석사학위논문

병렬적 3차원 체인메일 알고리즘의 최적화 연구

2024년

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

이 상 은

석 사 학 위 논 문
지도교수 계희원

병렬적 3차원 체인메일 알고리즘의 최적화 연구

A Study on the Optimization of Parallel 3D Chainmail Algorithm

2023년 12월 일

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

이 상 은

석 사 학 위 논 문
지도교수 계희원

병렬적 3차원 체인메일 알고리즘의 최적화 연구

A Study on the Optimization of Parallel 3D Chainmail Algorithm

위 논문을 공학 석사학위 논문으로 제출함

2023년 12월 일

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

이 상 은

이상은의 공학 석사학위 논문을 인준함

2023년 12월 일

심사위원장 김 진 모 (인)

심 사 위 원 계 희 원 (인)

심 사 위 원 이 지 은 (인)

국 문 초 록

병렬적 3차원 체인메일 알고리즘의 최적화 연구

한 성 대 학 교 대 학 원
컴 퓨 터 공 학 과
컴 퓨 터 공 학 전 공
이 상 은

환자들은 수술 전 정확한 진단을 위해 필수적으로 의료 영상 정보를 수집한다. 이러한 정보는 볼륨 가시화와 같은 기술을 통해 특정 환부에 대한 상세 정보를 추출하는 데 중점을 두고 처리된다. 또한, 시뮬레이션 기술의 지속적인 발전과 고령화 사회의 도래로 의료 시뮬레이션의 중요성이 점점 증가하고 있다. 다양한 변형체 알고리즘의 접근 중에서도, 3D 체인메일 알고리즘은 그 독특한 사슬 갑옷과 유사한 움직임을 모방하는 방식으로 주목받으며 발전을 거듭해왔다. 이 알고리즘의 진보된 형태인 HP(Heterogeneous Parallel)-체인메일은 GPU의 병렬 처리 능력을 활용하는 방식으로 구현되었다.

본 논문에서는 GPU 커널의 런치 비용에 대한 깊이 있는 분석을 바탕으로, CPU와 GPU에서의 반복을 통한 비용 절감 방안을 제안한다. 또한, HP-체인메일에서 제시된 블록 구조가 가진 문제점을 분석하고, 이를 극복하기 위해 GPU에 최적화된 블록 크기의 제한을 도입한다. 이에 따라 제한된 블록 크기는 GPU의 공유 메모리를 최대한 활용하여 성능을 극대화한다. 본 논문은 제안하는 방법들을 통해 GPU 최적화된 3D 체인메일 알고리즘을 개선하

여, 더욱 신속하고 사실적인 의료 시뮬레이션을 가능케 하는 새로운 지평을 열었다.

【주요어】 의료 시뮬레이션, 3차원 체인메일 알고리즘, 변형체 알고리즘, 최적화, GPU 병렬 컴퓨팅

목 차

표 목 차

그 림 목 차

Ⅰ. 서론

1.1 연구 배경 및 목적

급속도로 진행되는 고령화 현상으로 의료 산업의 중요도는 계속해서 상승하고 있다. 인구 고령화로 인해 환자의 연령대가 높아짐에 따라 수술과 같은 외과적 의료 행위는 필요성뿐만 아니라 위험성도 증가한다. 그러므로 고령 환자는 자신이 받을 수술을 숙련된 외과의가 집도하기를 바랄 것이다. 더욱이 자신이 받을 수술의 위험도가 증가할수록 초보 외과의가 수술을 집도하기를 꺼릴 것이다. 이는 외과 의료 전문가를 계속해서 양성하는 과정을 약화한다. 이러한 현상이 반복되면 숙련된 의사의 수는 감소하지만 미숙한 의사의 수가 증가하여 의료 행위의 신뢰도가 감소한다. 이 문제를 해결하기 위하여 다양한 곳에서 외과적 의료 시뮬레이션 시스템이 개발되고 있다. 이 시뮬레이션 시스템은 프로그램에서 구현된 수술 대상의 현실적인 변형과 즉각적인 변형을 필수적으로 요구한다.

현실적인 변형은 수술 대상의 물리적인 특성을 반영해야 하므로 상당한 계산이 필요하다. 지금까지 변형체를 조작하는 다양한 알고리즘들이 제시되었다. 하지만 의료 영상이라는 특수 분야는 비교적 데이터의 크기가 상당히 크기 때문에 변형체 알고리즘이 작동되면 수행 속도가 상당히 느리다는 문제를 지니고 있다. 3D 체인메일(Chainmail) 알고리즘(Gibson, S. F., 1997)은 실시간 의료 시뮬레이션에서 사용되기에 적합한 대표적인 변형체 알고리즘이다. 데이터에 변형이 일어나면 변형이 시작된 위치인 근원점(epicenter)를 중심으로 사슬 갑옷이 움직이듯이 작동한다. 3D 체인메일의 계속된 발전으로 GPU를 사용하여 병렬적으로 작동하는 HP-Chainmail 알고리즘(Rodríguez, A., 2016)이 제시되었다.

본 논문에서는 HP-Chainmail 알고리즘의 효율성과 성능을 극대화하기 위한 연구를 진행한다. 특히, 이 연구는 HP-Chainmail에서 미처 다루지 못했던 핵심적인 부분인 CUDA 커널의 성능과 최적화 방법에 대한 심층적인

분석을 제공한다. 먼저, 연구는 의료 영상의 크기가 변형체 알고리즘의 수행 속도에 미치는 영향을 면밀히 조사한다. 이는 병렬 처리 과정에서 데이터의 크기와 구조가 알고리즘의 효율성에 중요한 역할을 함을 시사한다. GPU 기반 병렬 프로그래밍에서는 CPU 영역에서 GPU로 작업을 지시하는 '커널 런치(kernel launch)' 과정이 필수적인데, 이 과정이 반복되면서 알고리즘의 전체 수행 시간에 상당한 영향을 미친다.

3D 체인메일 알고리즘은 한 번의 변형이 발생할 때마다 볼륨 데이터의 모든 영역에 변형이 전파되기까지 반복적인 커널 런치를 요구한다. 이러한 반복적인 런치는 전체 처리 시간에 상당한 지연을 야기하며, 이는 특히 대규모 데이터 세트를 처리하는 데 있어 심각한 문제가 된다. 본 논문에서는 이러한 문제를 해결하기 위해 GPU 커널의 내부에서 반복적인 처리를 통한 새로운 최적화 방법을 제시한다. 이 방법은 GPU의 스레드들을 효율적으로 활용하여 한 번의 커널 런치로 여러 단계의 처리를 동시에 수행할 수 있도록 함으로써, 전체 처리 시간을 대폭 줄일 수 있다.

또한, 이 연구는 의료 물리 시뮬레이션의 속도를 향상시키기 위해 HP-Chainmail에서 사용된 활성화 블록 기법을 개선한다. 기존의 활성화 블록 기법은 특정 영역의 변형에만 초점을 맞추어 효율성을 높였지만, 전체 데이터 세트에 대한 광범위한 최적화는 미흡했다. 이를 개선하기 위해, 본 연구는 GPU가 제공하는 특수 메모리를 보다 적극적으로 활용하는 방식을 도입하여, GPU의 처리 능력을 최대한 활용한다. 이러한 개선을 통해, GPU 기반의 병렬 처리에서 더욱 빠르고 효율적인 알고리즘 수행이 가능해졌으며, 결과적으로 실시간에 가까운 의료 시뮬레이션을 구현하는 데 중요한 발전을 이루었다. 이와 같은 연구 결과는 의료 시뮬레이션 분야에서의 응용 가능성을 크게 확장시키고, 실질적인 의료 현장에서의 활용도를 높이는 중요한 기여를 제공한다.

1.2 논문의 구성

본 논문의 구성은 다음과 같다. 2장에서는 변형체 알고리즘의 기본적인

설명 및 다양한 변형체 알고리즘을 소개한다. 변형체 알고리즘 중 하나인 3D 체인메일의 역사와 기본적인 작동 흐름인 전파와 안정화를 설명하고, 최근의 연구인 HP-Chainmail을 분석하여 서술한다. 그리고 GPU 병렬 프로그램에서 사용되는 NVIDIA의 CUDA를 간단하게 설명한다. 3장에서는 GPU 커널 내부 반복을 이용하여 3D 체인메일 알고리즘의 개선 방안을 설명한다. 4장에서는 GPU에 최적화된 블록 구조를 도입하여 활성화 블록 기법을 개선하는 방법을 설명한다. 5장에서는 앞선 3, 4장의 연구에 관한 실험 결과가 설명되어 있으며, 마지막으로 6장에서는 본 논문의 연구를 결론짓는다.

II. 관련 연구

2.1 변형체 알고리즘

물체를 변형하기 위해 다양한 방식의 변형체 알고리즘들이 제시되었다(Zhang, J., 2017)(Zhang, J., 2018a). 첫째는 질량-스프링 모델이다(Xavier, P., 1995). 이 모델에서는 물체를 질량을 갖는 원소들의 집합으로 여긴다. 각 원소는 이웃 원소들과 스프링 연결 상태를 지니며, 물리 법칙에서 사용되는 훅의 법칙(Hooke's Law)을 따른다. 스프링을 잡아당긴 후, 스프링 끝의 물체 위치가 수렴되기까지 위치 진동이 일어나듯이 이 모델의 원소 또한 자신의 위치가 수렴되기까지 진동이 발생한다. 이 모델을 이용한 다양한 후속 연구가 진행되었지만, 스프링 모델을 사용한 특성상 진동의 수렴이 늦어질 수 있다는 단점이 있다.

둘째는 유한요소법을 이용하는 알고리즘이다(Wolters, C.H., 2002)(Essa, I. A., 1992)(Metaxas, D., 1992). 유한요소법(FEM, Finite Element Method)은 기계공학의 다양한 분야에서 사용되는 방법의 일종으로, 물체를 임의의 자유도를 지닌 원소들의 집합으로 여긴다. 이때 물체의 특징에 알맞은 미분 방정식을 적용한 연립 방정식을 생성하고, 행렬을 이용한 풀이를 통해 물체에 변형이 적용된 시뮬레이션을 구현할 수 있다. 다양한 물질의 성질을 고려하여 변형을 계산할 수 있다는 장점이 있지만, 적용된 미분 방정식의 해를 구하는 과정의 비용이 많이 든다는 단점이 있다. 이 큰 단점으로 유한요소법을 사용한 알고리즘은 실시간 시뮬레이션에 적용하기에는 부적합하다.

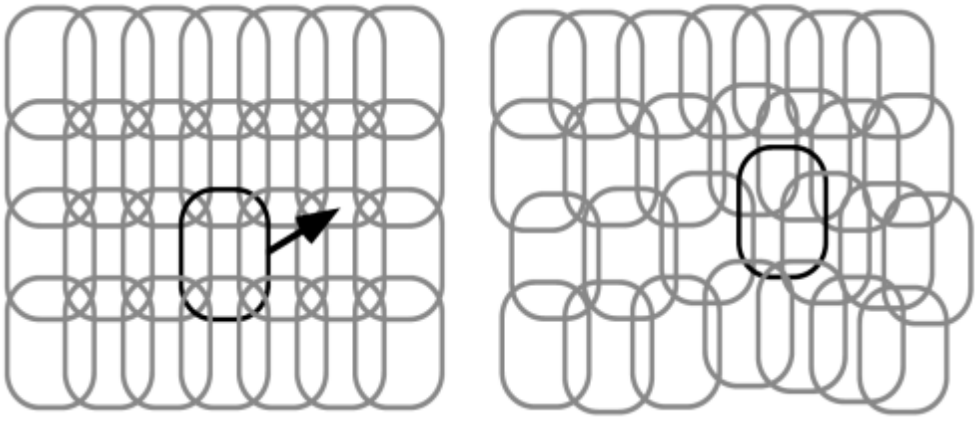
2.2 3D 체인메일 알고리즘

2.2.1 3D 체인메일 알고리즘의 발전 과정

3D 체인메일 알고리즘은 S.F. Gibson이 처음으로 제안했다(Gibson, S.F.,

1997). 체인메일(Chainmail)이라는 알고리즘의 이름처럼 사슬 갑옷의 움직임을 물체 변형에 적용하였다. 이 알고리즘은 물체를 이루고 있는 각각의 원소가 모두 고리 형태로 연결되어 있다고 가정한다. 선택된 원소의 위치가 변화하면 사슬이 움직이듯이 주변 원소들의 위치가 변화에 따라 움직이게 된다. 초기 3D 체인메일 알고리즘은 단순하게 한 가지의 성질을 지닌 물체에만 적용되었다.

이후 Enhanced Chainmail(Schill, M. A., 1998)에서 여러 가지 성질의 물질로 구성된 물체에 알고리즘을 적용할 수 있음을 보여주었으나 단순한 볼륨 구조에서만 적용할 수 있었다. Generalised Chainmail(Li, Y., 2003)에는 복잡한 위상을 가진 볼륨에 적용하였으나 단일한 성질을 가지고 있는 물체에만 적용하였다. 2015년에 SP(Sparse Parallel) Chainmail(Rodríguez, A., 2015)이 제안되어 3D 체인메일 알고리즘에 병렬화가 가능함을 보였다. 다만, 이 알고리즘은 단일한 성질, 단순한 위상 구조를 지녀야 한다는 단점을 가지고 있다. SP-Chainmail 알고리즘의 단점을 개선하기 위하여 이어진 후속 연구인 HP(Heterogeneous Parallel) Chainmail 알고리즘(Rodríguez, A., 2016)은 GPU 병렬화를 유지하면서도, 다양한 성질을 가지며 복잡한 위상 구조의 볼륨에도 적용할 수 있도록 발전되었다. 그 이후로도 TSVE(Time-Saving-Volume-energy conserved) 체인메일(Zhang, J., 2016)이나 타원체 모양으로 고리 구조를 변형하여 동작하는 Ellipsoid Bounding Region-based ChainMail(Zhang, J., 2017b) 등으로 다양하게 연구가 진행되고 있다.



