air meshes [36].

As the computational amount increased due to the complexity of the clothing and the increase in dimensions, several techniques were introduced to accelerate the collision handling. Some used spatial hashing [61, 50], volino diagram [47], GPU ray-tracing [29], or bounding volume hierarchy (BHV) [62, 22, 49, 68] to reduce triangle pairs that need to be checked in the high-level. Other works employed the triangle structure [12, 49, 45, 52, 67] or energy stored in the deformation [76] to cull the number of primitive tests, by its

connectivity or removing the duplicate tests. Hardware approaches were also made, use of multi-core architectures [4, 63, 53, 42] or GPU [51, 70, 54, 56, 68, 50, 57, 30] has been a promising work, as most collisions can be handled in parallel.

As powerful as it seems, continuous collision handling (CCH), i.e. CCD and its continuous collision response methods (CCR), has a critical weak spot. CCH is powerful when defending collisions, but it works under the following premise: when starting the current time step, the garment has to be in a collision-free state. Unfortunately, when used in real world situations (e.g., in real-time virtual fitting systems or animation studios), even the simulators equipped with a fully-functional CCH may produce anomalous results, due to the following violations of the premise.

The most frequent case is when a garment is initially created. As the constituent panels are seamed, the panels can interpenetrate each other, and also with the avatar. Since the penetration has already happened when the garment is born, the first frame of the simulation has to start carrying these problems. The similar situation also occurs when studies try to transfer a pre-simulated clothing to an another avatar [77, 14, 8, 28, 20, 23, 17]. In many cases, the animation of the avatar is not clothing-friendly, the body intersects with itself and creates a pinched area [2]. Clothing that is stuck between the pinching may be impossible to perform a valid CCR as it oscillates between the pinched area. Research [72] has been made to remove such a situation, but for the time being, we will assume that pinching can still happen.

Other than the above, sometimes the user can apply interactive manipulations that have to be prioritized before the CCH while the simulation is being performed. Also, system demands might have to override the CCH and call an administrative quit. 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.

Once a time step starts with some self-collisions unresolved (later also called as tanglements), 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 them. The preceding discussions imply that the simulator needs a discrete collision handling method (DCH) regardless of how concrete the CCH is.

Chapter 2

Related Works

When a triangular mesh is given, DCH tries to reconfigure the mesh until all intersections disappear. It is practically important for clothing simulation since, for CCH to work properly, simulation of each time step should start from an intersection-free configuration. In order for DCH to possibly untangle, it is essential to know which region is tangled. A common assumption in most DCH methods is the **majority rule** (i.e., the majority of the clothing vertices are not tangled), which turns out to accord well with the real situations thus solving most of the clothing cases.

2.1 Cloth Untangling: General

Baraff and Witkin [2] proposed global intersection analysis (GIA), which used the flood-fill algorithm to color the primitives (vertices, edges, triangles) of a smaller region based on the intersection analysis. After then, the coloring

could guide to which direction each vertex should move to resolve tanglement. Wicke et al. [71] pioneered the job of classifying intersections. Intersections were classified into seven types, each named after their unique shape in the uv-space (CLOSED, CROSS, EIGHT), or by a combination of B, L, and I (LL, BIBI, BBII, BLI). In the latter cases, B indicates that the intersection meets the boundary of the mesh, I indicates that the intersection ends in the interior of the mesh, and L indicates that the intersection passes a **loop vertex** (first introduced in [2]). Volino and Magnenat-Thalmann [64] proposed intersection contour minimization (ICM), which applied force/displacement to vertices in a way that would shorten the intersection in each triangle. Their work was further investigated by Ye and Zhao [75], who suggested some improvements such that the intersections can be resolved faster in a consistent direction when at least one of the intersecting objects has an orientation. Ye et al. [74] proposed a unified intersection resolver (UIR), which is a new pipeline of collision handling by harmoniously combining some of the previous methods. UIR calculated the minimal displacements for resolving the penetrations with the stencil-based formulation introduced in [73, 48]. Recently, Buffet et al. [9] 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.

2.2 Cloth Untangling: Multi-Garment

As more and more works try to dress an avatar with a pre-simulated data set of garments, the intersection between garments is inevitable. As a result, several works have focused on resolving intersections that can occur when multiple garments are worn on a single body. Zhong [77] projected the problematic area of the inner-layer (i.e., the area that is protruding the outer-layer) onto the body surface, then let the simulator relax the potential artifacts in the projection. Du et al. [14] constructed an AABB structure to pull out the outer layer that has penetrated inside. Brouet et al. [7] was able to transfer garments to different types of body shape-preserving the garment design. Hu et al. [20] formulated the problem into a minimization problem that searches for the optimal position for intersection-free vertices while considering minimal vertices displacement and smoothness of the corrected area. Buffet et al. [9] approximated garments as implicit surfaces, and converted the problem of untangling garments into resolving overlapping implicit surfaces. Han et al. [17] stratified garments such that the initial positioning does not have any garment-to-garment intersection, thus the conventional simulator could start from a sanitary configuration. All works are powerful when applied to layered clothes. However, intersections between different garments can generate only certain types of intersections, i.e., BIBI, BBII, and CLOSED.

2.3 Summary and Limitations

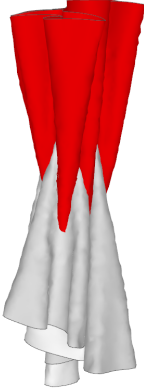

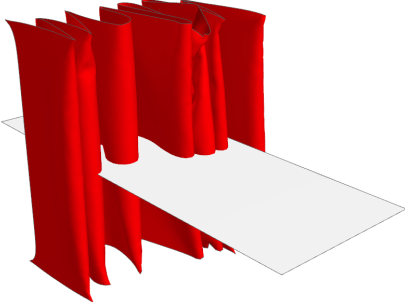
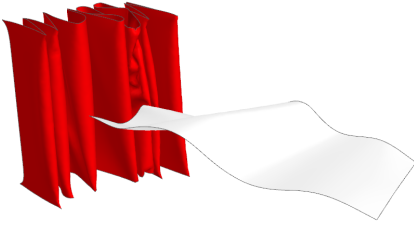
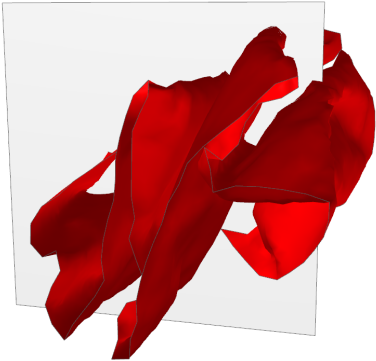

Type	Initial State	Expected (proposed method)
CLOSED		
BBII		
BIBI		

Table 2.1 Initial state and expected solution of CLOSED, BBII, and BIBI.



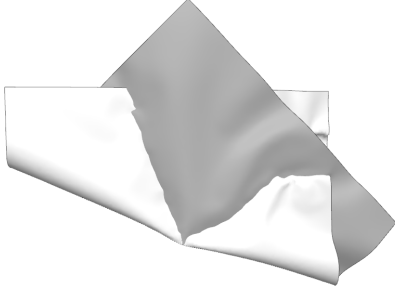
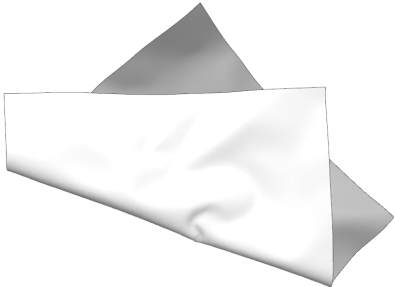
Type	Initial State	Expected (proposed method)
EIGHT		
CROSS		

Table 2.2 Initial state and expected solution of EIGHT and CROSS.

The limitation that the previous works have in common is first, the convergence/coverage was often not explicit as the table in Figure 2.1 shows. (Experimental results shown in Table 2.4 and 2.5 back up the table.) Therefore, with those methods, when a tanglement is persistent, it was difficult to judge whether the method needs more iteration or if the method is unable to cover the case. Second, for BLI, the majority rule is not valid and still remains most un-studied. This is not surprising. As a matter of fact, BLI is not a type of in-


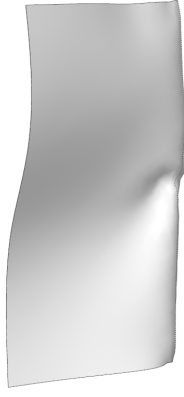
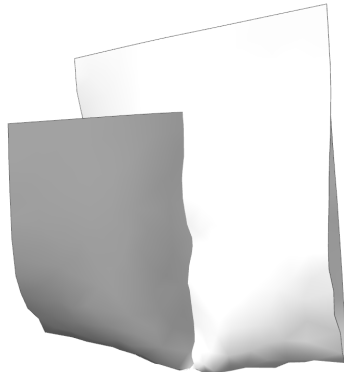
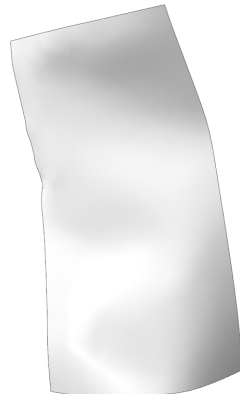
Type	Initial State	Expected (proposed method)
LL		
BLI		

Table 2.3 Initial state and expected solution of LL and BLI.

tersection that can be met frequently. By its character, it can only occur when a deformable, 2-manifold object self-intersects by *folding*. The experiments of the prior works were either not complex enough to include BLIs, or even if they included BLIs, they focused on the resolution itself, without addressing in which fashion the resolution should be done (hereafter we will call it the **resolution style**). Third, works that specifically target multi-garments have a strict range of applications. As its main power is based on the information

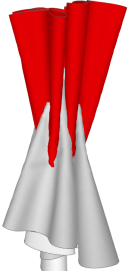
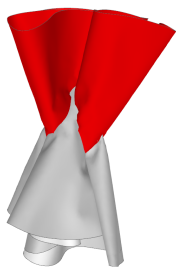
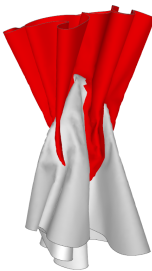
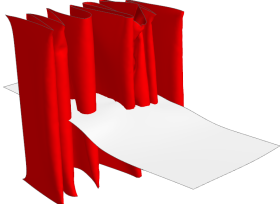

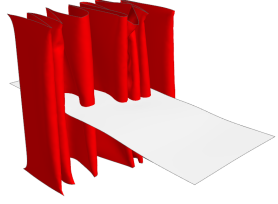
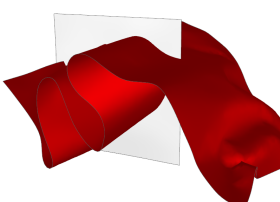
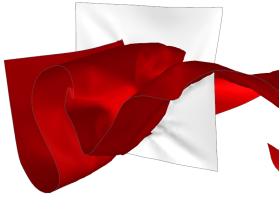
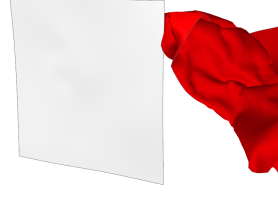
Type	GIA	Repulsive-ICM	UIR
CLOSED			
BBII			
BIBI			

Table 2.4 GIA, Repulsive-ICM, and UIR on CLOSED, BBII, and BIBI. All tests were simulated up to 1000 time-steps, to show that the number of iterations is not the cause of failure.

of layering order, the methods can only be applied to inter-garment intersections. It does not only unhandle intra-garment intersections, but also does not defend intra-garment collisions that can occur during its process. Thus, it is only a half-solution.


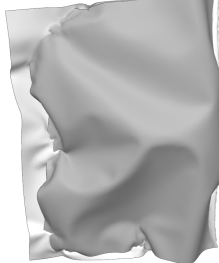

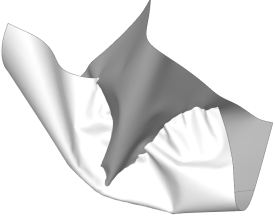
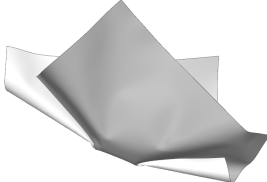
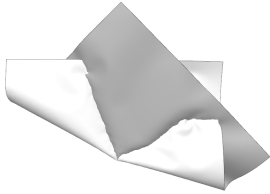



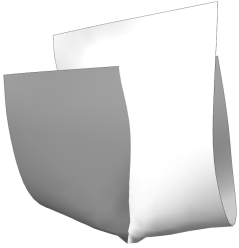
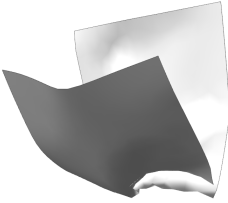
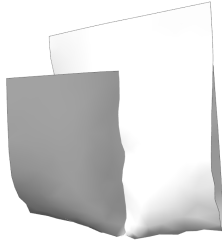
Type	GIA	Repulsive-ICM	UIR
EIGHT			
CROSS			
LL			
BLI			

Table 2.5 GIA, Repulsive-ICM, and UIR on EIGHT, CROSS, LL, and BLI. All tests were simulated up to 1000 time-steps, to show that the number of iterations is not the cause of failure.

2.4 Contribution of Proposed Work

This paper splits the intersections into two groups, the first group as (**CLOSED**, **BBII**, **BIBI**, **EIGHT**, **CROSS**, **LL**), and the second as **BLI** only. We find that

	method	Intersection type	CLOSED	BBII	BIBI	EIGHT	CROSS	LL	BLI
General	Baraff et al. [2003] GIA		△	-	-	△	-	-	-
	Volino and Thalmann[2006] ICM		△	△	△	△	△	△	-
	Ye et al. [2012] Repulsive-ICM		△	△	○	△	△	△	△
	Ye et al. [2017] UIR		△	○	○	△	-	△	-
Inter-layer	Yueqi Zhong [2012] Hu et al. [2017] Han et al. [2019] Buffet et al. [2019]		△	△	△	-	-	-	-

Figure 2.1 Coverage of previous methods in terms of categorization made in Wicke et al. [71]. The table is prepared based on the coverage explicitly stated in each paper or the theoretical limit of their methods. In the table, a circle means a complete (or theoretical) solution, a triangle means an experimentally shown partial solution, and a minus means that it is out of scope.

the two groups cannot be treated the same, thus requiring separate strategies. For the first group, this paper proposes **Edge-Shortening / Epsilon-Finishing (ESEF)** such that the method resolves them within a finite number of steps. With the novel coloring method and taking a new perspective of the CCD formulation, we make the simulator have the DCH capability by making simple modifications to the conventional simulator that has a full-proof CCH.

For the second group, we propose **Boundary-Loop-Interior Resolver (BLI-Resolver)**. For example, see Figure 5.6. Figure 5.6(a) shows an initial state where the collar has several BLIs, and Figure 5.6(b) ~ 5.6(b) are three different styles of resolving Figure 5.6(a). When counting the remaining intersections, all styles are valid, but from a practical point of view, resolving Figure 5.6(a) into Figure 5.6(b) would be most generally acceptable. People rarely wear a shirt with the collar folded inwards shown in Figure 5.6(c). The collar is not folded at all in Figure 5.6(d), which is not acceptable unless the

wearer intended it. The above discussion implies that, the given initial state (e.g., Figure 5.6(a)) alone is not a well-posed problem, but additionally needs to specify the desired resolution style. BLI requires a new approach covering starting from how to define the tangled region, which resolution styles to have, and finally defining the resolution algorithms to apply in order to obtain each of those resolution styles.

In common, both methods should operate under the premise addressed in Section 1. The method of both groups are designed to be built on top of a simulator equipped with a full-proof CCH as shown in Figure 2.2, such that we can still rely on CCH on where there are no tanglements, and at the same time, exploit its property and use it to aid the DCH process. In other words, the proposed DCH method does not tamper with CCH in a safe region, such that the simulation will not encounter any new tanglements by the resolution itself. But 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) that violate the premise, it would be impossible to technically discuss the completion of the resolution. We will assume this condition throughout this paper.

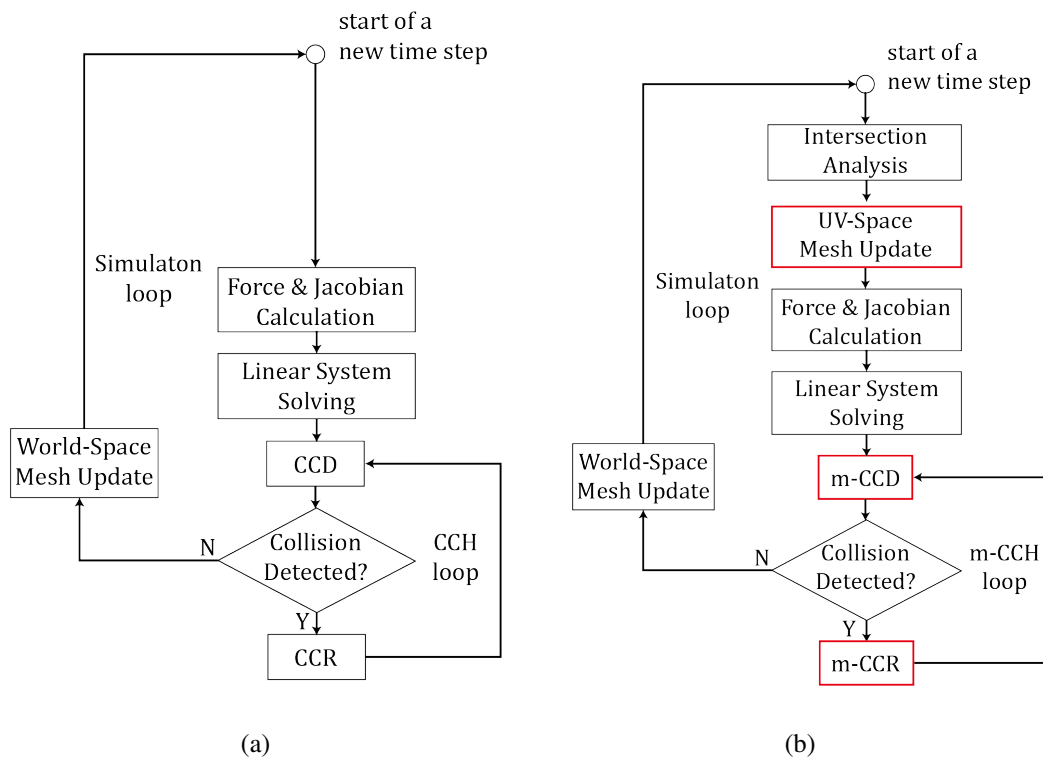


Figure 2.2 Simulation flowcharts. (a) shows the conventional simulator, (b) shows how the two proposed DCH methods (red boxes) fits into the conventional simulator to form the new simulator.

Chapter 3

Preliminary

In this paper, we consider only triangular meshes.

3.1 Edge-Shortening / Epsilon-finessing

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 the first group (**CLOSED**, **BBII**, **BIBI**, **EIGHT**, **LL**), the first step should be identifying the tangled regions¹, which can be achieved by

¹Determining 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 the majority of the clothing is on the right side (i.e., not tangled). Of course, a case can be fabricated such that the majority is tangled. Often, it is not a matter of right or wrong, but of the user's intention. Therefore, throughout the paper, for the sake of technicality of the discussions, we will assume that the clothing mesh is not tangled to

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 ([2, 71]).

– 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 Figure 3.1(a), 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 Figure 3.1(b), 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.

the extent the majority is tangled.

– **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 Figure 4.1(a)).

– 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 Figure 4.1(e)).

If the coloring was done as introduced in this section, the coloring itself contains the essential information in 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 Figure 3.1(a) and Figure 3.1(b), 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.

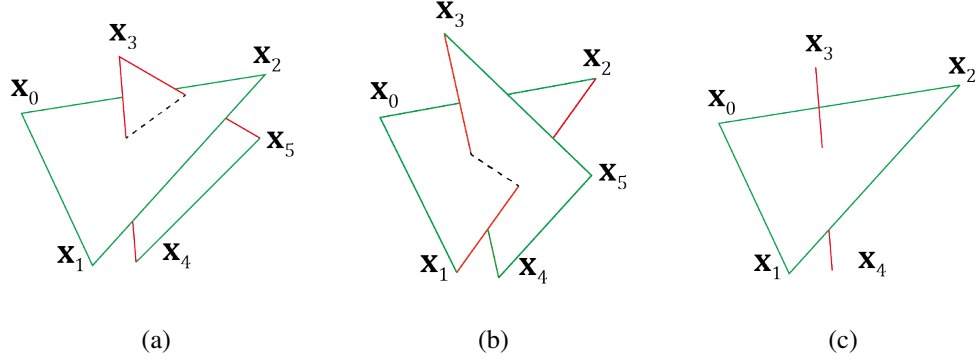


Figure 3.1 Two elementary tanglements. The intersection path is shown with black dashed lines. The edges are colored according to the proposed coloring scheme. (a) vertex-triangle tanglement, (b) edge-edge tanglement. (c) is a simplified drawing of (a), which is used when the discussion focuses on the behavior of $E(\mathbf{x}_3, \mathbf{x}_4)$ with respect to the triangle $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$.

