석사학위논문

고화질 볼륨 가시화의 가속화를 위한 GPU 스레드 동렬 투사 기법

2024년

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

김 나 연

석 사 학 위 논 문
지도교수 계희원

고화질 볼륨 가시화의 가속화를 위한 GPU 스레드 동렬 투사 기법

GPU Thread Alignment Projection Techniques for Accelerating High-Definition Volume Visualization

2024년 6월 일

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

김 나 연

석 사 학 위 논 문
지도교수 계희원

고화질 볼륨 가시화의 가속화를 위한 GPU 스레드 동렬 투사 기법

GPU Thread Alignment Projection Techniques for Accelerating High-Definition Volume Visualization

위 논문을 공학 석사학위 논문으로 제출함

2024년 6월 일

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

김 나 연

김나연의 공학 석사학위 논문을 인준함

2023년 6월 일

심사위원장 김진모 (인)

심 사 위 원 계희원 (인)

심 사 위 원 허준영 (인)

국 문 초 록

고화질 볼륨 가시화의 가속화를 위한 GPU 스레드 동렬 투사 기법

한 성 대 학 교 대 학 원
컴 퓨 터 공 학 과
컴 퓨 터 공 학 전 공
김 나 연

의료 영상 분석에서의 볼륨 렌더링은 미세한 혈관, 작은 종양과 같은 세부 구조를 관찰하기 위해 고화질로 수행되어야 한다. 이를 위한 주요 기법으로는 고차 보간을 통한 보간법 개선이 있다. 그러나 고차 보간법이 적용된 볼륨 렌더링은 메모리에 접근하는 횟수가 많아 대화적 속도로 수행하기에는 성능적 한계가 있다. 이를 극복하기 위해 GPU를 활용하는 등 다양한 최적화 방법이 연구되었으나, 비효율적인 메모리 접근으로 인한 성능 저하 문제는 여전히 존재한다. 이러한 문제는 메모리 접근 횟수가 많은 고차 보간 볼륨 렌더링에서 더욱 두드러진다.

본 논문에서는 비효율적인 메모리 접근 문제를 해결하기 위해 스레드 동렬 투사 기법을 제안한다. 스레드 동렬 투사 기법은 볼륨 데이터를 정렬하고 스레드와 데이터를 순차적으로 대응함으로써 볼륨 렌더링의 성능을 향상시킨다. 본 논문이 제시하는 기법은 모든 관찰 방향을 고려하여 세 개의 정렬된 볼륨 데이터를 사전에 생성하여, 주된 관찰 방향에 따라 적합한 데이터를 참

조하도록 유도한다. 또한 관찰 위치와 방향을 유지한 채로 관찰자 좌표계를 회전 변환함으로써 스레드와 데이터를 순차적으로 대응시켜, 화면의 왜곡이나 화질 저하 없이 가속화를 이루어 낸다. 결과적으로 3차 B-Spline 보간이 적용된 볼륨 렌더링이 제안 기법을 통해 대화적 속도로 수행할 수 있음을 보인다. 이는 의료 영상 분야에서 더욱 정확하고 빠른 진단을 가능케 하여, 볼륨 렌더링 기술의 실용성과 효용성을 증가시킨다.

【주요어】 볼륨 렌더링, 고화질 렌더링, 3차 B-Spline 보간, GPU 메모리 액세스 패턴, 관찰자 좌표계

목 차

제 1 장 서 론 ····· 1

제 1 절 연구 배경 및 목적 ····· 1
제 2 절 논문의 구성 ····· 3

제 2 장 관련 연구 ····· 4

제 1 절 볼륨 렌더링 ····· 4
제 2 절 B-Spline 근사 ····· 6
제 3 절 B-Spline 보간 ····· 9
제 4 절 CUDA ····· 11

제 3 장 CUDA 메모리 액세스 패턴 최적화 ····· 16

제 1 절 Coalesced Access ····· 16

제 4 장 스레드 동렬 투사 기법 ····· 22

제 1 절 개요 ····· 22
1) 전처리 단계 ····· 22
2) 볼륨 렌더링 단계 ····· 22
3) 후처리 단계 ····· 22
제 2 절 전처리 단계 ····· 24
제 3 절 관찰자 좌표계 재구성 ····· 26
1) 볼륨-광선 교차를 고려하지 않은 방법 ····· 27
2) 볼륨-광선 교차를 고려한 방법 ····· 29

제 5 장 구현 및 실험 결과 ····· 31

제 1 절 실험 환경 ····· 31

제 2 절 실험 결과 ····· 31

제 6 장 결 론 ····· 37

참 고 문 헌 ····· 39

ABSTRACT ····· 43

표 목 차

[표 1] 실험 수행 환경 ·····B
[표 2] CT 데이터 세트에 대한 렌더링 속도 최소값, 평균값, 최대값 ·····2 3
[표 3] 머리 CT 데이터에 대한 렌더링 속도 측정 결과 ·····53
[표 4] 복부 CT 데이터에 대한 렌더링 속도 측정 결과 ·····53
[표 5] 허리 CT 데이터 1에 대한 렌더링 속도 측정 결과 ·····63
[표 6] 허리 CT 데이터 2에 대한 렌더링 속도 측정 결과 ·····63

그 림 목 차

[그림 1] 볼륨 렌더링의 광선 투사법 ····· 5
[그림 2] 3차 B-Spline 근사 곡선 ····· 7
[그림 3] 3차 B-Spline 보간 곡선 ····· 9
[그림 4] 3차원에서의 선형 보간과 3차 보간 ····· 11
[그림 5] CPU와 GPU 구조 차이 ····· 12
[그림 6] CUDA의 메모리 구조 ····· 13
[그림 7] CUDA의 구조 ····· 14
[그림 8] coalesced access와 non-coalesced access ····· 17
[그림 9] 화면의 x축과 볼륨의 x축이 평행한 경우 ····· 19
[그림 10] 화면의 x축과 볼륨의 x축이 평행하지 않은 경우 ····· 20
[그림 11] 화면의 x축과 투영된 볼륨의 x축이 평행한 경우 ····· 21
[그림 12] 스레드 동렬 투사 기법 알고리즘 순서도 ····· 23
[그림 13] 스레드 동렬 투사 기법 적용을 위한 볼륨 데이터 정렬 ····· 25
[그림 14] 볼륨-광선 교차로 인한 non-coalesced access ····· 28
[그림 15] 볼륨-광선 교차로 인한 coalesced access ····· 29
[그림 16] 머리 CT 데이터에 대한 렌더링 결과 이미지 비교 ····· 33
[그림 17] 복부 CT 데이터에 대한 렌더링 결과 이미지 비교 ····· 33
[그림 18] 허리 CT 데이터 1에 대한 렌더링 결과 이미지 비교 ····· 43
[그림 19] 허리 CT 데이터 2에 대한 렌더링 결과 이미지 비교 ····· 43

제 1 장 서론

제 1 절 연구 배경 및 목적

볼륨 렌더링은 볼륨 데이터라고 불리는 3차원의 스칼라 데이터를 가시화하는 기술이다. 볼륨 렌더링은 의료 영상, 과학 시뮬레이션 등 다양한 분야에서 활용하며, 그 중 의료 영상 분야에서는 CT 혹은 MRI 촬영을 통해 얻은 수많은 단면 영상을 일일이 분석하기엔 어려움이 따르므로 볼륨 렌더링을 통해 3차원 영상으로 가시화하는 방법이 주로 사용된다. 의료진은 가시화된 의료 영상을 통해 환자의 신체 내부 조직이나 혈관을 관찰하여 질병을 발견하고 치료 계획을 수립한다. 이 과정에서 의료진은 미세한 혈관을 관찰하거나, 작은 종양을 정확히 포착해야 한다. 이러한 요구를 충족하기 위해서는 고화질 볼륨 렌더링 기술이 필요하다. 또한 의료진은 진단 과정에서 환자의 병변 부위를 확대하거나 다양한 각도로 관찰하며 분석을 수행하는데, 이를 위해서는 볼륨 렌더링을 대화적 속도로 수행해야 한다. 이는 의료진의 진단 및 치료 계획 수립 과정에서 필수적인 요구사항이다.

볼륨 렌더링은 화면을 구성하는 화소의 값을 계산하기 위해 가상의 광선을 투사해 일정한 간격으로 볼륨 데이터의 값을 조회하는데, 이를 샘플링 과정이라고 한다. 샘플링 과정에서는 배열 요소의 중앙에 위치한 지점만이 아닌 다른 위치의 값 또한 조회하기 때문에, 원하는 지점의 값을 얻어내기 위한 보간 과정이 필요하다. 일반적인 보간의 방법으로는 선형 보간을 각 축의 방향으로 수행하는 삼선형 보간이 있다(Hadwiger, M, 2008). 선형 보간은 값을 구하고자 하는 지점과 가깝게 위치한 두 값의 평균을 구하는 방식으로, 삼선형 보간은 이를 x축, y축, z축 방향으로 3번 수행해 근사값을 얻어낸다. 삼선형 보간을 수행하면 간단한 구현만으로 양질의 결과 영상을 얻을 수 있어 실시간 볼륨 렌더링에서 널리 사용된다. 그러나 삼선형 보간이 적용된 결과 영

상은 고화질 의료 영상 분야에서 요구하는 수준의 화질에는 미치지 못하므로, 이러한 문제에 대응하여 고차 보간을 사용하는 사례가 늘고 있다. 고차 보간은 선형 보간보다 더 많은 주변의 데이터를 참조하여 곡선을 생성해 값을 추정한다. 고차 보간은 원본 데이터의 특성을 보다 정확하게 반영하여 값을 근사하기 때문에 더 높은 품질의 결과 영상을 얻을 수 있다. 그러나 고차 보간은 곡선을 생성하기 위해 다수의 제어점이 필요하므로, 제어점으로 사용될 볼륨 데이터의 값을 참조하는 과정에서 렌더링 속도가 크게 떨어진다. 특히 3차원 공간에서의 3차 보간은 64개의 제어점을 사용해야 하므로, 8개의 제어점이 필요한 선형 보간에 비해 메모리 접근 횟수가 크게 증가하여 성능 문제가 심화된다. 이를 해결하기 위해 GPU를 활용한 병렬 연산을 비롯한 다양한 가속화 방법들이 연구되고 있으나, 비효율적인 GPU 메모리 접근 패턴에 따른 성능 제한 문제가 여전히 존재한다.

