옷감 찢기 시뮬레이션을 표현하는 GPU기반 동적 재메쉬

문성혁[O], 김종현[*]
[O]강남대학교 소프트웨어응용학부,
[*]강남대학교 소프트웨어응용학부
e-mail: jonghyunkim@kangnam.ac.kr

GPU-Based Dynamic Remeshing to Simulate Cloth Tearing

Seong-Hyeok Moon[O], Jong-Hyun Kim[*]
[O]School of Software Application, Kangnam University,
[*]School of Software Application, Kangnam University

● 요 약 ●

본 논문에서는 GPU 기반으로 옷감을 찢는 데 필요한 동적 재메쉬 기법에 대해서 제안한다. 일반적으로 메쉬를 파괴(Fracture)하거나 찢는 시뮬레이션에서는 안정적인 동역학 계산하는데 있어서 동적 재 메쉬과정에 매우 중요하며 이 과정이 계산양이 가장 크다. 본 논문에서는 GPU 친화적인 동적 메쉬 알고리즘을 새롭게 제안함으로써 옷감 찢기 시뮬레이션을 실시간으로 보여준다.

키워드: 옷감 시뮬레이션(Cloth simulation), GPU(Graphics Processing Unit), 동적 재메쉬(Dynamic remeshing), 옷감 찢기(Cloth tearing)

I. Introduction

가상환경에서 캐릭터와 에이전트를 사실적으로 표현하기 위해 다양한 물리 기반 시뮬레이션 기법들이 활용되고 있다[1] 그 중에서 특히 옷감 시뮬레이션은 패션 스튜디오, 옷감 제작 등 다양한 산업에 영향을 끼치고 있지만, 계산양이 크고 상태변화에 따른 옷감 형태가 달라지기 때문에 찢어지는 동적 재메쉬는 표현하기 어렵다[2-4]. 본 논문에서는 이러한 옷감의 재메쉬 효율적으로 처리하기 위한 GPU 기반 알고리즘을 제안한다.

II. The Proposed Scheme

2.1 Truncated information of elements

효율적으로 Remesh 문제를 풀기 위해 Face와 Edge에서 CutPoint가 두 개 이상은 될 수 없도록 함으로서 CutLine의 개수가 최소가 되도록 하였다. 여기서 CutLine는 자르는 경계선으로 인해 생기는 선을, CutPoint는 CutLine들이 교차하면서 생기는 점을 의미한다. Face와 Edge에서 CutPoint가 두 개 이상 존재한다면 점들의 평균위치를 CutPoint로 설정한다. Face와 Edge에서 생기는 CutLine의 정보를 CurInfo라고 하겠다. CutInfo는 Face에서는 6bit의 데이터, Edge에서는 2bit의 데이터로 표현할 수 있다.

Face는 3개의 Node와 Edge로 이루어져있는데 n번째 Node와 Edge는 서로 마주보고 있다. Face의 CutInfo의 3bit는 Edge와 Node 사이의 CutLine을 나타내고 나머지 3bit는 Edge와 Edge 사이의 CutLine을 나타낸다 (Fig. 1a 참조).

Edge에서의 CutLine은 해당 Edge의 일부분이다. Edge에 CutPoint가 있을 때 0번 Node부터 CutPoint까지 CutLine이면 CutInfo는 $01_{(2)}$이 되고 1번 Node부터 CutLine까지면 $10_{(2)}$이 된다. 전체가 CutLine이면 $11_{(2)}$이 된다 (Fig. 1b 참조). 단, Edge의 이웃한 삼각형의 개수가 한 개라면 0으로 통일한다.

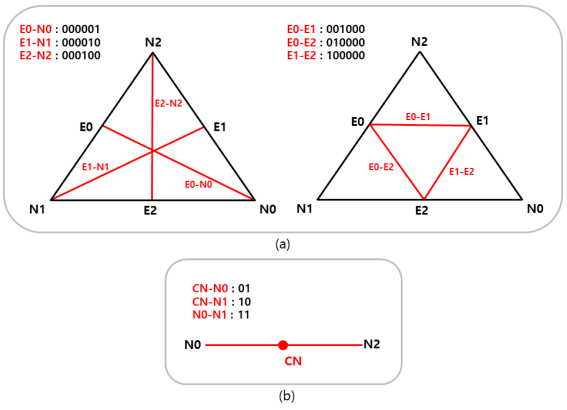

Fig. 1. (a) : CutInfo of Face (b) : CutInfo of Edge.