3.2 Boundary-Loop-Interior Resolver

For the second group (**BLI**), which is an open intersection, we cannot follow the majority principle that is commonly applied for closed intersections. Among open-loop intersections, recently, Ye et al. [74] pointed out a few properties of BIBI and BBII can be used to determine the region to resolve. However, BLI still remains unsolved.

3.2.1 Repulsive-ICM on BLI

In this section, we will repeat the formula introduced in [75], discuss a possible scenario when Repulsive-ICM can be stagnant, then show that BLI can be that scenario case. Referring to Figure 3.3, [75] formulates a directional vector \mathbf{G} along which the edge \mathbf{E} should move in order to shorten the intersection

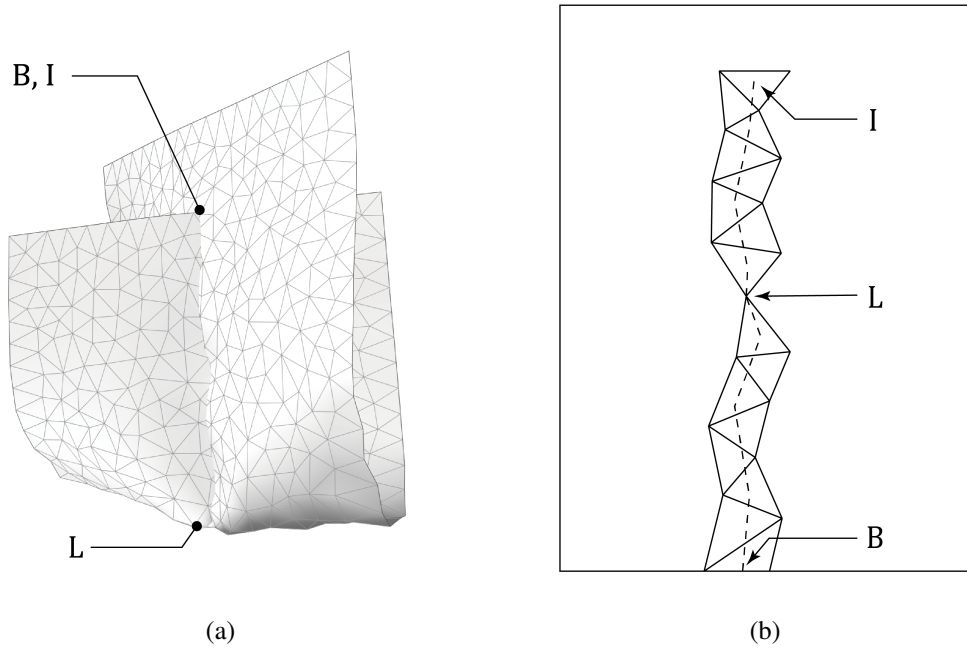


Figure 3.2 A BLI intersection. (a) shows it in 3D. (b) shows intersection analysis of (a) in 2D. Note that B and I coincide in 3D.

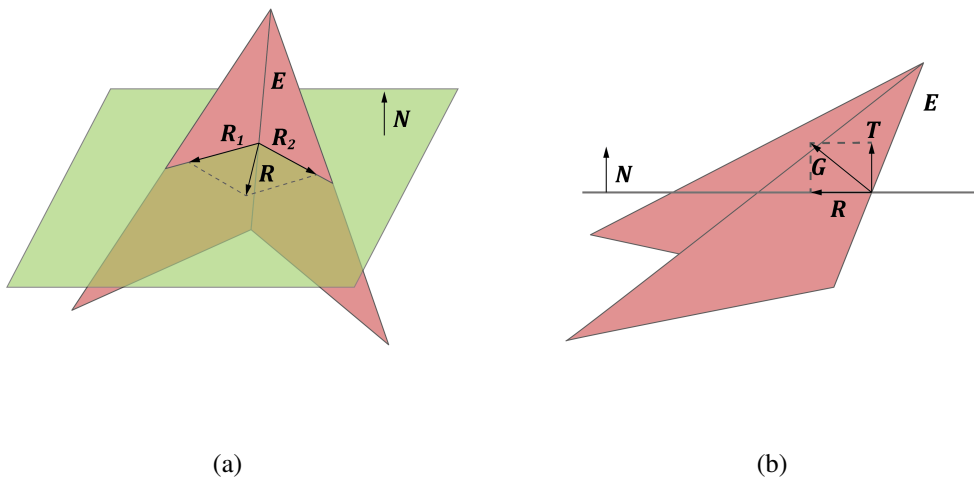


Figure 3.3 Diagrams taken from [75]. Two red triangles that share **E** are intersecting the green plane. (b) is the horizontal view of (a).

path. \mathbf{G} is defined as

$$\mathbf{G} = \mathbf{R} + \mathbf{T}, \quad (3.1)$$

where the two terms \mathbf{R} and \mathbf{T} are

$$\mathbf{R} = \sum_{i=1}^2 \mathbf{R}_i \mathbf{T} = -\frac{\mathbf{E} \cdot \mathbf{R}}{\mathbf{E} \cdot \mathbf{N}} \mathbf{N} \quad (3.2)$$

The first term \mathbf{R} is the sum of two unit vectors that point outwards from \mathbf{E} but are parallel to the surface \mathbf{E} is penetrating. Therefore, if there is no fold along \mathbf{E} , then $\mathbf{R} = 0$ and so becomes \mathbf{T} . We will call this case the **straight intersection path case**. In such a case, \mathbf{G} would not be meaningful. Also, the second term \mathbf{T} becomes zero when $\mathbf{E} \perp \mathbf{R}$. In fact, the above observations have been already pointed out in [75], but they did not explicitly address how the Repulsive-ICM would behave in such degenerate or near-degenerate cases.

The above degenerate or near-degenerate case can be easily met when resolving a BLI case. Let's suppose a typical BLI configuration shown in Figure 3.4, in which the tangled region takes approximately a cone-shape in 3D. Use of color in Figure 3.4 is related to Figure 3.3. The green-colored triangles in Figure 3.4 are doing the role of the green rectangular plane in Figure 3.3, and vice versa.

Figure 3.4(a) shows that the intersection path is smooth; it is a nearly straight intersection path case. Therefore \mathbf{R} will be close to zero. Figure 3.4(b) shows that, along the intersection path, the red triangles are almost orthogonal to the green triangles, thus $\mathbf{E} \cdot \mathbf{R}$ will tend to have a small value and so

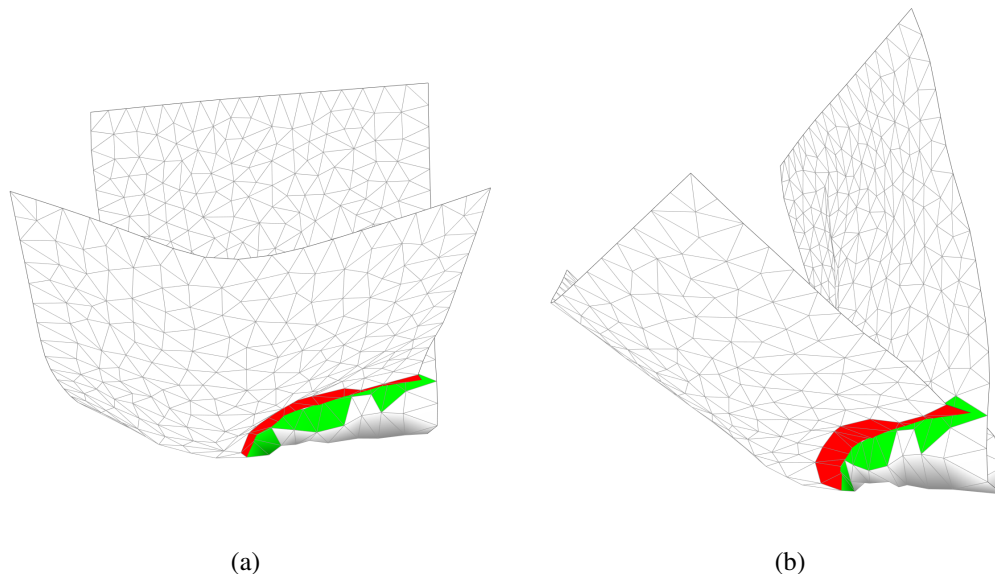


Figure 3.4 The behavior of the Repulsive-ICM for a cone-shaped BLI. (b) is the side view of (a).

will **T**. In fact, as we run Repulsive-ICM, we observe that, whenever a cone-shaped BLI is formed, the Repulsive-ICM loses its ability to create sufficient directional vector to resolve and the configuration gets stuck.

3.2.2 New Approach for BLI

As we follow the classification of Wicke and use their acronyms, considering the significance of BLI in this work, we review how the acronym BLI was tokened. When observed in 2D (i.e., referring to Figure 3.2(b)), BLI starts from the boundary (**B**) of a mesh, passes a loop vertex (**L**), and terminates at an interior point (**I**) of a mesh. By its nature, the triangles around the loop vertex have to be folded for the BLI to occur. The fold here does not need to be sharp as in a crease, but as shown in Figure 3.2(a), it can be round-shaped.

Hereafter, we will refer to each of them as sharp and round folds, respectively.

An interesting point we find is that, after the resolution of the BLI is complete, the decision of whether the fold should *unfold* or *remain* is crucial. Although the given case is the same, the two resolutions (i.e., **fold-unfolding** and **fold-remaining**) produce different outcomes. Note that the method itself should be different for the two resolutions.

In the later sections, we propose methods to resolve BLIs for both fold-unfolding and fold-remaining cases. We name the cases the user chooses the fold-unfolding and fold-remaining, in terms of the resolution algorithm, as **Mesh-Tearing** and **Regional-Flip** or **Crease-Flip**, respectively. For the time being, we will assume that the resolution style is given by the user. Later on, we will show that to some extent, the resolution style can be inferred from the garment input.

Chapter 4

Edge Shortening / Epsilon Finessing

4.1 Overview

In contrast to the previous works for untangling that applied external forces, displacements ([2, 64, 71]), or constraints ([74]) 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 untangles by itself after it goes through finite iterations of the simulation loop.

This section will refer to Figure 2.2(b) which shows how we obtain the DCH capability from the conventional simulator of Figure 2.2(a) with full-proof CCH capability. The simulation starts by performing the Intersection Analysis, reporting which types of intersections are in place. Then, it performs the UV-Space Mesh Update such that the simulation based on the resultant

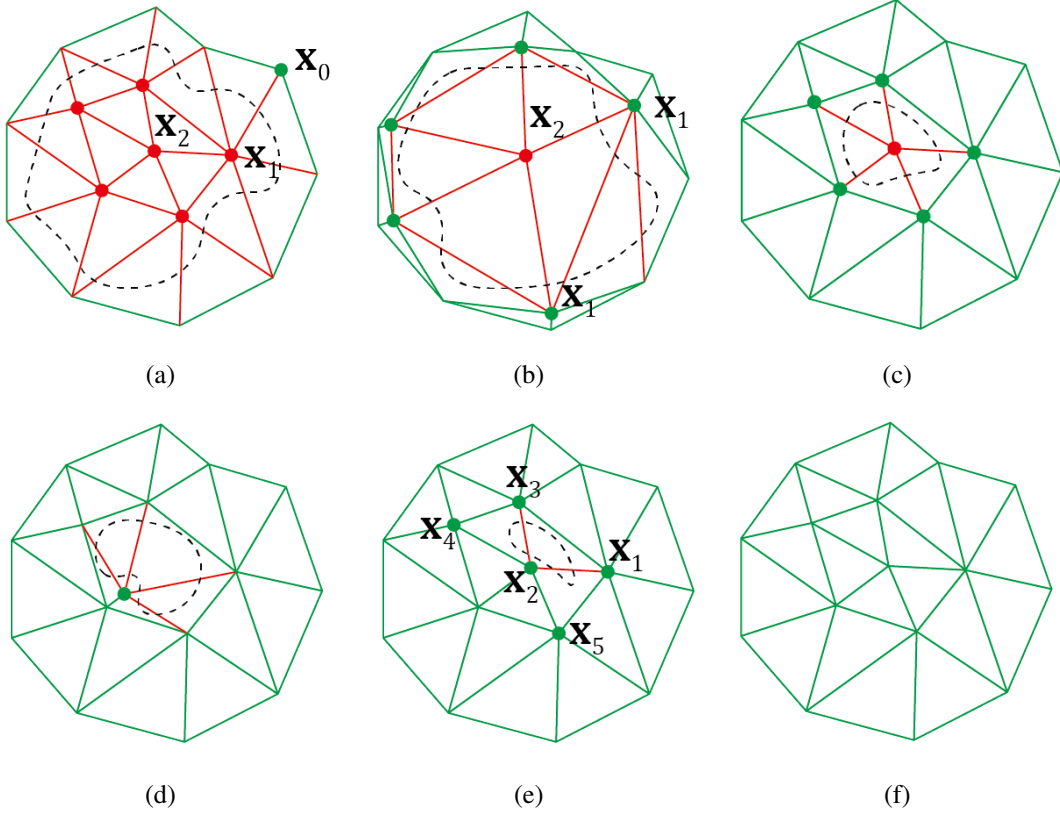


Figure 4.1 Operation of the proposed method in a cloth mesh fragment. (a) to (f) chronologically show various stages of the resolution.

mesh has a tendency to resolve the tangles. As detailed in Algorithm 1, the UV-Space Mesh Update colors the mesh according to the report of 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, 6 and 8). The details of those operations will be presented in Section 4.2.1.

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) is followed. Note that the results of the Force and Jacobian

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 γ
 - 4: **end for**
 - 5: **for** all out-most red vertices **do**
 - 6: Apply out-most red vertex pull by γ
 - 7: **end for**
 - 8: Revoke shrinkage and pull by $1/\gamma$ where needed
-

Calculations 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 Figure 2.2(a) as the **CCH loop**, and the counterpart in Figure 2.2(b) 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**.

Figure 4.1 focuses on the evolution of a uv -space mesh fragment as the simulator designed as Figure 2.2(b) operates. In Figure 4.1(a), \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 6) of of Algorithm 1) is achieved by applying the vertex pull operation (Section 4.2.1), 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. Figure 4.1(c) 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 Figure 4.1(d), and is revoked as shown in (Figure 4.1(e)). 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 Figure 4.1(e) are identified as red* triangles. For red* triangles, triangle shrinkage is applied. The resultant fully untangled mesh as shown in Figure 4.1(f).

4.2 Modifications to Conventional Simulator

Sections 4.2.1 and 4.2.2 describe more details of the UV-Space Mesh Update (Algorithm 1) and m-CCH, respectively. Sections 4.2.3 and 4.2.4 explain how the above two components work over the the simulation loop to resolve tangles in elementary and cloth meshes, respectively.

4.2.1 UV-Space Mesh Update

Section 4.2.1 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 4.2.1. Throughout this paper, $\gamma < 1$ is a user-controlled scale factor.

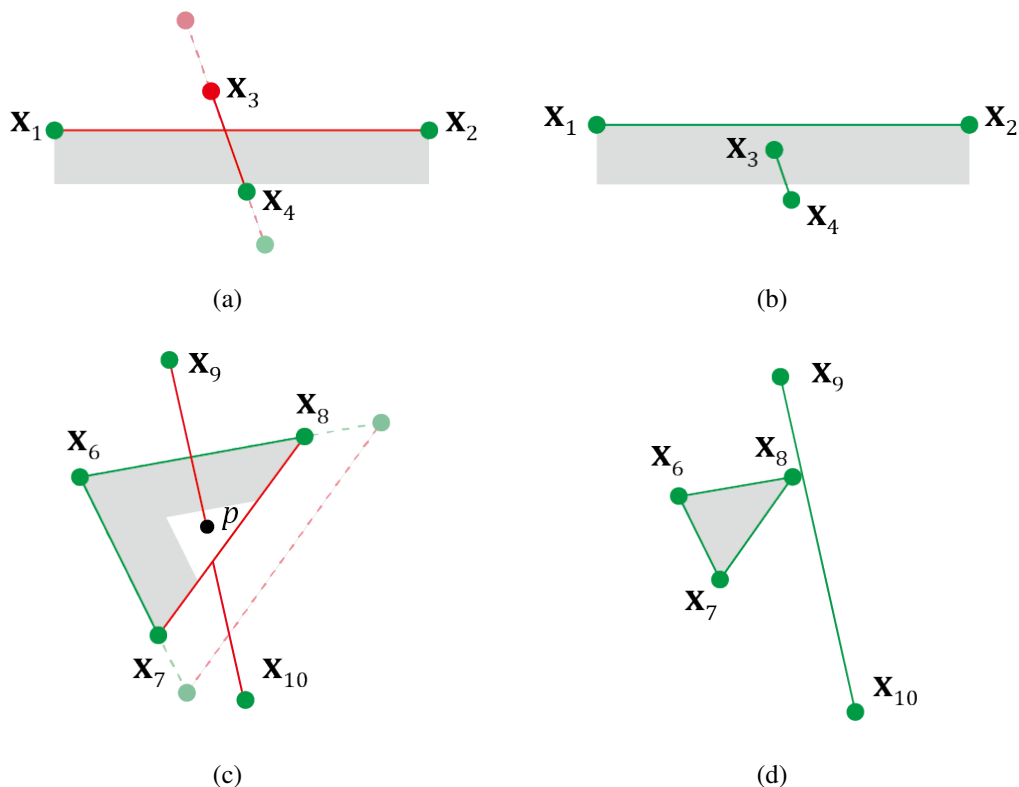


Figure 4.2 (a) and (c) show the outcome in the world-space mesh (after the World-Space Mesh Update) when the vertex pull and triangle shrinkage is applied to the elementary vertex-triangle tanglement (Figure 3.1(c)) and edge-edge tanglement (Figure 3.1(b)), respectively. Note that (a) is shown with the 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.

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 **tar-**

get vertex, and the red vertex as the **subject vertex** and the edge between the two as the **subject edge**. Figure 4.2(a) 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 Figure 4.2(c). 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**. Figure 4.2(c) 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).

Implementation of the Operations

For a 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 $\text{fan}(\mathbf{x}_i)$) is defined as the set of all the vertices adjacent to \mathbf{x}_i in the uv -space. For example, in Figure 4.3,

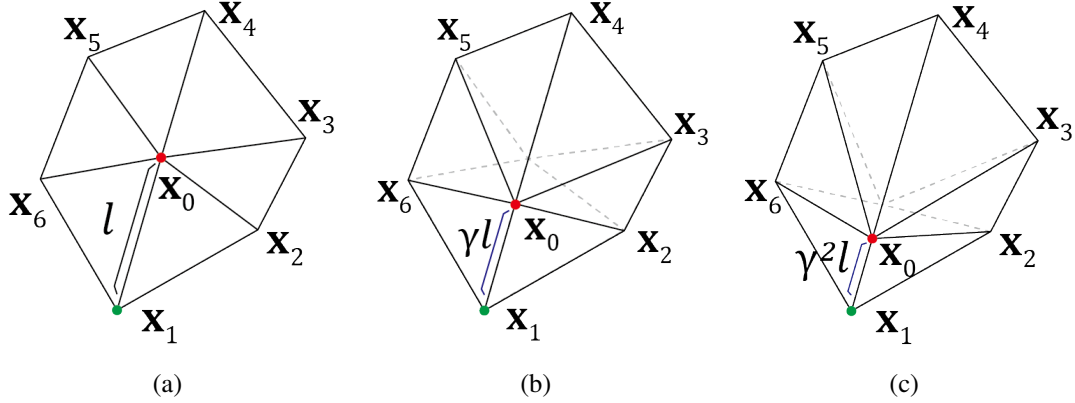


Figure 4.3 Shortening of the edge $E(\mathbf{x}_0, \mathbf{x}_1)$, which has the original length l , toward the target vertex \mathbf{x}_1 in the uv -space. (a) the mesh before the shortening. (b) the mesh after the edge is shortened by γ -scaling. Note that it results in modification of the whole fan(\mathbf{x}_0). (c) the mesh after another round of edge shortening.

$\text{fan}(\mathbf{x}_0) = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_6\}$.¹ In the context of pulling \mathbf{x}_i , we will call $\text{fan}(\mathbf{x}_i)$ 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 $\text{fan}(E(\mathbf{x}_i, \mathbf{x}_j))$) is the set of all the vertices adjacent to either of the edge vertices. For example, in Figure 4.8(a), $\text{fan}(E(\mathbf{x}_0, \mathbf{x}_1))$ 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**² rather than shortening the target edge alone. As shown in Figure 4.3, 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 $E(\mathbf{x}_0, \mathbf{x}_1)$).

¹Note that the set does not include itself, i.e., \mathbf{x}_0 .

²The 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.