광선 투사 기법을 사용하는 볼륨 렌더링은 화면의 각 화소에서 광선을 투영해 볼륨 데이터를 참조하고 근사하는 계산을 통해 화소의 값을 결정한다. 광선이 볼륨 데이터를 참조하는 과정에서 연속된 스레드들이 물리적으로 인접한 메모리 영역을 참조하는 규칙성을 지니면, 스레드가 메모리를 접근하는 횟수가 감소하여 볼륨 렌더링의 성능이 최적화된다. 이러한 현상은 기존 볼륨 렌더링을 특정한 관찰 방향으로 수행했을 때 일어난다.

이에 착안하여 본 논문은 광선의 샘플링 과정에서 스레드의 효율적인 GPU 메모리 참조 패턴을 유지하는 기법인 스레드 동렬 투사 기법을 제안한다. 스레드들이 연속적으로 데이터를 접근하기 위해서는 데이터의 인덱스 좌표 축이 스레드의 인덱스 좌표 축과 평행하게 정렬되어야 한다. 본 논문은 스레드 인덱스의 x축을 기준으로 데이터 인덱스의 x축을 정렬하여 기법을 적용한다. 일반적으로 스레드 인덱스의 x축은 화면의 x축에 대응되므로, 데이터 인덱스의 x축을 화면의 x축과 정렬시키면 스레드가 동렬의 위치를 접근하도록 유도할 수 있다.

본 논문은 관찰 방향과 상관없이 화면의 x축을 기준으로 데이터를 정렬하

기 위해 x축, y축, z축 방향을 기준으로 재정렬한 3개의 볼륨 데이터를 사전에 생성한다. 이후 관찰 방향에 따라 적절한 데이터를 선택하고, 관찰자 좌표계에서 화면의 가로 방향과 세로 방향을 정의하는 벡터를 회전 변환하면 두 축을 정렬할 수 있다. 스레드 동렬 투사 기법이 적용된 볼륨 데이터와 관찰자 좌표계는 인접한 스레드들이 데이터를 순차적으로 참조하도록 유도함으로써, 보간 과정에서 발생하는 메모리 효율을 증가시킨다. 결과적으로 렌더링 속도를 향상시켜 고차 보간의 볼륨 렌더링을 실시간으로 수행하는 데 기여한다.

제 2 절 논문의 구성

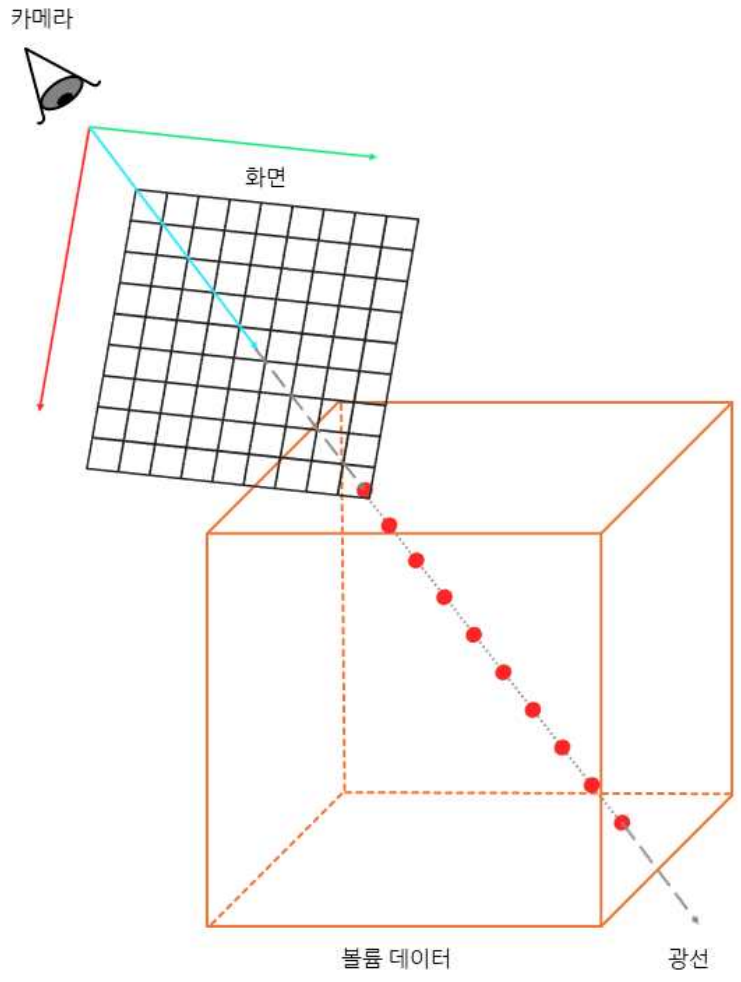
본 논문의 2장은 논문과 관련된 기존 연구를 설명한다. 볼륨 렌더링과 광선 투사법의 기본적인 내용, 고차 보간의 기법인 B-spline 근사와 B-spline 보간에 대해서, 그리고 이를 볼륨 렌더링에 적용하는 방법과 GPU 병렬 프로그래밍 모델인 CUDA에 대해 소개한다. 3장은 GPU 메모리 접근 패턴에 대해 설명하고, 4장은 본 논문이 제안하는 스레드 동렬 투사 기법을 개진한다. 5장은 제안 기법의 실험 결과를 제시하고, 마지막으로 6장은 본 논문의 결론을 정리하며 마무리한다.

제 2 장 관련 연구

제 1 절 볼륨 렌더링

볼륨 렌더링(Volume Rendering)(Drebin, R, A. et al, 1988)은 볼륨 데이터(Volume Data)라고 부르는 3차원 배열 형식의 스칼라 데이터를 2차원 이미지로 가시화하는 방법이다. 볼륨은 3차원 배열의 요소인 복셀(voxel)로 구성되어 있으며, 각 복셀에는 해당 위치의 스칼라 값[1]이 저장되어 있다. 이 스칼라 값을 볼륨 데이터라고 한다. 볼륨 데이터를 얻는 대표적인 예시로는 CT 촬영이 있다. 의료 영상 장비를 통해 신체의 단면을 연속적으로 촬영하고, 높이 방향으로 쌓아 올리면 볼륨 데이터를 만들 수 있다. 배열 요소에는 각 위치의 밀도 값이 저장되며, 이를 복셀이라고 정의한다. 이렇게 만들어진 복셀을 이용하면 객체의 내부 구조를 반영하여 영상을 가시화할 수 있다.

1) 밀도, 강도 등



[그림 1] 볼륨 렌더링의 광선 투사법

볼륨 렌더링 기법 중 가장 대중적으로 사용되는 방법은 광선 투사법(ray casting)(Levoy, M., 1988)이다. 광선 투사법은 카메라 위치에서 가상의 화면을 생성하고, 화면의 각 화소에서 관찰 방향으로 광선을 투사한다. 그 다음, 각 광선 경로에 위치한 복셀 값들을 일정한 간격으로 조회한다. 그러나 값을 조회하려는 위치와 복셀의 중심이 일치하지 않는 경우가 있으므로, 조회된 복셀 값을 보간 과정을 통해 조회하려는 위치의 값으로 계산한다. 이렇게 보간된 값들은 전이함수(transfer function)을 통해 RGBA[2] 값으로 변환한다. 전

2) red, green, blue, alpha

이함수는 사용자가 정의하는 함수로, 복셀 값을 원하는 색상과 불투명도로 대치시킬 수 있다. 이렇게 얻은 RGBA 값을 누적(compositing)하여 결과 영상을 이루는 화소의 색상이 결정된다. 이때 불투명도가 1에 가까워지면 해당 광선에 대한 계산을 조기 종료(early ray termination)하여 계산 효율을 높일 수 있다(Levoy, M, 1990).

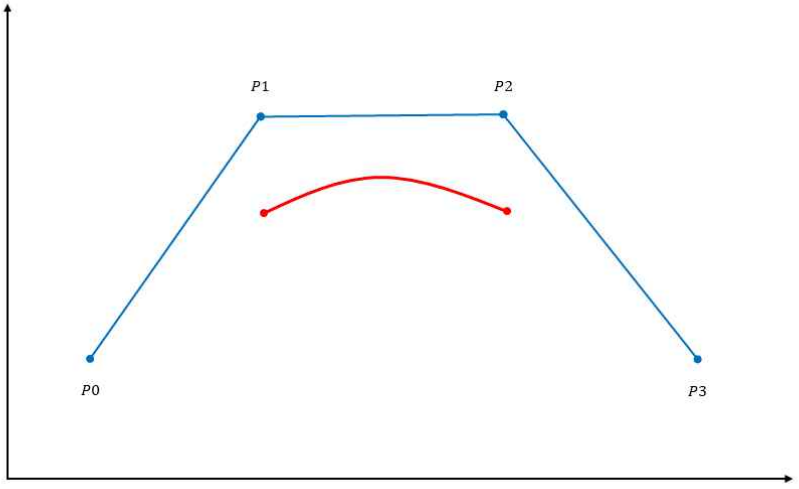
광선 투사법을 이용한 볼륨 렌더링은 상술한 과정을 화면을 구성하는 모든 화소에 대해 수행해야 하므로, 연산량이 많아 대화적 속도를 달성하기에는 어려움이 따른다. 이를 극복하기 위해 빈 공간 도약(empty space skipping), GPU 병렬 처리 등 다양한 최적화 기법들이 활발히 연구되고 있다(Beyer, J., et al, 2015). 특히, GPU를 활용한 병렬 처리는 볼륨 렌더링의 속도를 크게 향상시킬 수 있는 효과적인 방법으로 주목받는다. GPU의 대규모 병렬 처리 능력을 활용하면 다수의 광선에 대한 연산을 동시에 수행할 수 있어, 볼륨 렌더링의 성능을 크게 향상시킬 수 있다(Sugimoto, Y., et al, 2014).

