JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 29 No. 8, pp. 59-65, August 2024
https://doi.org/10.9708/jksci.2024.29.08.059

GPU-Optimized BVH and R-Triangle Methods for Rapid Self-Intersection Handling in Fabrics

Jong-Hyun Kim*

*Associate Professor, College of Software and Convergence (Dept. of Design Technology), Inha University, Incheon, Korea

[Abstract]

In this paper, we present a GPU-based acceleration of computationally intensive self-collision processing in triangular mesh-based cloth simulation. For Compute Unified Device Architecture (CUDA)-based parallel optimization, we propose 1) an efficient way to build, update, and traverse the Bounding Volume Hierarchy (BVH) tree on the GPU, and 2) optimize the Representative-Triangle (R-Triangle) technique on the GPU to minimize primitive collision checking in triangular mesh-based cloth simulations. As a result, the proposed method can handle self-collisions and object collisions of cloth simulation in GPU environment faster and more efficiently than CPU-based algorithms, and experiments on various scenes show that it can achieve simulation results that are 5x to 10x faster. Since the proposed method is optimized for BVH on GPU, it can be easily integrated into various algorithms and fields that utilize BVH.

▸ **Key words**: Graphics Processing Unit, Compute Unified Device Architecture, Cloth simulation, Self-collision, Collision handling

[요 약]

본 논문에서는 삼각형 메쉬 기반 옷감 시뮬레이션에서 계산이 큰 자기충돌(Self-collision) 처리를 GPU 기반으로 가속화하는 방법을 소개한다. CUDA(Compute Unified Device Architecture) 기반 병렬 최적화를 위해, 본 논문에서는 1) GPU에서 BVH(Bounding Volume Hierarchy) 트리를 효율적으로 구축, 업데이트 및 순회하는 방법을 제안하고, 2) 삼각형 메쉬 기반에서는 R-Triangle(Representative-Triangle) 기법을 GPU에서 최적화하여 프리미티브 충돌 검사를 최소화한다. 결과적으로, 제안된 방법은 GPU 환경에서 옷감 시뮬레이션의 자기충돌과 객체 충돌을 CPU기반 알고리즘에 비해 빠르고 효율적으로 처리할 수 있으며, 다양한 장면에서 실험한 결과 5배~10배정도 빠른 시뮬레이션 결과를 얻을 수 있다. 본 논문에서 제안하는 방법은 BVH를 GPU에서 최적화했기 때문에 BVH를 기반으로 활용하는 다양한 알고리즘과 분야에 쉽게 통합이 가능하다.

▸ **주제어**: 그래픽 처리 장지, 쿠다, 옷감 시뮬레이션, 자기충돌, 충돌처리

• First Author: Jong-Hyun Kim, Corresponding Author: Jong-Hyun Kim
*Jong-Hyun Kim (jonghyunkim@inha.ac.kr), College of Software and Convergence (Dept. of Design Technology), Inha University
• Received: 2024. 07. 08, Revised: 2024. 08. 08, Accepted: 2024. 08. 09.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

우리 주변에서 옷감은 흔히 볼 수 있는 재질이며, 강체와는 다르게 변형이 쉽게 일어나고 얇은 쉘(Shell)형태이기 때문에 자기충돌(Self-collision)도 많이 보이는 재질이다[1,2]. 최근에 메타버스와 가상현실이 이슈화가 되면서 패션, 게임, 의상디자이너 등 다양한 첨단 디지털 산업에 영향을 주는 대표적인 기술 중 하나이다[3,4]. 옷감은 쉽게 변형되는 탄성재질을 가지고 있기 때문에 다른 재질에 비해 자기충돌이 쉽게 발생한다. 특히, 실시간이나 인터랙티브하게 결과를 보면서 진행해야 하는 의상디자인이나 메타버스 콘텐츠에서는 시뮬레이션 움직임을 실시간으로 분석 및 검토해야 때문에 최적화된 병렬충돌 검사 기법을 필수적으로 요구된다.