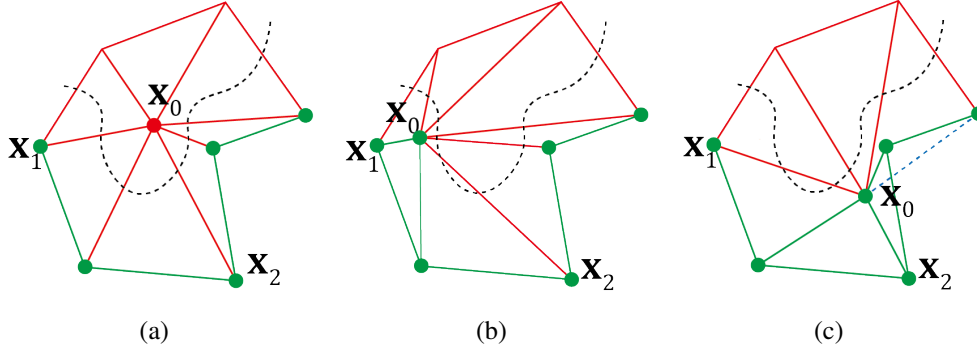


Figure 4.4 Determining the target vertex for edge shortening. (a) before shortening. (b) the shortened outcome when x_1 is chosen as the target vertex. (c) the shortened outcome when x_2 is chosen as the target vertex, which produces an **inverted triangle**.

An out-most red vertex can have multiple candidate green vertices for the pull as shown in Figure 4.4(a). 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., x_2 in Figure 4.4(c)) 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**.

4.2.2 CCH vs. 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) the barycentric coordinate of the penetration point if there was a collision. Both of the above require arithmetic operations that are exposed to round-off errors, so [6] and [69] used some small tolerance value ϵ_{CCD} to prevent false negatives. Some researchers used exact geometric computation [7, 65] or Bernstein sign classification [55] 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 Figure 2.2(a)), 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 Figure 4.5 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 Figure 4.5(a). 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 Figure 4.5(b).

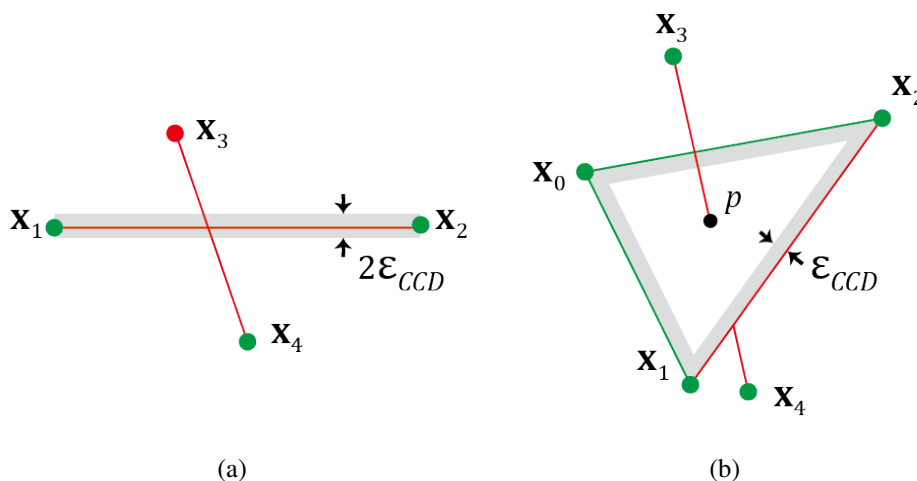


Figure 4.5 Effect of ϵ_{CCD} after the execution of full-proof CCH. (a) and (b) show the effect of ϵ_{CCD} acting as the thickness that prohibits elementary pairs from approaching too close. Note that the thickness in (a) exists also in (b), and vice versa.

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

- **Allowing red-red elementary pairs to cross:** The intersection path acts as the border that distinguishes 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 from 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 4.2.3. This modification (of ϵ_{CCD} value) applies to both CCD and CCR.

4.2.3 Resolution of Elementary Tanglements over Simulation Loop

A single application of vertex pull or triangle shrinkage may not resolve the tanglement, but the 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 Figure 4.2(a), 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 Figure 4.2(b) 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 the 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 Figure 4.2(c) 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 Figure 4.2(d). 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**.

Discussion on the Value of ε_{CCD}

In the VT-type ε_{CCD} -finesse shown in Figure 4.2(a) and 4.2(b), 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 $0.1 * \text{avg}(l)$, where $\text{avg}(l)$ is the average length of the mesh edges. (Further discussion on the value of ε_{CCD} is given in Appendix D.)

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.

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

In this section, we extend the explanation in Section 4.2.3 to the cloth mesh, i.e., we explain how repeated execution of the simulation loop resolves tangle-

ments in the cloth mesh. The following description is done referring to Figure 4.6.

Although the situation shown in Figure 4.6 for a cloth mesh may look complex (compared to Figure 3.1), it is just a 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 Figure 4.6(a) and 4.6(b). In this situation, the finesses need to be *scheduled* (which will be introduced in Section 4.3). 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 Figure 4.6(a). (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 Figure 4.6(b). The latter, which we will call 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 Figure 4.1.

By executing \mathbf{x}_0 -to- \mathbf{x}_1 finesse to the configuration shown in Figure 4.6(b), \mathbf{x}_0 crosses the intersection path, the result of which is shown in Figure 4.6(c). (The restoration step will relocate \mathbf{x}_0 to its original position, but that part

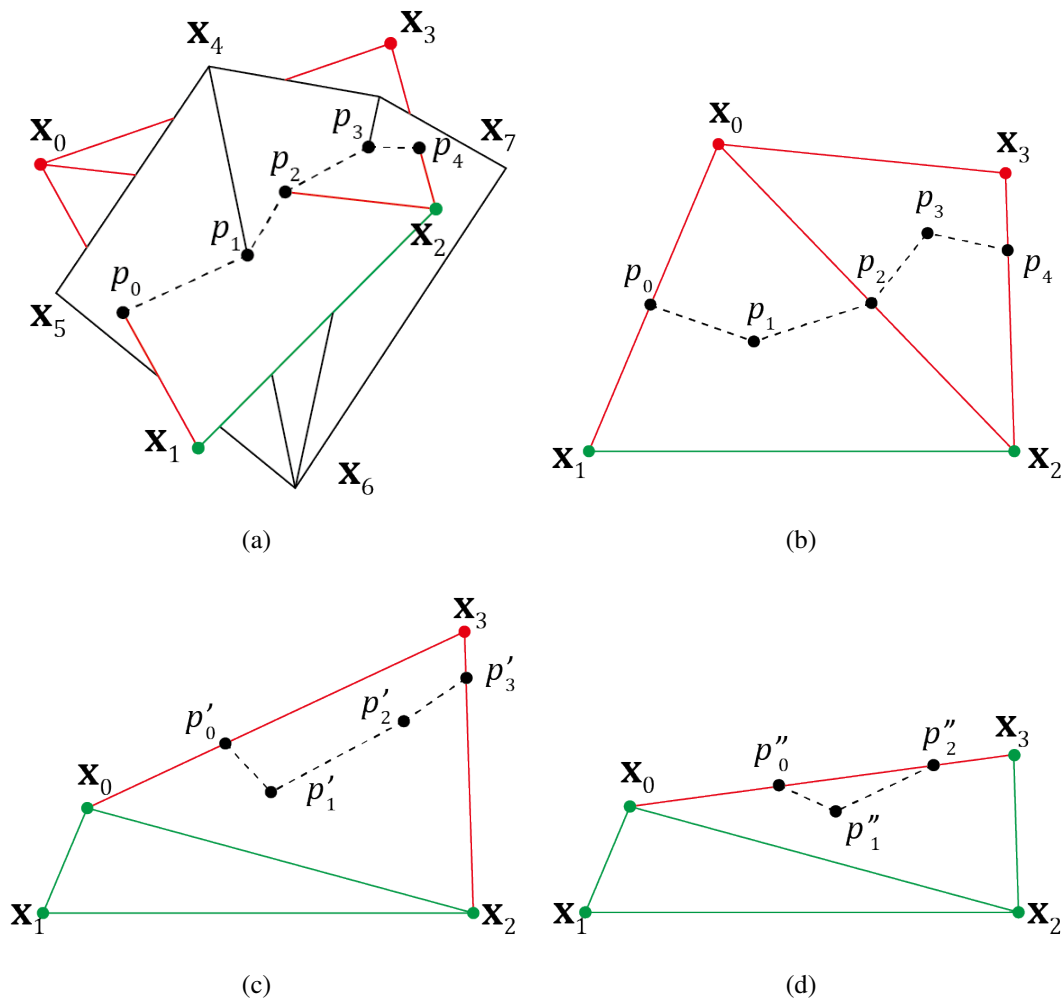


Figure 4.6 Applying ϵ_{CCD} -finesse to a cloth mesh. Black points represent the penetration points, and black dashed lines represent the intersection paths. Coloring of the vertices and edges is done based on the scheme described in Section 3. For visual comfort, the color is not shown or shown in black 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.

is omitted in the figure.) Now, in Figure 4.6(c), two possible VT-type ϵ_{CCD} -finesses (pulling x_3 to x_2 and x_3 to 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 Figure 4.6(d). 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.

4.2.5 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 the drawing on the right. We can set up 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 7 will give an experimental report on this issue.

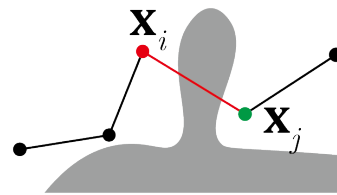


Figure 4.7 An example when edge shortening is caught between a sharp feature of a body.

4.3 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 out-most red vertices and red* triangles, respectively, it can introduce inverted triangles.³ 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 Figure 4.8(a) are shortened simultaneously, the two fans can **interfere**, which can result in an inverted triangle as shown in Figure 4.8(b).⁴ 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 re-

³Note that, whereas the TIT discussed in Section 4.2.1 checks the inversion within the subject fan, the inversions that are considered here are the ones when two neighboring fans are remeshed simultaneously.

⁴Note that if two subject vertices are not adjacent, the corresponding subject fans do not overlap, thus do not interfere each other.

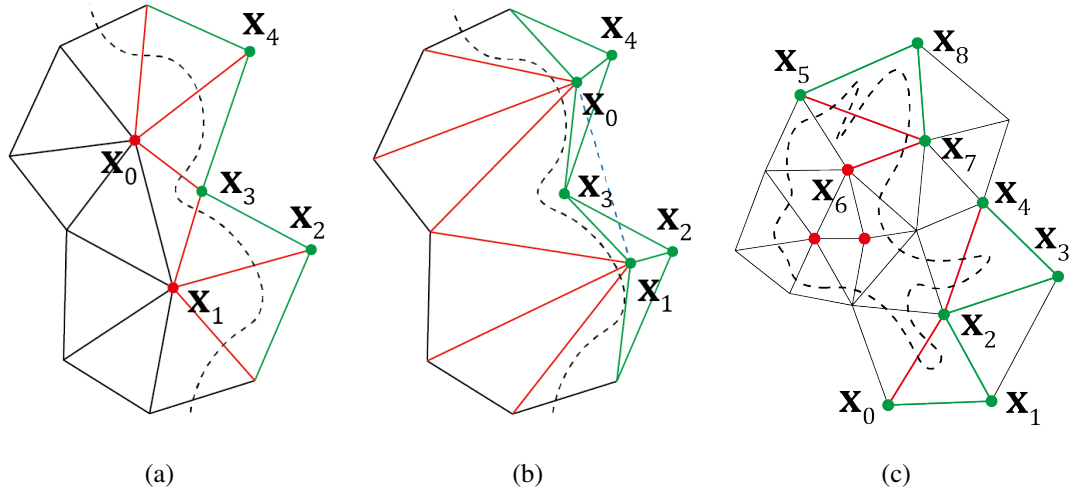


Figure 4.8 Cases that call for scheduling. (a) shows an example of Case 1. (b) when the two red vertices x_0 and x_1 of (a) are simultaneously pulled toward x_4 and x_2 , respectively. (c) shows examples of Cases 2 and 3.

stores to its original uv -position.