제 2 절 B-Spline 근사

볼륨에는 이산적인 데이터 값이 저장되어 있으므로, 볼륨 렌더링은 이를 연속적인 함수로 근사하기 위해 보간을 수행한다. 보간 과정에서 주로 사용되는 방법에는 최근접 이웃 보간법과 삼선형 보간법이 있다. 최근접 이웃 보간법은 조회하려는 지점의 값을 가장 가까운 복셀의 값으로 대체하는 방법으로, 계산 속도가 빠르지만 결과 영상의 화질이 낮은 단점이 있다. 삼선형 보간법은 선형 보간을 3차원 공간에서 수행하는 방법으로, 조회하려는 지점과 인접한 8개의 복셀 값으로 값을 추정하는 방법이다. 삼선형 보간법은 일반적으로 널리 사용되는 방식으로 계산 속도와 결과 영상의 화질에서 적절한 균형을 이룬다. 그러나 삼선형 보간의 화질은 의료 영상과 같이 고품질 볼륨 렌더링이 필요한 분야에서 요구하는 수준을 충족하지 못하므로, 더 나은 품질의 영상을 위해 고차 보간을 사용한다.

고차 보간은 여러 사례가 있는데, (Rössl, C., et al, 2004)는 B-Spline 근

사(Schoenberg, I. J. 1946)를 볼륨 데이터 보간에 사용하는 방안을 제시하였다. n차 B-Spline 근사는 1차원 당 (n+1)개의 제어점을 통해 부드러운 곡선을 생성하므로, 데이터 간의 연속성을 보장한다. 그 중 3차 B-Spline 곡선을 사용한 보간이 성능과 화질의 균형을 이루어 널리 사용되고 있다.



[그림 2] 3차 B-Spline 근사 곡선

B-spline 함수는 기저 함수(basis function)의 선형 조합으로 표현되며, 3차 B-spline 기저 함수 β_3는 다음과 같이 정의된다(Sigg, C. et al, 2005).

$$
\begin{aligned}
w_0(t) &= \left(\frac{1}{6}\right)(1-t)^3 \\
w_1(t) &= \left(\frac{1}{6}\right)(3t^3-6t^2+4) \\
w_2(t) &= \left(\frac{1}{6}\right)(-3t^3+3t^2+3t+1) \\
w_3(t) &= \left(\frac{1}{6}\right)t^3
\end{aligned}
\qquad \text{[수식 2]}
$$

이를 행렬 형태로 나타내면, $w(t)$는 다음과 같은 벡터가 된다.

$$w(t)= \begin{bmatrix} w_0(t) \\ w_1(t) \\ w_2(t) \\ w_3(t) \end{bmatrix} \qquad [수식\ 2]$$

따라서, 이를 전개하면 다음과 같다.

$$w(t)= \frac{1}{6}\begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 4 \\ 0 & -3 & 3 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \qquad [수식\ 3]$$

해당 수식에 제어점 벡터를 곱하면, 최종적으로 제어점이 $w(t)$에 의해 가중된 합인 3차 B-Spline 곡선을 얻을 수 있다.

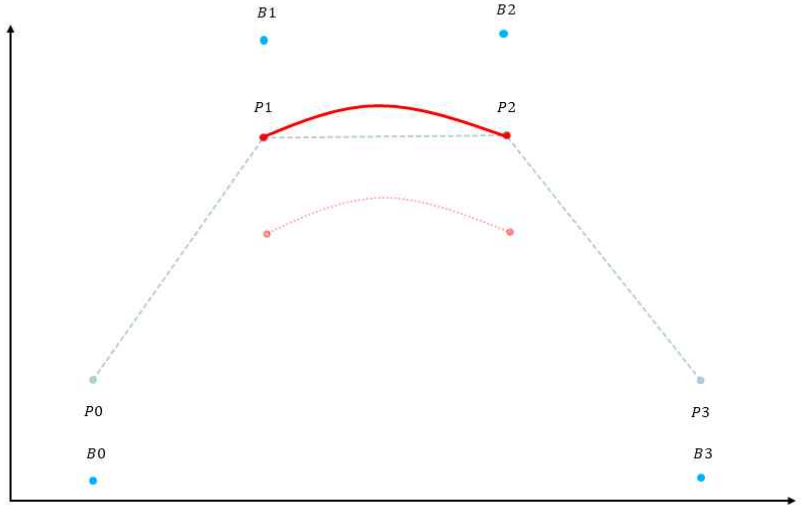
$$S_i(t)= w(t)\begin{bmatrix} P_{i-1} \\ P \\ P_{i+1} \\ P_{i+2} \end{bmatrix}= \begin{bmatrix} t^3 t^2 t 1 \end{bmatrix}\frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_{i-1} \\ P \\ P_{i+1} \\ P_{i+2} \end{bmatrix} \qquad [수식\ 4]$$

여기서 t는 매개변수고 P_{i-1}, P_i, P_{i+1}, P_{i+2}는 제어점이다. P_{i-1}, P_i, P_{i+1}, P_{i+2}의 값을 입력으로 넣으면 매개변수 $w(t)$에 따라 가중된 S_i를 얻을 수 있다. B-Spline 곡선은 부드러운 곡선의 형태를 띄고 있기 때문에, 볼륨 렌더링에 적용할 경우 매끈한 표면을 표현하는데 특화된 고품질의 영상을 출력할 수 있다. 또한 B-Spline 근사는 로우패스 필터링(lowpass filtering) 특성을 갖고 있어 볼륨 데이터의 노이즈를 제거하는데 효과적이다(Mihajlovic, Z., et al, 2004). 이는 의료 영상 등 잡음에 민감한 데이터를 다룰 때 유용하다. 그러나 B-Spline 곡선은 4개의 제어점의 가중합으로 결정되는데, 각 가중치 값은 1보다 항상 작으므로 곡선은 언제나 [그림 2]처럼 각 제어점을 지나지 않는다. 이는 볼륨 렌더링에서 샘플링

한 볼륨 데이터의 실제 값을 관찰하지 못하는 근사 오차의 원인이 된다.

제 3 절 B-Spline 보간

B-Spline 근사의 한계인 근사 오차와 오버 블러(OverBurr) 현상을 극복하기 위해 B-spline 보간이 제안되었다. B-spline 보간은 기존 제어점에 가중치를 부여한 새로운 제어점을 생성하여 연산하는 방법이다. 새로 생성된 제어점으로 연산을 수행하면 B-Spline 곡선이 주어진 기존의 제어점을 정확히 지나간다. 따라서 B-Spline 근사의 특징인 부드러운 결과 영상은 유지하면서 원본 데이터를 정확하게 반영할 수 있다.



[그림 3] 3차 B-Spline 보간 곡선

$x \in R$에서의 B-Spline 보간은 B-Spline 기저 함수 β_n을 B-Spline 계수 $c(k)$로 가중한 합이다. 따라서 다음 식이 모든 정수 x에 대해 만족하도록 $c(k)$의 값을 결정해야 한다(Ruijters, D. et al. 2008).

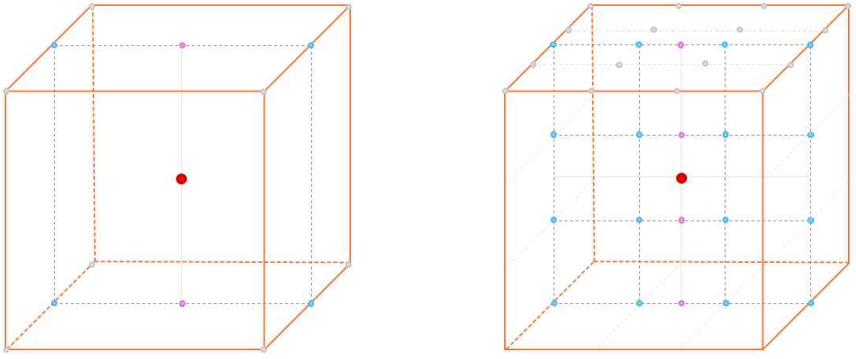
$$S_n(x) = \sum_{k \in Z} c(k)\beta_n(x - k) \qquad [수식\ 5]$$

3차 B-spline의 기저 함수는 구간별로 정의하면 다음과 같다.

$$\beta_3 = \begin{cases} 0, & 2 \leq |x| \\ \frac{1}{6}\cdot(2-|x|)^3, & 1 \leq |x| < 2 \\ \frac{2}{3} - \frac{1}{2}|x|^2\cdot(2-|x|), & |x| < 1 \end{cases} \qquad [수식\ 6]$$

B-spline은 제한된 구간에서만 0이 아닌 값을 가지므로, 위치 x에서의 보간에 영향을 미치는 계수 c(k)의 수는 제한된다. 주어진 위치에 대해 3차 B-spline 보간을 계산하기 위해선 주변 4개의 제어점에 각각과 대응하는 계수를 가중하여 합해야 한다.

B-spline 보간은 높은 품질의 결과를 제공하지만, 계산 비용이 크다는 단점이 있다. 선형 보간의 경우 1차원 기준으로 2개의 제어점을 사용하므로, 삼선형 보간으로 한 점을 보간하기 위해 $2^3 = 8$번의 데이터 참조가 필요하다. 반면, 3차 B-spline 보간은 1차원 기준 4개의 제어점을 사용하므로, 이를 3차원에서 수행할 경우 $4^3 = 64$번의 데이터 참조가 발생한다. 이는 계산 비용을 크게 증가시키는 요인이 된다.