2.2 Generate new nodes

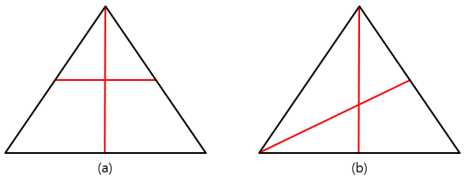

Fig. 2. Case of CutLines Crossing.

새로운 Node의 개수는 Face, Edge, Node에서 병렬적으로 계산된다. Face에서는 CutPoint가 있을 때만 새로운 Node의 개수를 계산한다. CutPoint는 Face 내 CutLine들이 교차하는 경우 존재한다. Face 내 CutLine들이 교차하는 경우는 CutInfo의 값이 $100001_{(2)}$, $010010_{(2)}$, $001100_{(2)}$이거나 (Fig. 2a) Node와 Edge 사이의 CutLine의 개수가 2개 이상인 경우다 (Fig. 2b). 이때 CutPoint와 CutLine으로 연결되어있는 요소들의 개수가 새로운 노드의 개수다.

Edge에서는 Face에서와 마찬가지로 CutPoint가 있는 경우에만 새로운 Node의 개수를 계산한다. Edge에서의 새로운 Node 개수는 CutPoint와 연결된 해당 Edge와 이웃 Face의 CutLine의 개수이다. 단, Edge의 이웃 Face의 개수가 한 개일 CutLine의 개수가 하나 더 있는 것과 같게 처리되므로 한 개를 추가한다 (Fig. 3a 참조).

Node에서는 Node와 연결된 이웃 Face와 Edge의 CutLine 개수가 새로운 Node의 개수다. 단, Node가 Face로 완전히 둘러싸여있지 않다면 Node와 연결된 CutLine이 하나 더 있는 것과 동일하게 처리되므로 한 개를 추가한다 (Fig. 3a 참조).

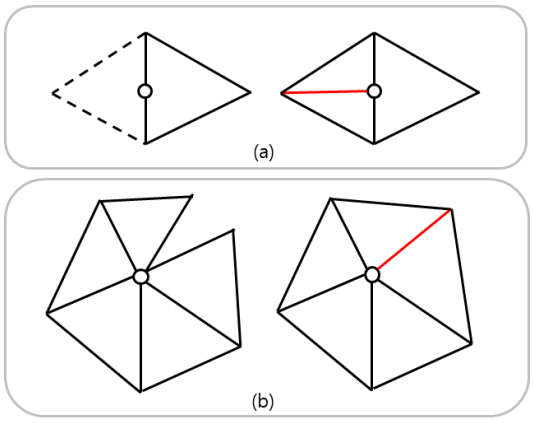

Fig. 3. (a) : When the edge has one neighbor triangle.
(b) : When the node is not completely surrounded by faces.

각 요소에서 구한 새로운 Node 개수는 구간 합(Prefix Sum) 알고리즘을 통해 해당 요소에서 생성할 Node의 인덱스를 저장한다. 생성되는 총 개수 크기의 새로운 Node 배열을 생성하고 각 요소에서 해당하는 Node 인덱스에 Node 정보를 저장한다 (Fig. 5 참조).

2.3 Generate new faces

새로운 Face의 개수는 Face에서 CutInfo를 통해 계산된다. Face에 CutPoint의 유무에 따라 두 가지 방법으로 나뉜다. 또한 Face에 CutPoint가 없을 경우 Edge-Node CutLine의 유무에 따라 나뉜다 (Fig. 4 참조).

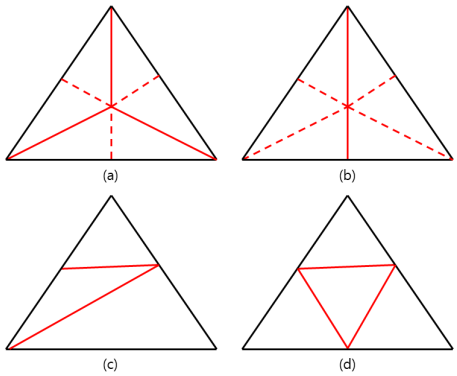

Fig. 4. Case of CutLines Crossing.