- **Case 2 – When two red* triangles are adjacent:** In Figure 4.8(c), the two red* triangles $T(x_0, x_1, x_2)$ and $T(x_2, x_3, x_4)$ share x_2 thus adjacent. If we shrink $T(x_0, x_1, x_2)$, x_2 will move toward x_1 . But if we shrink $T(x_2, x_3, x_4)$, x_2 will move toward x_3 . In such a conflicting case, they need to be *sequentialized*, i.e., either $T(x_0, x_1, x_2)$ or $T(x_2, x_3, x_4)$ 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 x_6 and the red* triangle $T(x_5, x_7, x_8)$ in Figure 4.8(c). 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, $\text{fan}(\mathbf{x}_6)$ includes \mathbf{x}_5 and \mathbf{x}_7 , and $\text{fan}(E(\mathbf{x}_5, \mathbf{x}_7))$ 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  $\text{fan}(E(\mathbf{x}_j^*, \mathbf{x}_k^*))$  is in its original  $uv$ -position then
4:     if  $\text{fan}(E(\mathbf{x}_j^*, \mathbf{x}_k^*))$  passes the TIT then
5:       Perform triangle shrinkage
6:     end if
7:   end if
8: end for
9: for all out-most red vertices  $\mathbf{x}_r$  do
10:  if  $\text{fan}(\mathbf{x}_r)$  is in its original  $uv$ -position then
11:    if  $\text{fan}(\mathbf{x}_r)$  passes TIT then
12:      Perform vertex pull
13:    end if
14:  end if
15: end for
16: for all green vertices  $\mathbf{x}_g$  do
17:  if  $\mathbf{x}_g$  is not in its original  $uv$ -position then
18:    if  $\mathbf{x}_g$  is not a subject vertex then
19:      Revoke any operations applied
20:    end if
21:  end if
22: end for

```

Algorithm 2 is one possible implementation of the above sequentializations and priority. If it is replaced with 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 9~ 12 should not be parallelized. Therefore Line 10 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 11 is to yield to the next TIT-passable out-most red vertex. Finally, Lines 16~ 19 are for restoring the green vertices that used to be red.

4.3.1 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 3.1).
- 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. 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). We have never met such a case so far.

4.4 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 4.3 enables multiple finesses to run without interfering with 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 4.2.5), and (ii) there is at least one TIT-passable fan (Section 4.3.1).*

Proof. As discussed in Section 4.2.5 and Section 4.3.1, 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 Figure 4.2(a), 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.⁵ A similar argument can be made for the red edges using Figure 4.2(c)

⁵In addition to the total number, we note that, by m-CCH, a red vertex can remain red or

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.

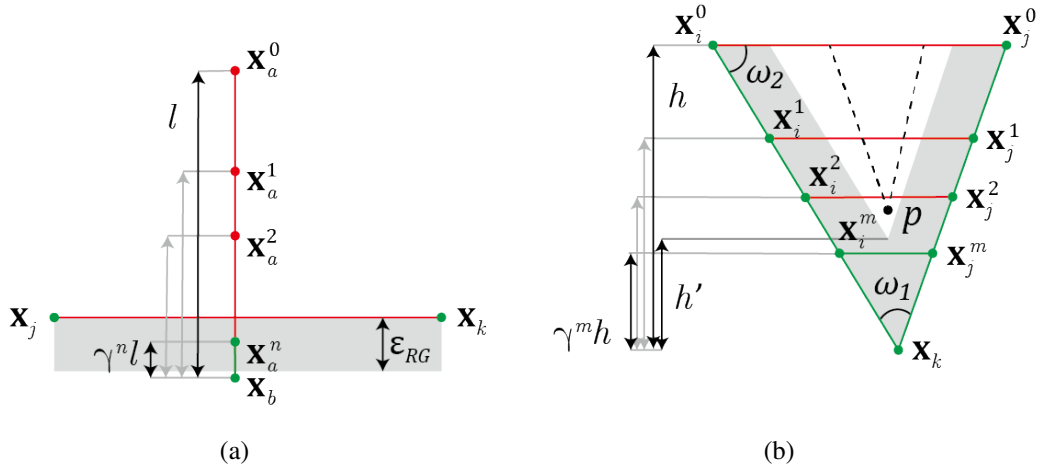


Figure 4.9 ϵ_{CCD} -finesse to vertex-triangle tanglement and edge-edge tanglement. (a) VT-type ϵ_{CCD} -finesse. (b) EE-type ϵ_{CCD} -finesse, in which the black dashed lines represent the intersection path.

Now we prove the second part. Let's suppose that the VT-type ϵ_{CCD} -finesse does not successfully pull an out-most red vertex out of the intersection path in finite time steps. Figure 4.9(a) shows the situation, in which \mathbf{x}_a is the out-most 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 become green, but a green vertex can never become red.

after n time steps,

$$\gamma^n l < \epsilon_{RG} \quad (4.1)$$

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. Figure 4.9(b) 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

$$\gamma^m h < h' \left(= \frac{\sin(0.5\omega_1 + \omega_2)}{\sin(0.5\omega_1)} \epsilon_{RG} \right) \quad (4.2)$$

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. \square

4.5 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 Figure 4.10, 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 Figure 4.10, 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

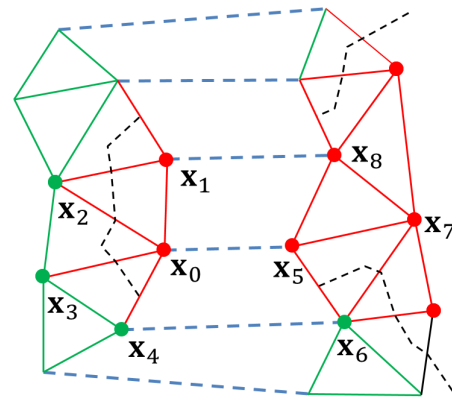


Figure 4.10 An example in which the intersection path exists across multiple panels.

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.

For $\text{fan}(\mathbf{x}_0) = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ and $\text{fan}(\mathbf{x}_5) = \{\mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\}$ in Figure 4.10, it is clear that not only $\text{fan}(\mathbf{x}_0)$ but $\text{fan}(\mathbf{x}_5)$ 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 $\text{fan}(\mathbf{x}_0)$ and $\text{fan}(\mathbf{x}_5)$ 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 $\text{fan}(\mathbf{x}_5)$ during the simulation. We put the detailed procedure in Appendix B.⁶

The procedures presented in Appendices A and B are developed such that

⁶The 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, and is presented in Appendix B.

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 an identical result with the one produced with the algorithms presented up to Section 4.4. Here, we note that the extensions in Appendices A and B 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.

Chapter 5

Boundary-Loop-Interior Resolver

5.1 Overview

Mesh-Tearing directly solves BLI, by editing the uv -mesh to unlock the mesh biting into itself. It omits certain triangles to free the lock. On the other hand, Crease-Flip and Regional-Flip is a rather more indirect approach, it converts a BLI into a different type of path such that the study in Section 4 can resolve the left-over. Crease-Flip is intuitive, by comparing the user-given crease angle and the current angle during the simulation, it can distinguish which side is folded in the opposite direction. Crease-Flip detects it and positions it at a correct angle. Regional-Flip removes the flipped area between the coupled two BLIs.

5.2 Modifications to Conventional Simulator

This section will once again refer to Figure 2.2(b) and show which parts of the simulation flow need to be modified to implement the three algorithms for BLI. Mesh-Tearing deliberately omits certain triangles which are beneficial to the resolution. Omitted triangles are reflected to the simulator by editing the uv-mesh in the UV-Space Mesh Update. Same with ESEF, it is also essential to allow crossing primitives when it is orchestrated by DCH. Crease-Flip and Region-Flip resolve BLI by intentionally crossing over two regions, thus the action should be allowed in the m-CCH Loop.

5.3 Mesh-Tearing

Once the user chooses fold-unfolding (i.e., B *sliding* towards L) for the given BLI case, some non-physical operation should be allowed if it is critical to the resolution. In this work, what we propose is to temporarily omit triangles along the intersection path (as shown in Figure 5.1) that can benefit the untangling. Chronological progress is visualized in Figure 5.1(a) ~ 5.1(d), by coloring the omitted triangles in sky blue. The sky blue triangle strip can be considered as stitches holding the blue and red regions of the mesh. The blue and red regions will be called as the **wings** hereafter. The remaining mesh colored in white will be called as the **body**. If the triangle strip is omitted, each wing will be free and independently unfold, and when it is unfolded enough, we let the omitted strip be re-identified. We call the above operation as the **Mesh-**

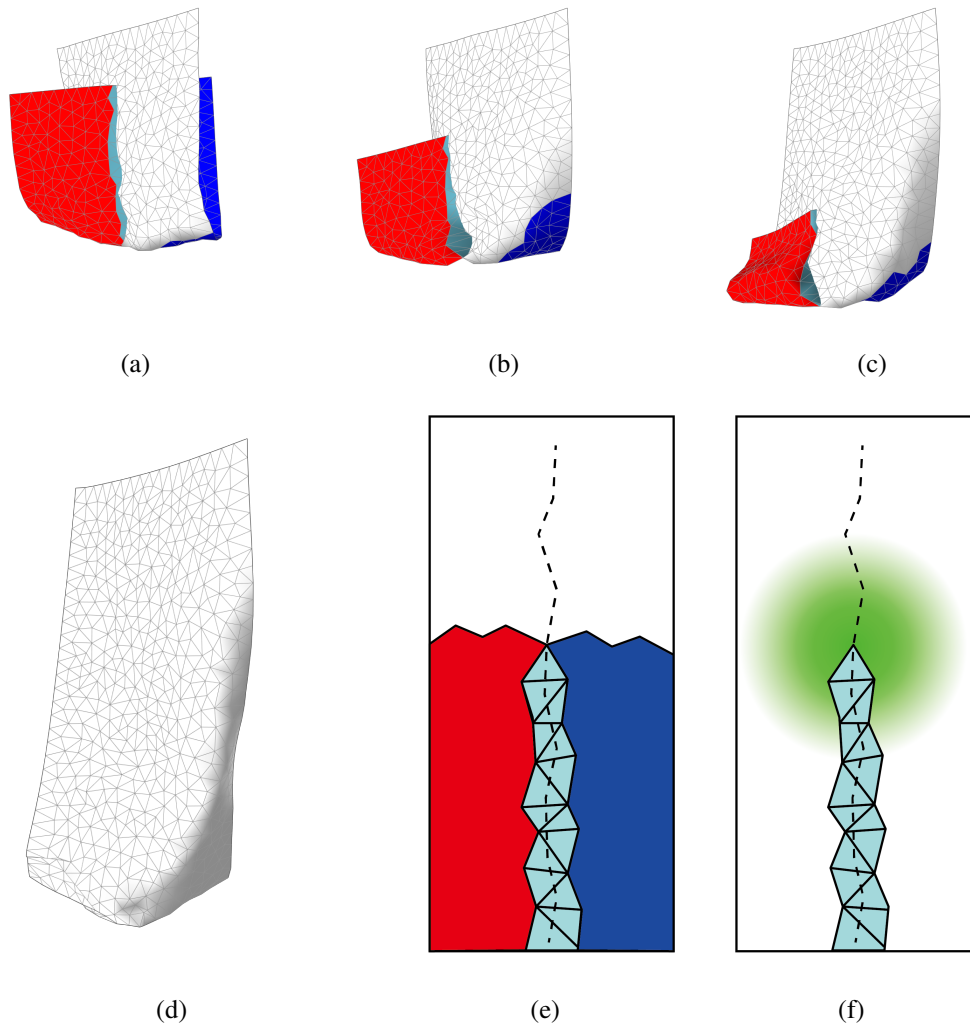


Figure 5.1 Operation of Mesh-Tearing. (a)~(d) chronologically shows how it works. (e) is a diagrammatic depiction of (a) in 2D, where the black dashed line is the intersection path, and the sky blue triangles are the omitted triangles, which enclose the intersection path from L to B. The green region in (f) shows where the bending stiffness is increased, which facilitates the loop vertex to slide out to B. In fact, (a) is identical to the third case in Figure 9 of [64].

Tearing. Its details are as follows.

Mesh Tearing is achieved by the following two operations. First, triangles that lie under the intersection path between B and L in Figure 3.2(b) are omitted, thus are excluded in the collision handling process of the simulation as shown in Figure 5.3. As the simulation proceeds, the wings slide out towards L (as shown in Figure 5.1(a) ~ 5.1(d)) and, as a consequence, the intersection path becomes shorter. The situation can change after every simulation time step. As the fold unfolds and B slides out toward L (in Figure 3.2(a)), L will move toward B (in Figure 3.2(b)). Then the previous BLI will not stay the same anymore. Thus we update the set of omitted triangles accordingly. By the update, triangles that used to be omitted may be no longer omitted. We will call those triangles as the **revived** triangles.

Second, the physical property of triangles around the loop vertex is varied to aid/accelerate the unfolding. If the unfolding relies entirely on the simulation, the progress can be hindered by external factors (e.g., gravity, penalty forces created by collision, etc...). To encourage the sliding out, we give a boost in the bending stiffness to a few neighboring triangle rings around the loop vertex.¹ In the remainder of this section, we will look a little more details of the operation of Mesh-Tearing.

¹These rings are selected in proportion to the L-to-B intersection depth, in terms of the number of the triangles the path is over.

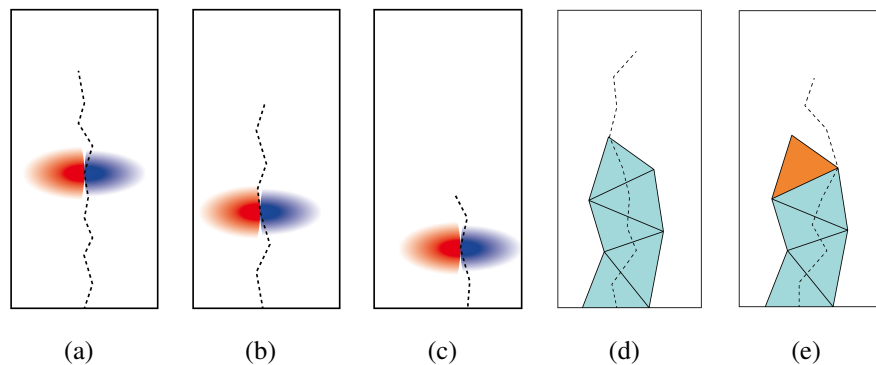


Figure 5.2 Operation of Mesh-Tearing in 2D. (a)~(c) are the 2D version of Figure 5.1(a) ~ 5.1(c). The color indicates the winding direction, and the color thickness indicates the bending curvature. In (e), the revived triangle is shown in orange. Take a note of where the new loop vertex comes.

5.3.1 L-to-B Propagation

The main power that drives the fold to propagate downward in Figure 3.2(a) is the boost in the bending stiffness around L. The above raises a question: if the *fold* propagates downward, would the *loop vertex* L propagate toward B?

A BLI is created when a mesh is wound-up in opposite directions (in Figure 5.1, the red and blue wings). Therefore there must be a vertex where the wings meet, known as the loop vertex. At that single point, we note that the curvatures of the two wings are opposite in sign as shown in Figure 5.2(a) ~ 5.2(c). (We will call that single point as the *crossing point*.) Therefore, as the fold propagates as shown in Figure 5.1(a) ~ 5.1(d), the crossing point L propagate towards B.

An interesting fact here is that only the vertices of the omitted triangle strip can be the new loop vertex. During Mesh-Tearing, history-based collision han-

dling prevents wings from penetrating the body, which makes the intersection path between L and B of the BLI to be kept inside the omitted triangle strip. Therefore, the loop vertex propagates towards B via vertices of the omitted triangle strip as shown in Figure 5.2(d) ~ 5.2(e).

5.3.2 Revived Triangles

Referring to the simplified flowchart of the simulator shown in Figure 5.3, during simulation, we note that Mesh-Tearing makes the intersection analysis module and the collision handling module run with different meshes. The intersection analysis module works with the *un-omitted* mesh, but the collision handling module receives the omitted version (i.e., the mesh without the omitted triangle strip). The intersection analysis module is the only one that is aware that the cloth has a BLI.

We let a triangle be revived only when the intersection analysis module has found out that the triangle no longer participates in creating a BLI. It can be surprising to the collision handling module as a triangle suddenly appears, but the new intersection it creates is not of the BLI type. Since the other six types of intersections can be resolved, the revived triangle does not cause any significant complications.

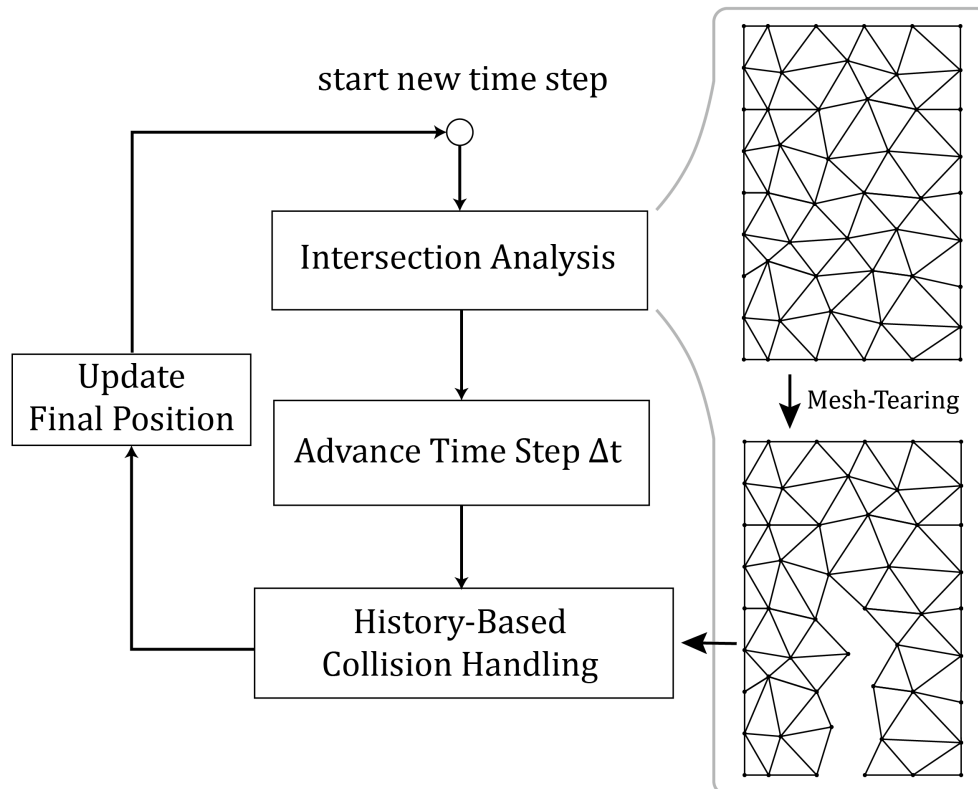


Figure 5.3 A simplified flowchart of the simulator. From the start of a new time step, the Intersection Analysis module reports intersections, and applies necessary measures to resolve them. Then, the simulator advances by Δt , i.e., calculates the mesh accounting for the time flow. The History-Based Collision Handling module verifies if the resultant mesh is valid, and if needed, resolves collisions. The Intersection Analysis module works with un-omitted mesh, then based on the BLI intersections present, it generates the omitted mesh for the History-Based Collision Handling module.

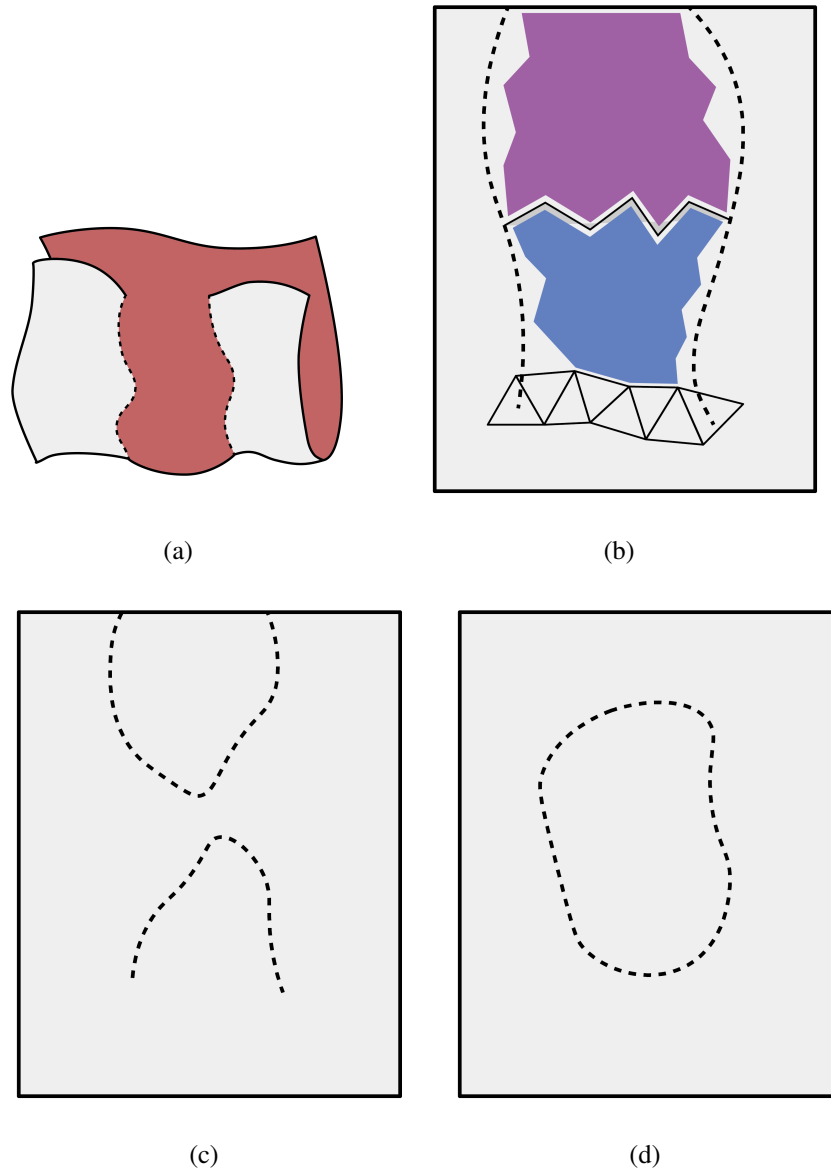


Figure 5.4 BLIs being resolved by Regional-Flip. (a) shows two BLIs on a round fold. Assuming that the grey-side cloth should be above the red-side cloth, (b) shows the situation in 2D, where purple and blue colored triangles are those that need to be switched by the proposed method. (c) shows possible resultant intersections as the BLIs are approaching each other. (d) shows another possible resultant intersection.

5.4 Regional-Flip

When BLI paths exist as a pair (an example shown with black dotted lines in Figure 5.4(a)), the region between the two paths needs to be resolved. If *fold-remaining* is the resolution style, switching or flipping the position of the two problematic areas (let's call the area the **cross-over region**) in 3D (e.g., by applying *attraction* forces, make the outer surface go underneath the inner surface and the inner come out of the outer) can be a simple solution. In this work, we take that approach. We perform the resolution in two parts: (1) defining the cross-over region, (2) performing flip for that region.