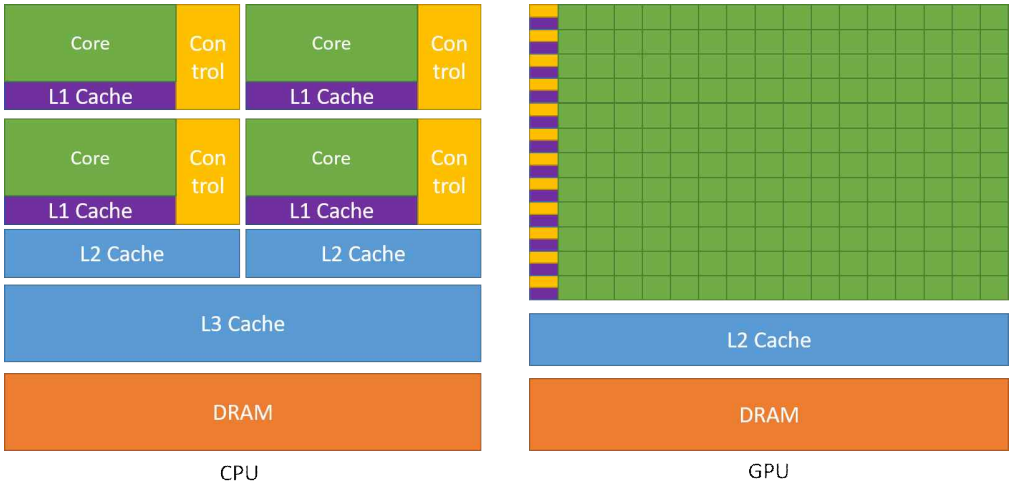
[그림 4] 3차원에서의 선형 보간과 3차 보간

3차 B-spline 보간의 성능 문제를 해결하기 위해 다양한 가속화 기법이 연구되었다. Rössl, C., et al, (2004) 은 Type-6 사면체 분할을 통해 64개의 제어점을 사용하지 않고도 B-Spline 보간과 유사한 품질을 얻을 수 있는 새로운 보간 방법을 제안하였다. Ruijters, D. et al. (2008) 에서는 룩업 테이블 생성을 통해 B-Spline 보간 과정을 사전에 수행하여 볼륨 렌더링의 성능을 향상시켰다. 신용하, 계희원. (2016) 은 B-Spline 곡선을 조각적 베지어 곡선으로 분해해 빈공간 도약을 적용하여 볼륨 렌더링의 성능을 향상시켰다. 그러나 선행된 연구 모두 B-Spline 곡선의 제어점이 될 값을 조회하는 샘플링 과정에서 일어나는 비효율적인 GPU 메모리 접근 패턴 문제는 해결하지 못했다.

제 4 절 CUDA

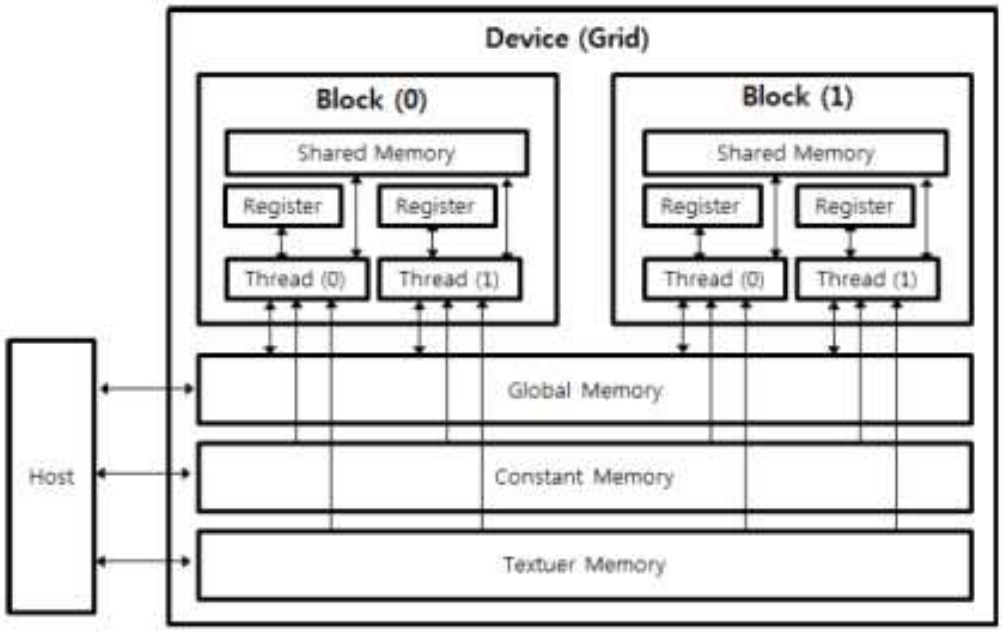
CPU와 GPU는 구조적으로 차이가 있다. CPU는 복잡한 구조의 코어(Core)를 적게 갖고 있어 어려운 연산을 순차적으로 빠르게 수행하는 데 적

합하다. 반면, GPU는 단순한 구조의 코어를 많이 갖고 있어 대량의 연산을 병렬적으로 수행하는 데 특화되어 있다. NVIDIA는 이러한 GPU의 특성을 이용해, 병렬 컴퓨팅 플랫폼 및 프로그래밍 모델인 CUDA(Compute Unified Device Architecture)를 개발했다. 개발자는 CUDA를 통해 GPU의 많은 스레드를 활용하여 연산을 효율적으로 병렬 처리할 수 있다.



[그림 5] CPU와 GPU 구조 차이

또한, CUDA는 다양한 메모리 공간을 제공한다. 각 메모리 공간은 서로 다른 특성과 용도를 가지고 있으므로, 이를 활용하여 프로그램의 성능을 더욱 최적화할 수 있다.



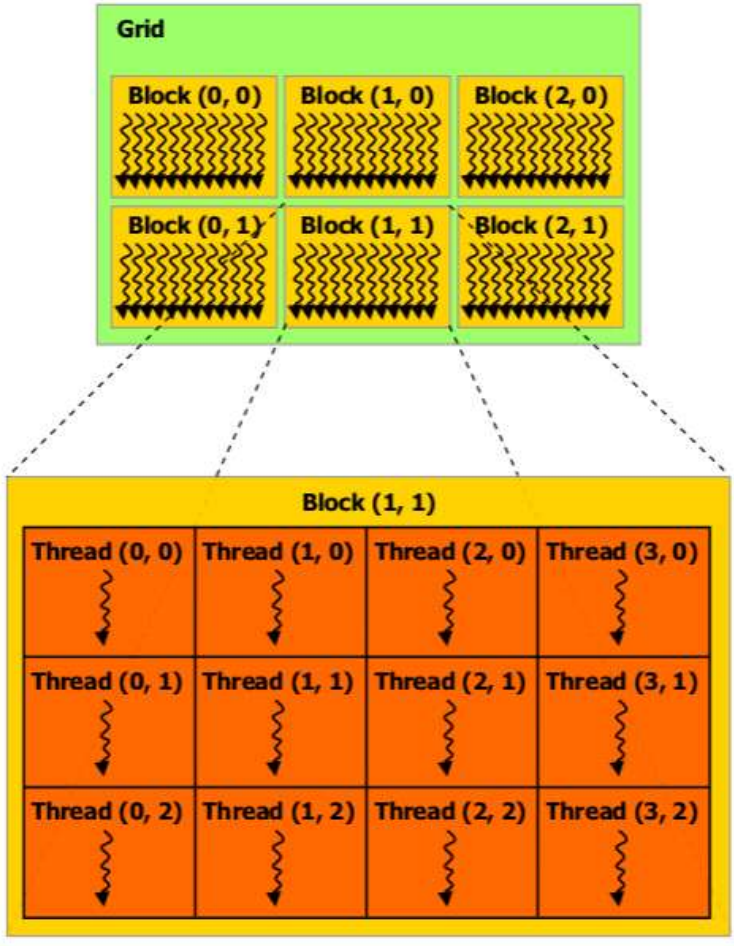
[그림 6] CUDA의 메모리 구조

전역 메모리(Global Memory)는 GPU에서 가장 큰 용량을 가지는 기본적인 메모리로, 모든 스레드에서 접근 가능하다. 대부분의 데이터는 전역 메모리에 저장되며, CPU와 GPU 간의 데이터 전송도 전역 메모리를 통해 이루어진다. 하지만 전역 메모리는 높은 대기 시간(Latency)과 낮은 대역폭을 갖고 있어, 프로그램 성능 저하의 주요한 원인이 된다.

공유 메모리(Shared Memory)는 블록 내의 스레드들이 협력적으로 사용할 수 있는 고속 메모리이다. 각 블록마다 독립적인 공유 메모리가 할당되며, 블록 내의 모든 스레드는 이 공유 메모리에 접근할 수 있다. 공유 메모리는 전역 메모리에 비해 매우 빠르지만, 크기가 제한적이다.

텍스처 메모리(Texture Memory)는 2D 또는 3D 텍스처 데이터를 효율적으로 접근하기 위해 사용되는 특수한 메모리다. 텍스처 메모리는 전역 메모리에 위치한 특수한 영역으로, 읽기 전용으로 동작한다. 텍스처 메모리는 캐시 메커니즘을 통해 데이터 지역성을 활용하므로, 반복된 좌표나 인접한 텍스처 좌표에 대해 접근할 경우 성능이 향상된다. 볼륨 렌더링에서 볼륨 데이터는 일반적으로 텍스처 메모리의 3D 텍스처로 저장되어 사용된다.

상수 메모리(Constant Memory)는 읽기 전용 메모리로, 커널 실행 중에 값이 변경되지 않는 데이터를 저장하는 데 사용된다. 상수 메모리는 모든 스레드에서 접근 가능하며, 캐시를 통해 빠른 접근 속도를 제공한다. 그러나 상수 메모리의 크기는 작게 제한되므로, 대량의 데이터를 저장하는 데는 적합하지 않다.