[그림 2-1] 3D 체인메일의 2차원에서의 동작 예시(Gibson, S.F., 1997)

2.2.2 3D 체인메일의 전파(Propagation)

3D 체인메일은 전파(Propagation)와 안정화(Relaxation)의 두 가지 과정을 반복하여 수행한다. 전파는 원소의 특성에 따라 설정한 제한 값(constraint)을 고려하여 원소가 고리 모양의 이동 범위 제한 안에서 움직이는 것이다. 3D 체인메일은 선택된 고리가 중심이 되어 후보자(candidate), 즉 주변 원소들의 위치를 제어한다. 예를 들어, 선택된 고리인 원소는 후보자인 우측 이웃 원소와 제한 거리를 비교하고 제한 거리 바깥에 후보자가 존재한다면 해당 후보자의 위치를 자기 자신과 제한 거리 이내로 조정한다. 또한, 위치가 변경된 원소는 자기 자신이 다시 전파 변형의 중심이 되어 반복된다.

2.2.3 3D 체인메일의 안정화(Relaxation)

전전파 과정이 볼륨 전체에 수행된 이후에는 물체가 변형된 직후의 불안정한 상태를 해소해야 한다. 또한, 물체가 가지고 있는 탄성을 전혀 고려하지 않은 움직임이므로 더욱 현실적인 움직임을 위해 안정화(Relaxation)를 수행한다. 안정화에서는 탄성을 반영된 움직임을 보이기 위해서, 스프링의 움직임을 표현하는 물리 법칙인 훅의 법칙을 사용한다. 모든 이웃 원소에 대해 내부

적인 퍼텐셜 에너지가 최소한이 되는 위치로 원소의 위치를 조정하게 된다.

```
for (all non-zero object elements) {
    determine lowest energy position for the element
        relative to its (top, bottom, right, left, front,
        and back) neighbors
    move element towards that position by step size
}
```

[그림 2-2] 3D 체인메일 알고리즘의 안정화 의사 코드(pseudo code)
(Frisken-Gibson, S.F., 1999)

2.3 HP-체인메일 알고리즘

HP-Chainmail(이하 HP-체인메일)이란 3D 체인메일이 지닌 문제점인 병렬화의 어려움과 다양한 물성을 가진 이형질의 물체에 대한 변형 문제를 해결하기 위해 제시된 알고리즘이다. HP-체인메일은 앞선 연구들의 난점들을 돌파하기 위해 여러 가지 해법을 제안한다.

앞선 연구에서 변형의 중심인 원소가 이웃 원소들의 위치를 조정했지만, 병렬화를 적용하기 위해서는 해당 방법을 고수할 수 없다. 병렬화는 여러 개의 스레드를 사용하는 멀티스레드 방식의 동작을 의미한다. 그리고 병렬화를 적용한 체인메일 알고리즘에서 하나의 스레드는 한 원소의 위치를 계산한다. 여기서 각 스레드는 자신에 해당하는 원소의 이웃 원소의 위치 정보를 토대로 움직이는데, 이는 이전의 위치 조정 방식처럼 근원점을 중심으로 이웃 원소를 조정하는 순차적인 알고리즘과는 전혀 다른 구조이다.

이렇게 순차적인 알고리즘과 다르게 멀티스레드를 사용한 병렬적 체인메일 알고리즘은 변형 순서가 어긋나는 문제가 발생할 수 있다. 기초적인 체인메일 알고리즘에서 한 번의 발생한 변형으로 각 원소는 여러 방향으로부터 영향을 받더라도, 결과적으로 한 번의 전파를 수행하고 한 번 위치를 이동한다.

그렇다면 원소의 위치 변화는 단 한 번만 발생하는 것이 가장 효율적일 것이다. 이때, 우리가 기준으로 삼아야 할 방향은 가장 빠르게 위치 변화한 방향이 되어야 할 것이다. 왜냐하면, 물리적 시뮬레이션 알고리즘의 특징인 현실 반영이라는 측면으로 바라볼 때, 가장 빠르게 위치 변화한 방향을 기준으로 삼는 것은 현실의 물리 법칙을 따르기 때문이다. 물리적으로 변형을 일으키고자 물체에 힘을 가하면 그 위치로부터 파동처럼 힘이 전달된다. 이때 전달되는 힘은 단단한 물질에서 더 빠르게 전파된다. 하지만 이 법칙을 지키려면 모든 원소가 전파의 순서상 정렬이 되어 있다는 가정이 필요하다. 즉 변형의 전파는 빠르게 변형이 도달한 원소를 따라서 순차적으로 이루어지지만, 멀티스레드 환경은 전파에 따른 순서가 어느 원소가 더 빠른지 구분할 수 없으므로 변형 순서가 어긋나는 문제가 생길 수 있다.

특히 이 문제는 이형질 물체에서 두드러질 수 있다. 병렬적으로 구현한 알고리즘은 각 스레드가 독립적으로 변경 위치를 계산한다. 이때 물리적 성질이 다른 이형질의 물체에 전달된 변형 파동은 모두 같은 속도로 전달되지 않고 물질의 성질에 따라 다른 속도로 전달될 것이다. 그렇다면 어떤 한 원소에 빠르게 도달한 변형 전파로 계산된 위치는 느리게 도달한 변형 전파로 인해 두 번 이상 연산 되어야 한다. 즉 한번 가해진 변형에 한 번의 원소 이동은 한 변형 파동으로 두 번 이상의 변형이 이루어지는 문제가 생기는 것이다.

이 난점을 해결하기 위해서 HP-체인메일은 타임스탬프(timestamp)를 도입한다. 여러 파동이 순차적으로 원소에 도달하면 그중 가장 타임스탬프에 기록된 시간이 빠른 파동에 대해서만 변형을 수행하기 위해서이다. 변형이 수행되면 해당 파동과 원소의 성질에 따라 설정된 시간 정보를 이용하여 도달 시간을 계산하여 다시 전파한다. 이와 같은 과정을 거쳐 주변 원소 중 도달 시간이 가장 빠른 변형 파동에 대해서만 변형이 이루어진다. 예를 들어, 사람의 팔에 해당하는 데이터가 있고 변형이 발생했을 때, 이웃 뼈 원소의 전파는 단단한 성질을 가지기 때문에 더 빠르게 변형 파동을 전파한다. 뼈 원소에서 전파된 변형 파동은 이웃 피부 원소의 전파보다 빠르게 도달하므로 팔의 변형은 뼈의 변형을 우선하여 이루어지게 된다.

이어서 HP-체인메일 또한 안정화를 거친다. 기존의 3D 체인메일이 가정

하였던 훅의 법칙을 따라서 내부 에너지를 감소시키는 방향으로 움직인다. 내부의 다양한 성질을 가진 물체는 성질이 다른 원소들이 다른 제한 값(constraint)을 가진다. 이러한 제한 값을 이용하여 x, y, z 방향으로 편미분 방정식을 풀고 원소가 조정되어야 하는 위치를 계산한다.

병렬적으로 알고리즘을 수행하는 것은 수행 속도 향상의 굉장한 발전을 가져왔지만, 움직이지 않아도 되는 원소들에 대해서도 병렬처리하여 GPU의 스레드가 낭비되는 경우가 존재한다. HP-체인메일은 이전 연구인 SP-체인메일의 방법을 이어서 활성화 블록(active block)을 이용한다. 활성화 블록을 적용하지 않게 되면 계산이 필요하지 않은 영역도 GPU 커널에 전달하여, GPU 자원을 낭비하게 되므로 활성화 블록 방법이 필요함을 알 수 있다. 물체를 구역으로 나누어 일정한 원소를 묶어 블록 구조를 가진다. 변형이 필요한 위치의 블록의 활성화 플래그(flag)가 켜지고, 해당 블록의 모든 원소가 변형이 종료될 때까지 플래그가 꺼지지 않는다. 이러한 방법들을 이용하여 HP-체인메일은 다양한 부분에서 3D 체인메일 알고리즘을 개선했다.

2.4 CUDA

NVIDIA의 CUDA(Calculate Unified Device Architecture)는 병렬 컴퓨팅 플랫폼 및 프로그래밍 모델로서, 개발자가 NVIDIA의 GPU를 사용하여 일반적인 처리를 위한 병렬 컴퓨팅 작업을 수행할 수 있도록 설계되었다. CUDA는 C, C++, 그리고 Fortran 같은 고수준 프로그래밍 언어를 지원하여, GPU 상에서 수백에서 수천 개의 스레드를 동시에 실행할 수 있는 강력한 병렬 처리 능력을 제공한다.

CUDA는 개발자가 대규모 병렬 프로세서의 계산 능력을 활용할 수 있도록 하는데, 이는 특히 과학적 계산과 그래픽 처리에서 그 효과가 두드러진다. CUDA 프로그래밍 모델은 '커널(Kernel)'이라 불리는 함수들을 통해 구현되며, 이 커널들은 GPU의 수천 개의 스레드에서 독립적으로 실행된다. 각 스레드는 그리드와 블록이라는 구조로 구성되며, 스레드 블록 내에서는 공유 메모리를 이용하여 데이터를 공유할 수 있다. 이러한 구조는 데이터의 대량 병렬

처리를 가능하게 하며, 전통적인 CPU 기반의 시스템보다 훨씬 빠른 속도로 복잡한 계산을 수행할 수 있게 한다.

CUDA는 GPU의 메모리 계층 구조를 활용하여 효율적인 데이터 관리와 접근을 제공한다. 이는 공유 메모리, 상수 메모리, 텍스처 메모리 등을 포함하는데, 각각의 메모리 유형은 특정 애플리케이션의 요구 사항에 따라 최적화될 수 있다. 예를 들어, 공유 메모리는 블록 내 스레드 간의 빠른 데이터 교환을 위해 사용되며, 상수 메모리는 변경되지 않는 데이터에 대한 효율적인 접근을 제공한다.

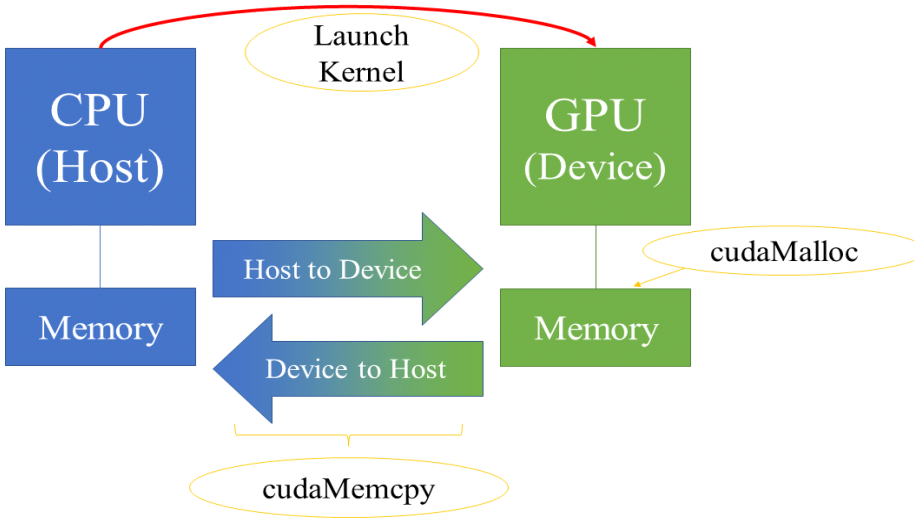
또한, CUDA는 전역 메모리에 대한 접근을 최적화하는 메모리 접근 패턴을 제공하여, 메모리 뱅크 충돌을 피하고, 메모리 대역폭을 최대한 활용할 수 있도록 한다. 이는 메모리의 공간적 및 시간적 지역성을 고려하여, 데이터 전송과 처리를 최적화함으로써 전체적인 프로그램의 성능을 향상시킨다.

종합적으로, CUDA는 GPU의 병렬 처리 능력을 고도로 활용하는 효율적이고 유연한 방법을 제공함으로써, 과학적 계산, 데이터 분석, 기계 학습, 그리고 그래픽스 렌더링 등의 분야에서 혁신을 가능하게 하는 핵심 기술로 자리 잡고 있다.

III. GPU 커널 런치의 비용을 최소화하는 향상된 3D 체인메일 알고리즘

3.1 개요

CUDA 기반 GPU 프로그래밍의 작동 원리는 [그림 3-1]에 의해 개략적으로 설명된다. 이 과정에서 CPU는 GPU에 병렬 실행을 지시하는데, 이를 '커널 런치'(kernel launch)라고 한다. Jez(2014)에 따르면, GPU 커널을 런치하는 데는 대략 10μs의 시간 비용이 소모된다. 기존의 체인메일 알고리즘은 한 번의 함수 호출을 통해 하나의 원소에 대한 연산만을 수행하는 방식으로 구현되어왔다. 이러한 설계는 각 원소에 대해 개별적으로 알고리즘을 적용하는 데 초점을 맞추었기 때문에 발생한다. 그러나 병렬 처리 기법을 적용한 HP-체인메일은 GPU 커널을 한 번 런치하는 것으로도 다수의 원소에 대한 연산을 동시에 진행할 수 있도록 발전되었다.