Let's assume that the user selected two BLIs to be coupled as shown in Figure 5.4(a). Then, the cross-over region is defined by creating a connecting path between the two I's of the BLI paths via the mesh *triangles* as shown in Figure 5.4(b) (hereafter we will call it **I-to-I connection**), which produces a closed area when combined with the coupled BLIs. Now, in 2D, the cross-over region is divided into two, by creating a connecting path via triangle *edges* between two L's (hereafter we will call it **L-to-L connection**) as shown in Figure 5.4(b), which produces purple and blue regions. Regional-Flip is not sensitive to the choice of both L-to-L or I-to-I connection as long as they do not meet, such that the cross-over region can be defined.

Flipping the cross-over region in a single step can be difficult; the two regions can be distant in the 3D space, or the intersection path can be too complex to define a feasible flip. So in our implementation, flip occurs over a

number of simulation time steps, starting from the vertices near the intersection path, such that the two BLI paths approach each other. Two exemplar results are shown in Figure 5.4(c) and 5.4(d). Depending on where the BLI paths meet, BLIs are converted into other types of intersection paths: BBII if the two L's meet first, LL if the two B's meet first, and both BBII and LL if the intersection meets at anywhere else. Notice that the above is an indirect approach, converting a pair of BLIs into other type(s) of intersection path(s) that can be resolved with the previous methods.

5.4.1 Crease-Flip

We sometimes encounter a special case of Regional-Flip. When *fold-remaining* is the resolution style of the BLI, the fold can be supposed to be sharp (e.g., crease for the design purpose in which the rest angle with respect to the fold is predefined) as shown in Figure 5.5(a). Fig 5.5(a) shows that, in 3D, the front red wing needs to be flipped to behind the grey body. Since the fold should remain, resolving this case (we call it Crease-Flip) has some similarities with the Regional-Flip. A difference from the Regional-Flip is that for this case, we do not need to define the cross-over region explicitly; the red sharp fold in Figure 5.5(b) is comparable to the L-to-L connection of the Regional-Flip, which can be identified by comparing the current fold angle and the predefined fold angle at the rest state; Crease-Flip does not require I-to-I connection since the step-by-step resolution is performed from the single BLI intersection path.

Other than the above differences, Crease-Flip proceeds similarly to the Regional-

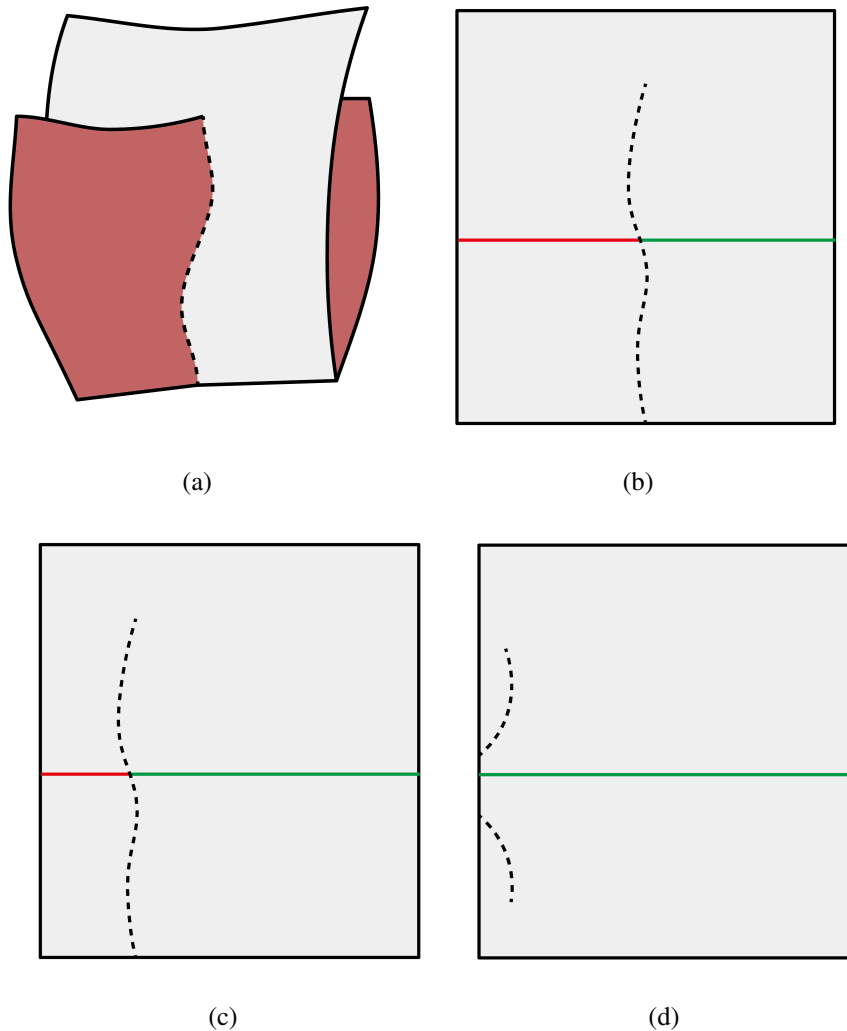


Figure 5.5 BLI being resolved by Crease-Flip. (a) shows the situation in 3D. (b) shows the same situation in 2D; notice that compared to Regional-Flip, the fold is significantly sharper. The horizontal lines in (b)~(d) represent the fold; the lines are colored in green if the current angle is close to the rest angle, and in red if not. (c) and (d) show a possible progression after (b), in which the BLI is converted into BIBI.

Flip. One possible step-by-step progression is shown in Figure 5.5(b) ~ 5.5(d), where the loop vertex translates through the red fold until it meets the end of

mesh, the BLI possibly disappearing or converting into BBII or BIBI.

5.5 Selecting Resolution Style/Algorithm

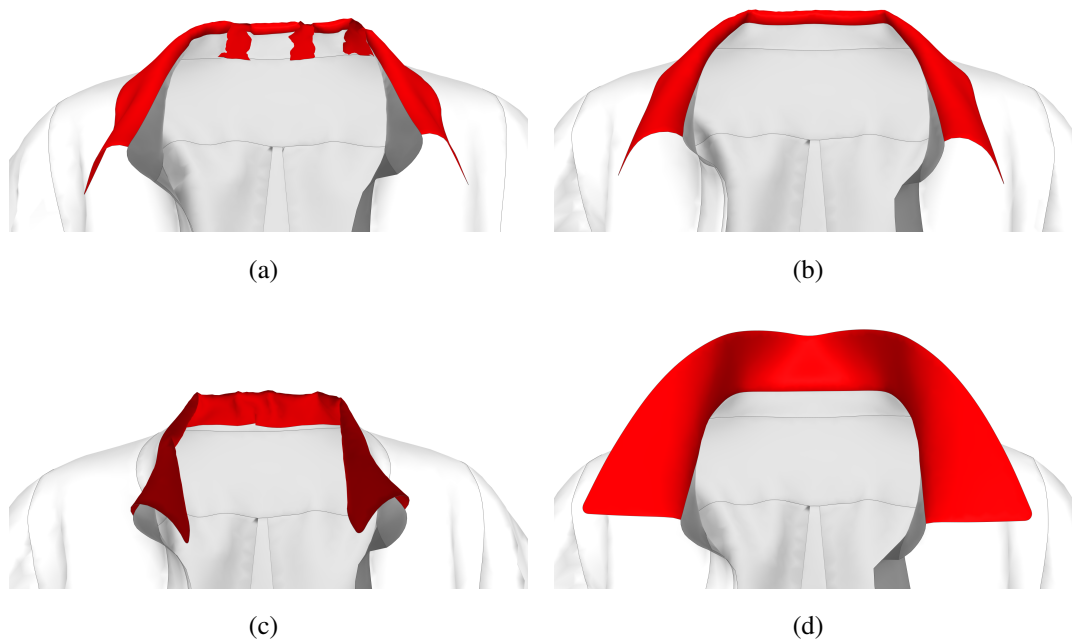


Figure 5.6 Shirts with a collar on a transparent avatar. (a) shows the initial situation in which multiple BLIs exist, (b) BLIs are all resolved with Regional-Flip, (c) BLIs are resolved with Regional-Flip but the collar is folded inward, (d) BLIs are resolved with Mesh-Tearing but the collar is raised upward. All three resolutions are intersection-free, but (b) is in general considered most appropriate.

In developing resolution algorithms for BLI, we assumed that the user selects the style, i.e., one among {fold-unfolding, fold-remaining} and the algorithm, i.e., one among {Mesh-Tearing, Regional-Flip, Crease-Flip}. In this section, we consider the possibility that the selection of the style/algorithm can

be done heuristically.

The most obvious case would be Crease-Flip, as its condition is clear. If a BLI is over a sharp fold (we can judge this by examining if the clothing design has a crease along the fold), Crease-Flip should be the algorithm to select. The above means that if there is no crease defined along the fold, Crease-Flip should not be selected (Figure 5.7).

The decision whether to apply Mesh-Tearing or Regional-Flip needs to be made. The originally intended way of wearing the garment (which may involve the *cultural* aspect) now matters. For example, in the case shown in Figure 5.6, BLIs occur in the *collar* of the shirt. Therefore, among Figure 5.6(b) ~ 5.6(d), Figure 5.6(b) is what is conventionally expected. For that purpose, we will assume that, where the intended way needs to be noted, the clothing data contain some kind of label to transfer that information to the program. For example, we assume that the red-colored sewing pattern in Figure 5.6 is labeled as ‘collar’. The above means the Regional-Flip (rather than Mesh-Tearing) should be selected as the resolution algorithm for this case. Note that Regional-Flip can produce either **folding-out** (Figure 5.6(b)) and **folding-in** (Figure 5.6(c)) depending on which/how BLIs are coupled. Since folding-in rarely occurs in wearing clothes, this work will assume folding-out is the correct choice for every Regional-Flip case.²

Referring to the flowchart in Figure 5.7, we determine the resolution style/algorithm

²Although we have never met such a case, if a Regional-Flip case has to be folded in, the situation should be somehow notified.

in the following way. Any arbitrary two neighboring BLIs from the left overs³ (excluding BLIs that can be processed by Crease-Flip) are examined whether they can be the BLI couple for the Regional-Flip. If applying Regional-Flip to the BLI couple results in folding-out, we process that BLI couple by the Regional-Flip. But if it results in folding-in, we process each of the two BLIs by Mesh-Tearing. (How to find whether applying Regional-Flip to the BLI couple results in folding-out or folding-in is explained in Appendix E.) The above should be done to every possible BLI pair, until no further valid couple is left. All remaining BLIs are now resolved by Mesh-Tearing.

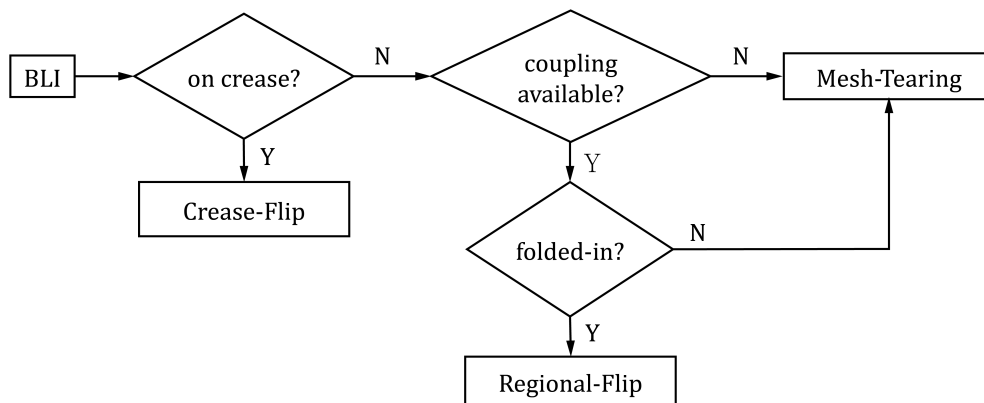


Figure 5.7 Flowchart for determining the resolution algorithm.

³In normal situations, there will be a small number of left over pairs.

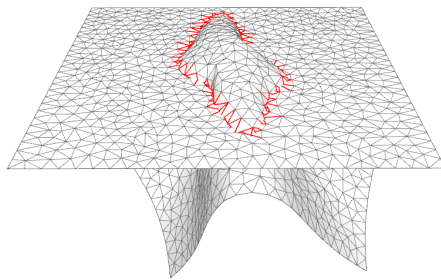
Chapter 6

Experiment Results

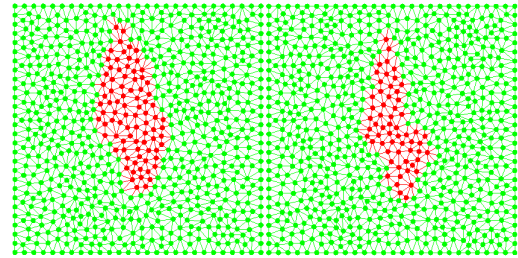
6.1 Overview

We built three types of simulators; all three simulators having the same basic architecture, by a combination of prior simulation models ([1, 11, 6]), with CCD acceleration methods ([43, 49, 12, 52, 67]). For reference, the first simulator was equipped with previous DCH methods ([2, 64, 75, 74]) with necessary measures to fit into the force-based simulator. The second and third simulator was each equipped with ESEF, ESEF & BLI-Resolver respectively. We will refer to the first as **SIM-PREV**, second as **SIM-ESEF** and third as **SIM-BLIR** later on. From Section 6.2 to Section 6.4, we compare the results between **SIM-PREV** and **SIM-ESEF**. From Section 6.5 to Section 6.8, we compare the results between **SIM-PREV** and **SIM-BLIR**.

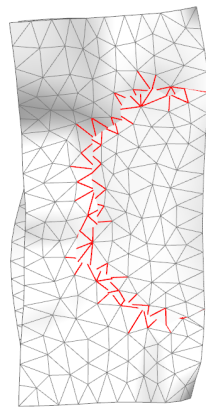
We note two important points in **SIM-BLIR**. [75] is practically the only



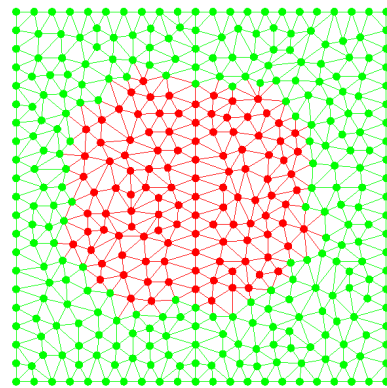
(a) CLOSED (3D)



(b) CLOSED (2D)



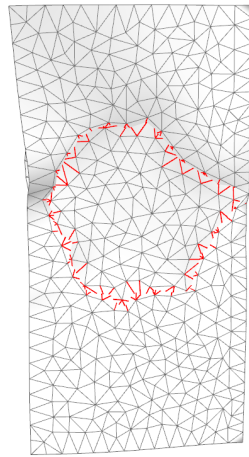
(c) LL (3D)



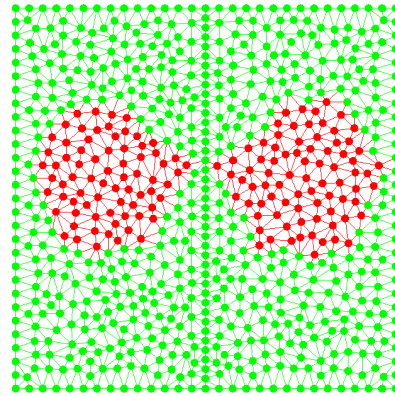
(d) LL (2D)

Figure 6.1 CLOSED & LL. In 3D, every edge that penetrates a triangle is shown in red as in Figure 3.1(a) and Figure 3.1(b), which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.

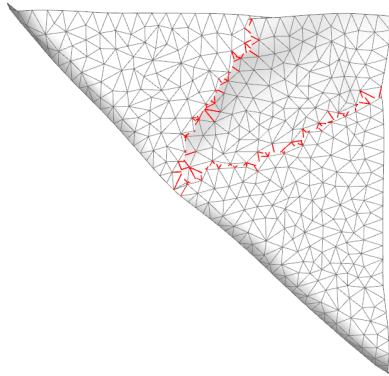
competitor to BLI-Resolver. Although [75] did not explicitly mention BLI, experiments show that it is capable of taking care of some BLIs. So to be scientific, we should compare [75] with BLI-Resolver only on BLIs. However, as



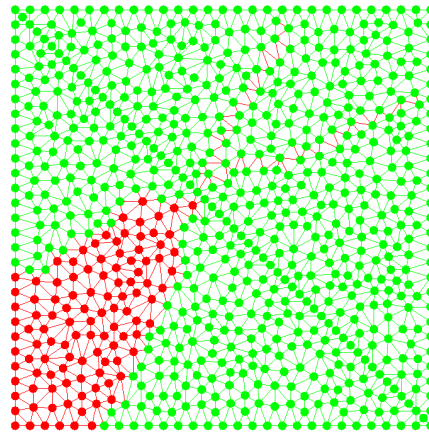
(a) EIGHT (3D)



(b) EIGHT (2D)



(c) CROSS (3D)



(d) CROSS (2D)

Figure 6.2 EIGHT & CROSS. In 3D, every edge that penetrates a triangle is shown in red as in Figure 3.1(a) and Figure 3.1(b), which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.

the clothing simulation progresses, the situation cannot be controlled such that only BLIs occur. Under the circumstances, resolving BLIs only can leave the clothing tangled due to the other types of intersections, making the comparison

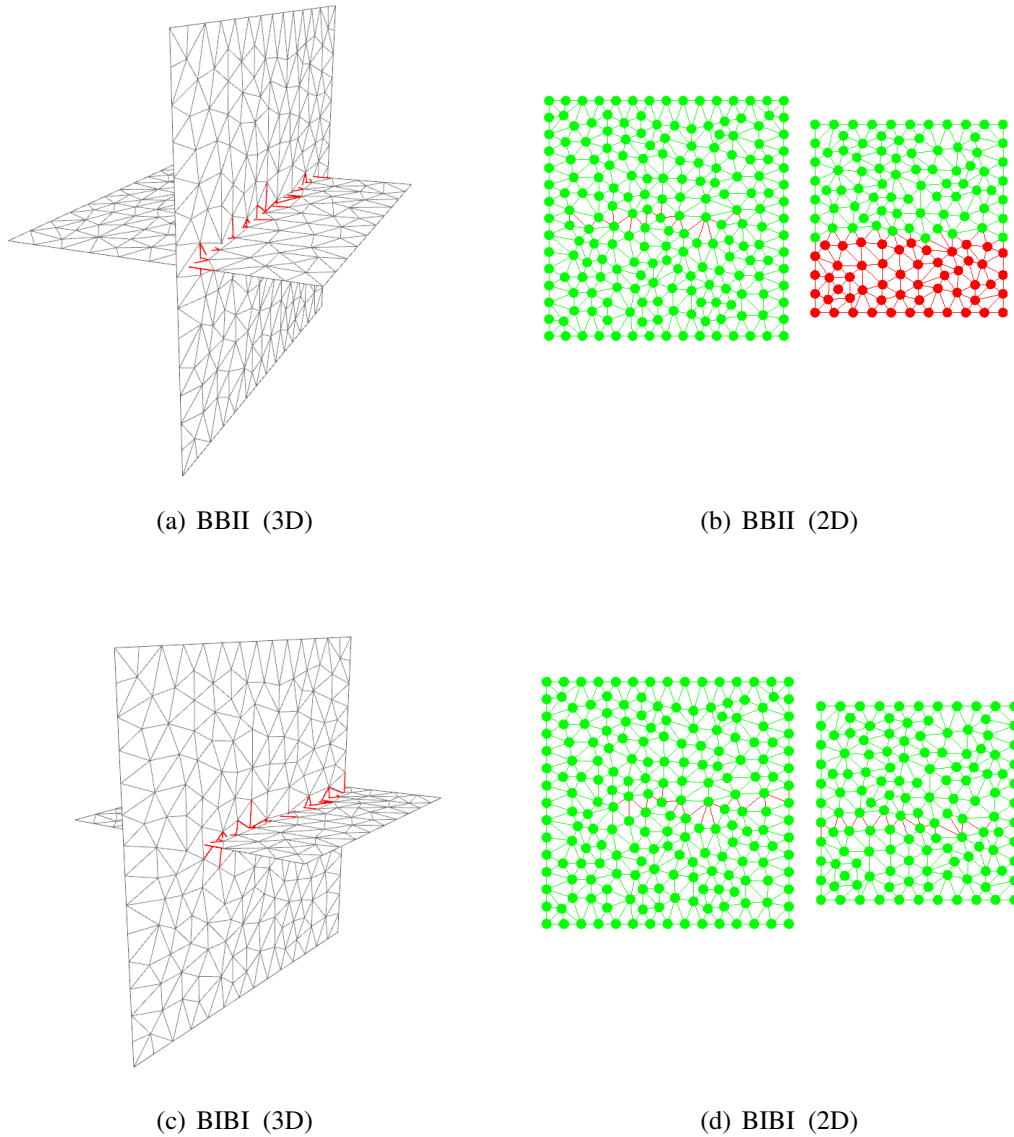


Figure 6.3 BBII & BIBI. In 3D, every edge that penetrates a triangle is shown in red as in Figure 3.1(a) and Figure 3.1(b), which conspicuously labels where the tanglements exist. In 2D, elements are colored according to the scheme introduced in Section 3.

itself difficult. This is why SIM-BLIR is not only equipped with BLI-Resolver, but also ESEF. Other than the above, when running BLI-Resolver, the resolu-

tion algorithms for BLIs were automatically selected according to the flowchart in Figure 5.7 except for Section 6.8.

6.2 Rudimentary Cases

We set up a handkerchief (or two handkerchiefs) to produce the above six cases as shown in Figure 6.1, 6.2 and 6.3. 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. ESEF 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.

6.3 Exploded Handkerchief

In this experiment, a handkerchief consisting of 388 triangles was shuffled 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

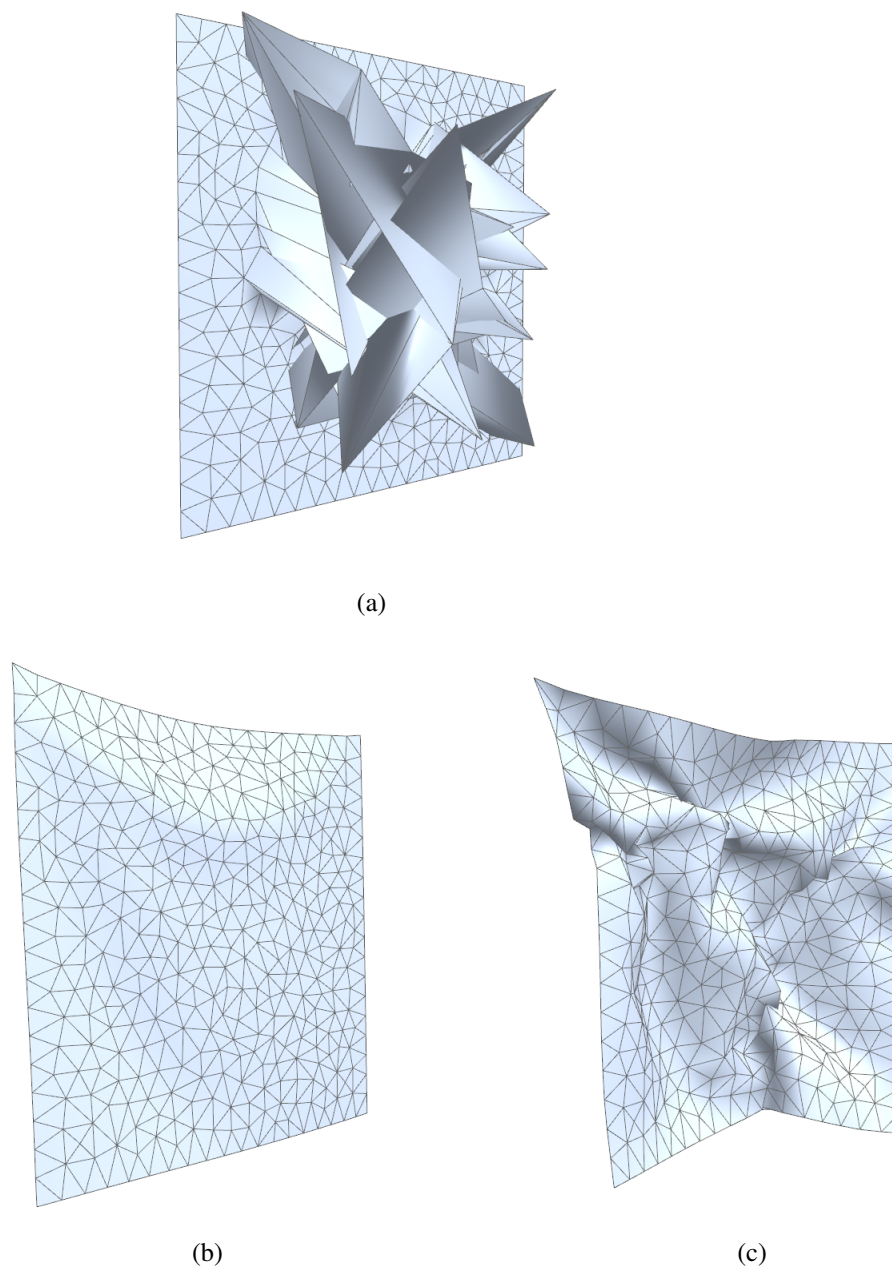


Figure 6.4 Exploded handkerchief. (a) initial tanglement, (b) after running the proposed method 75 time steps, (c) after running SIM-PREV additional 2,000 time steps after the first 75 time steps.

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. On average, one red triangle intersected with 47 other red triangles. The proposed method resolved tanglements without any failure (Figure 6.4(b)) in 75 time steps. With SIM-PREV, three LL cases were left unresolved even after 2000 time steps as shown in Figure 6.4(c) 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) Repulsive-ICM may not produce correct/consistent vertex movement direction for the resolution.

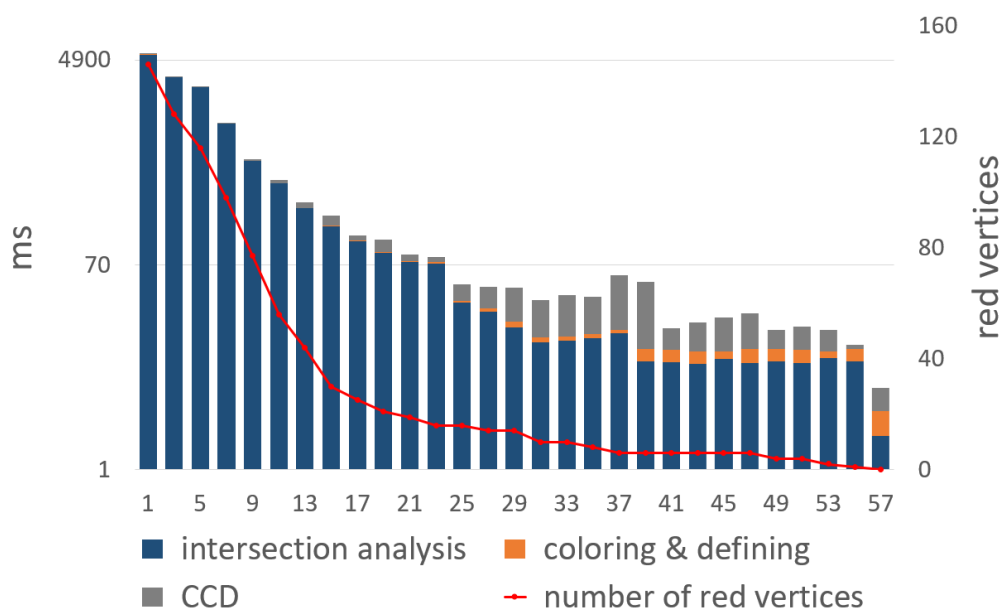


Figure 6.5 Plotting of the resolution time and red vertex count. The bar graph represents the time taken for the resolution in log scale, and the red curve represents the number of remaining red vertices.

Figure 6.5 shows how the proposed method performed in resolving the

scene shown in Figure 6.4. The x -axis represents the frame number. Figure 6.5 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. A more detailed discussion about the choice of the γ value is placed in Appendix D.

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 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.



Figure 6.6 Experiments on clothing cases. (a)~(d) shows the initial tanglements. (e) is a capture during a session of interactive tanglement generation and resolution.

6.4 Clothes

Tanglements in four ensembles shown in Figure 6.6(a) ~ 6.6(d) were created by deliberately disabling self-collision handling for a few time steps. BLIs could be introduced, which were manually removed before running SIM-ESEF and SIM-PREV. As shown in Figure 6.7(a) ~ 6.7(d), the proposed method resolved all cases successfully. With SIM-PREV, however, there were resolution



Figure 6.7 Experiments on clothing cases. (a)~(d) show the results after executing SIM-ESEF to Figure 6.6(a) ~ 6.6(d), respectively. (e) shows the results after 500 times steps of simulation with the proposed method.

failures in all four cases, as shown in Figure 6.8(a) ~ 6.8(d), even after we ran 1,000 additional time steps compared to those required for Figure 6.7(a) ~ 6.7(d), respectively.

Figure 6.6(e) 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 Figure 6.7(e), except for a single



Figure 6.8 Experiments on clothing cases. (a)~(d) show the results after executing SIM-PREV to Figure 6.6(a) ~ 6.6(d), respectively. (e) shows the results after 500 times steps of simulation with SIM-PREV.

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 on average. The accompanying video contains the capture of the interactive session. With SIM-PREV, however, some non-BLI cases were left unresolved as shown in Figure 6.8(e).¹

¹In this test, in which new tanglements were interactively generated while some were being resolved, we could not run the proposed method and SIM-PREV in the strictly same condition.

In the clothing examples, the failure included CLOSED, BBII, and BIBI cases, as well as LL. Since the methods in SIM-PREV have upper bounds in the force/displacement they can apply for the resolution, when there exists a tanglement that calls for a larger force/displacement, the resolution can fail. The above implies that the failure of SIM-PREV is not necessarily limited to {LL, CLOSED, BBII, BIBI}.² 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 4.5. 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 Appendix A and B, respectively. Table 6.1 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 was never needed.