게임이나 영상특수효과에서는 옷감 편집 툴로 대부분 마블러스 디자이너(Marvelous Designer)라는 저작도구를 사용하고 있다. 이 툴에서는 사용자가 옷감 도면을 편집하면 시뮬레이션을 통해 3차원에서 옷감 형태가 실시간으로 표현되기 때문에 산업에서 많이 활용되고 있다. 하지만, 충돌 검출 및 처리의 프리미티브 단위가 정점(Vertex), 에지(Edge), 페이스(Face)이기 때문에 계산양이 크고, CPU 기반 병렬화를 적용해도 성능이 장면 복잡도에 의존하기 때문에 사용자가 만족할만한 성능을 보여주는 것이 쉽지 않다. 특히, 실크 같은 천 재질은 자기충돌이 많이 발생하기 때문에 같은 장면임에도 불구하고 재질에 따라 계산 차이가 많이 크다. 본 논문에서는 CUDA를 이용하여 BVH를 최적화 시킬 수 있는 방법과 충돌처리 과정에서 중복 연산을 피할 수 있는 방법을 GPU 병렬화함으로써 옷감의 자기충돌이나 객체의 충돌을 빠르게 처리할 수 있는 방법을 제안한다.

변형 가능한 모델 간의 빠른 충돌감지는 물리 기반 시뮬레이션에서 여전히 이슈인 주제이다. 특히 빠르고 안정적인 충돌 처리는 옷감과 가상 수술 시뮬레이션을 포함한 많은 응용 분야에서 중요한 부분이다. 대부분 충돌감지가 주요 병목 현상을 하나를 나타내기 때문이다[5,6]. 변형 가능한 모델 간의 충돌 감지를 위한 대부분의 알고리즘은 BVH[7,8] 또는 공간 해싱[9,10]을 사용한다. BVH기반 병렬 알고리즘은 계산을 가속화하기 위해 기본 BVH의 프론트(Front)를 유지하거나 업데이트한다[8,11]. 그러나 BVH의 프론트를 저장하는 방식은 저장에 필요한 계산양이 크며, 수십 개 또는 수백 개의 삼각형으로 구성된 복잡한 모델에는 몇 GB의 메모리가 필요할 수 있다. 게다가 이러한 저장 오버헤드는 GPU 성능에 큰 영향을 미칠 수 있다.

다른 접근법으로 콘(Cone)을 기반으로 다양한 접근법들이 제안되어 왔지만[12,13,14], 순차 충돌 감지에 한정적으로 사용될 수 있다는 한계점이 있다. 대부분의 GPU 기반 충돌 감지 및 천 시뮬레이션에서는 콘을 기반으로 자기충돌을 검사하지 않는다. 결과적으로 컬링(Culling) 효율이 좋지 않으며, 충돌 검사의 정확성이 떨어진다. 본 논문에서는 제안하는 방법은 다음과 같다 :

- GPU에서 BVH(Bounding Volume Hierarchy) 트리를 효율적으로 구축, 업데이트 및 순회하는 방법을 제안한다.
- 콘이 아닌 경계상자 기반의 BVH와 R-Triangle 기법을 GPU로 최적화하여 빠르게 자기충돌을 할 수 있는 기법을 제시한다. 즉, 삼각형 메쉬 기반에서는 R-Triangle(Representative-Triangle) 기법을 GPU에서 최적화하여 프리미티브 충돌 검사를 최소화한다.

결과적으로, 제안된 방법은 GPU 환경에서 옷감 시뮬레이션의 자기충돌과 객체 충돌을 CPU기반 알고리즘에 비해 빠르고 효율적으로 처리할 수 있으며, 다양한 장면에서 실험한 결과 빠른 시뮬레이션 결과를 얻을 수 있다. 본 논문에서 제안하는 방법은 BVH를 GPU에서 최적화했기 때문에 BVH를 기반으로 활용하는 다양한 알고리즘과 분야에 쉽게 통합이 가능하다.

II. Related Work

공간 해싱(Spatial hashing)은 충돌감지를 위해 많이 사용하는 알고리즘이며, GPU에서 쉽게 병렬화할 수 있다[15]. 이 알고리즘은 근접한 모든 삼각형을 쿼리하는 데 일정한 시간 복잡도(Constant time complexity)를 가지고 있다. Pabst 등은 삼각형 메시에 대해 GPU에서 Broad phase culling을 적용하기 위해 정규 격자를 사용했다[16]. 장면을 구성하고 있는 기하학 프리미티브(Geometry primitives)의 비균등한 분포를 처리하기 위해 Faure 등은 정규 격자를 2-레이어 격자(Two-layer grids)로 확장했다[17]. 계층적 격자는 공간 분할 기반 충돌 감지 알고리즘의 효율성을 더욱 향상시키기 위해 사용되어왔다[10,17,18]. 계층적 격자는 장면 프리미티브의 비균등한 분포 처리를 해결할 수 있지만, 옷감 모델 같이 비교적 평평한 재질에 대해서는 좋은 컬링 성능을 보이지 않는다.