[그림 7] CUDA의 구조

CUDA는 그리드, 블록, 스레드로 구성된 계층적 구조로 이루어져 있다. 스레드는 커널(Kernel)이라 불리는 함수를 독립적으로 실행하는 기본 단위다.

여러 스레드가 모여 블록을 형성하며, 같은 블록 내의 스레드들은 공유 메모리와 레지스터를 공유한다. 이 블록들이 모여 하나의 그리드를 구성한다. 이러한 스레드 배치 구조는 실제 하드웨어 상 배치 구조가 아닌 소프트웨어 상의 논리적 구조다. 개발자는 스레드와 블록의 차원 형태를 최대 3차원까지 선택할 수 있으며, 블록당 스레드 수와 그리드당 블록 수를 결정할 수 있다. 한편, 하드웨어 상에서 스레드 인덱스의 연속된 x값을 가진 스레드들은 32개씩 묶여 워프라는 단위를 형성한다. 같은 워프에 속한 스레드들은 SIMD(Single Instruction, Multiple Data) 방식으로 동작하여 동일한 명령을 동시에 수행한다.

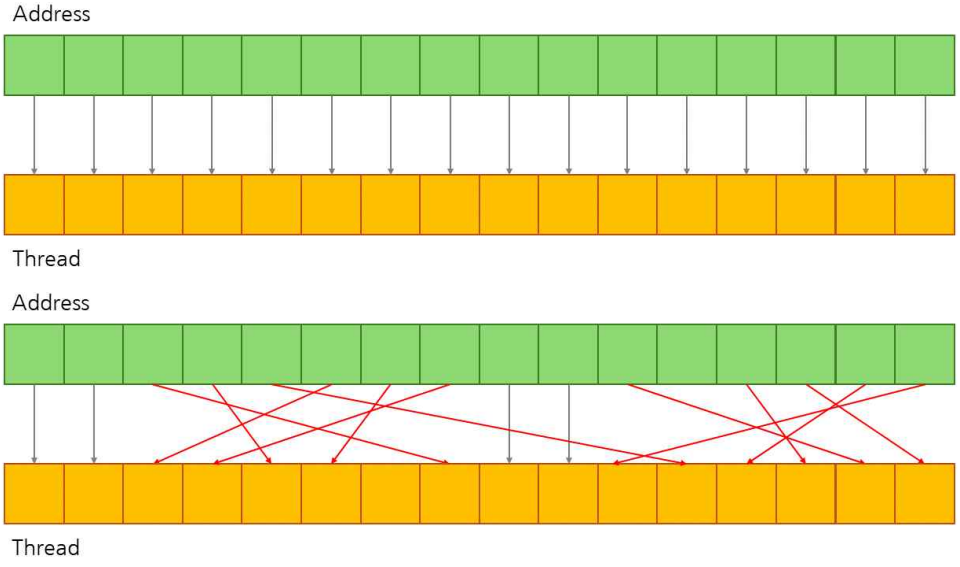
볼륨 렌더링은 CUDA를 활용한 대표적인 응용 분야 중 하나이다. 볼륨 렌더링은 광선 투사법 과정에서 대량의 볼륨-광선 교차 검사와 색상 누적 연산이 필요하므로, CUDA의 메모리 계층과 병렬 처리 능력을 통해 최적화하는 연구가 꾸준히 이루어지고 있다. (Wang, J. et al. 2017)은 하나의 워프에 위치한 스레드들이 동일한 메모리를 접근하도록 샘플 간격을 조절해, 렌더링 성능을 개선하였다. (Sugimoto, Y. et al. 2014)는 카메라와 볼륨 축 사이의 각도에 따라 스레드의 모양을 동적으로 선택해, 렌더링 성능을 향상시켰다. 이처럼 CUDA는 볼륨 렌더링을 비롯한 다양한 그래픽 응용 분야에서 GPU 가속을 위한 핵심 기술로 활용되고 있다.

제 3 장 CUDA 메모리 액세스 최적화

제 1 절 Coalesced Access

볼륨 렌더링에서는 일반적으로 볼륨 데이터가 3차원 텍스처로 저장되며, 이는 전역 메모리에 위치한다. 볼륨 렌더링 커널에서는 샘플링 과정에서 볼륨 데이터를 빈번하게 액세스하는데, 이 때 전역 메모리의 느린 액세스 속도가 성능 저하를 유발한다. 따라서 전역 메모리 액세스를 최소화하고, 가능한 한 빠른 메모리를 활용하는 것이 중요하다.

CUDA에서 Coalesced Access란 인접한 스레드들이 전역 메모리의 인접한 위치에 동시에 액세스하는 것을 말한다. Coalesced Access가 발생하면 메모리 컨트롤러가 여러 개의 메모리 요청을 하나로 통합하여, 해당되는 메모리 영역을 한 번에 읽어올 수 있다. 반면, Non-Coalesced Access는 인접한 스레드들이 흩어져 있는 메모리 위치에 액세스할 경우를 뜻한다. 이 경우, 메모리 컨트롤러가 요청을 통합할 수 없어 전역 메모리를 여러 번 액세스하게 되며 성능의 저하로 이어진다.



[그림 8] coalesced access와 non-coalesced access

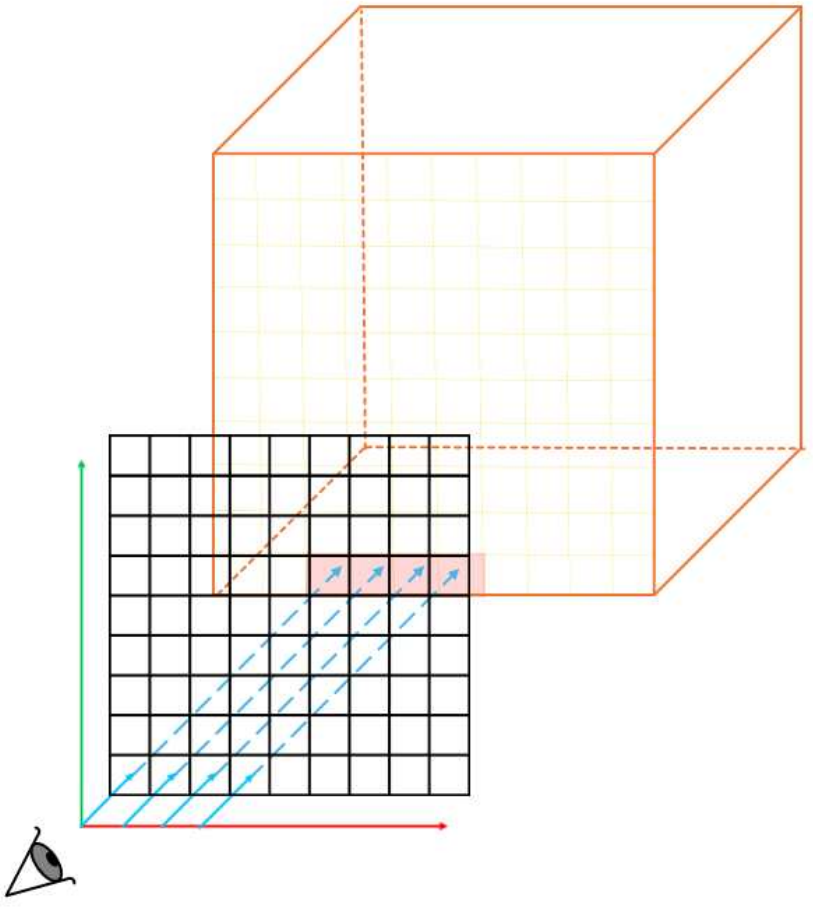
광선 투사법을 이용한 볼륨 렌더링에서는 각 스레드가 화면의 한 화소를 담당하며, 해당 화소를 통과하는 광선을 추적하면서 볼륨 데이터에 접근한다. 이러한 샘플링 과정에서 하나의 워프에 속한 스레드들이 메모리의 인접한 위치를 참조한다면, coalesced access가 발생해 메모리 접근 패턴을 최적화할 수 있다.

이를 위해서는 하나의 워프에 속한 스레드들이 물리적으로 인접한 데이터에 접근할 수 있도록 스레드 인덱스와 데이터 인덱스를 적절히 대응해야 한다. 볼륨 렌더링에서는 일반적으로 관찰자 좌표계를 이용하여 스레드 인덱스와 데이터 인덱스를 짝 지으며, 이는 관찰자 좌표계가 메모리 접근 패턴에 영향을 미치는 원인이 된다. 관찰자 좌표계는 관찰 방향 벡터 dir, 화면의 가로 방향인 u, 세로 방향인 v로 이루어져 있다. u와 v를 통해 스레드 인덱스의 x는 화면의 x 방향 위치와 대응되고 y는 y 방향 위치와 대응되어, 하나의 스레드가 하나의 화소를 담당하게 된다.