A curiosity here is whether the handling of the non-planar cases (i.e., OnSeam-

However, we note that, as we perform the test multiple times, if we exclude BLI, the proposed method was always successful while SIM-PREV was not.

²If experimented in different conditions, SIM-PREV on the exploding handkerchief case may also have produced CLOSED, BBII, BIBI (and possibly other new) failures.

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 6.1 shows, OnSeam took only about 1.76 additional time steps at most compared to OffSeam. It experimentally indicates that the procedures proposed for OnSeam emulated that for OffSeam quite well.

Scene	vertex type	#(*)	%(*)	t(*)
Figure 6.6(a)	OffSeam	571	92.24	6.10
	OnSeam-A	42	6.79	7.86
	OnSeam-B	6	0.97	6.30
Figure 6.6(b)	OffSeam	1390	91.27	9.88
	OnSeam-A	121	7.94	10.57
	OnSeam-B	12	0.79	11.01
Figure 6.6(c)	OffSeam	898	90.25	7.53
	OnSeam-A	75	7.54	7.10
	OnSeam-B	22	2.21	6.90
Figure 6.6(d)	OffSeam	1,451	91.09	6.84
	OnSeam-A	68	4.27	7.02
	OnSeam-B	74	5.64	6.59
Figure 6.6(e)	OffSeam	2,192	82.81	5.76
	OnSeam-A	357	13.49	5.90
	OnSeam-B	98	3.70	5.11

Table 6.1 #(*), %(*) and t^* represent the occurrences of each case, percentage, and average number of time steps required, respectively.

6.5 Round Folds

A round fold in the collar is a vulnerable spot to collision failures, as shown in Figure 5.6(a). According to Section 5.5, in such a case, Regional-Flip was selected, and that algorithm successfully produced the result shown in Figure 5.6(b).

6.6 Sharp Folds

Lapel and collar are prone to create collision failures with the bodice. When the avatar is suddenly switched to another avatar or the garment is switched to another size (in the context of virtual try-on techniques such as [17]), the situation shown in Figure 6.9(a) can occur, in which the lapel/collar intersects with the bodice, creating multiple BLIs along with a number of other types of intersections. With the provided design information (i.e., the crease along the lapel and collar), the proposed method applied the Crease-Flip along the lapel/collar fold line, and quickly produced the configuration shown in Figure 6.9(b).

6.7 User Interactions

In some cases, the user may grab and drag a part of the clothing (in order to obtain a proper fit or to create more natural wrinkles), which can create new intersections including BLIs. Figure 6.10 shows the case when the user

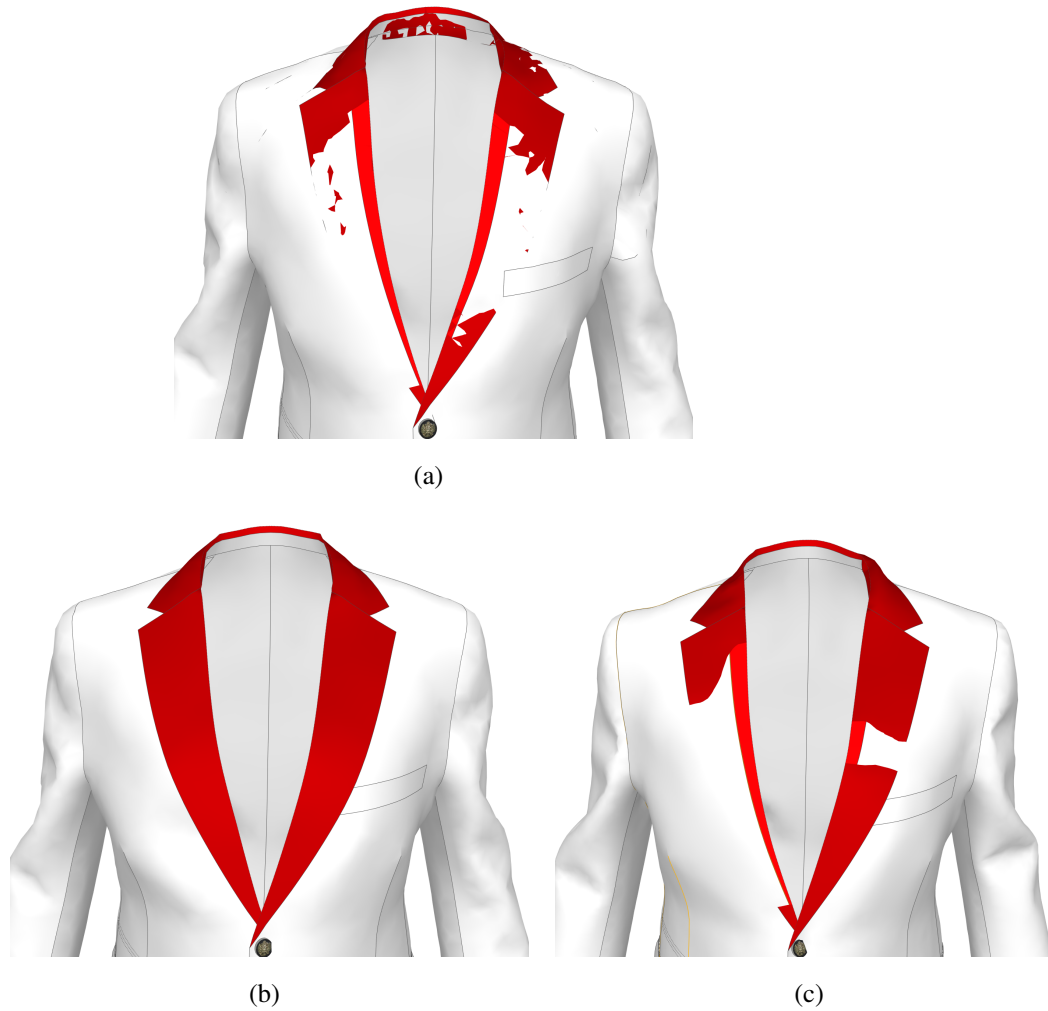


Figure 6.9 Crease-Flip applied to sharp-folded lapel and collar. (a) is the given setup which contains multiple intersections including BLIs. (b) shows the resolution with the SIM-BLIR, and (c) shows the resolution with the SIM-PREV.

drags up a sleeve. 78 intersections were created, among which 7 were BLIs. The panels apart from the sleeves were made to remain fixed to their initial draping, in order to provide a fair condition to this test. SIM-PREV left 3 BLIs unresolved (Figure 6.10(d)), but SIM-BLIR resolved all tanglesments

(Figure 6.10(b)). Throughout the execution of SIM-BLIR, for this particular case, the program selected only the Mesh-Tearing (according to the flowchart shown in Figure 5.7).

As we repeat the sleeve test, we note that it was not resolved as shown in Figure 6.10(b) in all cases. This does not mean that the proposed method failed (since there were no intersections left), but that the sleeve was not fully extended to its original shape. As the implemented simulator has an asymmetric bending model([11]), depending on which wrinkle is formed first, the outcome had a little variance as shown in Figure 6.10(c).

As another user interaction case, the user pulled up the bottom of a skirt as shown in Figure 6.11(a), which created 125 intersections including 12 BLIs. As we ran the resolution programs, SIM-PREV left 4 BLIs unresolved (Figure 6.11(c)), while SIM-BLIR resolved all the tanglements including the BLIs (Figure 6.11(b)). Throughout the execution of SIM-BLIR, again the program selected only the Mesh-Tearing.

6.8 Exploded Handkerchief

In this experiment, a handkerchief consisting of 388 triangles was shuffled to be in a tangled state. In contrast with Section 6.3, we now no longer need to worry about BLIs, thus the re-positioning is not limited to the central area. All vertices except for the two vertices on the top left and right corner were re-positioned; intersection analysis performed at the first frame (Figure 6.12(a))

resulted in 32 BLIs and 3549 other types of intersections. We fixed the resolution algorithm to Mesh-Tearing for this particular case.

SIM-BLIR untangled the handkerchief completely (Figure 6.12(b)), whereas SIM-PREV left 7 BLIs unresolved (Figure 6.12(c)). We repeated the Exploded Handkerchief test with new randomized setups detailed in the video. SIM-BLIR produced complete resolution with no exception, but SIM-PREV left 2 to 15 BLIs unresolved.

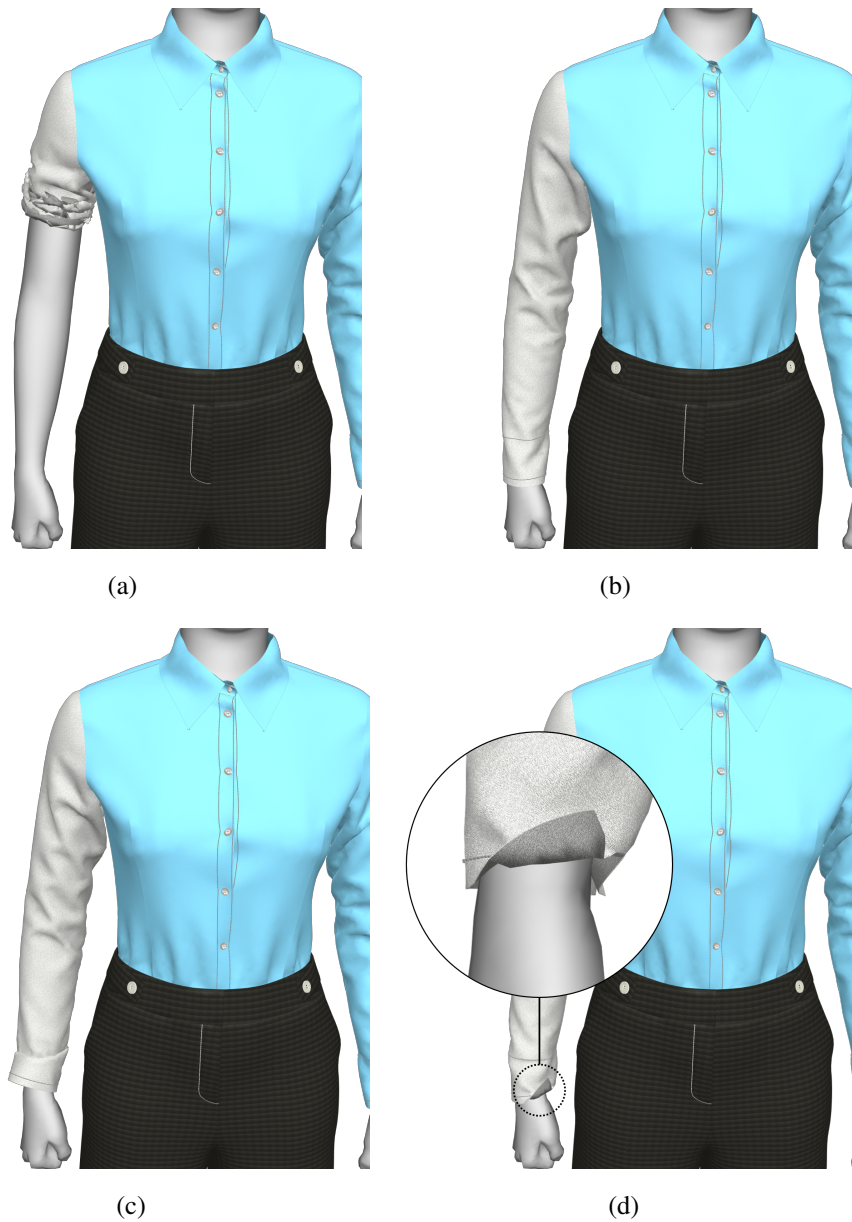


Figure 6.10 Mesh-Tearing applied to sleeve. Panels in blue indicate that they remain fixed during the test. (a) shows the initial state in which the sleeve is dragged up. (b) shows the result with SIM-BLIR, which left no intersection. (c) shows the result of another trial with SIM-BLIR, in which no intersection is left but the sleeve is left folded up. (d) shows the result with SIM-PREV, in which some BLIs are left unresolved.

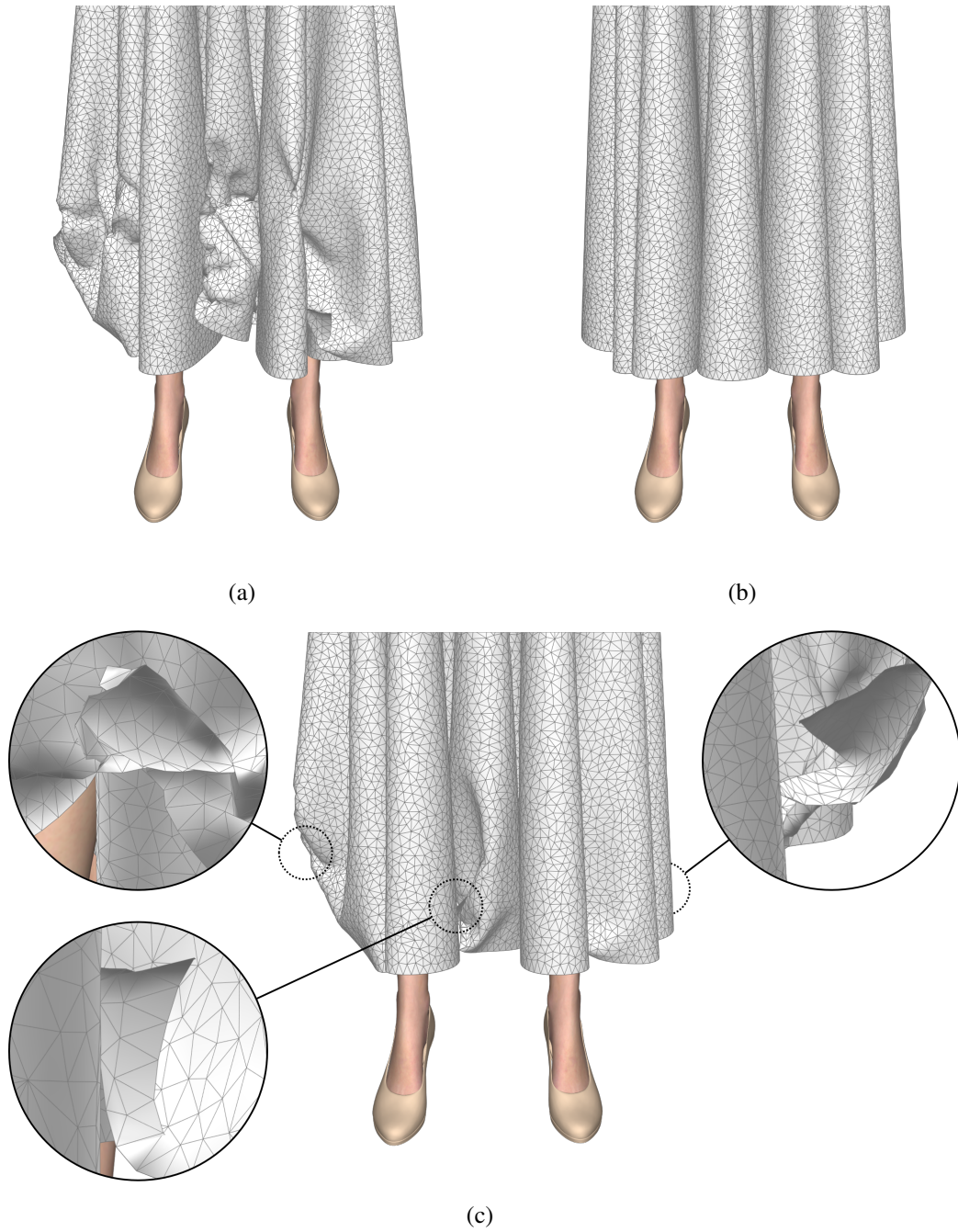
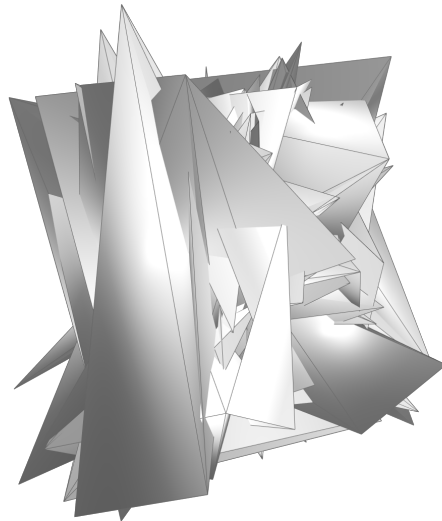
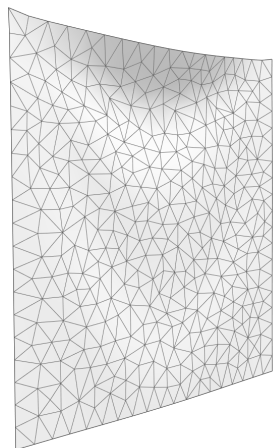


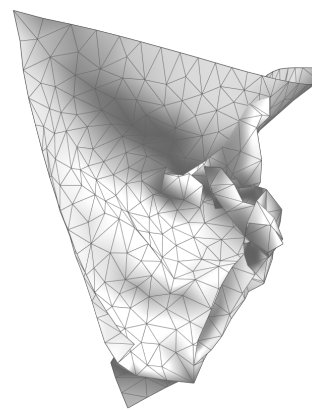
Figure 6.11 Mesh-Tearing applied to the skirt. (a) shows the initial state in which the bottom of the skirt is pulled up. (b) shows the result with SIM-BLIR. (c) shows the result with SIM-PREV.



(a)



(b)



(c)

Figure 6.12 Exploded handkerchief. (a) shows the initial state, (b) with SIM-BLIR (c) with SIM-PREV.

Chapter 7

Conclusion

In this paper, we split intersections into two groups (non-BLI and BLI), find that they require different strategies, and proposed ESEF and BLI-Resolver to handle each of them respectively. The first method ESEF resolves its target group in finite time steps by re-meshing in the uv-space according to the novel coloring method and taking a new perspective of the CCD formulation. The key ideas of ESEF were to (1) utilize the tolerance value in the CCD formulation (ϵ_{CCD}) for the purpose of tanglement resolution, and (2) re-mesh in the uv-space to generate a sufficient size of force that pressures the simulation to untangle.

By the coloring, ESEF starts untangling from the edge of the tanglement in an out-to-in manner, thus the operation can be considered as if *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 try to

resolve the whole tanglements at once, our method can be slower. But we would like to note that the method does not require an impractically large amount of computation. In typical clothing setup of Figure 6.6(a) ~ 6.6(d), it took 161 time steps in average for the complete resolution.

The theorem inside ESEF holds for intrinsically planar meshes, and is extended to non-planar cases with two heuristic procedures. We have never encountered a case in which condition (ii) of the theorem is not met. As for condition (i), it is 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 second method BLI-Resolver, resolves BLI by proposing three resolution algorithms and a decision-making flowchart of when to apply which algorithm. The key idea of BLI-Resolver was to (1) deliberately ignore certain triangles that were holding back the resolution, and (2) involve the clothing design to provide the optimal resolution direction.

Certain types of clothing (jacket, shirt, suit, etc...) are easily exposed to BLI, but yet the subject has not been studied much. With the premise that ESEF is available for the other six types of intersections, BLI-Resolver can fully focus on BLI. It analyzed how BLIs occur, and noted that the desired form of resolution (i.e., resolution style) can vary depending on the type or particular region of the garment. We identified there should be two resolution styles for BLI, namely, fold-unfolding and fold-remaining, and developed three resolution algorithms, namely, Mesh-Tearing, Regional-Flip, and Crease-Flip.

In summary, with ESEF, we can guarantee the resolution of non-BLI tangles for intrinsically planar cases in finite time steps, when shortening operation is not hindered, and when there is at least one TIT-passable fan. In addition to BLI-Resolver, resolution style/algorithms expand the DCH capability, when information is extracted from the clothing design. BLI-Resolver covers the blind spot of ESEF. With the two combined, we can now say that the whole spectrum of intersections is covered. Both methods can be implemented with only a small modification to the conventional simulator (that has a full-proof CCH). Under the assumption that the garments are given with some necessary information that is relevant to the resolution styles/algorithms, ESEF and BLI-Resolver can create conventionally plausible simulation results unattended.

Appendix A

Edge Shortening When Intersection

Path Exists Across Multiple Panels

When an out-most red vertex to be pulled lies on the seam, we can use the procedure presented in this section for that pull by regarding the subject out-most red vertex as \mathbf{x}_0 below. Note that the red* triangle shrinkage consists of two vertex pulls. Therefore, if either (or both) of the target vertices lie on the seam, we can process it (or them) with the same procedure.

Let's suppose that three panels P_0 , P_1 , and P_2 are seamed in a garment and the vertex \mathbf{x}_0 needs to be pulled. Figure A.1(a) draws only the immediately involved triangles of the panels, i.e., $F_0 = \text{fan}(\mathbf{x}_0)$, $F_1 = \text{fan}(\mathbf{x}_1)$, and $F_2 = \text{fan}(\mathbf{x}_2)$. In the figure, the seamed line pairs are drawn in the same color. The vertices of F_0 , F_1 , and F_2 have been resolved in their own uv -coordinate systems C_0 , C_1 , and C_2 , respectively. To be more specific about the above pulling,

Appendix A. Edge Shortening When Intersection Path Exists Across Multiple Panels⁹⁰

let's suppose that \mathbf{x}_0 needs to be pulled along the unit vector \mathbf{y}_0 by η in the coordinate system C_0 .

We define the **left** and **right** as we look out the fan circumference from the fan center. For example, in F_0 of Figure A.1(a), we can say (1) the red edge is on the right side of the yellow edge, and (2) the red edge is at the right extreme, etc. (We will use the above definitions also in the sector forms below.)

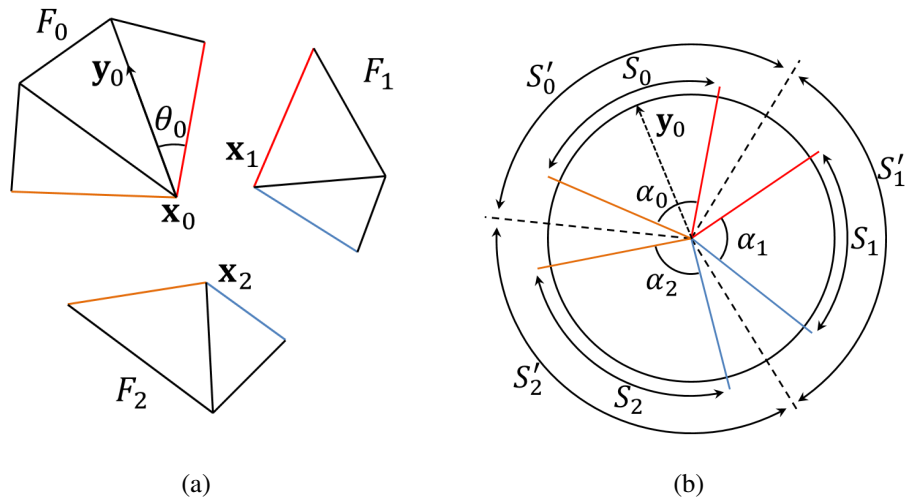


Figure A.1 Handling of the case when three panels are seamed at the subject vertex \mathbf{x}_0 . (a) the triangle fans that are directly adjacent to \mathbf{x}_0 . (b) the sector form version of (a).

Now, referring to Figure A.1(a), let's suppose that the angle \mathbf{y}_0 makes with the right extreme edge of F_0 is θ_0 . Then, the question is, to conform with the pulling of \mathbf{x}_0 in F_0 , how \mathbf{x}_1 in F_1 and \mathbf{x}_2 in F_2 should move in their own coordinate systems. We assume that the same length (i.e., η) is moved. Then, we need to know only the moving direction.

Appendix A. Edge Shortening When Intersection Path Exists Across Multiple Panels 91

Figure A.1(b) is an abstraction of Figure A.1(a), which is constructed in the following way:

1. Let's call the angle made by the left and right extreme edges of a fan as the **span** of the fan. We abstract F_i ($i = 0, 1, 2$) to the **sector forms** S_0 , S_1 , and S_2 shown in Figure A.1(b), in which only the span matters. (The other details such as the number of triangles each fan is comprised of does not make any difference.)
2. While preserving its orientation (of F_0), translate S_0 so that the corner \mathbf{x}_0 comes to the origin, as shown in Figure A.1(b). Note that the position of S_0 is *fixed* hereafter.
3. Let the spans of S_0 , S_1 , and S_2 be α_0 , α_1 , and α_2 , respectively. The basic idea here is to expand (or shrink) the spans so that they cover 360° . For that purpose, we will put some allowance in both sides of S_i . The resultant allowed sector forms (shown with dashed lines in A.1(b)) will be denoted as S'_0 , S'_1 , and S'_2 , and their spans will be denoted as α'_0 , α'_1 , and α'_2 , respectively. We find a constant k such that $k(\alpha_0 + \alpha_1 + \alpha_2) = 360^\circ$. Then, we use $\alpha'_i = k\alpha_i$, for $i = 0, 1, 2$.
4. The allowances are calculated in the following way. For S'_1 and S'_2 , we calculate the left and right allowances as

$$A_i^L = A_i^R = \frac{1}{2}\alpha_i(k-1) \quad (i = 1, 2) \quad (\text{A.1})$$

Appendix A. Edge Shortening When Intersection Path Exists Across Multiple Panels 92

For S'_0 , however, we calculate the right allowance as

$$A_0^R = \frac{\theta_0}{\alpha_0} \alpha_0 (k-1) \quad (\text{A.2})$$

and the left allowance as

$$A_0^L = \left(1 - \frac{\theta_0}{\alpha_0}\right) \alpha_0 (k-1) \quad (\text{A.3})$$