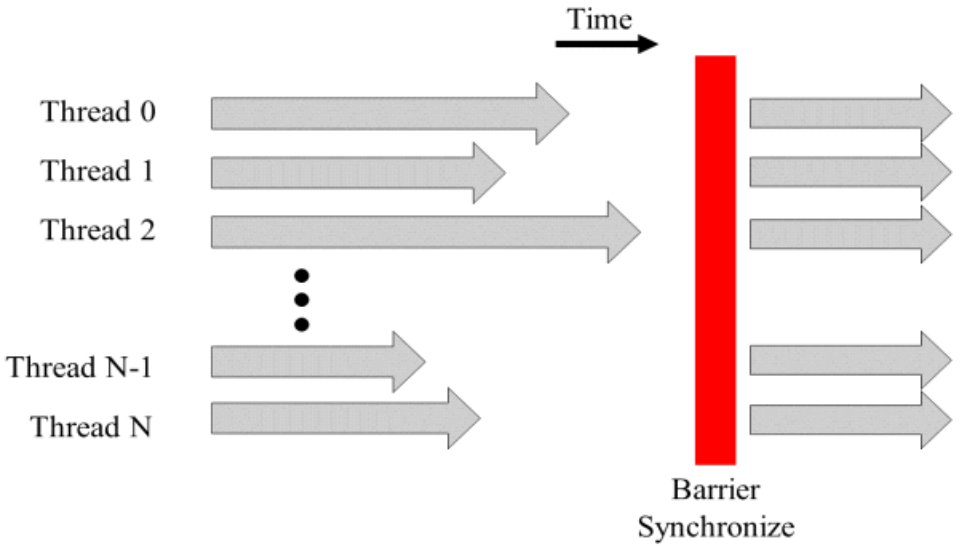
[그림 3-1] CUDA GPU 프로그래밍의 개요

3.2 커널 런치 비용 최소화를 위한 GPU 커널의 내부 반복

커널 런치의 시간 비용이 상대적으로 낮은 10μs에 불과하지만, 의료 시뮬레이션에서 복잡한 데이터 처리를 단일 원소 연산에 의존하는 방식은 심각한 제약으로 여겨진다. 체인메일 알고리즘에 의한 3차원 변형 파동 전파는 대각선 방향으로는 이루어지지 않으며, 오직 상, 하, 좌, 우, 전, 후의 직교 방향을 따라 진행된다. 이는 변형 파동이 각 커널 런치마다 6개의 주요 방향으로 한 단위씩 전파됨을 의미하며, 원소 간에 건너뛰는 전파는 발생하지 않는다. 결과적으로, 데이터의 각 축에 대한 최대 맨해튼 거리만큼의 반복 런치가 요구되며, 특히 의료 영상에서 활용되는 대규모 데이터 세트는 이에 따른 상당한 수의 커널 런치를 필요로 한다.

본 논문에서는 500x500x300의 이상의 크기를 가진 데이터를 사용한다. 이 데이터의 경우, 마주 보는 꼭짓점 사이의 맨해튼 거리는 1300이다. 따라서 변형 파동이 전체 볼륨 데이터에 도달하기 위해서는 최소 1,300번의 커널 런치가 필요하다. 이로 인해 발생하는 반복적인 커널 런치의 시간 비용은 10μs * 1,300회 = 13,000μs = 13ms이다. 걸리는 시간은 변형의 전파 과정에서 발생하는 커널 런치 횟수만 고려하여 계산한 것으로, 체인메일 알고리즘의 안정화를 위한 커널의 수행까지 고려하면 훨씬 큰 시간 비용이 발생한다. 안정화를 위해 한 번의 전파 과정이 끝난 후에 여러 번의 안정화가 필요하기 때문이다.

이 연구는 GPU 커널의 런치 횟수를 획기적으로 감소시키기 위한 새로운 방법을 제안한다. 이 방법은 커널 내부에서의 반복에 중점을 둔다. 다만, 이 방법을 적용하기 위해서는 의료용 볼륨 데이터를 재구조화하고 GPU의 전역 메모리 공간에 저장되어 있어야 한다. 이러한 저장 방식은 GPU의 모든 스레드가 데이터에 접근할 수 있도록 한다. 따라서 이론적으로 커널 내부에서 반복문을 적용하면 한 번의 커널 런치만으로도 최대 맨해튼 거리만큼 반복을 실행할 수 있으며, 이는 커널 런치 횟수를 획기적으로 줄일 수 있다는 것을 의미한다.



[그림 3-2] Barrier synchronization.

하지만 단순한 커널 내부에서의 반복문 수행은 장점만 가지고 있지 않다. 왜냐하면, GPU 커널 내부의 스레드들은 단독 구조로 존재하지 않고 실행 흐름도 각자 다르게 흐르기 때문이다. GPU 커널의 스레드들은 블록이라는 이름의 동작 단위 구조로 이루어져 있다. [그림 3-2]와 같이 한 블록의 스레드는 각각의 실행 흐름이 존재한다. 이 스레드들은 전역 메모리 공간에 있는 데이터를 참조할 수 있다. 하지만 스레드가 동기화 없이 연속적으로 작업을 수행하면, 스레드 간 동작 시점의 차이, 즉 실행 시간 지연이 누적되어 한 블록 내의 스레드끼리 메모리 접근 시점이 크게 달라지는 문제가 생길 수 있다. 접근 시점의 차이는 마침 변형이 도달한 원소를 참조하는 경우, 실행 시간 지연은 변형이 도달하지 않았을 때의 위치를 참조하게 한다. 이러한 오차가 누적되면, 변형이 일어나는 경우와 변형이 일어나지 않은 경우가 혼재하여 변형이 비정상적으로 보이게 된다. 이것은 알고리즘을 구현한 프로그램 작성자가 의도한 결과가 아니다.

단일 스레드 블록의 동기화 문제뿐만 아니라, 모든 스레드 블록이 동기화된 것은 아닌 문제가 존재한다. 스레드 블록 내에서의 동기화는 커널 내부에서 __syncthreads() 함수를 이용하여 이루어지며, 스레드 블록 간의 동기화는 cudaDeviceSynchronize() 함수를 통해 외부에서 수행된다. 그러나 스레드 블

록 간의 동기화는 커널 내부에서 실행할 수 없다. 이 함수는 호스트 영역에서 호출되는 코드로, 커널 외부에서 모든 스레드 블록의 종료 시점을 일치시킨다. 따라서 커널 내부에서는 모든 스레드 블록의 실행 흐름을 정렬할 수 없다.

이 문제를 해결하기 위해 본 연구에서는 GPU와 CPU 동기화를 혼합하는 방법을 사용한다. 이 방법은 양쪽의 이점을 취하여 GPU에서 효율적으로 스레드를 활용하면서 동시에 정확도 문제를 해결하기 위해 CPU에서 추가적인 동기화를 수행한다. GPU에서는 스레드 내 동기화를 효과적으로 수행하면서도, 정확도 문제를 회피하기 위해 CPU에서 전역적인 동기화를 추가로 수행한다. 이렇게 함으로써 CPU와 GPU의 협력 구조를 디자인하여 거동의 문제를 최소화하고 효율성을 극대화한다. 이러한 CPU와 GPU 간의 협력 구조를 설계하는 데에는 정확도와 성능 간의 균형을 맞추기 위한 적절한 비율을 고려해야 한다. 이 방법은 CPU와 GPU 간의 동기화를 통해 더욱 효과적인 결과를 얻을 수 있게 해준다.

Algorithm 1 : CPU 영역에서의 반복 알고리즘

input : 체인메일 알고리즘을 위해 구조화된 데이터 elems

```
1   For 데이터 elems의 최대 3차원 L1 거리 Do
2       propagate ≪ GRID, BLOCK ≫ (elems)
3       CudaDeviceSynchronize()
4       For 안정화 반복 횟수 Do
5           relax ≪ GRID, BLOCK ≫ (elems)
6           CudaDeviceSynchronize()
7       End For
8   End For
```

[표 1] CPU 영역에서의 반복 알고리즘

본 연구에서 제시된 CPU 영역의 반복 알고리즘은 [표 1]에 제시된 의사 코드를 통해 상세히 설명된다. 체인메일 알고리즘의 구현을 위해 볼륨 데이터는 특정한 구조적 변환을 거쳐 사용되며, 이는 각 원소의 공간적 위치 정보와 함께 변형 전파가 해당 원소에 도달한 시간을 기록하는 타임스탬프, 그리고 각 원소의 물질적 특성을 반영하는 정보를 포함한다.

이 구조는 전파 과정을 최대 L1 거리에 해당하는 횟수만큼 반복적으로 수행함으로써, 변형이 전체 볼륨 데이터에 효과적으로 전파되도록 한다. 또한, CPU 영역에서의 안정화 반복은 각 원소가 자신의 물성과 일치하는 최적의 위치로 조정될 수 있도록 충분한 반복 횟수를 가진다.

Algorithm 2 : 전파 과정에서의 GPU 커널 내부 반복 알고리즘

input : 체인메일 알고리즘을 위해 구조화된 데이터 elems,
스레드 블록 내의 x, y, z 방향 스레드 tx, ty, tz

```
1   For 커널 내부 반복 횟수 Do
2       e ← elems[tz][ty][tx]
3       fastestNeighbor, fastestNeighborDir
4       For 이웃 원소 n에 대하여 Do
5           newTime ← n.time + pTime(element, n)
6           If e.time > newTime Do
7           fastestNeighbor ← n
8           fastestNeighborDir ← n.dir
9           End If
10      End For
11      If fastestNeighbor에 대하여 Do
12          e.time ← fastestNeighbor.time + pTime(e, fastestNeighbor)
13          shiftElement(e, fastestNeighbor, fastestNeighborDir)
14      End If
15  End For
16  __syncthreads()
```

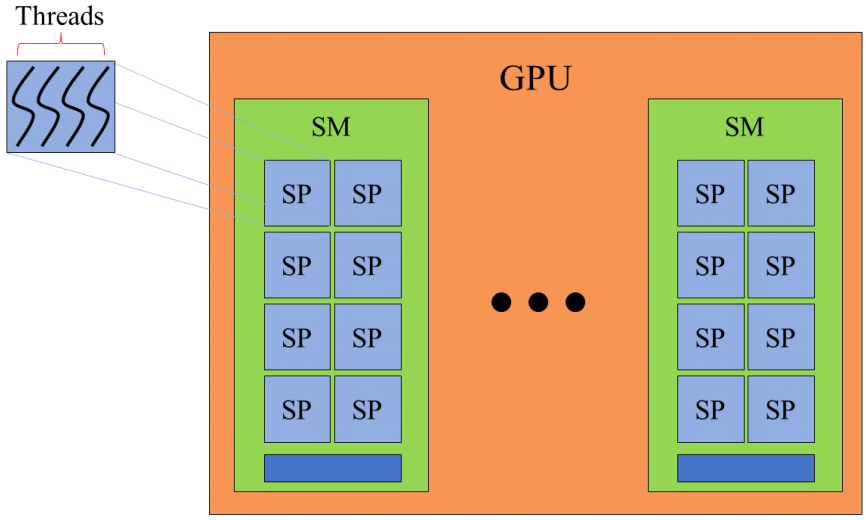
[표 2] 전파 과정에서의 GPU 커널 내부 반복 알고리즘

GPU 커널 내부에서의 반복적인 연산을 통해 변형 전파 과정의 효율성을 개선하고 있다. 이 접근법은 [표 2]에 도식화된 의사 코드에서 제시된 바와 같이, 커널 실행 도중에 원소의 위치 정보를 연속적으로 갱신하는 메커니즘에 기반한다. 이는 HP-체인메일 알고리즘에서의 방법론과 일치하며, 각 원소의 위치 정보 갱신은 해당 알고리즘의 정의에 따라 수행된다.

이 안정화에서도 커널 내부 반복의 원리를 적용한다. 여기서 안정화는 정해진 반복 횟수 동안 각 원소가 퍼텐셜 에너지를 최소화하는 방향으로 움직이도록 구성된다. 이때 퍼텐셜 에너지의 계산은 HP-체인메일 알고리즘에 기술된 바와 동일한 방식으로 이루어진다. 커널 내부 반복을 사용함으로써, 전파와 안정화 모두에서 불필요한 외부 커널 호출을 줄이고, 이로 인한 오버헤드를 최소화하여 전체적인 알고리즘 수행 시간을 단축시키는 효과를 달성하였다.

3.3 GPU 작동 원리에 따른 효율적 스레드 블록 이용 방법

HP-체인메일은 의미 없는 반복을 피하려고 커널에서 움직임이 필요 없는 원소들을 블록(block) 구조로 구획화하는 방법을 도입하였다. 그러나, HP-체인메일이 제안한 블록의 크기들은 모두 GPU 커널의 하나의 블록에서 처리할 수 있는 스레드의 숫자를 초과한다. 이를 고려할 때, 스레드 블록 단위로 GPU 커널을 런치하지 않았을 가능성이 커 보인다. 따라서 HP-체인메일에서 주장한 블록 구조는 GPU의 구조 및 작동 원리에 따라 최적화가 필요한 부분으로 판단된다. 최적화를 통해 GPU의 효율성을 높일 수 있을 것으로 예상하며, 이는 계속하여 블록 구조의 적절한 크기 및 GPU 커널의 효율적인 활용에 대한 중요한 고려 사항으로 다가올 것이다.



[그림 3-3] 단순화한 고전적인 GPU 구조

CUDA 커널은 전통적으로 GPU의 계층적 구조에 기반하여, 스레드를 그리드와 블록이라는 두 가지 주요 구조로 조직한다. 여기서 블록은 스레드들의 집합으로, 병렬적 체인메일 알고리즘의 문맥에서는 데이터 블록과 혼동을 방지하기 위해 '스레드 블록'으로 지칭한다. [그림 3-3]은 GPU 구조를 단순화하여 전통적인 형태로 표현하고 있다.