Selle 등은 멀티코어 플랫폼을 위한 병렬 옷감 시뮬레이션 알고리즘을 제안했다[19]. 이 방법은 16코어 워크스테

이션에서 50만개의 삼각형을 가진 옷감 메시의 경우 한 프레임당 최대 30분이 소요될 수 있다. Tang 등은 GPU에서 규칙적인 형태의 고해상도 옷감 시뮬레이션을 위한 스트리밍 알고리즘을 제안했다[20]. 이 알고리즘은 임의의 토폴로지 구조를 가진 옷감 시뮬레이션을 위해 확장되었으며[21], 일반적인 GPU에서 100만개 삼각형들을 가진 옷감의 경우 한 프레임 당 최대 35초가 소요될 수 있다. 이러한 병렬 알고리즘은 병렬 충돌 검사를 위해 BVTT(Bounding volume traverse tree)를 **사용**하지만, 큰 메모리 오버헤드를 초래할 수 있다.

III. The Proposed Scheme

1. GPU-based BVH(Bounding volume hierarchy) Tree Structure

제한된 리소스에서 병렬화를 개선해야 하는 GPU환경에서 BVH를 최적화하기 위해서는 다음과 같은 특징을 고려해야 한다. 첫 번째로 알고리즘의 병렬성을 최대화하고, 두 번째로 메모리를 최소화하여 가속화하는 것이다.

트리의 업데이트나 DFS(Depth-first search)같은 트리 탐색의 경우 다른 레벨의 노드들을 순서에 맞게 계산해야 한다. 하지만 BFS(Breadth-first search)는 같은 레벨의 노드들을 병렬적으로 계산할 수 있다. 본 논문에서는 이점을 최대한 활용하기 위해 같은 레벨의 노드들을 한곳에 모으는 너비 우선으로 노드들을 저장한다. 트리의 형태가 완전이진트리(Complete binary tree)라고 가정했을 때, 현재 레벨을 l, 레벨이 바뀌는 인덱스는 2^l-1로 계산할 수 있다. 또한 현재 노드의 인덱스를 n이라고 할 때 자식노드의 인덱스를 $2n+1$, $2n+2$로 계산할 수 있다. 반대로 완전이진트리가 아니라면 노드들의 레벨이 바뀌는 인덱스의 정보와 자식 및 부모노드의 인덱스를 가지는 데이터를 따로 저장해야 한다. 따라서 메모리를 좀 더 효율적으로 최소화할 수 있도록 BVH 트리를 완전이진트리 형태로 트리를 구축할 수 있다.

BVH 트리에서 리프노드의 개수는 삼각형의 개수와 같고 완전이진트리 형태로 구축하기 때문에 삼각형의 개수만 알면 트리의 구조를 알 수 있다.

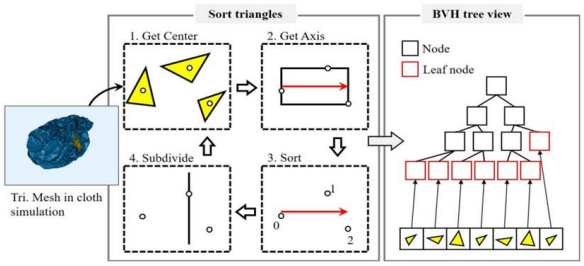

Fig. 1. An overview of GPU-based BVH tree construction.

따라서 Fig. 1에서 보듯이 삼각형의 중심 위치에 따라 정렬하는 과정과 정렬된 삼각형을 트리의 리프노드에 배치하는 과정으로 나뉜다. 삼각형의 중심 위치에 따라 정렬하는 과정은 4가지 CUDA 커널(Kernel)함수로 세분화된다(Fig.1의 4가지 단계 참조). 먼저 삼각형들의 중점을 이용한 경계상자(AABB, Axis-aligned bounding box)를 찾는다. 그리고 가장 긴축을 분할 축으로 선정한다. 선정된 분할 축을 기준으로 삼각형을 정렬하고 트리의 모양에 따른 개수로 나눠준다. 이 과정에서 분할 기준은 다양한 방법들이 있으며 본 논문에서는 Popov 등이 제안한 방법을 이용했다[21]. 삼각형들을 정렬하는 과정에서는 GPU 환경에서 효율적이라고 알려진 바이토닉(Bitonic sort)를 이용했고, 나눠주는 과정은 다음과 같다.