Since the position of S_0 has been fixed in Step 2, the above calculation of A_0^R and A_0^L *determines* the position of S'_0 . Then, by referencing the adjacency of the panels in 3D, we can now determine the position of S'_1 and S'_2 (since, together with S'_0 , they should fill up 360°). Referencing A_i^L and A_i^R ($i = 1, 2$), we can now determine S_1 and S_2 . The intuition behind the above steps is we span-wisely expand (or shrink) S_0 , S_1 , and S_2 to create a *planar* situation. The reason we use different allowance formulae for A_0^L and A_0^R is to make \mathbf{y}_0 point the same direction after the expansion.

5. In terms of S'_0 , S'_1 , and S'_2 , the situation is planar. Thus in this expanded configuration, as \mathbf{x}_0 is pulled along \mathbf{y}_0 , we can make F_1 and F_2 conform to it by making both \mathbf{x}_1 and \mathbf{x}_2 translated along that direction in Figure A.1(b). Since Figure A.1(b) is the result of applying transformations, more useful information would be the angle θ_i that \mathbf{y}_0 makes with the left extreme edge of each fan, which will have the same meaning in its

Appendix A. Edge Shortening When Intersection Path Exists Across Multiple Panels⁹³

own coordinate system. It is straightforward to derive

$$\theta_1 = A_1^L + A_0^R + \theta_0, \quad (\text{A.4})$$

$$\theta_2 = A_2^L + A_1^R + \alpha_1 + \theta_1. \quad (\text{A.5})$$

6. In summary, \mathbf{x}_i ($i = 1, 2$) should be translated by η with angle θ_i from the left extreme edge of the fan, measured in the counterclockwise direction.

The above derivation was done for the three fan case, but can be straightforwardly generalized to the case where the number of seamed fans is arbitrary.

Appendix A. Edge Shortening When Intersection Path Exists Across Multiple Panels⁹⁴

Appendix B

Edge Shortening When Intersection

Path Exists Across the Dart Opening

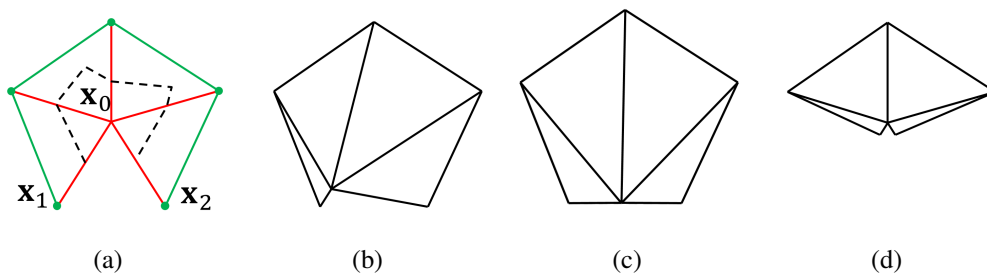


Figure B.1 x_1 , x_0 , and x_2 form a dart, in which $E(x_0, x_1)$ is seamed with $E(x_0, x_2)$. We will call it the **dart seam**. (a) initial tanglement. (b) pulling x_0 toward x_1 without considering the dart seam. (c) pulling x_0 downward while keeping the length of $E(x_0, x_1)$ and $E(x_0, x_2)$ the same. (d) pulling x_1 and x_2 toward x_0 .

In the dart, seamed edges can share a common vertex in the uv -space (e.g., x_0 in Figure B.1(a)). Let's suppose that the fan center x_0 needs to be pulled out and $E(x_0, x_1)$ is chosen as the subject edge. If we process this case naïvely,

Appendix B. Edge Shortening When Intersection Path Exists Across the Dart Opening⁹⁶

\mathbf{x}_0 will approach \mathbf{x}_1 . However, note that $E(\mathbf{x}_0, \mathbf{x}_1)$ and $E(\mathbf{x}_0, \mathbf{x}_2)$ are identical in 3D. Applying edge shortening to $E(\mathbf{x}_0, \mathbf{x}_1)$ without accounting for that situation can lead to the configuration shown in Figure B.1(b), in which $E(\mathbf{x}_0, \mathbf{x}_2)$ can resist the shortening of $E(\mathbf{x}_0, \mathbf{x}_1)$. This section presents how to cope with this situation.

Until now, re-meshing of the subject fan (for pulling out the red vertex at the center) has been done by changing the position of the center vertex while leaving the fan vertices stationary. But in the current case, no matter where we position the center vertex (i.e., \mathbf{x}_0), either (1) the other corresponding seamed edge resists, or (2) if we shorten while maintaining the length of the two edges the same, the shortening may not be done in the full extent as shown in Figure B.1(c), since $E(\mathbf{x}_0, \mathbf{x}_1)$ can shorten only up to the half of the original \mathbf{x}_1 -to- \mathbf{x}_2 distance.

In this case, we leave the center vertex stationary, but change the position of the two circumference vertices. More specifically, as demonstrated in Figure B.1(d), shortening of $E(\mathbf{x}_0, \mathbf{x}_1)$ (and $E(\mathbf{x}_0, \mathbf{x}_2)$) should be achieved by (1) moving \mathbf{x}_1 toward \mathbf{x}_0 , and at the same time (2) moving \mathbf{x}_2 toward \mathbf{x}_0 by the same amount. The above edge shortening will not experience the resistance discussed above.

Appendix C

Convexification

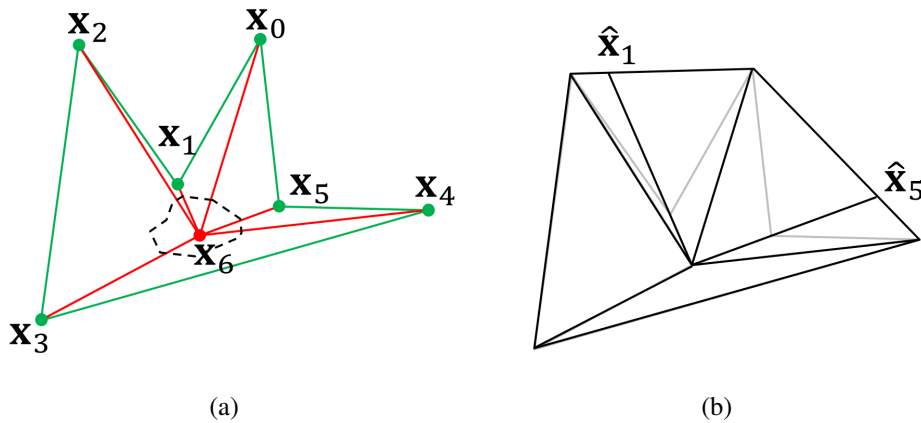


Figure C.1 A case that needs the convexification heuristic. (a) $\text{fan}(x_6)$ is not TIT-passable. (b) the result of convexification.

In the process of resolution, we can encounter an out-most red vertex x_i for which $\text{fan}(x_i)$ is not TIT-passable. In such a case, the scheduling algorithm in Section 6 postpones the processing of x_i and processes other TIT-passable out-most red vertices, which may after all make $\text{fan}(x_i)$ TIT-passable. But in a certain case, waiting cannot be an option. Figure C.1 is a case in which

only one red vertex is left and the fan is not TIT-passable. This is a violation of the condition (i) of the theorem, thus is out of the scope of the theorem. However, we suggest a simple (but not complete) heuristic procedure that can be used when such a case ever occurs.

The idea is to make the fan convex by re-positioning the vertices that create the concavity, as shown in Figure C.1(b). \mathbf{x}_1 and \mathbf{x}_5 are the problematic vertices, so we translate them outwards until they do not create concavity anymore, i.e., to $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_5$ in Figure C.1(b).

In an extreme case, we cannot rule out the possibility the new positioning (e.g., $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_5$) produces a triangle inversion, for which this paper does not have a remedy. We report that, in the experiments performed so far, we have not encountered any occasion that calls for the convexification heuristic in the first place.

Note that the red* triangle shrinkage can be viewed as consisting of two vertex pulls (let's call those two vertices as \mathbf{x}_i and \mathbf{x}_j), thus the above procedure can be modified to cover the red* triangle shrinkage by convexifying the $\text{fan}(E(\mathbf{x}_i, \mathbf{x}_j))$.

Appendix D

Discussion on the Values of ϵ_{RG} and γ

The stability of the simulator was not considered in the theorem itself. As a matter of fact, if the stability is not an issue, as long as ϵ_{RG} and γ are taken from the range $(0,1)$, the theorem holds regardless of the particular choice of ϵ_{RG} and γ . But the stability has to be considered in the actual implementation, since we assume a typical force-based simulator that can become unstable if (1) a vertex is abruptly displaced too far, or (2) an edge is shortened too rapidly. In this section, we discuss which values of ϵ_{RG} and γ in the range $(0,1)$ do not cause simulation instability. Note that, the soundness of the proposed resolution algorithm (Section 7) still holds if, for each of ϵ_{RG} and γ , we can find a non-empty set within $(0,1)$ which is free from simulation instability.

As explained in Section 5.3.1, we use $\epsilon_{RG} = 0.1 * \text{avg}(l)$ in the implementation. Using an even larger ϵ_{RG} may produce further speed up in the finesse, but it may cause some instability in the simulation.

When a red-red elementary pair turns into a red-green elementary pair, the tolerance value abruptly increases, from zero to ϵ_{RG} , which causes the pair to be separated by ϵ_{RG} , without accounting for how much internal energy the re-positioning incurs. An example is shown in Figure D.1, which chronologically shows the changes in the world-space mesh. Figure D.1(a) shows the situation just after the Intersection Analysis of time step n . You can see that \mathbf{x}_5 is at least ϵ_{RG} away from $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ due to m-CCR performed for time step $n - 1$. \mathbf{x}_4 is then pulled toward \mathbf{x}_5 , and Figure D.1(b) shows the situation after the World-Space Mesh Update of time step n . \mathbf{x}_5 is still ϵ_{RG} away, but ϵ_{RG} 's effect on that vertex is not shown since we now focus on the tolerance applied to \mathbf{x}_4 . It turns out that \mathbf{x}_4 has crossed $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$, but this fact is not known to the simulator until it runs the Intersection Analysis of time step $n + 1$ which changes the color of \mathbf{x}_4 to green as shown in Figure D.1(c). Since $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ is still red, the tolerance value ϵ_{RG} should be used between $T(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ and \mathbf{x}_4 . Therefore, after m-CCR of time step $n + 1$, \mathbf{x}_4 experiences a sudden displacement as shown in Figure D.1(d). Here, if ϵ_{RG} is too large, instability can occur in the force-based simulation. According to experiments, however, the instability never occurred as long as $\epsilon_{RG} \leq 0.3 * \text{avg}(l)$.¹²

¹In terms of the stability range, the simulation is stable if we take ϵ_{RG} from the range $(0, 0.3 * \text{avg}(l)]$.

²We note that taking ϵ_{RG} from that range is just one way to avoid simulation instability related to ϵ_{RG} . For example, for each was-green-now-red vertex, we may use the sequence $(\frac{1}{3}\epsilon_{RG}, \frac{2}{3}\epsilon_{RG}, \epsilon_{RG})$, i.e., gradually increasing ϵ_{RG} over three time steps. Then, we can use a larger value for ϵ_{RG} . The essence of the proposed method is to apply *unlimited shortening* until the given tanglement is resolved. In doing so, the tolerance/rate control (i.e., the control of ϵ_{RG} and γ) for stability is an implementation-related issue.

Apart from the simulation instability, the sudden tolerance value changes (zero-to- ϵ_{RG} or ϵ_{RG} -to- ϵ_{CCD}) result in jerky movements in the cloth when watching the resolution process. We do not claim that our resolution is physical, but still prefer the process to be as gentle as possible. For this reason, we picked $0.1 * \text{avg}(l)$ for ϵ_{RG} for the implementation.

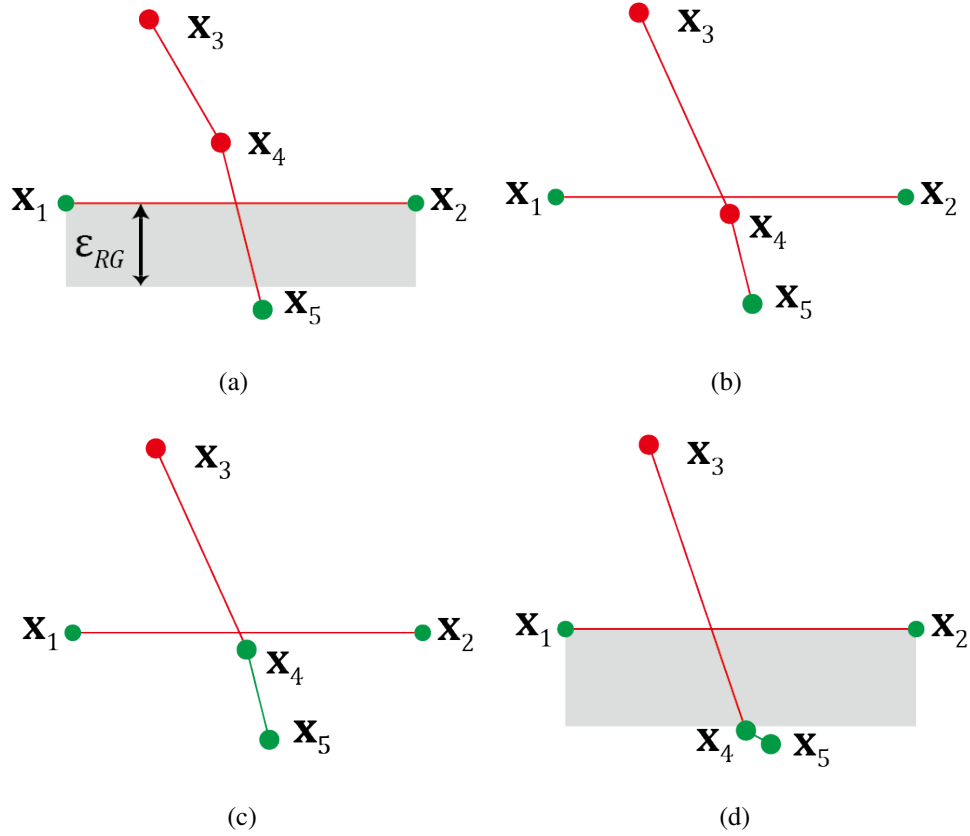


Figure D.1 The scenario in which ϵ_{RG} can produce a sudden vertex displacement. (a) after the Intersection Analysis of time step n . (b) after World-Space Mesh Update of time step n , (c) after Intersection Analysis of time step $n + 1$, (d) after m-CCR of time step $n + 1$.

We now discuss the instability that can be caused by the value of γ . If γ is too small, the uv -space deformation will occur rapidly. The world space

result may not be able to catch up the uv -space deformation in the given time step, and the out-date can accumulate as the simulation progresses. This leads to extraordinarily large forces/accelerations in force-based simulators, making the simulator unstable.

When using the time step $\Delta t = 0.02s$, we found that γ_{min} , the smallest value of γ that is safe from instability, is 0.8 from experiments.³ Here, we note that γ_{min} depends on Δt . Let $\gamma_{min}(\Delta t)$ be the γ_{min} when the time step size is Δt . If the simulation should run with $\Delta t = 0.01s$ instead of $0.02s$, then we note that $\gamma_{min}(0.01s)$ should be $\gamma_{min}(0.01s) = \sqrt{\gamma_{min}(0.02s)}$, so that the shortening over two time steps with $\Delta t = 0.01s$ match the shortening in a single step with $\Delta t = 0.02s$. For the general Δt , the compensation should be

$$\gamma_{min}(\Delta t) = (\gamma_{min}(0.02s))^{\frac{\Delta t}{0.02}}. \quad (D.1)$$

³In terms of the stability range, with $\Delta t = 0.02s$, the simulation is stable if we take γ from the range $[0.8, 1)$.

Appendix E

Details of BLI Coupling for Regional-Flip

Two cases are shown in Figure E.1, in which the black dotted lines represent BLIs and the dark grey volumes in the back represent the human body. It visualizes the cut-away situation that omits the middle part of the BLI-to-BLI interval, which enables us to see the surface normal along the cross-sectional cut-edge, which curls out in Figure E.1(a) and curls in Figure E.1(b). Note that, if the cross over regions are flipped, Figure E.1(a) and E.1(b) will result in folding-in and folding-out, respectively. We conclude that, the **curl analysis** (i.e., finding the curl of the surface normal along the cross-sectional cut-edge of the cross-over region) can tell us whether applying Regional-Flip to the region between two BLIs results in folding-in or folding out. If it is out-curling, it results in folding-in, and vice versa.

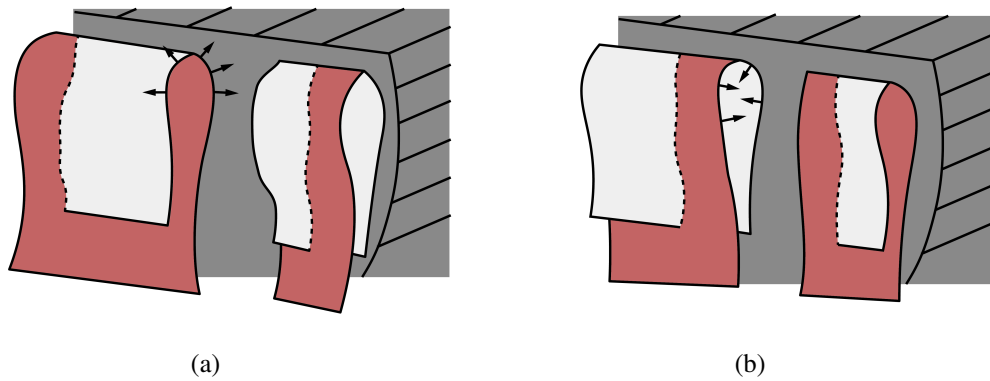


Figure E.1 Two different ways of coupling BLIs in Regional-Flip. Black dotted lines are BLIs, which define the cross-over regions. Both figures show the cut-away view, i.e., they do not show the middle parts to allow us to see the cross-section of the mesh. In (a) and (b), if the cross-over regions are flipped, it will result in folding-in and folding out, respectively. Note that, in (a) and (b), the surface normal along with the cross-section curls out and in, respectively.

Bibliography

- [1] BARAFF, D., AND WITKIN, A. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), pp. 43–54.
- [2] BARAFF, D., WITKIN, A., AND KASS, M. Untangling cloth. *ACM Trans. Graph.* 22, 3 (July 2003), 862–870.
- [3] BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. Projective dynamics: Fusing constraint projections for fast simulation. *ACM transactions on graphics (TOG)* 33, 4 (2014), 1–11.
- [4] BOXERMAN, E., AND ASCHER, U. Decomposing cloth. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), pp. 153–161.