CUDA 커널 아키텍처는 GPU의 스트리밍 멀티프로세서(SM)를 기반으로 구축된 스레드 블록의 개념을 중심으로 설계되었다. 각 스레드 블록은 단일 SM에 대응되며, 이 SM은 다수의 스트리밍 프로세서(SP)로 구성되어 있다. 이 구조에서 GPU 커널의 그리드 크기는 사용 가능한 SM의 총 개수로 정의되며, 스레드 블록의 크기는 해당 SM에 할당된 스레드의 수를 나타낸다.

여기에서 주목할 점은 CUDA에서의 스레드 스케줄링 단위인 '워프'가 32개의 스레드로 구성된다는 사실이다. 워프 단위로 실행되는 스레드들은 같은 실행 타이밍을 공유하므로, 워프를 기준으로 그룹화함으로써 데이터 처리의 효율성을 극대화할 수 있다. 따라서, 효과적인 CUDA 프로그래밍을 위해서는 워프의 크기를 고려하여 스레드 개수를 2의 거듭제곱으로 설정하는 것이 바람직하며, 이는 데이터 블록과 스레드 블록 간의 일대일 대응을 통한 알고리즘 수행의 정확성을 보장한다.

HP-체인메일 알고리즘은 16x16x16 및 16x4x4의 블록 크기를 실험적으로 적용하여, SP-체인메일의 초기 연구에서 탐색된 32x32x32와 8x8x8 크기의 블록들과 비교하였다. 이러한 크기들은 2010년 이후에 출시된 CUDA의 계산 능력(compute capability) 2.0 이상을 가진 GPU에서 스레드 블록당 허용되는 최대 스레드 수인 1,024개의 제한을 넘어서는 문제를 내포한다. 특히 HP-체인메일이 제안하는 16x16x16 블록은 단일 스레드 블록에서 처리할 수 있는 범위를 초과하는 4,096개의 원소를 포함함으로써, 병렬 처리 구조에 대한 충분한 고려 없이 제시되었다고 볼 수 있다.

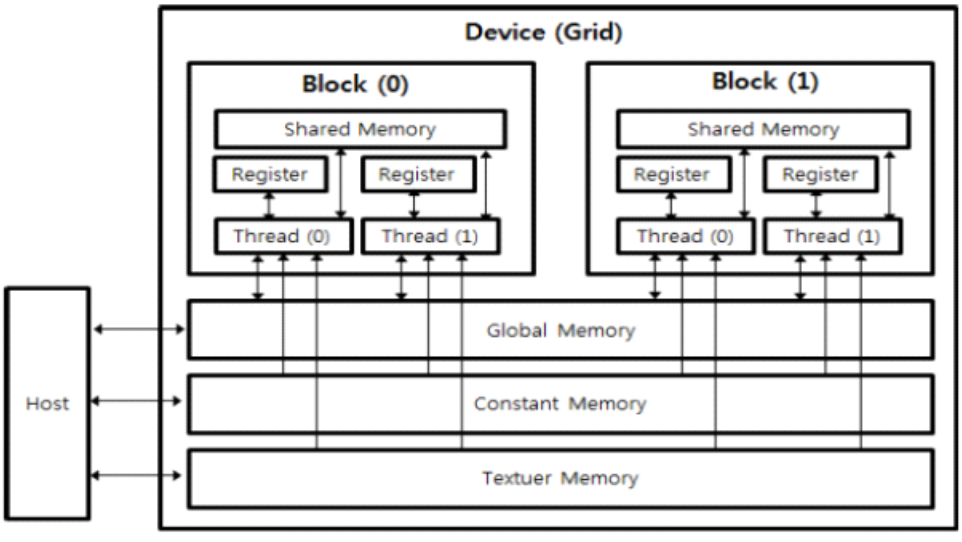
본 연구에서는 스레드 블록과 데이터 블록의 일대일 대응과 권장되는 CUDA 프로그래밍 방법론을 따르기 위하여 스레드 블록의 크기를 8x8x8로 설정할 것을 제안한다. 이는 CUDA의 워프 크기에 적합하며, 동시에 스레드 블록의 크기 제한을 준수한다는 장점을 가진다.

본 연구의 제안된 방법론은 스레드 블록과 데이터 블록의 일대일 대응을 통해 각 스레드가 독립적인 데이터 원소를 처리하도록 구성된다. 이를 통해 블록 내의 스레드들이 개별적으로 원소의 움직임을 책임지며, CUDA 라이브러리의 스레드 개수 제한을 고려하여 구현된 8x8x8 크기의 스레드 블록을 통해 복잡한 병렬 처리 작업을 보다 효율적으로 수행할 수 있음을 입증한다.

IV. 최적화된 활성화 블록을 적용한 3D 체인메일 알고리즘 최적화

4.1 개요

NVIDIA의 GPU 아키텍처에서 공유 메모리(Shared Memory)의 활용은 병렬 프로그램의 성능 향상에 필수적인 요소로 인식되고 있다. 공유 메모리는 병렬 컴퓨팅 환경 내에서 다수의 스레드가 데이터를 효율적으로 교환할 수 있도록 설계된 고속 메모리이다. 이는 스레드 간의 정보 공유를 용이하게 하여, 전역 메모리(Global Memory)와 비교하면 상대적으로 낮은 지연시간으로 메모리 접근이 가능하게 함으로써, 성능 최적화에 중요한 역할을 한다. 본 문서의 [그림 4-1]은 GPU 메모리 계층의 구조를 상세히 나타내고 있으며, 이를 통해 공유 메모리가 전체 메모리 아키텍처 중 어떤 위치에 배치되어 있는지, 그리고 다른 메모리 계층과 어떻게 상호 작용하는지를 명확히 이해할 수 있다.



[그림 4-1] GPU의 메모리 계층 구조(riverrun17, 2015)

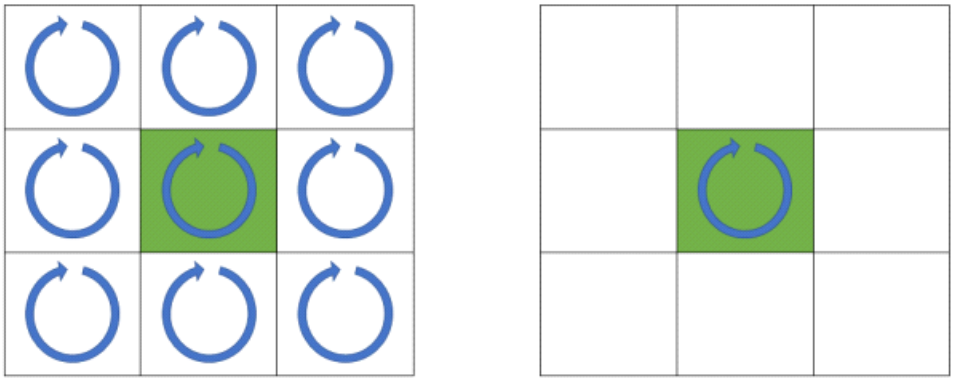
GPU 병렬 프로그래밍의 맥락에서, 공유 메모리의 전략적 활용은 CPU 기반 병렬 프로그래밍과 유사하게 성능 향상의 잠재력을 내포하고 있다. GPU 아키텍처 내에서 제공되는 공유 메모리는 프로그램의 실행 속도를 향상시킬 수 있는 중요한 자원으로 간주되며, 그 사용은 병렬 처리 효율을 극대화하는 데 있어 권장되고 있다. 그러나 공유 메모리의 한정된 용량과 메모리 뱅크 충돌과 같은 복잡성은 개발자에게 주의 깊은 관리를 요구한다. 이러한 맥락에서 본 논문은 공유 메모리를 활용한 3D 체인메일 알고리즘의 최적화 방안을 탐구한다. 이는 메모리 접근 패턴을 조정함으로써 메모리 뱅크 충돌을 최소화하고, GPU의 병렬 처리 능력을 보다 효과적으로 활용하는 방법을 제안한다.

4.2 공유 메모리를 사용한 활성화 블록 알고리즘

HP-체인메일 알고리즘은 '활성화 블록'이라는 기법을 도입함으로써 블록화 구조 기반의 알고리즘의 효율성을 높이고자 하였다. 본 방법은 볼륨 데이터를 블록 단위로 구분하고, 변형이 전파되어 움직임이 필요한 블록들에 한하여 이들을 활성화시켜 동작하도록 설계된 알고리즘 접근법이다. 이전 제3장에서 논의된 작동 메커니즘에 따르면, 이 기법은 볼륨 데이터의 모든 원소가 변형으로 발생한 전파 및 안정화를 수행할 필요를 배제한다. 변형이 도달하지 않은 블록들에 대해서는 불필요한 연산을 방지함으로써, 전체 알고리즘의 수행에 있어서 불필요한 연산을 제거한다. 또한, CPU와 GPU 사이에서 데이터를 이동시키는 과정에는 불가피하게 오버헤드가 발생하는데, 이는 메모리가 물리적으로 분리된 위치에 존재하기 때문이다. 따라서 변형이 감지된 블록들만을 선별하여 GPU의 병렬 처리 능력을 이용해 알고리즘을 수행하는 것이 훨씬 효율적이다.

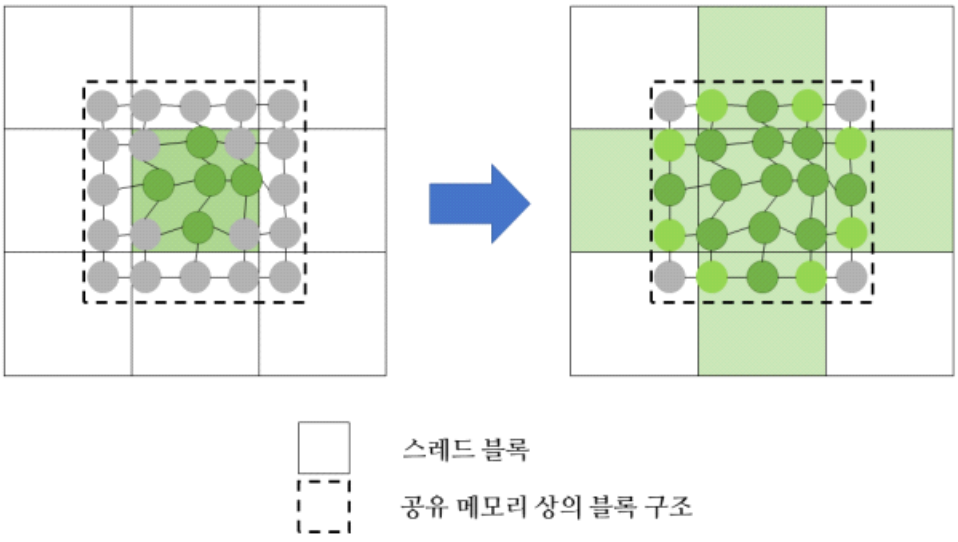
[그림 4-2]는 활성화 블록 메커니즘의 구체적인 동작 방식을 시각적으로 설명하고 있다. 그림에서 초록색으로 표시된 블록들은 활성화된 블록들을 나타낸다. 왼쪽의 그림은 블록화 구조를 채택하고 있지만, 활성화되지 않은 블록들에 의해 전체 볼륨이 불필요하게 커널 동작을 수행하는 상황을 보여준다.

반면에 오른쪽 그림은 활성화된 블록들만이 연산을 수행함으로써 연산적 이점이 있음을 시사한다.



[그림 4-2] 활성화 블록 구조의 동작 개요

비록 활성화 블록 기법이 병렬 프로그래밍의 효율성을 대폭 향상시킬 수 있는 잠재력을 지니고 있음에도 불구하고, 이 방식은 블록 간 통신이라는 중요한 제약 사항을 안고 있다. 체인메일 알고리즘에서는 블록 간 통신이 필수적이며, 이는 한 블록에서 처리된 변형 파동이 인접 블록으로 전달되어야 할 상황에서 특히 중요하다. 만약 이러한 통신 메커니즘이 적절히 구현되지 않는다면, 변형 파동이 이웃 블록 또는 전체 블록으로 효과적으로 전달되지 않을 위험이 있다. 본 연구에서 서술한 바와 같이, GPU의 공유 메모리를 활용한다면 이러한 블록 간 통신의 문제를 해결할 수 있으며, 이는 변형 파동의 전파가 각 이웃 블록에 빠르고 효율적으로 도달하도록 보장하는 방법으로 제시된다.



[그림 4-3] 2차원 체인메일에서 활성화 블록 구조의 통신을 위하여 공유 메모리를 적용한 예시

본 연구에서는 블록 구조의 통신 효율성을 개선하는 방안으로, 각 블록이 인접 블록의 가장자리 영역 원소를 포함하여 데이터를 처리하는 방법을 제안한다. 2차원 모델을 기준으로 할 때, 두 인접 블록이 외곽 원소의 정보를 교환하고자 하면, 데이터 접근에 있어서 순서가 중요한 선후 관계가 형성된다. 이러한 상황에서 동일한 메모리 영역에 다수의 스레드가 접근하려 할 때 발생하는 경쟁 상태(race condition)는 병렬 프로그래밍에서 심각한 문제를 일으킬 수 있다. 이를 해결하기 위해, 각 코드가 해당 메모리 영역에 접근하기 전에 실행 흐름을 제어하는 잠금(lock) 메커니즘이 필요하다. 그러나 이러한 잠금은 교착 상태(deadlock) 또는 기아 상태(starvation)와 같은 동시성 문제를 유발할 위험이 있으므로, 세심한 주의가 요구된다.