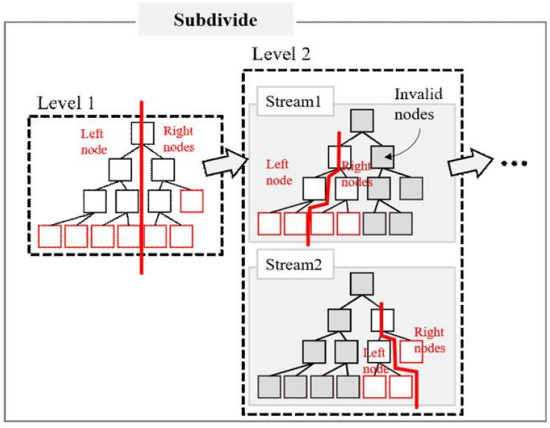

Fig. 2. Node subdivision.

트리의 최대 레벨을 L, 현재 나눠지고 있는 레벨을 l, $L-l$을 L', 삼각형의 개수이자 리프노드의 개수를 k라 할 때 한쪽으로 나눠줘야 할 삼각형의 개수는 다음과 같다(수식 1 참조).

$$\begin{cases} k-2^{L'-2} & \left(k<\frac{3}{4}2^L\right) \\ 2^{L'-1} & \left(k\geq\frac{3}{4}2^L\right) \end{cases} \quad (1)$$

이 과정을 반복하여 삼각형들을 나눠주면, 이 과정에서 삼각형들이 여러 묶음으로 나눠지게 된다. 이때 CUDA 스트림(Stream)을 이용하여 병렬적으로 계산해 가속화한다.

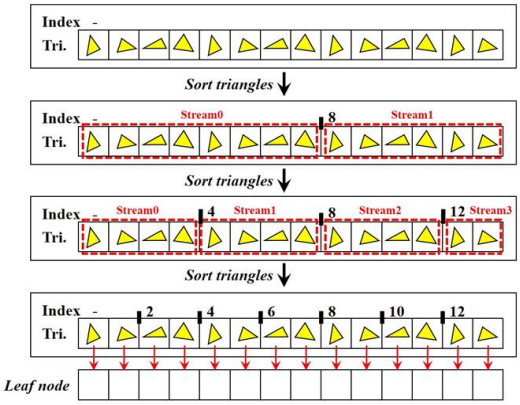

Fig. 3. BVH construction.

본 논문에서 제안한 방법을 기반으로 BVH를 구성하는 예제 다음과 같다. 우선 옷감을 구성하는 삼각형들을 인덱스 기반으로 계층적으로 정렬하고 이때 CUDA 스트림을 통해 각 구간별로 병렬화를 진행한다. 이 과정은 각 삼각형이 **리프**노드가 될 때까지 반복한다 (Fig. 3 참조).

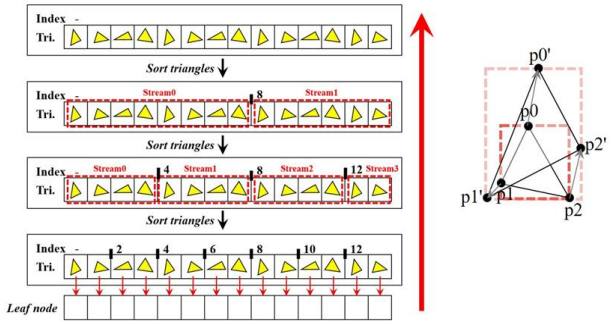

Fig. 4. Refitting BVH.

만약 옷감이 변형되어 삼각형 정점인 $(p0, p1, p2)$의 위치가 수축이나 확장되는 경우 완전이진트리로 구성된 BVH는 리프노드로부터 **루트**노드 순서로 AABB의 크기를 업데이트한다 (Fig. 4 참조).