- [5] BRANDT, C., EISEMANN, E., AND HILDEBRANDT, K. Hyper-reduced projective dynamics. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

- [6] BRIDSON, R., FEDKIW, R., AND ANDERSON, J. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (ToG)* 21, 3 (2002), 594–603.
- [7] BROCHU, T., EDWARDS, E., AND BRIDSON, R. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 96.
- [8] BROUET, R., SHEFFER, A., BOISSIEUX, L., AND CANI, M.-P. Design preserving garment transfer. *ACM Transactions on Graphics* 31, 4 (2012), Article–No.
- [9] BUFFET, T., ROHMER, D., BARTHE, L., BOISSIEUX, L., AND CANI, M.-P. Implicit untangling: a robust solution for modeling layered clothing. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.
- [10] CARIGNAN, M., YANG, Y., THALMANN, N. M., AND THALMANN, D. Dressing animated synthetic actors with complex deformable clothes. *ACM Siggraph Computer Graphics* 26, 2 (1992), 99–104.
- [11] CHOI, K.-J., AND KO, H.-S. Stable but responsive cloth. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 604–611.
- [12] CURTIS, S., TAMSTORF, R., AND MANOCHA, D. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games* (2008), ACM, pp. 61–69.

- [13] DESBRUN, M., SCHRÖDER, P., AND BARR, A. Interactive animation of structured deformable objects. *Graphics Interface 99*, 5 (1999), 10.
- [14] DU, H., HUANG, K., AND WANG, R. Research and implementation of penetration resolving for multi-layered virtual garment dressing. In *2012 Fourth International Conference on Digital Home (2012)*, IEEE, pp. 326–330.
- [15] EBERHARDT, B., ETZMUSS, O., AND HAUTH, M. Implicit-explicit schemes for fast animation with particle systems. In *Computer Animation and Simulation 2000*. Springer, 2000, pp. 137–151.
- [16] GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. Efficient simulation of inextensible cloth. In *ACM SIGGRAPH 2007 Papers (New York, NY, USA, 2007)*, SIGGRAPH '07, Association for Computing Machinery, p. 49–es.
- [17] HAN, D.-H., CHA, I.-H., LEE, K.-H., AND KO, H.-S. Garment pre-conditioning and regional simulation omission for trying-on virtual ensembles. In *Proceedings of the 32nd International Conference on Computer Animation and Social Agents (2019)*, pp. 53–58.
- [18] HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 Papers (New York, NY, USA, 2008)*, SIGGRAPH '08, Association for Computing Machinery.

- [19] HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, Association for Computing Machinery.
- [20] HU, S., WANG, R., AND ZHOU, F. An efficient multi-layer garment virtual fitting algorithm based on the geometric method. *International Journal of Clothing Science and Technology* (2017).
- [21] HU, X., WEI, L., AND LI, D. A modified numerical integration method for deformable object animation. *Asian Simulation Conference* (2007), 375–383.
- [22] HUTTER, M., AND FUHRMANN, A. Optimized continuous collision detection for deformable triangle meshes. *J. WSCG 15* (2007), 25–32.
- [23] JIANG, L., YE, J., SUN, L., AND LI, J. Transferring and fitting fixed-sized garments onto bodies of various dimensions and postures. *Computer-Aided Design 106* (2019), 30–42.
- [24] KANG, Y.-M., CHOI, J.-H., AND CHO, H.-G. Real-time animation technique for flexible and thin objects.
- [25] KANG, Y.-M., CHOI, J.-H., CHO, H.-G., AND LEE, D.-H. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer 17*, 3 (2001), 147–157.

- [26] KIM, T. A finite element formulation of baraff-witkin cloth. *Computer Graphics Forum* 39, 8 (2020), 171–179.
- [27] LAFLEUR, B., MAGNENAT-THALMANN, N., AND THALMANN, D. Cloth animation with self-collision detection. *Modeling in computer graphics* (1991), 179–187.
- [28] LEE, Y., MA, J., AND CHOI, S. Automatic pose-independent 3d garment fitting. *Comput. Graph.* 37 (2013), 911–922.
- [29] LEHERICEY, F., GOURANTON, V., AND ARNALDI, B. Gpu ray-traced collision detection for cloth simulation. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology* (2015), pp. 47–50.
- [30] LI, C., TANG, M., TONG, R., CAI, M., ZHAO, J., AND MANOCHA, D. P-cloth: interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.
- [31] LIU, T., BARGTEIL, A. W., O’BRIEN, J. F., AND KAVAN, L. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–7.
- [32] LÖSCHNER, F., LONGVA, A., JESKE, S., KUGELSTADT, T., AND BENDER, J. Higher-order time integration for deformable solids. *Computer Graphics Forum* 39, 8 (2020), 157–169.

- [33] MACKLIN, M., MÜLLER, M., AND CHENTANEZ, N. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (2016), pp. 49–54.
- [34] MACKLIN, M., STOREY, K., LU, M., TERDIMAN, P., CHENTANEZ, N., JESCHKE, S., AND MÜLLER, M. Small steps in physics simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2019), pp. 1–7.
- [35] MULLER, M., CHENTANEZ, N., KIM, T., AND MACKLIN, M. Strain based dynamics. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2014), The Eurographics Association, p. 2.
- [36] MÜLLER, M., CHENTANEZ, N., KIM, T.-Y., AND MACKLIN, M. Air meshes for robust collision handling. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.
- [37] MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.
- [38] MÜLLER, M. Hierarchical Position Based Dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)* (2008), F. Faure and M. Teschner, Eds., The Eurographics Association.

- [39] NARAIN, R., OVERBY, M., AND BROWN, G. E. Admm \supseteq projective dynamics: fast simulation of general constitutive models. *Symposium on Computer Animation 1, 2* (2016), 2016.
- [40] OTADUY, M. A., TAMSTORF, R., STEINEMANN, D., AND GROSS, M. Implicit contact handling for deformable objects. *Computer Graphics Forum* 28, 2 (2009), 559–568.
- [41] OVERBY, M., BROWN, G. E., LI, J., AND NARAIN, R. Admm \supseteq projective dynamics: fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2222–2234.
- [42] PABST, S., KOCH, A., AND STRASSER, W. Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. *Computer Graphics Forum* 29, 5 (2010), 1605–1612.
- [43] PROVOT, X. Collision and self-collision handling in cloth model dedicated to design garments. *Computer Animation and Simulation'97* (1997), 177–189.
- [44] PROVOT, X., ET AL. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface* (1995), Canadian Information Processing Society, pp. 147–147.
- [45] SCHVARTZMAN, S. C., PÉREZ, A. G., AND OTADUY, M. A. Star-contours for efficient hierarchical self-collision detection. In *ACM SIG-*

- GRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, Association for Computing Machinery.
- [46] SELLE, A., SU, J., IRVING, G., AND FEDKIW, R. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics* 15, 2 (2009), 339–350.
- [47] SUD, A., GOVINDARAJU, N., GAYLE, R., KABUL, I., AND MANOCHA, D. Fast proximity computation among deformable models using discrete voronoi diagrams. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 1144–1153.
- [48] SUN, L., NYBERG, T. R., XIONG, G., AND YE, J. Minimum displacements for cloth-obstacle penetration resolving. In *Eurographics (Short Papers)* (2016), pp. 53–56.
- [49] TANG, M., CURTIS, S., YOON, S.-E., AND MANOCHA, D. Iccd: Interactive continuous collision detection between deformable models using connectivity-based culling. *IEEE Transactions on Visualization and Computer Graphics* 15, 4 (2009), 544–557.
- [50] TANG, M., LIU, Z., TONG, R., AND MANOCHA, D. Psc: Parallel self-collision culling with spatial hashing on gpus. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 1–18.

- [51] TANG, M., MANOCHA, D., LIN, J., AND TONG, R. Collision-streams: Fast gpu-based collision detection for deformable models. *Symposium on interactive 3D graphics and games* (2011), 63–70.
- [52] TANG, M., MANOCHA, D., AND TONG, R. Fast continuous collision detection using deforming non-penetration filters. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2010), ACM, pp. 7–13.
- [53] TANG, M., MANOCHA, D., AND TONG, R. Mccd: Multi-core collision detection between deformable models using front-based decomposition. *Graphical Models* 72, 2 (2010), 7–23.
- [54] TANG, M., TONG, R., NARAIN, R., MENG, C., AND MANOCHA, D. A gpu-based streaming algorithm for high-resolution cloth simulation. *Computer Graphics Forum* 32, 7 (2013), 21–30.
- [55] TANG, M., TONG, R., WANG, Z., AND MANOCHA, D. Fast and exact continuous collision detection with bernstein sign classification. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 186.
- [56] TANG, M., WANG, H., TANG, L., TONG, R., AND MANOCHA, D. Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. *Computer Graphics Forum* 35, 2 (2016), 511–521.

- [57] TANG, M., WANG, T., LIU, Z., TONG, R., AND MANOCHA, D. I-cloth: Incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–10.
- [58] TERZOPOULOS, D., AND FLEISCHER, K. Deformable models. *The visual computer* 4, 6 (1988), 306–331.
- [59] TERZOPOULOS, D., AND FLEISCHER, K. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (1988), pp. 269–278.
- [60] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), pp. 205–214.
- [61] TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANTES, D., AND GROSS, M. H. Optimized spatial hashing for collision detection of deformable objects. In *Vmv* (2003), vol. 3, pp. 47–54.
- [62] TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., ET AL. Collision detection for deformable objects. *Computer graphics forum* 24, 1 (2005), 61–81.

- [63] THOMASZEWSKI, B., PABST, S., AND BLOCHINGER, W. Parallel techniques for physically based simulation on multi-core processor architectures. *Computers & Graphics* 32, 1 (2008), 25–40.
- [64] VOLINO, P., AND MAGNENAT-THALMANN, N. Resolving surface collisions through intersection contour minimization. *ACM Trans. Graph.* 25, 3 (July 2006), 1154–1159.
- [65] WANG, H. Defending continuous collision detection against errors. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.
- [66] WANG, H. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–9.
- [67] WANG, T., LIU, Z., TANG, M., TONG, R., AND MANOCHA, D. Efficient and reliable self-collision culling using unprojected normal cones. *Computer Graphics Forum* (2017).
- [68] WANG, X., TANG, M., MANOCHA, D., AND TONG, R. Efficient bvh-based collision detection scheme with ordering and restructuring. *Computer Graphics Forum* 37, 2 (2018), 227–237.
- [69] WANG, Z., TANG, M., TONG, R., AND MANOCHA, D. Tightccd: Efficient and robust continuous collision detection using tight error bounds. *Computer Graphics Forum* 34, 7 (2015), 289–298.

- [70] WEBER, D., BENDER, J., SCHNOES, M., STORK, A., AND FELLNER, D. Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. *Computer Graphics Forum* 32, 1 (2013), 16–26.
- [71] WICKE, M., LANKER, H., AND GROSS, M. Untangling cloth with boundaries. In *Proc. of Vision, Modeling, and Visualization* (2006), pp. 349–356.
- [72] WONG, A., EBERLE, D., AND KIM, T. Clean cloth inputs: Removing character self-intersections with volume simulation. In *ACM SIGGRAPH 2018 Talks* (New York, NY, USA, 2018), SIGGRAPH '18, Association for Computing Machinery.
- [73] YE, J. History-free collision response for deformable surfaces. *CoRR abs/1409.2081* (2014).
- [74] YE, J., MA, G., JIANG, L., CHEN, L., LI, J., XIONG, G., ZHANG, X., AND TANG, M. A unified cloth untangling framework through discrete collision detection. *Computer Graphics Forum* 36, 7 (2017), 217–228.
- [75] YE, J., AND ZHAO, J. The intersection contour minimization method for untangling oriented deformable surfaces. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), Eurographics Association, pp. 311–316.

- [76] ZHENG, C., AND JAMES, D. L. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–12.
- [77] ZHONG, Y. Fast penetration resolving for multi-layered virtual garment dressing. *Textile Research Journal* 79, 9 (2009), 815–821.

초 록

수십 년 동안 그래픽스 필드에선 의상 시뮬레이션 중에 발생하는 자가 충돌 처리 실패(엥킴)를 해결하기 위한 여러가지 방법이 제안되었다. 그러나 제안된 방법들은 간단한 의상(티셔츠, 바지)에 대해서만 동작하고, 실제 가상 피팅이나 애니메이션 제작에 등장하는 복잡한 의상에선 대다수가 엥킴 해결에 실패하였다. 본 논문에서는 엥킴을 두 그룹으로 나누고, 각각에 대한 새로운 이산 충돌 처리 방법을 제안하며, 엥킴이 있는 복잡한 의상에 적용하는 실험을 통해 제안된 방법의 효용성을 입증한다.

첫번째 그룹, BLI를 제외한 6가지 엥킴에 대해서는 **ESEF(변-압축 / 입실론-압출)**를 제안하였다. 6가지 엥킴은 잘못된 영역이 확정적으로 정의됨을 이용하며, 가장 바깥부분부터 서서히 해결하는 아웃투인 방식으로 엥킴을 점진적으로 해결하였다. 이를 위해 매 타임 스텝마다 의상 메쉬의 엥킴 분석을 수행하고, 그 결과를 정점, 변, 삼각형을 채색하는 형태로 저장하였다. 이후 채색을 참조하여 메쉬의 필요한 영역에 두가지 기법 **삼각형-수축**과 **정점-당기기**를 가하였고, 최종적으로 모든 엥킴이 없어질때까지 이를 반복적으로 적용하였다. 삼각형-수축과 정점-당기기는 연속 충돌 처리에서 통상적으로 반올림 오류를 보정하기 위해 사용되어 왔던 입실론 값의 의미를 재해석하였다. 입실론의 효용을 반올림 오류 방어로 한정하지 않고, 더 나아가서 이산 충돌 처리에 응용하여 특정 조건에서 유한한 타임 스텝안에 엥킴 해결을 보장할 수 있게 되었다.

두번째 그룹, BLI에 대해서는 **BLI-Resolver**를 제안하였다. 먼저 BLI의

특징과 어떤 상황에서 발생하는지 분석하고, 이를 통해 원하는 엉킴 해결의 형태(스타일)가 의상의 분류 또는 특정 영역에 따라 달라져야 함을 보였다. 따라서 각각의 스타일에 대응하기 위해 BLI를 해결할 세 가지 알고리즘, **메쉬-찢기**, **영역-교차**, **접힘-교차**를 제안하였다. 메쉬-찢기는 의상 메쉬를 필요에 따라 임시로 몇몇 삼각형들을 누락 후 재구성하여 엉킴 해결에 유리한 메쉬로 변경하였다. 영역-교차, 접힘-교차는 BLI를 직접적으로 해결하지 않고, 다른 6가지 엉킴으로 변환하여, ESEF가 해결할 수 있게 해주었다.

제안된 두가지 방법(ESEF, BLI-Resolver)을 통합하여 엉킴의 스펙트럼을 모두 다룰 수 있게 되어, 의상 시뮬레이션 속의 이산충돌처리의 마침표를 찍게 되었다. 이 방법들은 기존의 연속 충돌 처리가 구현되어 있는 시뮬레이터에 쉽게 통합이 가능하며, 시뮬레이터의 종류에 영향을 받지 않는 특징이 있다. 또한 의상의 복잡도나 종류에 구애받지 않고 유한한 타임스텝 내로 엉킴이 풀림을 보장할 수 있으며, 의상의 디자인에 대한 정보가 제공된 경우 엉킴이 디자인에 적합한 방향으로 해결된다. 최종적으로 실험을 통해 이전의 방법으로 해결할 수 없었던 다양하고 실용적인 의복에서의 엉킴이 해결됨을 보였다.

주요어: 의상 시뮬레이션, 이산 충돌 처리, 엉킴 풀기

Keywords: Clothing Simulation, Discrete Collision Handling, Intersection, Tanglement, Untangling

학 번: 2014-21691