CUDA의 공유 메모리를 통해 이러한 문제를 해결할 수 있는데, 공유 메모리는 스레드 간의 공유된 자원 접근을 용이하게 한다. [그림 4-3]은 공유 메모리를 활용하여 블록 간 통신을 진행하고, 인접 영역의 원소들을 효과적으로 처리하는 방법을 설명한다. 여기에서 제시하는 방법은 블록의 크기를 기존보다 확장하여, 한층 더 넓은 가장자리 원소를 포함시킴으로써 블록의 크기를 재설정하는 것이다.

앞서 스레드 블록과 데이터 블록 간의 최적 크기를 8x8x8로 결정함으로써, GPU 커널의 병렬 처리 효율성을 극대화하고자 하였다. 이와 더불어, 변형이 발생한 가장자리 원소들에 대한 추가적인 공간, 즉 패딩을 고려하여 공유 메모리를 사용하는 블록의 크기를 10x10x10으로 확장하는 것을 제안한다. 이러한 확장은 가장자리 원소들에서 발생하는 블록을 넘어가는 변형 파동의 전달을 용이하게 하고, 공유 메모리를 활용함으로써 메모리 접근 횟수를 줄이고 전역 메모리 접근에 따른 지연을 최소화한다.

앞서 탐구한 8x8x8이 블록 크기를 지닌 기존 방법에서는 공유 메모리를 사용하지 않았기 때문에, 활성화된 블록의 경계에 있는 원소들 사이의 정보 전달이 느리고 비효율적이었다. 특히 변형 파동이 블록의 경계를 넘어 인접 블록으로 전파될 때, 전역 메모리를 통한 여러 차례의 추가적인 접근이 요구되며, 이로 인해 발생하는 지연은 전체 알고리즘의 수행 속도를 저하시키는 주요한 요인이다. 반면에, 제안하는 데이터 패딩을 사용한 10x10x10 크기의 블록은 공유 메모리를 활용하여 이러한 경계 원소들 사이의 정보 전달을 신속하게 처리한다. 공유 메모리에 저장된 원소는 타임스탬프를 활용하여 가장 최신의 변형 정보를 신속하게 비교하고, 필요한 경우 전역 메모리에 업데이트를 수행함으로써 변형 정보의 일관성과 신속한 전파를 보장한다.

따라서, 본 연구에서 개발한 방법은 기존 방법에 비해 메모리 접근 횟수를 상당히 줄이며, 활성화 블록의 경계에서 발생하는 정보 전달의 복잡성을 감소시킨다. 이는 전체 알고리즘의 수행 시간을 획기적으로 단축시키는 동시에, 변형체 알고리즘의 실시간 실행 가능성을 한층 더 향상시킨다는 점에서 혁신적이다.

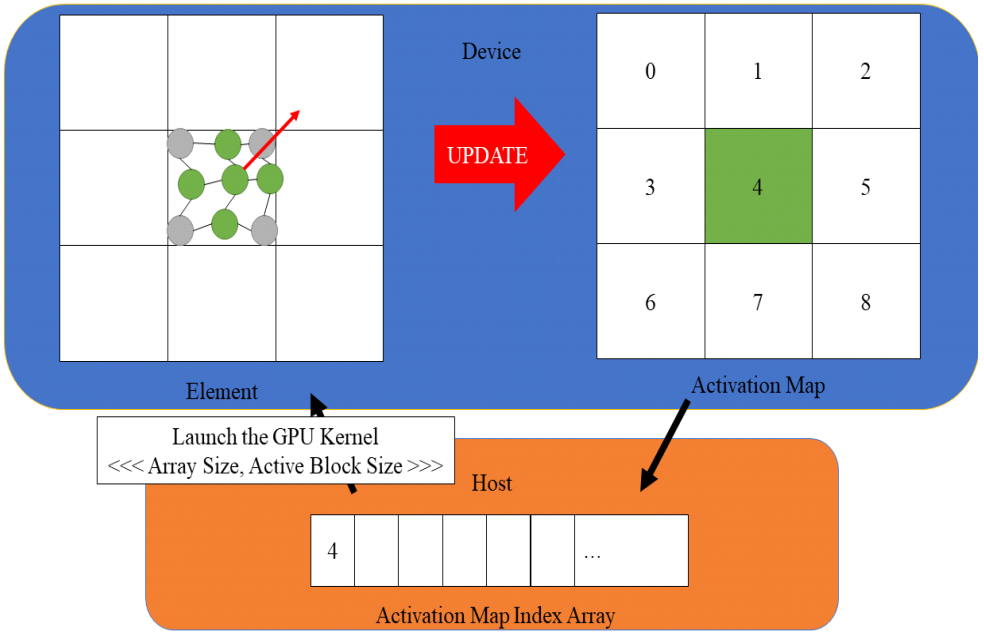
4.3 GPU 최적화된 3D 체인메일 알고리즘의 수행 과정

본 연구에서 제안하는 GPU에 최적화된 3D 체인메일 알고리즘의 구조는 변형 이벤트 발생 시 변형의 근원점이 포함된 활성화 블록의 인덱스를 효과적으로 기록하고 관리하는 '활성화 맵 인덱스 배열'(Activation Map Index

Array)을 중심으로 한다. 이 배열은 가변적인 길이를 가지며, 변형 이벤트 발생 시 그 범위 내에 있는 블록들의 인덱스를 저장함으로써, GPU 커널의 효율적인 런치를 가능하게 하는 핵심적인 정보를 제공한다.

GPU 커널의 런치 과정에서, 가변 배열에 저장된 활성화 블록의 인덱스는 GPU가 개선된 체인메일 알고리즘을 수행해야 할 특정 블록을 식별하는 데 사용된다. 추가적인 변형이 필요한 블록이 감지되면, 해당 블록은 '활성화 맵'(Activation Map)에 플래그되어, 후속 연산에서 고려될 수 있도록 한다. 이는 변형이 필요한 블록을 동적으로 식별하고, 필요한 연산을 수행할 블록들을 효과적으로 재정렬하는 순환적 과정을 포함한다.

이 과정은 변형이 더 이상 필요하지 않게 될 때까지, 즉 가변 배열 내에 인덱스가 더는 존재하지 않을 때까지 반복된다. 알고리즘의 실행이 완료되면, GPU 리소스의 효율적 활용을 통해 전체 수행 속도가 획기적으로 향상된다.



[그림 4-4] 제안하는 활성화 블록 방법을 적용한
개선된 3D 체인메일 알고리즘의 동작 과정

[그림 4-4]는 제안된 알고리즘의 실행 과정을 도식화하여 보여준다. 이

그림은 가변 배열인 '활성화 맵 인덱스 배열'을 통해 알고리즘의 실행이 필요한 블록들만을 선별적으로 GPU 커널에 전달하는 구조를 시각적으로 설명한다. 가변 배열의 크기는 GPU 커널의 그리드 크기를 결정하는 중요한 요소이며, 각 스레드 블록은 할당된 인덱스에 따라 수행할 알고리즘을 결정한다.

이 구조는 효율적인 자원 할당과 관리를 가능하게 하며, 병렬 처리 시스템의 성능 최적화를 위한 새로운 패러다임을 제시한다. GPU 기반의 복잡한 계산 과정에서 연산 부하를 분산시키고, 전체적인 처리 시간을 단축시키는 혁신적인 접근 방식으로 평가될 수 있다.

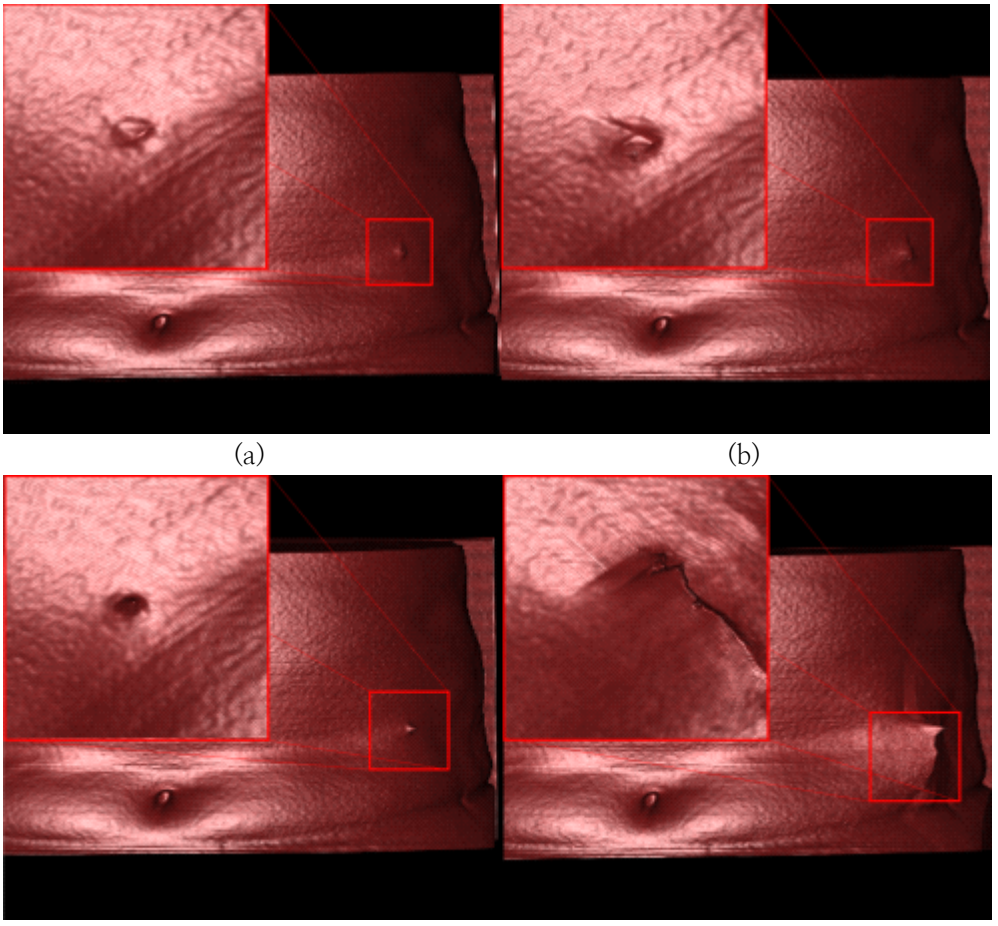
V. 구현 및 실험 결과

5.1 실험 환경

이번 장에서는 구체적인 구현 방법과 실험 결과를 설명한다. 본 실험은 Intel i7-11800H CPU, 16GB RAM, NVIDIA GeForce RTX 3060 Laptop GPU 환경에서 수행되었으며, CUDA Toolkit 11.4로 제작하였다. 실험에 사용된 데이터는 512x512x300 크기의 복부 CT 데이터이다.

5.2 볼륨 데이터 크기 최적화 실험

앞서 제안한 연구의 실험을 진행하기에 앞서 볼륨 데이터의 크기에 따른 변형 적용의 성능과 이미지 결과의 비교를 위해 다양한 크기로 다운샘플링된 데이터를 실험하였다. 원본 데이터를 기준으로 하여, 각각의 볼륨 데이터를 x, y, z축에 대해 1/2, 1/4, 1/8로 축소한 다운샘플링 버전들을 생성하였다. [그림 5-1]에서 (a)는 원본 데이터에 변형을 적용한 결과를, (b)는 데이터 크기를 각 축에 대해 1/2로 줄인 경우의 결과를, (c)는 1/4로 줄인 경우, 그리고 (d)는 1/8로 줄인 경우의 결과를 보여준다. 실험은 모든 볼륨 데이터에 대해 최대 맨해튼 거리를 전파 횟수로, 그리고 전파의 두 배에 해당하는 횟수를 안정화를 적용하여 수행되었다. 예를 들어, 512 x 512 x 300 크기의 데이터의 경우, 전파는 1,324회, 안정화는 2,648회 수행되었다. 결과 이미지는 병렬 리샘플링 알고리즘(Aguilera, A. R., 2015)을 적용하여 렌더링한 것이다.



(a) (b)

(c) (d)

[그림 5-1] 데이터 크기에 따른 변형 결과 및 확대 사진.
(a) 500x500x300, (b) 256x256x150,
(c) 128x128x75, (d) 64x64x37

결과 이미지들의 비교 분석을 통해 [그림 5-1]의 (b)는 원본 데이터의 절반 크기를 사용한 볼륨의 변형 결과로, 원본 데이터인 (a)에 가장 근접한 시각적 결과를 나타내는 것을 확인할 수 있다. (c)의 경우에도 데이터 축소에도 불구하고, 상대적으로 원본 이미지에 유사한 결과를 제공한다. 그러나 (d)의 결과는 원본 데이터와 비교하면 변형이 볼륨의 훨씬 더 넓은 범위에 걸쳐 불안정하게 발생하였음을 보여준다. 이는 주로 변형체의 해상도 감소로 인해 복잡한 조직의 경계에서 높은 오차가 발생했기 때문으로 해석된다.