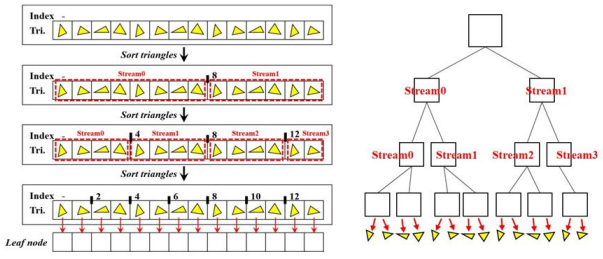

Fig. 5. Tree visualization of BVH.

Fig. 5는 본 논문에서 제안하는 완전이진트리를 이용한 GPU 기반 BVH 트리를 시각화한 결과이다. BFS는 병렬화하기 **쉬우나**, 트리의 크기가 커지면 메모리가 많이 필요할 수 있기 때문에 본 논문에서는 GPU 기반 DFS로 이 문제를 해결한다. GPU에서 DFS를 구성하는 방법은 각 스레드에 위치에 맞는 리프노드의 삼각형을 배치하고, 해당 삼각형에 대해 BVH를 순회하며 충돌검사를 진행한다 (Fig. 6 참조).

2. GPU-Based BVH Tree Updates and Recursion

BVH 트리의 업데이트는 리프노드로부터 루트노드 순서로 AABB의 크기를 업데이트한다. 따라서 각 레벨의 노드 업데이트를 병렬적으로 계산하여 GPU 아키텍처에서 효율적일 수 있도록 설계한다.

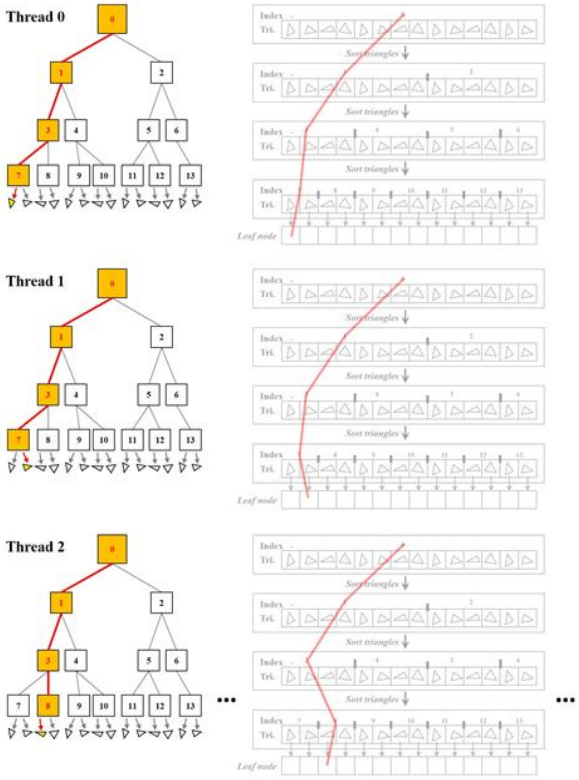

Fig. 6. Collision detection.

트리의 순환은 BFS가 병렬적으로 계산되기 쉽지만, 다음 레벨에 탐색해야 할 노드에 대한 데이터를 저장하는 과정이 필요하다. 이러한 과정은 제한적인 GPU 환경에서는 친화적이지 않기 때문에 장면 복잡도에 따라 메모리를 초과하는 경우가 발생한다. 따라서 본 논문에서는 메모리에 제한적이지 않는 CUDA 기반의 DFS 방식으로 순환하는 방법을 제안한다.

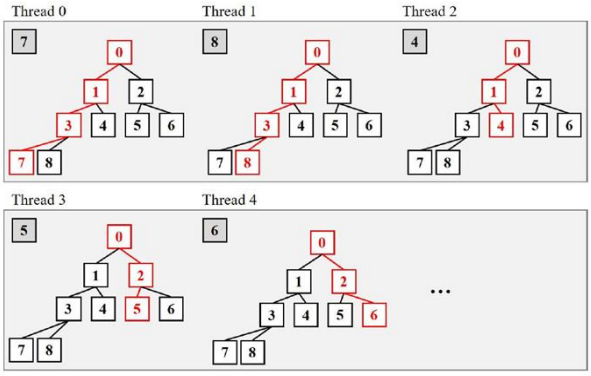

Fig. 7. Depth-first search on GPU.