먼저 CutPoint가 있을 경우 최소 세 개의 삼각형으로 나눠진다. 각 삼각형에 있는 Face의 Edge에 CutPoint가 있을 경우 두 개의 삼각형으로 한 번 더 나눠진다 (Fig. 4a 참조). CutPoint가 없고 Edge-Node CutLine이 있을 경우 최소 두 개의 삼각형으로 나누어진다. 각 삼각형에 있는 Face의 Edge에 CutPoint가 있을 경우 삼각형은 세 개로 나눠진다 (Fig. 4b 참조).

마지막으로 CutPoint가 없고 Edge-Node CutLine 또한 없을 경우 Edge-Edge CutLine이 한 개 있을 경우 (Fig. 4c) 세 개의 삼각형, 두 개 이상 있을 때(Fig. 4d)는 네 개의 삼각형으로 나눠진다. 단. Edge-Edge CutLine이 한 개 있을 때 CutLine에 연결되어있지 않은 Edge에 CutPoint가 있을 경우 Edge-Edge CutLine이 두 개 이상 있을 때와 동일하게 네 개의 삼각형으로 나눠진다.

2.2절과 동일하게 각 Face에서 생성될 새로운 Face의 인덱스와 총 개수 크기의 새로운 Face 배열을 생성한다 (Fig. 5 참조).

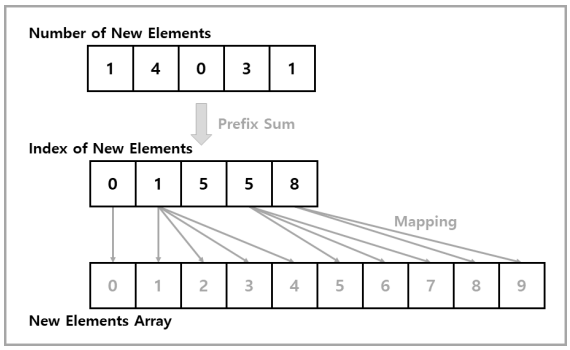

Fig. 5. Stores the number of new elements to be created from existing elements. Then, the index to be created from the element is obtained through the interval sum algorithm. Map new data to the corresponding index in each element.

2.4 Face, node information data

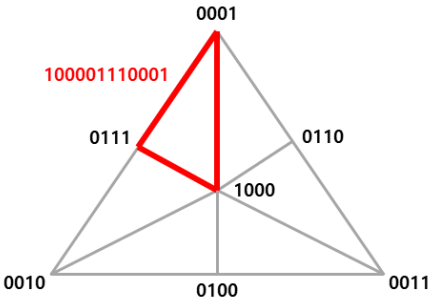

Fig. 6. (red) is new face Information data and (black) is node information data.

각 요소에서 새로운 Face 배열에 해당하는 Node 인덱스를 매핑하기 위해서는 Face의 세 Node에 대한 정보를 알 수 있어야 한다. Node가 위치하는 요소의 종류와 인덱스 정보를 가지고 있는 데이터를 Node 정보 데이터라 하겠다. 그리고 새로운 Face의 세 Node에 대한 정보를 가지고 있는 데이터를 Face 정보 데이터라 하겠다.

Node 정보 데이터는 4bit의 데이터로 구성되어있다 (Fig. 5 참조). 2bit는 Face 내에서 요소의 인덱스(0~2)를 나타내고 나머지 2bit는 요소의 종류(0 : Node, 1 : Edge, 2: Face)를 나타낸다. 이때 요소의 종류가 Face일 경우 인덱스를 나타내는 2bit의 값은 0으로 통일한다. 기존 Face에서 Face 정보 데이터를 새로운 Face 배열 크기와 위치에 맞게 생성한다.

2.4 Node mapping

기존 요소들에서 Face정보 데이터를 참조하여 새로운 Face배열에 Node 인덱스를 매핑한다. Face, Edge에서는 CutPoint 중심으로, Node에서는 해당 Node 중심으로 둘러싸고 있는 새로운 Face를 순서대로 매핑한다. 이때 새로운 Face 사이의 Edge가 CutLine이면 해당 요소에서의 새로운 Node 인덱스를 변경한다.

마지막으로 생성된 Node와 Face를 통해 Edge를 포함한 이웃정보들을 생성한다.