스레드 인덱스 상에서 연속된 스레드들은 화면 상에서 연속된 화소들을 담당하므로, 볼륨 데이터를 메모리에 저장할 때에는 인접한 데이터가 메모리

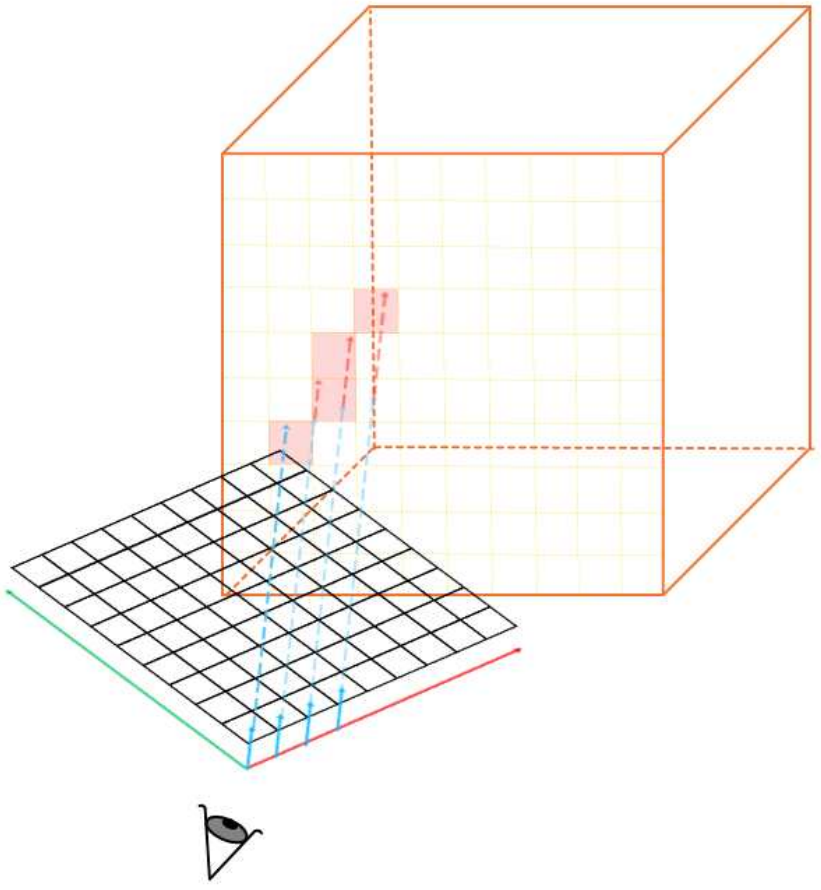
의 연속된 공간에 위치되어야 coalesced access가 발생할 확률이 높아진다. 일반적으로 볼륨 데이터의 경우, (x, y, z) 순서로 데이터를 저장한다. 이 경우, x축 방향으로 인접한 데이터가 메모리 상에서도 인접하게 위치하게 된다.

데이터가 x축을 기준으로 정렬되었을 경우, 데이터 인덱스의 x축이 정렬된 데이터의 x축과 평행을 이룰 때 연속된 스레드가 메모리 공간 상에서 연속되게 위치한 데이터에 접근하게 된다. 일반적으로 데이터 인덱스의 x축은 화면의 가로 축과 대응된다. 따라서 화면의 가로 축인 u가 볼륨의 x축이 이루는 각도는 샘플링 과정에서의 메모리 접근 패턴에 직접적인 영향을 미친다.



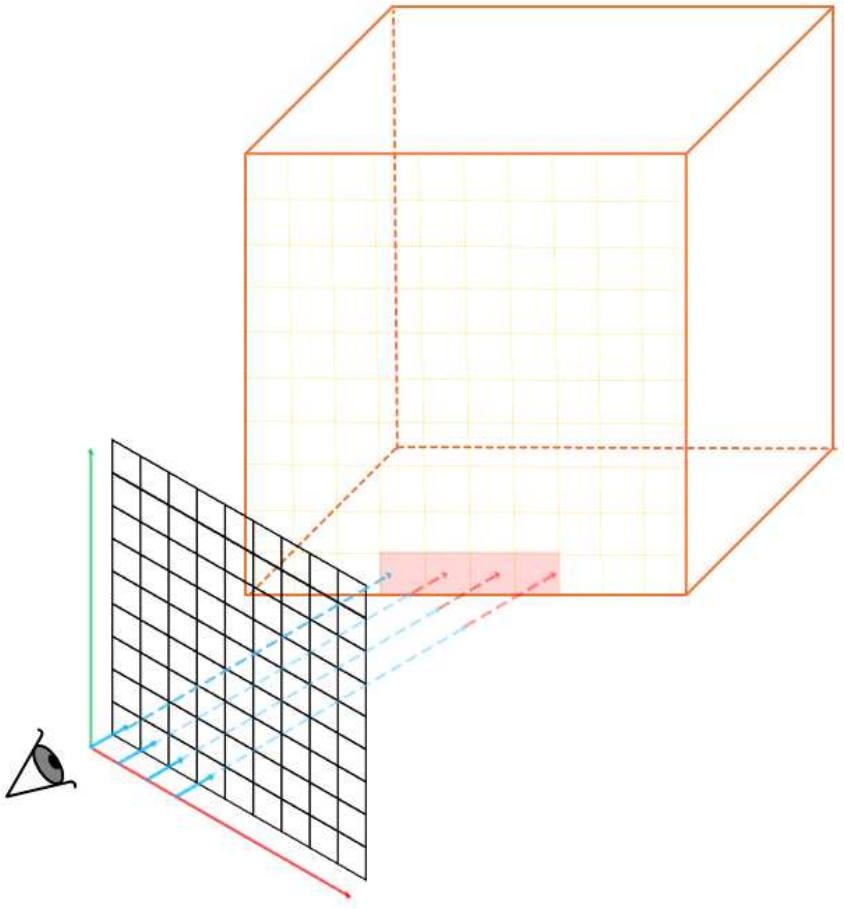
[그림 9] 화면의 x축과 볼륨의 x축이 평행한 경우

예를 들어 u와 볼륨의 x축과 완전히 평행한 경우, 스레드가 연속적으로 x축 방향으로 이웃한 데이터를 접근하므로 접근 패턴이 규칙적이고 연속적이다. 이 경우 coalesced access가 발생하여 캐시 적중률이 높아지고 메모리 참조 패턴의 효율이 좋아진다.



[그림 10] 화면의 x축과 볼륨의 x축이 평행하지 않은 경우

반면, u이 볼륨의 x축과 임의의 각도를 이루는 경우, 화면의 연속된 화소가 서로 떨어져 있는 위치의 데이터를 참조하게 된다. 이는 non-coalesced access로 이어져 메모리 효율이 저하되는 원인이 된다. u와 x 사이의 각도가 커질수록 메모리 참조 패턴은 비효율적이게 된다.



[그림 11] 화면의 x축과 투영된 볼륨의 x축이 평행한 경우

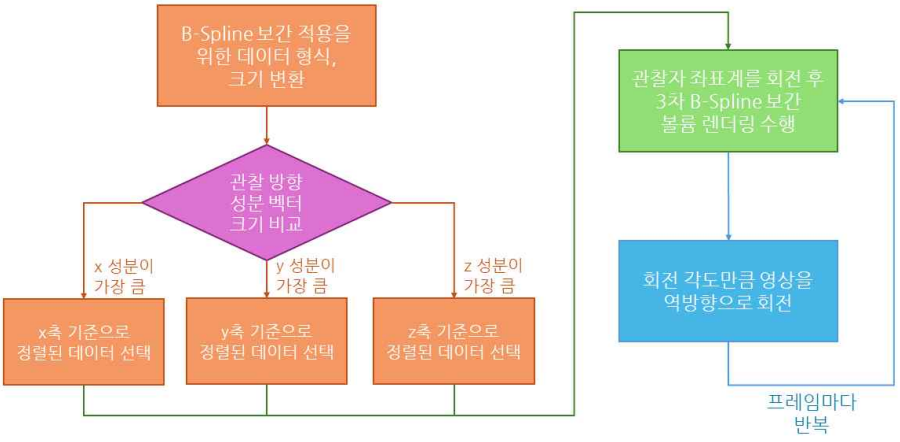
coalesced access가 발생하기 위해 u와 볼륨의 x축이 3차원 공간 상에서 반드시 평행할 필요는 없다. 실제로는 스레드 인덱스와 데이터 인덱스의 대응 관계가 중요시되는데, 예를 들어 화면에 투영된 볼륨의 x축이 화면의 x축과 평행을 이루면 광선을 투사하는 과정에서 스레드 인덱스는 연속된 위치의 데이터를 참조하므로 coalesced access가 발생한다. 이러한 현상은 그림 11에서 확인할 수 있다.

제 4 장 스레드 동렬 투사 기법

제 1 절 개요

3장에서는 볼륨 데이터가 적절한 순서로 저장되어 있을 때, 관찰자 좌표계를 구성하는 화면의 축과 화면에 투영된 볼륨 데이터의 축을 정렬한다면 coalesced access를 유도할 수 있음을 살펴보았다. 앞선 내용을 바탕으로, 본 장에서는 볼륨 데이터의 정렬과 스레드 인덱스의 축 정렬을 통해 볼륨 렌더링의 메모리 접근 패턴을 최적화하는 스레드 동렬 투사 기법을 제안한다. 본 논문에서는 화면의 x축과 화면에 투영된 볼륨의 x축을 평행하게 정렬시키는 방식으로 스레드 동렬 투사 기법을 선보인다. 볼륨 렌더링 과정에는 3차 B-Spline 보간을 적용하여, 스레드 동렬 투사 기법이 고화질 볼륨 렌더링을 대화적 속도까지 가속화할 수 있음을 보인다.

본 논문이 수행하는 알고리즘 순서도는 다음과 같다.



[그림 12] 스레드 동렬 투사 기법 알고리즘 순서도

1) 전처리 단계

볼륨 렌더링을 수행하기에 앞서, 3차 B-spline 보간을 적용할 수 있도록 볼륨 데이터를 크기와 데이터 타입을 변환한다. 또한 관찰 방향과 관계없이 스레드 동렬 투사 기법을 적용할 수 있도록 여러 개의 볼륨 데이터를 사전에 정렬하고 생성하는 전처리 작업이 필요하다. 예를 들어, 관찰자가 볼륨의 yz 평면을 주로 관찰한다면 데이터는 (y, z, x) 순서로 정렬되어야 coalesced access를 최대한으로 유도할 수 있다. 즉 주된 관찰 방향에 따라 서로 다르게 정렬된 3개의 데이터가 필요하므로, 크기와 데이터 타입이 변환된 볼륨 데이터를 회전하여 각각 x축, y축, z축에 수직인 순서로 저장한 3개의 데이터를 생성하는 것이다.