GPU에서 DFS를 구현하는 과정은 다음과 같다. 각 스레드에 위치에 맞는 리프노드의 삼각형을 배치한다 (Fig. 7 참조). 그리고 해당 삼각형에 대해 BVH 트리를 순환하며 충돌검사를 한다. 이때 스레드의 위치에 맞게 삼각형을 배치함으로써 BVH 트리 특성상 비슷한 위치의 삼각형이 같은 워프(Warp)에 위치하게 된다. 따라서 거의 비슷한 경로로 순환하기 때문에 병렬적으로 계산하게 되어 더 좋은 성능을 기대할 수 있다.

3. Representative-Triangle

메시의 구조를 보면 정점과 에지를 여러 삼각형이 공유하고 있고, 이러한 문제 때문에 충돌 검사를 할 때 계산양이 증가하게 된다. 본 논문에서는 R-triangle(Representative-triangle)을 통해 이 문제를 해결했으며[22], 이 과정 역시 GPU 커널에서 동작하도록 설계한다.

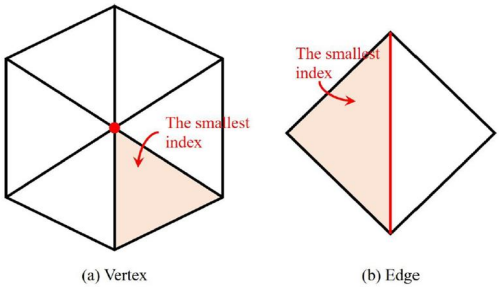

Fig. 8. Collision candidates collected based on the R-triangle.

R-triangle 기법에서 정점과 에지의 대표 삼각형을 효율적으로 설정하는 방법이 제안되었다. 하지만 본 논문에서는 효율적으로 처리하기 위해 공유하고 있는 삼각형 중 제일 작은 인덱스를 가지고 있는 삼각형을 대표 삼각형으로 설정한다 (Fig. 8 참조).

IV. Experiment and Results

본 연구의 결과들을 만들기 위해 실험한 환경은 Intel Core i7-7700K CPU, 32GB RAM, Geforce GTX 1080Ti GPU가 탑재된 컴퓨터를 이용하였다. 사용한 언어는 C++와 OpenGL을 이용하였고, 렌더링을 OpenGL을 이용하였으며, 별도의 쉐이더처리를 하지는 않았다.

Fig. 9. Self-collision handling with our method.

Fig. 7은 본 논문에서 제안하는 GPU기반 BVH 트리와 기존 CPU기반 BVH 트리를 비교한 결과이다. 옷감을 다양한 해상도로 표현하고, 자기충돌(Self-collision)이 많이 발생하는 장면을 구성하여 결과를 비교했다. Table 1~3은 BVH 트리의 구성시간, 업데이트 그리고 탐색시간이다.

Table 1. Construction of BVH tree.

# of triangles	GPU (ours)	CPU
2,244	23.94 ms	5.71 ms
27,030	59.12 ms	45.18 ms
35,992	0.12 s	0.98 s

Table 2. Update(refit) of BVH tree.

# of triangles	GPU (ours)	CPU
2,244	0.16 μs	0.31 μs
27,030	0.18 μs	1.28 μs
35,992	0.35 μs	4.81 μs

Table 3. Search(traverse) of BVH tree.

# of triangles	GPU (ours)	CPU
2,244	1.19 ms	10.28 ms
27,030	12.52 ms	162.92 ms
35,992	0.41 s	8.65 s

옷감 메시의 삼각형 개수가 3만개 미만일 때는 메모리 할당 시간 때문에 GPU가 CPU보다 느리다. 하지만 3만개 이상부터는 5~10배 속도가 향상되는 결과를 얻었다 (Table 1 참조). BVH 트리의 업데이트는 메모리 할당이 필요 없고 커널함수만 실행된다. 또한 병렬적으로 계산되

기 때문에 계산양이 적음에도 불구하고 좋은 성능을 보여주고 있으며, CPU보다 최대 10배 이상의 속도가 향상되었다 (Table 2 참조). BVH 트리의 탐색시간은 너비 우선탐색이 아닌 깊이 우선방법으로 탐색하기 때문에 병렬성이 떨어지지만 그럼에도 불구하고 10~20배 속도가 향상되는 높은 효율을 보여주었다.