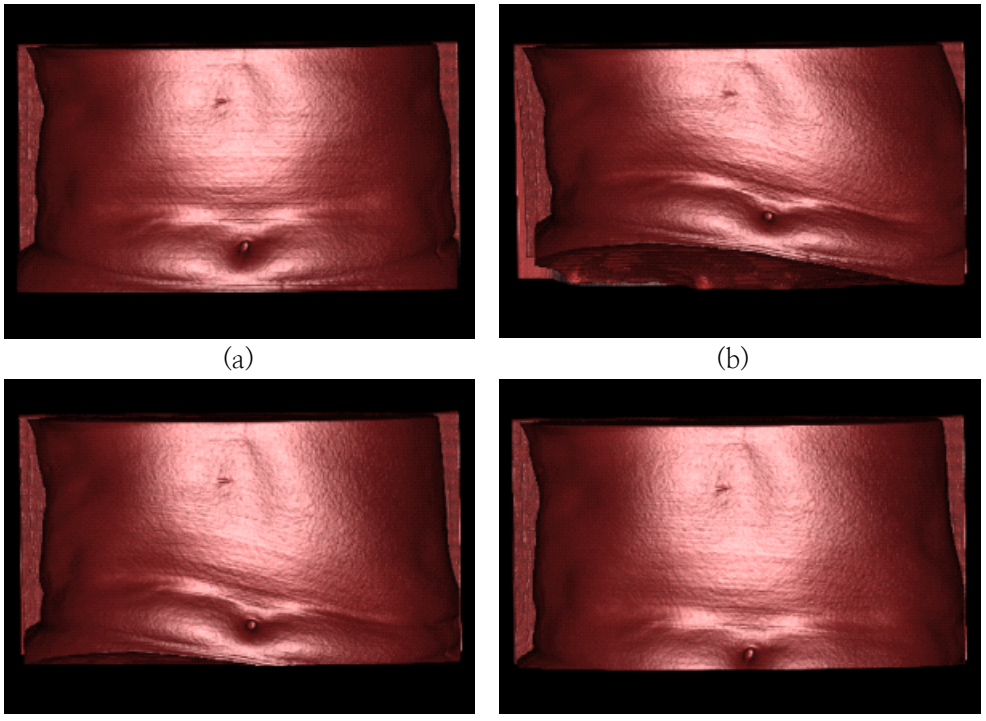
[표 3] 볼륨 데이터 크기와 체인메일 알고리즘 수행 시간 사이의 관계성 실험 결과

Volume Size	(a) 512x512x300	(b) 256x256x150	(c) 128x128x75	(d) 64x64x37
Time(ms)	46885.27	2894.56	196.96	22.61

[표 3]은 다양한 크기의 볼륨 데이터에 반복적인 변형을 적용했을 때 체인메일 알고리즘의 평균 수행 시간을 나타낸다. 가장 빠른 수행 속도를 보인 (d)는 원본 데이터 크기보다 약 2,073배의 성능 향상을 보였다. 그러나 [그림 5-1] (d)에서 관찰된 변형 결과 이미지는, 원본 데이터를 사용한 경우와 비교할 때 뚜렷한 차이와 비정상적인 변화를 보였다. 비록 (d)의 실험에서 알고리즘 수행 시간 측면에서 상당한 효율성을 달성했음에도 불구하고, 이미지 생성 결과는 원본과의 상당한 차이를 나타내어 결과의 정확성 면에서는 손실을 입증하였다.

결론적으로, 본 연구에서 [그림 5-1]과 [표 3]에 제시된 결과를 종합해 볼 때, 원본 이미지 크기의 1/4인 128x128x75 크기로 시뮬레이션을 구성했을 때, 원본 이미지에 근접한 품질을 유지하면서도 상대적으로 빠른 수행 속도를 보였다. 이에 따라, 원본 이미지의 1/4 크기를 사용하는 것이 효과적인 접근 방법으로 권장된다.

5.3 GPU 커널 런치의 비용을 최소화하는 향상된 3D 체인메일 알고리즘의 구현 및 실험 결과



[그림 5-2] 원본 데이터 영상과 변형을 적용된 볼륨 데이터의 실시간 변형 예.
(a) 원본 데이터 영상, (b)~(d) 시간에 따른 볼륨의 변형 모습 영상

[그림 5-2]는 체인메일 알고리즘을 적용하여 변형을 주고 위치 변화가 이루어지는 과정을 담은 순간 영상들을 시각적으로 나타낸다. (a)는 변형을 가하지 않은 원본 데이터의 상태를 보여주며, (b)부터 (d)에 이르기까지는 알고리즘이 적용된 후의 데이터 변형 과정을 순차적으로 보여준다. 특히, 영상의 왼쪽 하단에서 발생한 주요 변형이 원형으로 회복되어가는 부드러운 과정을 감상할 수 있다. 이는 체인메일 알고리즘의 핵심 특성인 연결된 사슬 고리가 자연스럽게 변형을 받고 이에 따라 움직이는 모습을 생동감 있게 표현하고 있다.

본 연구에서는 커널 내 반복적인 변형 적용을 실험하였다. [표 4]는 체인메일 알고리즘이 적용된 데이터의 크기와 변형을 주는 반복 횟수에 따른 평균 수행 시간을 정리한 것이다. 이 실험에서는 GPU의 블록 구조에 기반을 둔 작동 원리는 고려하지 않았으며, 변형은 총 100회에 걸쳐 연속적으로 적

용되었다.

[표 4] 블록 구조가 아닌 GPU 커널에서 데이터 크기와 내부 반복 크기에 따른 변형 1회의 평균 수행 시간

		Data Size	
		128×128×75	64×64×37
Iteration Size	0	3,762.2ms	220.8ms
	2	2,743.9ms	160.5ms
	4	2,300.7ms	135.2ms
	8	2,042.6ms	121.0ms
	16	1,863.0ms	112.5ms
	32	1,800.0ms	107.6ms
	64	1,759.0ms	83.9ms

[표 4]에서는 커널 내부 반복 횟수의 증가에 따른 성능 변화가 명확하게 관찰된다. 여기서 0회 반복은 단일 커널 런치로 단 한 번의 원소 변형이 발생하는 상황을 의미하며, 이는 HP-체인메일 알고리즘의 초기 변형 방식에 해당한다. 그러나 반복 횟수를 2, 4, 8회로 증가시킬 경우, 실행 속도의 점진적 향상을 확인할 수 있다. 특히, 커널 내에서 64회 반복을 수행했을 때, 약 2.13에서 2.63배의 속도 향상이 관찰되었다. 동일한 전파 및 안정화를 적용했음에도 불구하고 성능 차이가 발생하는 것은, 기존 변형 방식에서의 커널 런치 오버헤드가 상당히 크다는 점을 시사한다. 본 연구에서 제안된 방법론은 커널 런치 횟수를 2배에서 최대 64배까지 감소시켜, 오버헤드를 현저히 줄임으로써 전체 실행 시간을 대폭 단축했다.

또한, 적절한 커널 내 반복 횟수의 설정은 알고리즘의 효율성을 극대화하는 데 중요한 요소로 간주된다. [표 4]에 따르면, 커널 내 반복 횟수가 너무 적으면 성능 향상이 미미하고, 반대로 너무 많으면 데이터 변형 과정에서 오

류를 유발할 수 있다. [그림 5-2]에서 제시된 비현실적인 변형 결과는 이러한 오류의 사례를 시각적으로 보여준다. 실험에서는 데이터를 상하좌우로 25회씩 이동시키며 변형을 관찰하였다. 원본 데이터 크기에서 8회 반복 적용한 결과는 안정적인 변형을 보여주었으나, 16회 반복부터는 안정화되지 않은 결과가 나타났다. 이는 과도한 반복으로 인해 변형 전파가 적절히 이루어지지 않은 것으로 해석된다. 따라서 속도 향상을 고려하면서도 안정적인 결과를 얻기 위해 8회의 커널 내 반복을 적용하는 것이 가장 적절하다고 결론지을 수 있다.

추가로, [표 4]에서는 두 가지 크기의 변형체 데이터에 대한 실험을 수행하였다. 64x64x37의 크기는 빠른 변형 속도를 보여주었으나, 세부 정보의 손실로 인해 원본 데이터의 크기에 비해 적절치 않은 것으로 평가되었다. 이상은(2023b)에 따르면, 수행 속도와 시각적 효용성을 모두 고려할 때, 알고리즘 수행에 효과적인 데이터 크기는 128x128x75로 나타났다. 이 크기는 HP-체인메일에서 적용된 전체 볼륨 데이터 실험의 데이터 크기인 126x126x126과 가장 유사하며 적절한 볼륨 데이터 크기로 여겨진다. 반면, 원본 데이터 크기인 512x512x300과 이를 1/2로 축소한 256x256x150은 내부 반복 적용에도 불구하고 긴 수행 시간을 요구했다. 각각 64회 반복 적용 시 450,697.0ms와 28,370.8ms의 수행 시간을 기록하여 일반적인 실시간 시뮬레이션에는 부적합한 것으로 평가되었다.

[표 5] 커널 내부 반복 횟수에 대한 복셀 거리 오차 비교

			Data Scale 1/4	
			position error	error rate(%)
Iteration Size	0	x	0	0
		z	0	0
	2	x	0.000236	0.000737
		z	0.000317	0.000689
	4	x	0.001018	0.003181
		z	0.001583	0.003441
	8	x	0.007061	0.022066
		z	0.009384	0.020400
	16	x	0.008301	0.025940
		z	0.107498	0.233691
	32	x	0.009214	0.028793
		z	4.079495	8.868454
	64	x	4.282505	8.868455
		z	12.709732	27.629813

이에 더하여 물리적 시뮬레이션 프로그램에서의 효율성과 정확성을 평가하기 위해 선행 연구에서 추천한 1/4 스케일로 반복 횟수에서의 오차율을 분석하였다. 오차는 해당 반복 횟수에서 0회 반복의 x, z축 상의 거리 차이이다. 또한, 오차율은 (거리 오차) / (0회 반복의 위치)로 계산한다. 실험은 ±5만큼의 변형을 먼저 x 방향으로 50회, z 방향으로 50회 연속으로 가하였다. 위 표는 반복 횟수 0부터 64까지 변화에 따른 x, y, z 좌표의 변동 및 오차율을 상세히 보여준다. 반복 횟수 0회를 기준으로 할 때, 8회의 반복에서 가장 효율적인 결과를 도출하였음을 관찰할 수 있다. 이는 x, y, z 좌표에 대한 오차율이 상대적으로 낮으면서도, 반복 횟수의 증가함에 따른 계산 비용 대비

효율성이 최적화된 지점을 나타낸다.

x 좌표의 경우, 반복 횟수 64회에서의 오차가 4.282505로 급격히 증가하였으며, 이에 대한 오차율은 13.38%에 이르는 것으로 나타났다. z 좌표 또한 64회의 반복 횟수에서 12.709732의 오차와 27.63%의 오차율을 보이며, 상대적으로 큰 값을 기록하였다. 이러한 결과는 물리적 시뮬레이션 프로그램에서 오차가 적을수록 더 신뢰할 수 있는 결과를 도출한다는 점에서 중요하다. 위 표에서 x 방향의 오차가 더 적은 이유는 z 방향으로 가해진 50회의 변형 과정에서 값의 큰 변화 없이 안정화를 계속 수행할 수 있었기 때문이다.

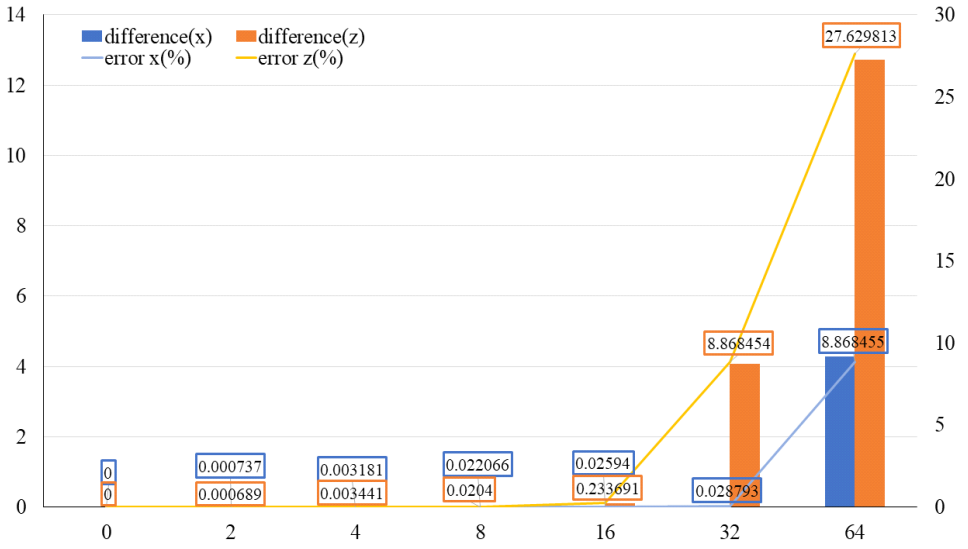
따라서, 본 연구에서는 물리적 시뮬레이션의 정확성과 효율성 사이의 균형을 고려하여, 1회 변형에 GPU 커널 내부 반복 횟수 8회를 가장 이상적인 설정으로 제안한다. 이러한 설정은 낮은 오차율을 유지하면서 계산 비용을 최소화하는 방법을 탐색하는 데 있어서 중요한 기준점을 제공한다.

가장 비교해 볼 수 있는 지점은 8회 반복과 16회 반복 사이이다. 물리적 시뮬레이션의 정밀도 및 효율성 측면에서 8회 반복이 더욱 우수한 선택으로 판단된다. 8회의 경우, x, z 좌표의 오차율이 각각 0.0221%, 0.0204%로 계산되었다. 반면, 16회 반복에서는 x 좌표의 오차율이 0.0259%, z 좌표의 오차율이 0.2337%로 측정되어, 특히 z 좌표에서 상당한 오차율의 증가를 보인다.