2) 볼륨 렌더링 단계

스레드 인덱스와 데이터 인덱스의 축을 정렬하기 위해 볼륨 데이터를 회전시키는 방식은 계산 비용의 소모가 커 비효율적이다. 따라서 관찰자 좌표계를 변환하여 두 축을 정렬해야 한다. 본 논문은 앞서 언급한 바와 같이 회전 변환을 통해 화면을 이루는 x 방향의 축과 화면에 투영된 볼륨의 x축을 정렬

한다. 우선 카메라의 위치와 초점으로 결정되는 관찰 방향 벡터 dir의 성분 크기에 따라 사용할 데이터를 선택한다. 다음으로 일반적인 방식의 관찰자 좌표계를 계산하고, 계산된 관찰자 좌표계를 회전시켜 스레드 동렬 투사 기법이 적용된 새로운 관찰자 좌표계를 생성한다. 마지막으로 생성된 관찰자 좌표계를 사용하여 3차 B-spline 보간을 적용한 볼륨 렌더링을 수행한다.

3) 후처리 단계

렌더링 과정에서 화면을 이루는 벡터를 회전하였으므로, 결과 영상은 기존 영상과 달리 회전되어 출력된다. 이를 해결하기 위해, 결과 영상을 앞서 회전한 각도만큼 역방향으로 회전시킨다. 화면의 회전 외에는 결과 영상에 어떠한 왜곡도 일어나지 않았으므로, 최종적으로 원본 영상의 내용과 품질이 그대로 유지된다.

앞서 제시된 2단계와 3단계를 프레임마다 반복하면, 마우스의 움직임에 따라 관찰 방향이 달라지는 등의 상호 작용이 가능한 볼륨 렌더링을 대화적 속도로 수행할 수 있다.

제 2 절 전처리 단계

본 논문이 제시하는 알고리즘을 수행하기 위해서는 두 가지의 사전 작업이 필요하다. 첫 번째는 B-Spline 보간을 위한 자료형 및 크기 변환 작업이고, 두 번째는 논문에서 제시하는 스레드 동렬 투사 기법을 적용하기 위한 볼륨 데이터 정렬 작업이다.

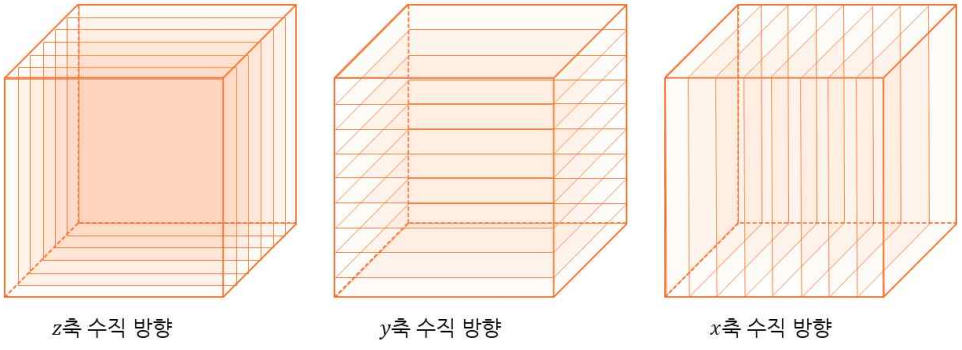
3차원 볼륨 데이터에 대한 B-spline 보간은 각 축의 방향으로 1차원 B-Spline 보간을 순차적으로 적용하여 수행한다. 다시 말해, x축, y축, z축 방향으로 각각 B-spline 보간을 수행한다. 보간 과정에서는 볼륨 데이터의 각 복셀을 제어점으로 사용하여 연속적인 스칼라 함수를 보간한다. 최종적으로

보간된 함수의 값은 샘플링 과정에서 사용된다. 3차원에서의 3차 B-spline 보간 함수는 다음과 같이 표현할 수 있다(Theußl et al. 2000).

$$S(x,\ y,\ z) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} C_{ijk}\beta_3(x-i)\beta_3(y-j)\beta_3(z-k) \qquad [수식\ 7]$$

B-Spline 보간에서 데이터의 경계값을 처리하기 위해서는 추가적인 제어점이 더 필요하다. 1차원 보간의 경우, 매 구간의 양 끝에 1개씩의 제어점이 추가로 사용된다. 3차원 데이터에 대한 적용은 이 과정을 y축, z축에 대해 추가로 수행하면 된다. 따라서, 볼륨의 크기가 *width*, *height*, *depth*라고 할 때, 3차 B-spline 보간을 위해서는 $(width+2)$, $(height+2)$, $(depth+2)$ 크기의 메모리 공간이 필요하다(신용하, 계희원. 2016).

또한 B-Spline 보간을 정확하게 수행하기 위해선 적용할 볼륨 데이터의 형식을 float 형식으로 변환해야 한다. 일반적으로 의료 영상 데이터는 unsigned short와 같은 2byte의 정수 형식으로 저장되나, B-Spline 보간 연산은 정밀한 근사를 위해 부동 소수점 연산을 기반으로 수행되므로 데이터 변환 과정이 반드시 필요하다.



[그림 11] 스레드 동렬 투사 기법 적용을 위한 볼륨 데이터 정렬

앞선 3장에서는 볼륨 데이터가 (x, y, z) 순서로 정렬되어 있고, 관찰자는 xy 평면을 바라볼 때 coalesced access가 유도되는 원리를 살펴보았다. 그러나 실제 볼륨 렌더링 환경에서는 관찰자의 관찰 방향에 따라 주로 관찰되는 평면이 다르므로, 이를 반영하는 과정이 필요하다. 예를 들어, dir의 x 성분이 가장 크다면 볼륨의 yz평면을 주로 관측하고 있음을 의미한다. 이때 데이터는 (y, z, x) 순서로 저장되어야, 한 워프 안의 스레드들이 y 방향으로 인접한 데이터를 참조하게 되어 coalesced access를 가장 이상적으로 유도할 수 있다. 따라서 제시된 기법의 적용을 위해 x축, y축, z축에 각각 수직인 방향으로 저장된 3개의 볼륨 데이터를 전처리 과정으로 생성한다.

이후, 알고리즘은 볼륨 렌더링 과정에서 관찰자 좌표계를 통해 주로 관찰하는 평면을 판단하고 적절한 데이터를 선택하여 샘플링 과정을 수행한다. 화면의 x축과 투영된 볼륨 렌더링의 x축을 평행하게 정렬하여 진행하면, 관찰 방향에 상관없이 연속된 스레드가 주로 관찰되는 평면의 방향으로 인접한 데이터에 접근할 것이다.

제 3 절 관찰자 좌표계 재구성

스레드 동렬 투사 기법을 적용하기 위해서는 볼륨 렌더링 과정에서 스레드 인덱스의 축과 데이터 인덱스의 축을 정렬해야 한다. 데이터 인덱스의 축이 스레드 인덱스의 축과 평행을 이루도록 볼륨 데이터를 회전하는 방식은 상호 작용형 볼륨 렌더링에서 계산 비용이 높아 비효율적이다. 따라서 본 논문은 관찰자 좌표계를 재구성하여 스레드 인덱스 축을 데이터 인덱스 축 기준으로 정렬하는 방법을 제안한다. 이 방법은 관찰 방향 벡터를 유지하면서, 나머지 벡터를 방향 벡터에 대한 회전으로 변형시켜, 결과 영상의 왜곡 없이 두 축을 정렬할 수 있다.

광선 투사법을 이용한 볼륨 렌더링은 스레드 인덱스의 x, y축을 관찰자

좌표계에서 화면을 이루는 벡터 u, v와 대응시키므로, u와 v를 조작하는 것으로 두 축을 정렬하는 것이 가능하다. 제안하는 방식은 기존의 관찰자 과표계를 계산하고 수식을 적용하여 u와 v를 재계산하는 과정으로 이루어진다. 기존의 관찰자 좌표계는 사용자가 카메라의 위치와 초점 등을 통해 dir를 정의하면, 세 벡터는 서로 수직인 특성을 이용해 나머지 벡터를 구할 수 있다. u는 임의의 벡터인 업벡터와 dir의 외적으로 계산되며, v는 dir과 u의 외적으로 구해진다. 이후, u와 화면에 투영된 볼륨의 x축이 평행을 이루는 각도를 계산하여 회전하면 최종적으로 스레드 인덱스의 x축과 데이터 인덱스의 x축을 정렬할 수 있다.

1) 볼륨-광선 교차를 고려하지 않은 방법

벡터 u를 임의의 각도 θ만큼 회전하여 화면의 x축과 평행하게 정렬하는 과정을 다음과 같이 수식으로 표현할 수 있다.

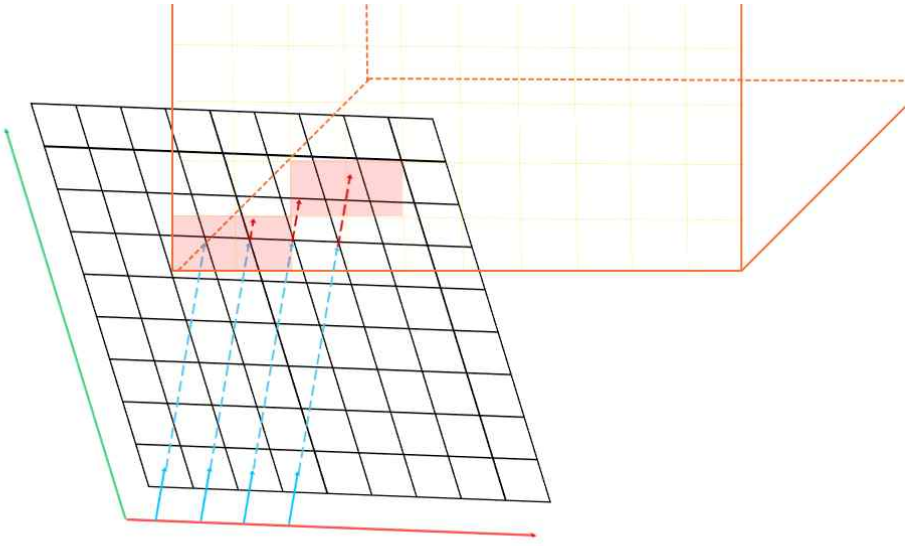
$$(u'_x,\ u'_y) = u \cdot cos\theta + v \cdot sin\theta = (t,\ 0) \qquad \text{[수식 8]}$$

여기서 u'_y, u'_z는 이므로 y나 z를 기준으로 방정식을 전개하면 θ를 얻을 수 있다.

$$\begin{aligned} u_y + v_y \cdot \tan\theta &= 0 \\ \theta &= \operatorname{atan}(-u_y + v_y) \end{aligned} \qquad \text{[수식 9]}$$

새로운 벡터 u', v'는 다음과 같이 계산된다.