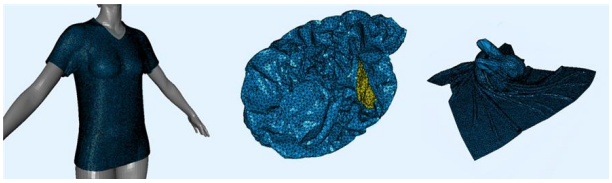

Fig. 10. Various collision scenes with our method.

Fig. 10은 본 논문에서 제안하는 방법을 좀 더 다양한 장면에서 실험한 결과이다. 앞에서 보여준 자기충돌 뿐만 아니라 객체-옷감 충돌에서도 안정적으로 수행되었다.

V. Conclusion

본 논문에서는 옷감 시뮬레이션에서 계산양이 큰 자기충돌을 GPU 기반으로 가속화하는 방법을 소개했다. 일반적으로 충돌검사에서 많이 사용하는 BVH를 GPU로 최적화하기 위해, 본 논문에서는 1) GPU에서 BVH 트리를 효율적으로 구축, 업데이트 및 순회하는 방법을 제안하고, 2) 삼각형 메쉬에서 R-Triangle을 GPU 커널에 맞게 계산양이 최소화하는 방법을 제안했다. 이 기법은 병렬성을 최대한 유지하고 가속화하는 것에 초점을 맞추었다. 본 논문에서 실험한 모든 결과에서 CPU기반 알고리즘에 비해 빠르고 효율적으로 충돌검사를 할 수 있음을 보여주었고, 옷감의 자기충돌 뿐만 아니라 객체-옷감과의 상호작용에서도 빠른 결고를 얻을 수 있었다.

그럼에도 불구하고 순회하는 과정에서 병렬성과 메모리를 동시에 최적화하는 방법은 찾아내지 못했다. 향후 이 메모리 문제를 해결하여 너무 우선 탐색으로 충돌검사를 최적화할 수 있도록 알고리즘을 확장할 계획이다.

ACKNOWLEDGEMENT

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00254695).

REFERENCES

[1] Bridson, Robert, Ronald Fedkiw, and John Anderson. "Robust treatment of collisions, contact and friction for cloth animation." In Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 594-603. 2002. DOI: 10.1145/566654.566623

[2] Tang, Min, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. "PSCC: Parallel self-collision culling with spatial hashing on GPUs." Proceedings of the ACM on Computer Graphics and Interactive Techniques 1, no. 1 (2018): 1-18. DOI: 10.1145/3203188

[3] Dionisio, John David N., William G. Burns Iii, and Richard Gilbert. "3D virtual worlds and the metaverse: Current status and future possibilities." ACM Computing Surveys (CSUR) 45, no. 3 (2013): 1-38. DOI: 10.1145/2480741.2480751

[4] Shams, Mahmoud Y., Omar M. Elzeki, and Hanaa Salem Marie. "Towards 3D virtual dressing room based user-friendly metaverse strategy." In The Future of Metaverse in the Virtual Era and Physical World, pp. 27-42. Cham: Springer International Publishing, 2023. DOI: 10.1007/978-3-031-29132-6_2

[5] Tang, Min, Huamin Wang, Le Tang, Ruofeng Tong, and Dinesh Manocha. "CAMA: Contact-aware matrix assembly with unified collision handling for GPU-based cloth simulation." In Computer Graphics Forum, vol. 35, no. 2, pp. 511-521. 2016. DOI: 10.1111/cgf.12851

[6] Jiang, Chenfanfu, Theodore Gast, and Joseph Teran. "Anisotropic elastoplasticity for cloth, knit and hair frictional contact." ACM Transactions on Graphics (TOG) 36, no. 4 (2017): 1-14. DOI: 10.1145/3072959.3073623

[7] Heo, Jae-Pil, Duksu Kim, Joon-Kyung Seong, Jeong-Mo Hong, Min Tang, and Sung-Eui Yoon. "FASTCD: Fracturing-aware stable collision detection." In ACM SIGGRAPH 2010 Posters, pp. 1-1. 2010. DOI: 10.1145/1836845.1836961

[8] Zhang, Xinyu, and Young J. Kim. "Scalable collision detection using p-partition fronts on many-core processors." IEEE Transactions on Visualization and Computer Graphics 20, no. 3 (2013): 447-456. DOI: 10.1109/TVCG.2013.239