물리적 시뮬레이션에서 정확성은 결과의 신뢰도를 결정하는 핵심 요소이다. 과도한 오차는 시뮬레이션의 예측 가능성과 현실적 반응을 모사하는 능력을 저하시킨다. 16회 반복에서 관찰된 z 좌표의 오차율은 시뮬레이션의 신뢰성을 손상시킬 수 있는 수준이다. 또한, 오차율의 증가는 연산 과정에서 불안정성이 증가했음을 암시하며, 이는 시뮬레이션의 예측 불가능성으로 이어질 수 있다.

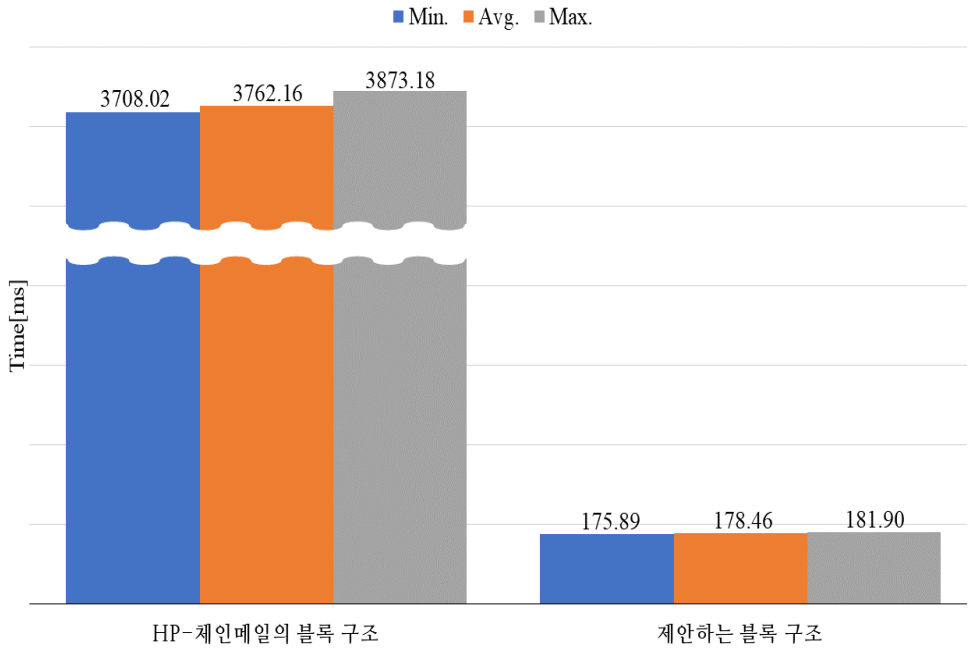
반면, 8회 반복에서는 상대적으로 낮은 오차율을 유지하며, 이는 물리적 현상을 보다 더욱 모사할 수 있음을 의미한다. 물리적 시뮬레이션의 목적은 현실 세계의 동작을 정확하고 일관되게 재현하는 것이기 때문에, 커널 내부 반복 8회가 더 적합한 선택이라 할 수 있다. 아래의 [그림 5-3]은 [표 5]의 결과를 시각적으로 표현한다. 이러한 결과는 시뮬레이션의 정확도를 최대화하

면서 계산 비용을 효율적으로 관리할 수 있는 방법론을 모색하는 데 중요한 기준을 제시한다.



[그림 5-3] 커널 내부 반복 횟수에 대한 복셀 거리 오차 비교

두 번째 실험에서는 HP-체인메일의 블록 구조와 GPU 구조에 최적화된 블록 구조를 적용한 알고리즘의 수행 시간을 비교 분석하였다. 이 실험에서 사용된 데이터 크기는 모두 128x128x75이며, 블록의 크기는 8x8x8로 설정되었다. 변형은 100회 연속 적용되었으며, 실험 결과의 영상은 안정화가 모두 완료된 후의 상태를 반영하였다.



[그림 5-4] HP-체인메일의 블록 구조와 제안하는 블록 구조 간의 알고리즘 수행 시간 비교

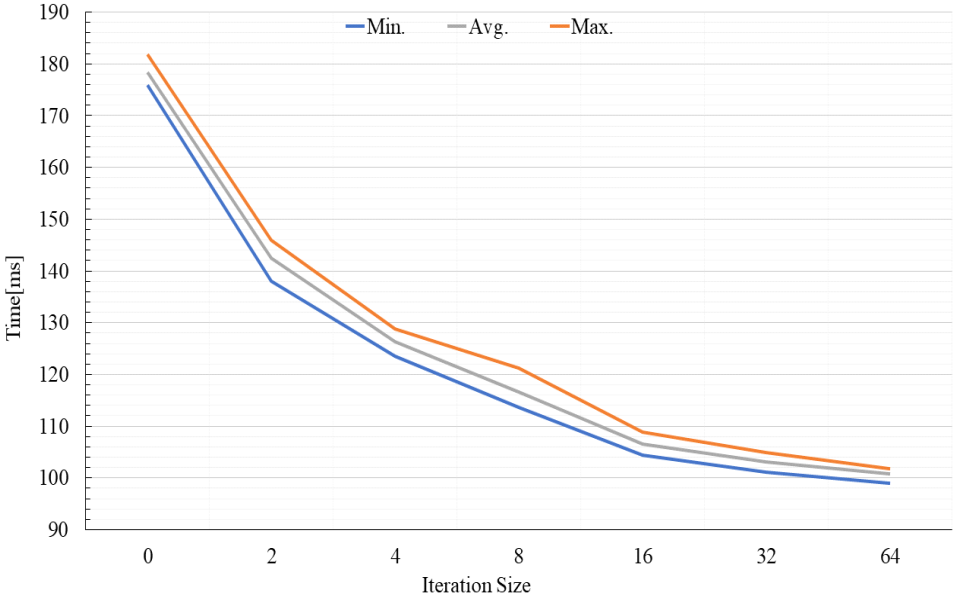
[그림 5-4]에 제시된 데이터에 따르면, HP-체인메일 알고리즘을 전통적인 블록 구조로 구현했을 경우 평균 처리 시간이 3762.16ms로 측정되었다. 반면, 본 논문에서 제안된 블록 구조는 GPU 구조에 보다 최적화되어 있으며, 이에 따라 HP-체인메일의 방식과 비교하면 약 21.1배 빠른 실행 속도를 나타내었다. 이는 커널의 그리드에 데이터 블록을 할당하는 대신 커널의 블록에 직접 대응시키는 단순한 변화만으로도 상당한 성능 향상을 이룰 수 있음을 시사한다.

추가로, 커널 내 반복문의 도입과 결합된 제안된 블록 구조에 대한 성능을 평가하였다. 실험 조건은 이전 실험과 동일하게 데이터 크기 128x128x75, 블록 크기 8x8x8로 설정하고, 100회의 변형을 적용하였다. [표 6]에 실험 결과를 요약하고, [그림 5-4]에서는 해당 결과를 그래프로 시각화하여 보여준다.

[표 6] 제안하는 블록 구조와 GPU 커널에서의 내부 반복을 적용한 실험의 반복 횟수 - 수행 시간 표

		Loop Size						
		0	2	4	8	16	32	64
Time (ms)	Min.	175.89	138.10	123.57	113.59	104.42	101.14	98.93
	Avg.	178.46	142.48	126.25	116.67	106.54	103.05	100.77
	Max.	181.90	145.97	128.79	121.29	108.83	104.99	101.85

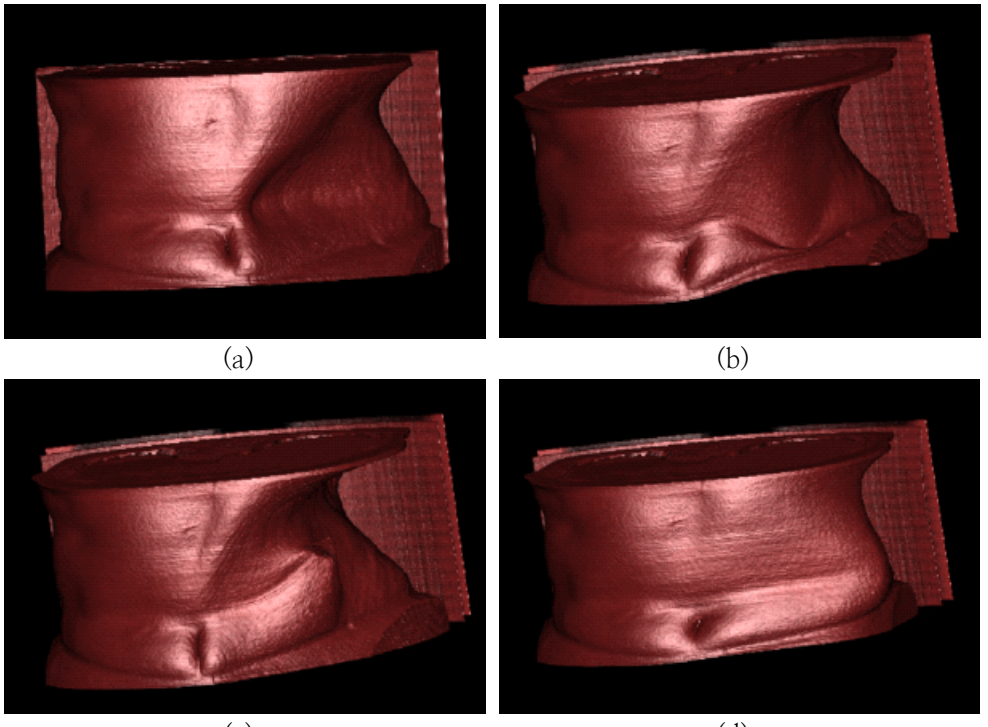
본 실험에서 GPU 커널 내의 8회 반복은 반복이 전혀 없는 경우에 비해 약 1.5배의 속도 향상을 달성했다. HP-체인메일의 블록 구조를 사용하고 커널 내 반복이 없는 상황과 비교했을 때는 3,645.49ms의 시간 단축을 이루어, 약 32.2배의 속도 향상을 나타냈다. 볼륨 데이터의 변형 처리에 100ms대의 시간만 소요되는 것은 실시간 대화형 프로그램 개발에 매우 적합한 성능이다. 마지막으로 이루어진 실험 결과는 본 연구에서 제안된 최적화 방안이 실질적이고 유효함을 입증한다. 또한, [그림 5-5]의 그래프 분석에 따르면, 16회 이상의 반복 적용 시 속도 향상에 따른 수행 시간 감소의 폭이 줄어들어, 반복 횟수 증가에 따른 이득이 점차 감소하는 경향을 보인다.



[그림 5-5] 제안하는 블록 구조와 GPU 커널에서의 내부 반복을 적용한 실험의 반복 횟수 - 수행 시간 그래프

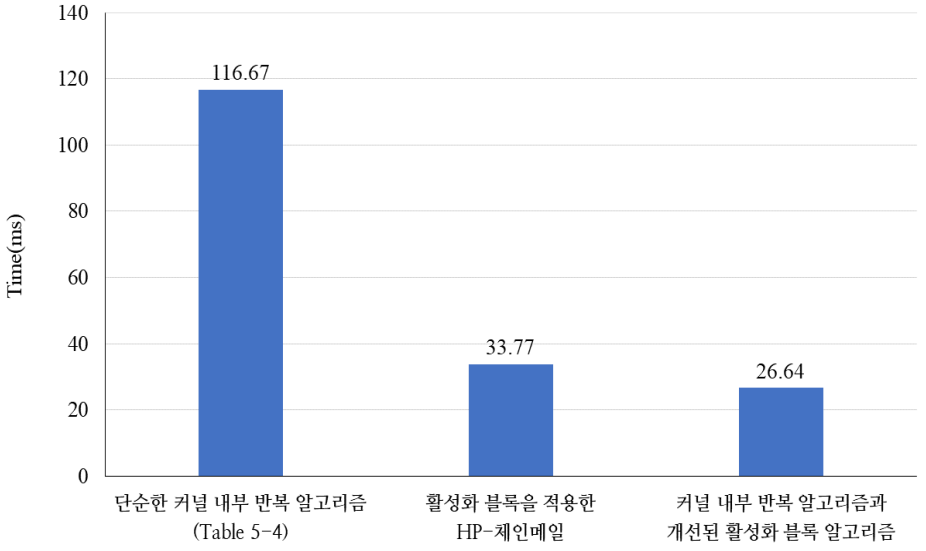
5.4 개선된 활성화 블록 알고리즘 구현 및 실험 결과

본 실험은 앞선 연구 결과들을 종합하여 내부 반복 횟수를 최적화된 8회로 설정하고 진행되었다. [그림 5-6]은 실험을 통해 생성된 영상을 통해 볼륨 데이터에 적용된 변형이 부드럽게 처리되는 과정을 시각적으로 보여준다. [그림 5-6]의 (a)에서 (d)까지의 흐름은 체인메일 알고리즘을 다양한 변형 방향으로 연속적으로 적용한 결과를 나타낸다. 변형은 단순한 수평적 움직임뿐만 아니라 수직 방향으로도 효과적으로 적용되었으며, 특히 (b)와 (c)의 영상에서는 변형의 방향을 급격하게 바꾸어도 변형이 자연스럽게 이루어지는 것을 확인할 수 있다. 이는 알고리즘이 물리적으로 타당하고 유연하게 작동함을 시사하며, 이러한 결과는 체인메일 알고리즘의 강건함(robustness)을 잘 보여주는 예시로 해석될 수 있다.