III. Result

본 논문에서 옷감 시뮬레이션을 Constraint가 자유롭게 바뀌어도 부담 없고 병렬프로그래밍에 친화적인 Jacobian Projective Dynamics를 통해 구현했다. 또한 Wang이 제안한 Chebyshev 반복법으로 가속화 하였다. 구간 합 알고리즘을 CUDA thrust 라이브러리의 exclusive_scan을 이용하여 계산했다.

Fig. 7은 다른 해상도를 가진 세 개의 옷감을 여섯 방향으로 자른 테스트 장면이다. 해당 옷감의 Face 개수와 성능은 Table. 1에 나와있다.

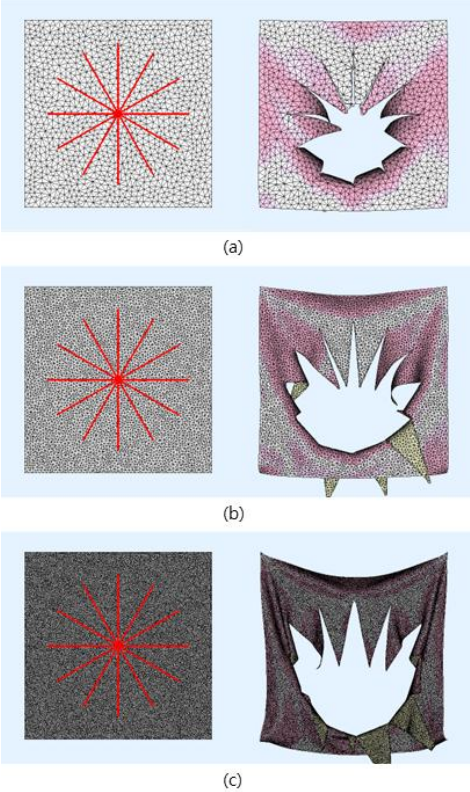

Fig. 7. A scene where the fabric is cut in six directions. (a) low resolution, (b) medium resolution, (c) high resolution.

Table 1. Performance.

Model	# of Face.	Time
Low	2,822	4.3 msec
Medium	9,022	9.1 msec
High	38,298	9.1 msec

Fig. 8 은 실시간으로 고해상도 옷감을 자르는 장면이다.

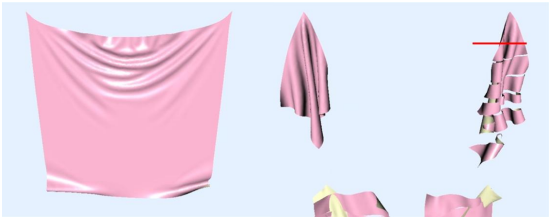

Fig. 8. Real-time cloth cutting scene.

IV. Conclusions

본 논문에서 옷감을 찢는 시뮬레이션을 효율적으로 하기 위한 방법과, 이를 GPU로 개발할 수 있는 새로운 프레임워크를 제안했다. 향후, 우리는 옷감을 3차원 볼륨메트릭 메쉬로 확장하여 파괴 시뮬레이션 실시간 처리할 수 있도록 알고리즘을 확장할 계획이다.

REFERENCES

[1] Meseure, Philippe, Jérôme Davanne, Laurent Hilde, Julien Lenoir, Laure France, Frédéric Triquet, and Christophe Chaillou. "A physically-based virtual environment dedicated to surgical simulation." In International Symposium on Surgery Simulation and Soft Tissue Modeling, pp. 38-47. Springer, Berlin, Heidelberg, 2003.

[2] Tan, Qingyang, Zherong Pan, Lin Gao, and Dinesh Manocha. "Realtime simulation of thin-shell deformable materials using CNN-based mesh embedding." IEEE Robotics and Automation Letters 5, no. 2 (2020): 2325-2332.

[3] Metaaphanon, Napaporn, Yosuke Bando, Bing-Yu Chen, and Tomoyuki Nishita. "Simulation of tearing cloth with frayed edges." In Computer Graphics Forum, vol. 28, no. 7, pp. 1837-1844. Oxford, UK: Blackwell Publishing Ltd, 2009.

[4] Souza, Marco Santos, Aldo Wangenheim, and Eros Comunello. "Fast simulation of cloth tearing." SBC Journal on Interactive Systems 5, no. 1 (2014): 44-48.