[9] Fan, Wenshan, Bin Wang, Jean-Claude Paul, and Jiaguang Sun. "A hierarchical grid based framework for fast collision detection." In Computer Graphics Forum, vol. 30, no. 5, pp. 1451-1459. Oxford, UK: Blackwell Publishing Ltd, 2011. DOI: 10.1111/j.1467-8659.2011.02019.x

[10] Weller, René, Nicole Debowski, and Gabriel Zachmann. "kDet: Parallel constant time collision detection for polygonal objects." In Computer Graphics Forum, vol. 36, no. 2, pp. 131-141. 2017. DOI: 10.1111/cgf.13113

[11] Tang, Min, Dinesh Manocha, and Ruofeng Tong. "MCCD: Multi-core collision detection between deformable models using

front-based decomposition." Graphical Models 72, no. 2 (2010): 7-23. DOI: 10.1016/j.gmod.2010.01.001

[12] Tang, Min, Sean Curtis, Sung-Eui Yoon, and Dinesh Manocha. "Interactive continuous collision detection between deformable models using connectivity-based culling." In Proceedings of the 2008 ACM symposium on Solid and physical modeling, pp. 25-36. 2008. DOI: 10.1109/TVCG.2009.12

[13] Schvartzman, Sara C., Álvaro G. Pérez, and Miguel A. Otaduy. "Star-contours for efficient hierarchical self-collision detection." In ACM SIGGRAPH 2010 papers, pp. 1-8. 2010. DOI: 10.1145/1833349.1778817

[14] Wang, Tongtong, Zhihua Liu, Min Tang, Ruofeng Tong, and Dinesh Manocha. "Efficient and Reliable Self‐Collision Culling Using Unprojected Normal Cones." In Computer Graphics Forum, vol. 36, no. 8, pp. 487-498. 2017. DOI: 10.1111/cgf.13095

[15] Lefebvre, Sylvain, and Hugues Hoppe. "Perfect spatial hashing." ACM Transactions on Graphics (TOG) 25, no. 3 (2006): 579-588. DOI: 10.1145/1141911.1141926

[16] Pabst, Simon, Artur Koch, and Wolfgang Straßer. "Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces." In Computer Graphics Forum, vol. 29, no. 5, pp. 1605-1612. Oxford, UK: Blackwell Publishing Ltd, 2010. DOI: 10.1111/j.1467-8659.2010.01769.x

[17] Faure, Xavier, Florence Zara, Fabrice Jaillet, and J-M. Moreau. "An Implicit Tensor-Mass solver on the GPU for soft bodies simulation." In Eurographics Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS). 2012.

[18] Wong, Tsz Ho, Geoff Leach, and Fabio Zambetta. "An adaptive octree grid for GPU-based collision detection of deformable objects." The Visual Computer, Vol. 30, No. 6, pp. 729-738, 2014. DOI: 10.1007/s00371-014-0954-1

[19] Selle, Andrew, Jonathan Su, Geoffrey Irving, and Ronald Fedkiw. "Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction." IEEE Transactions on Visualization and Computer Graphics, Vol. 15, No. 2 pp. 339-350, 2008. DOI: 10.1109/TVCG.2008.79

[20] Tang, Min, Huamin Wang, Le Tang, Ruofeng Tong, and Dinesh Manocha. "CAMA: Contact‐aware matrix assembly with unified collision handling for GPU‐based cloth simulation." In Computer Graphics Forum, Vol. 35, No. 2, pp. 511-521. 2016. DOI: 10.1111/cgf.12851

[21] Popov, Stefan, Johannes Günther, Hans‐Peter Seidel, and Philipp Slusallek. "Stackless KD‐tree traversal for high performance GPU ray tracing." In Computer Graphics Forum, Vol. 26, No. 3, pp. 415-424, 2007. DOI: 10.1111/j.1467-8659.2007.01064.x

[22] Curtis, Sean, Rasmus Tamstorf, and Dinesh Manocha. "Fast collision detection for deformable models using representative-triangles." In Proceedings of the 2008 symposium on Interactive 3D graphics and games, pp. 61-69. 2008. DOI: 10.1145/1342250.1342260

Authors

Jong-Hyun Kim received the B.A. degree in the Department of Digital Contents at Sejong University in 2008. He received M.S. and Ph.D. degrees in the Department of Computer Science and Engineering at Korea University, in 2010 and 2016. Prof. Kim is an Associate Professor in the College of Software and Convergence (Dept. of Design Technology) in Inha University. His current research interests include fluid animation and virtual reality.