[그림 5-6] 변형이 적용된 볼륨 데이터의 실시간 변형 예.
(a)~(d) 시간에 따른 볼륨의 변형 모습 영상

공유 메모리를 활용하는 개선된 활성화 블록 알고리즘은 실행 속도를 현저히 개선한다는 논리적 결과를 도출해낸다. 이전 연구에서는 GPU의 최적화 기법인 공유 메모리의 활용이 충분히 이루어지지 않아 알고리즘의 수행 속도는 향상되었으나 실시간 조작에 있어서 원활한 영상을 제공하지 못하는 한계를 드러냈다. 그러나 이 실험에서는 이러한 제약을 극복하고 더 빠른 실행 속도를 실현함으로써, 개선된 활성화 블록 알고리즘의 우수성을 입증한다. [그림 5-7]에 나타난 막대 그래프와 그에 해당하는 수치적 결과들을 비교 분석함으로써, 알고리즘의 효율성 향상을 명확히 확인할 수 있다.



[그림 5-7] 기존 알고리즘과 개선된 활성화 알고리즘의 실행 속도 비교

[그림 5-7]에서 좌측 막대 그래프는 [표 5-4]의 결과를 반영한 것으로, 중앙에 있는 그래프는 활성화 블록을 적용한 HP-체인메일 알고리즘의 성능을 나타낸다. HP-체인메일 논문에서 제시된 블록 최소 크기는 16x16x16이지만, 이 크기는 CUDA API에서 정의한 스레드 블록의 최대 크기 1,024를 초과하므로, 3장에서 제안된 하나의 스레드와 하나의 원소의 일대일 대응 규칙에 부합하지 않는다. 실험의 일관성을 위해, 스레드 블록과 활성화 블록이 일대일로 대응되지 않음에도 불구하고, 동기화 문제가 존재할 수 있지만, HP-체인메일의 최소 블록 크기를 그리드 크기에 맞추어 구현하였다. 이 조건에서 HP-체인메일 알고리즘의 수행 시간은 약 33.77ms로 측정되었으며, 이는 빠른 처리 속도를 나타낸다. 하지만 공유 메모리를 활용하지 못하며, 블록 내 원소들의 동기화가 일치하지 않는다는 문제점을 가진다.

본 논문에서 제안된 개선된 활성화 블록 알고리즘은 공유 메모리의 사용을 통해 GPU의 성능을 최대화하고자 하였으며, 스레드 블록 크기를 고려하여 주장한 8x8x8의 블록 크기로 블록 내부의 원소들의 동기화 문제를 해결하였다. 이에 따른 성능 결과는 우측 그래프에서 확인할 수 있으며, 26.64ms의 더 짧은 수행 시간을 기록하였다. 이러한 실험 결과는 실시간 변형이 적용

될 때 이전 연구 대비 부드럽고 현실적인 변형이 신속하게 이루어질 수 있음을 입증한다. 따라서 GPU 최적화를 적용한 공유 메모리 활용의 활성화 블록 알고리즘을 통해 HP-체인메일이 효과적으로 개선되었음을 확인할 수 있다.

VI. 결론

3D 체인메일 알고리즘은 변형을 독특한 방식으로 처리하는 메커니즘을 가진 혁신적인 변형체 알고리즘이다. 이 알고리즘은 GPU의 병렬 처리 능력을 활용하여 변형이 실시간으로 이루어지는 단계까지 발전하였다. 최근 연구 중 하나인 HP-체인메일은 GPU를 활용한 3D 체인메일의 병렬화 접근법을 도입하였으나, 변형을 전체 볼륨 데이터가 아닌 특정 위치에서 발생한 비율의 데이터에만 적용하는 방식으로 한계를 드러냈다. 이에 본 연구에서는 볼륨 데이터 전체에 변형이 신속하게 전파되고 현실적인 결과를 도출할 수 있는 데이터 크기를 탐색하는 연구를 수행하였다. 더불어 HP-체인메일은 GPU의 공유 메모리를 비롯한 여러 최적화된 하드웨어 메모리 자원을 활용하지 못하는 구조로 구현되었으며, 이는 GPU의 병렬 프로그래밍에서 중대한 결점으로 작용한다.

본 연구에서는 커널 내 반복 사용을 통해 변형 전파에 필요한 반복적인 커널 런치의 비용을 대폭 절감하였다. 또한, HP-체인메일에서 제안된 활성화 블록 방식이 GPU 최적화에 적합하지 않다고 판단하고, GPU 하드웨어의 특성을 극대화할 수 있는 새로운 블록 구조를 제안하였다. 공유 메모리의 적극적인 사용을 통해 구현된 개선된 활성화 블록 알고리즘은 처리 속도의 향상은 물론, 현실성 있는 변형 결과를 성공적으로 제시하였다. 이 연구 결과는 변형이 완료된 후에도 지속적인 안정화가 이루어질 수 있을 정도로 알고리즘의 수행 속도가 개선되었음을 보여준다.

의료 시뮬레이션의 발전을 위한 현재의 요구 사항을 고려할 때, 본 논문에서 제시하는 3D 체인메일 알고리즘의 개선은 주목할 만하다. 더 나아가, 사용자 친화적인 UI 개발과 같은 추가 연구가 수행된다면, 이 알고리즘은 실제 의료 시뮬레이션 프로그램에도 효과적으로 적용될 수 있는 잠재력이 있다.

참 고 문 헌

1. 국내 문헌

이상은, 계희원. (2023a). 3D 체인메일을 이용한 변형체의 GPU 최적화 연구. 『멀티미디어학회논문지』, 26(2), 131-139.

이상은, 계희원. (2023b). *3D 체인메일을 이용한 변형체의 가시화 효율성 연구*. 『한국HCI학회 학술대회』, 921-923.

이세인, 계희원. (2019). CPU 기반의 볼륨 변형을 위한 다형질 Chainmail 모델. 『멀티미디어학회논문지』, 22(7), 759-769.

2. 국외 문헌

Aguilera, A. R., Salas, A. L., Perandrés, D. M. and Otaduy, M. A. A. (2015). A parallel resampling method for interactive deformation of volumetric models. *Computers & Graphics*, 53, pp. 147-155.

Castro-Pareja, C. R., Daly, B. and Shekhar, R. (2006). Elastic registration using 3D chainmail: Application to virtual colonoscopy. In *Medical Imaging 2006: Image Processing,* 6144, pp. 947-955. SPIE.

Essa, I. A., Sclaroff, S. E. and Pentland, A. (1992). *Physically-based modeling for graphics and vision*. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology.

Fortmeier, D., Mastmeyer, A. and Handels, H. (2013). Image-based Palpation Simulation with Soft Tissue Deformations using Chainmail on the GPU. *Bildverarbeitung für die Medizin 2013: Algorithmen-Systeme-Anwendungen*, *Proceedings des Workshops vom 3. bis 5. März 2013 in Heidelberg*, Springer, Berlin, Heidelberg, pp. 140-145.

Frisken-Gibson, S.F. (1999). Using Linked Volumes to Model Object Collisions, Deformation, Cutting, Carving, and Joining. *IEEE Transactions on Visualization and Computer Graphics*, 5(4), pp. 333-348.

Gibson, S. F. (1997). 3D chainmail: a fast algorithm for deforming volumetric objects. *Proceedings of the 1997 symposium on Interactive 3D graphics*, pp. 149-ff.

Le Fol, T., Acosta-Tamayo, O. and Haigron, P. (2007). Angioplasty simulation using ChainMail method. In *Medical Imaging 2007: Visualization and Image-Guided Procedures*, 6509, pp. 1007-1014. SPIE.

Levoy, M. (1988). Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(3), pp. 29-37.

Li, Y. and Brodlie, K. (2003). Soft Object Modelling with Generalised ChainMail—Extending the Boundaries of Web-based Graphics. *Computer Graphics Forum*, 22(4), pp. 717-727. Oxford, UK: Blackwell Publishing, Inc.

Mensmann, J., Ropinski, T. and Hinrichs, K. (2008). Interactive Cutting Operations for Generating Anatomical Illustrations from Volumetric Data Sets. *The Journal of WSCG*, 16(2), pp. 89-96.

Metaxas, D. and Terzopoulos, D. (1992). Dynamic Deformation of Solid Primitives with Constraints. *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pp. 309-312.

Neubauer, D. I. A. (2005). *A ChainMail algorithm for direct volume deformation in virtual endoscopy applications*. PhD thesis.

Rodríguez, A., León, A., Arroyo, G. and Mantas, J. M. (2015). SP-ChainMail: a GPU-based sparse parallel ChainMail algorithm for deforming medical volumes. *The Journal of Supercomputing*, 71(9), pp. 3482-3499.

Rodríguez, A., León, A. and Arroyo, G. (2016). Parallel deformation of heterogeneous ChainMail models: Application to interactive deformation of large medical volumes. *Computers in Biology and Medicine*, 79, pp. 222-232.

Rössler, F., Wolff, T. and Ertl T. (2008). Direct GPU-based Volume Deformation. *Proceedings of the CURAC*, Leipzig, Germany, 24-26 September 2008, pp. 65-68.

Schill, M. A., Gibson, S. F., Bender, H. J. and Männer, R. (1998). Biomechanical simulation of the vitreous humor in the eye using an enhanced chainmail algorithm. *Medical Image Computing and*

Computer-Assisted Intervention—MICCAI'98: First International Conference Cambridge, MA, USA, October 11–13, 1998 Proceedings 1, pp. 679–687. Springer Berlin Heidelberg.

Schill, M. A., Wagner, C., Hennen, M., Bender, H. J. and Männer, R. (1999). Eyesi–a simulator for intra-ocular surgery. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI'99: Second International Conference, Cambridge, UK, September 19-22, 1999. Proceedings 2.* pp. 1166–1174. Springer Berlin Heidelberg.

Schulze, F., Bühler, K. and Hadwiger, M. (2007). Interactive deformation and visualization of large volume datasets. In *International Conference on Computer Graphics Theory and Applications,* 2, pp. 39–46. SCITEPRESS.

Tagliabue, E. (2022). *Patient-specific simulation for autonomous surgery.* PhD Thesis, UNIVERSITÀ DEGLI STUDI DI VERONA.

Xavier, P. (1995). Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour, *Proceeding of Graphics Interface*, Canadian Information Processing Society, pp. 147–154.

Wang, X. and Fenster, A. (2004). A haptic-enhanced 3D real-time interactive needle insertion simulation for prostate brachytherapy. In *Medical Imaging 2004: Visualization, Image-Guided Procedures, and Display,* 5367, pp. 781–789. SPIE.

Wolters, C.H., Kuhn, M., Anwander, A. and Reitzinger, S. (2002). A Parallel Algebraic Multigrid Solver for Finite Element Method Based Source Localization in the Human Brain. *Computing and Visualization in Science*, 5(3), pp. 165–177.

Zhang, J. (2017). *Real-time simulation of soft tissue deformation for surgical simulation*. PhD Thesis. RMIT University.

Zhang, J., Zhong, Y., Smith, J. and Gu, C. (2016). A new ChainMail

approach for real-time soft tissue simulation. *Bioengineered*, 7(4), pp. 246-252.

Zhang, J., Zhong, Y., Smith, J. and Gu, C. (2017). ChainMail based neural dynamics modeling of soft tissue deformation for surgical simulation. *Technology and Health Care*, 25(S1), pp. 231-239.

Zhang, J., Zhong, Y. and Gu, C. (2018a). Deformable Models for Surgical Simulation: a Survey. *IEEE reviews in biomedical engineering*, 11, pp. 143-164.

Zhang, J., Zhong, Y. and Gu, C. (2018b). Ellipsoid bounding region-based ChainMail algorithm for soft tissue deformation in surgical simulation. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 12, pp. 903-918.

Zhang, L., Wahib, M. and Matsuoka, S. (2019). Understanding the Overheads of Launching CUDA Kernels. *ICPP19*, pp. 5-8.

3. 기타 자료

CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/ (accessed October 6, 2023)

Jez. (2014). How Bad is It to Launch Many Small Kernels in CUDA?. https://stackoverflow.com/a/27038751 (accessed October 6, 2023)

riverrun17. (2015). Cuda 메모리 계층 구조. https://blog.naver.com/riverrun17/220420579990 (accessed December 10, 2023)

ABSTRACT

A Study on the Optimization of Parallel 3D Chainmail Algorithm

Lee, Sangeun

Major in Computer Engineering

Dept. of Computer Engineering

The Graduate School

Hansung University

With the advancement of simulation technology and the growing reality of an aging population, the importance of medical imaging data for surgical preparation has never been more pronounced. The enhancement of specific anatomical region details through volume visualization techniques is a critical step in this process. Among the various algorithmic advancements, the 3D Chainmail algorithm has emerged as a notable technique due to its distinctive emulation of chain armor dynamics and its ongoing evolution. The refined variant, known as Heterogeneous Parallel (HP)-Chainmail, leverages GPU parallelization to potentiate its application.

This paper delves into optimizing volume data resolution in the context of contemporary GPU hardware, which has significantly outpaced the capabilities of hardware from the era of initial studies. We conduct a

thorough analysis of the GPU kernel launch overhead and introduce a 'kernel internal loop' as a strategic solution to minimize such costs. We further scrutinize the block structure challenges inherent in the HP-Chainmail approach and recommend constraining the block size to surmount these challenges. By tailoring the block size for GPU optimization, we fully harness the capabilities of shared memory, an intrinsic feature of GPU architecture. The methods proposed herein culminate in the enhancement of GPU-optimized 3D Chainmail, thereby facilitating swifter and more lifelike medical simulations.

【Keyword】Medical Simulation, 3D Chainmail Algorithm, Deformation Algorithm, Optimization, GPU Parallel Computing