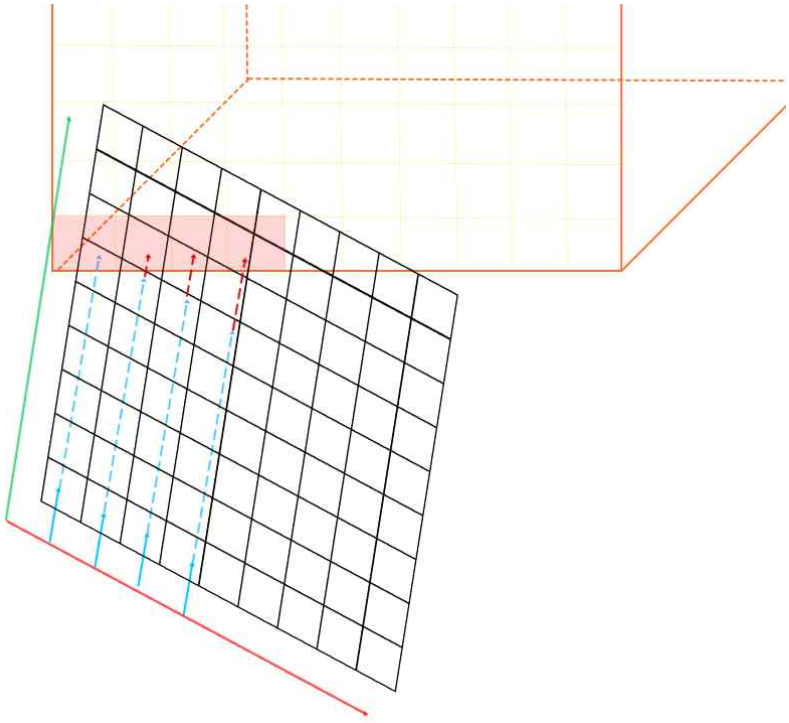
$$\begin{aligned} u' &= \cos\theta \cdot u + \sin\theta \cdot v \\ v' &= -\sin\theta \cdot u + \cos\theta \cdot v \end{aligned} \qquad \text{[수식 10]}$$



[그림 14] 볼륨-광선 교차로 인한 non-coalesced access

그러나 볼륨-광선 교차가 적용된 볼륨 렌더링의 경우, 앞서 제시한 수식만으로는 스레드들이 순차적으로 데이터에 접근하기 어렵다. 이는 볼륨-광선 교차가 적용된 볼륨 렌더링에서 광선들이 화면의 화소가 아닌 볼륨-광선 교차점부터 전진한다는 특성을 고려하지 않았기 때문이다. 볼륨-광선 교차는 광선이 볼륨 데이터의 경계와 처음 만나는 지점을 찾고, 해당 교차점부터 광선을 전진하여 샘플링을 수행하는 방식이다. 이는 볼륨 외부의 빈 공간을 뛰어넘어 렌더링 속도를 증가시키는 이점이 있다.

재구성된 화면의 가로 축인 u'이 투영된 볼륨의 x축이 평행하더라도, 실제 볼륨 렌더링 과정에서는 각 광선이 볼륨에 도달하는 거리가 다르기 때문에 광선은 dir 방향으로 제각각 전진하게 된다. 광선이 샘플링을 시작하는 볼륨-광선 교차점을 선으로 이은 축 l은 볼륨 데이터의 x축과 평행하지 않으므로, coalesced access가 불완전하게 유도된다.



[그림 15] 볼륨-광선 교차로 인한 coalesced access

따라서 볼륨-광선 교차가 적용된 볼륨 렌더링에서는 dir 방향으로의 추가적인 전진을 고려하여 관찰자 좌표계를 재계산해야 한다. 볼륨 렌더링을 수행하기 전에는 u''와 투영된 볼륨의 x축이 평행하지 않을 수 있으나, 볼륨-광선 교차를 수행하면 l과 x가 평행을 이루어 완전한 coalesced access을 유도할 수 있다.

2) 볼륨-광선 교차를 고려한 방법

u''은 임의의 각도 θ만큼 회전한 u'과 dir을 Δw만큼 가산하여 합했다고 가정할 수 있다. 이를 수식으로 작성하면 다음과 같다.

$$u \cdot cos\theta + v \cdot sin\theta + \Delta w \cdot dir = (t,\ 0,\ 0) \qquad [수식\ 11]$$

$\Delta w \cdot dir$를 d라고 가정하면 식 11를 다음과 같이 전개할 수 있다.

$$\tan\theta = \frac{u_y - d \cdot u_z}{-v_y + d \cdot v_z} \qquad [수식\ 12]$$

식 11는 다음과 같이 d에 대한 식으로도 전개할 수 있다.

$$\frac{u_y \cdot \cos\theta + v_y \cdot \sin\theta}{u_z \cdot \cos\theta + v_z \cdot \sin\theta} = \frac{\Delta w \cdot dir_y}{\Delta w \cdot dir_z} \qquad [수식\ 13]$$

식 12과 식 13을 통해 θ를 구할 수 있다.

$$\theta = \operatorname{atan}\theta = \frac{u_y - \frac{dir_y}{dir_z} \cdot u_z}{-v_y + \frac{dir_y}{dir_z} \cdot v_z} \qquad [수식\ 14]$$

최종적인 수식은 다음과 같다.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$u'' = R \cdot u + \frac{u_y \cdot \cos\theta + v_y \cdot \sin\theta}{dir_y} \cdot dir$$

$$v'' = R \cdot v + \frac{u_z \cdot \cos\theta + v_z \cdot \sin\theta}{dir_z} \cdot dir$$

[수식 15]

구해진 각도 θ를 이용하여 u와 v를 회전시키면 u'', v''를 구할 수 있다. 이렇게 변환된 관찰자 좌표계에서 광선 투사를 수행하면, 인접한 스레드들이 볼륨의 x축 방향으로 메모리를 순차적으로 접근할 수 있다. 이는 coalesced access로 이어지고, 메모리 병목 현상을 완화시켜 렌더링 성능

을 향상시킨다.

제 5 장 구현 및 실험 결과

제 1 절 실험 환경

본 실험은 다음과 같은 환경에서 수행되었다.

구분	개발 환경
CPU	AMD Ryzen 5600 6-Core Processor
GPU	NVIDIA GeForce RTX 4060 Ti 8GB
Memory	32GB
OS	Windows 11 Education
IDE	Visual Studio Community 2019
API	CUDA 11.4

[표 1] 실험 수행 환경

제 2 절 실험 결과

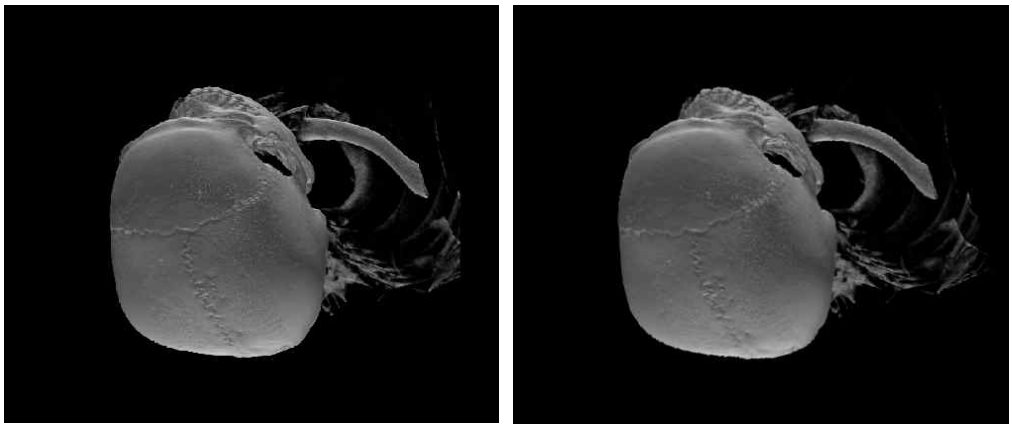
실험에 사용된 데이터 세트는 다양한 CT 데이터를 포함하고 있으며, 각 데이터의 크기를 제안된 방식으로 변형하여 사용하였다. 성능 측정을 위해 900번의 렌더링을 수행하였으며, 소요된 시간의 평균값을 기록하였다. 이 과정을 20회 반복하여 수행 시간의 최소값, 최대값, 평균값을 산출하고 비교 분석하였다.

		기존 방식	제안 기법 적용
Head 01 (512 x 512 x 552)	Min.	244.37 ms	163.90 ms
	Avg.	247.86 ms	165.40 ms
	Max.	255.31 ms	169.59 ms
Abdomen 01 (512 x 512 x 300)	Min.	98.66 ms	63.22 ms
	Avg.	105.68 ms	71.65 ms
	Max.	127.10 ms	82.30 ms
River 01 (512 x 512 x 370)	Min.	189.68 ms	174.81 ms
	Avg.	191.61 ms	174.99 ms
	Max.	193.71 ms	176.49 ms
Lower 01 (512 x 512 x 341)	Min.	176.35 ms	133.30 ms
	Avg.	177.40 ms	134.48 ms
	Max.	179.00 ms	135.40 ms

[표 2] CT 데이터 세트에 대한 렌더링 속도 최소값, 평균값, 최대값

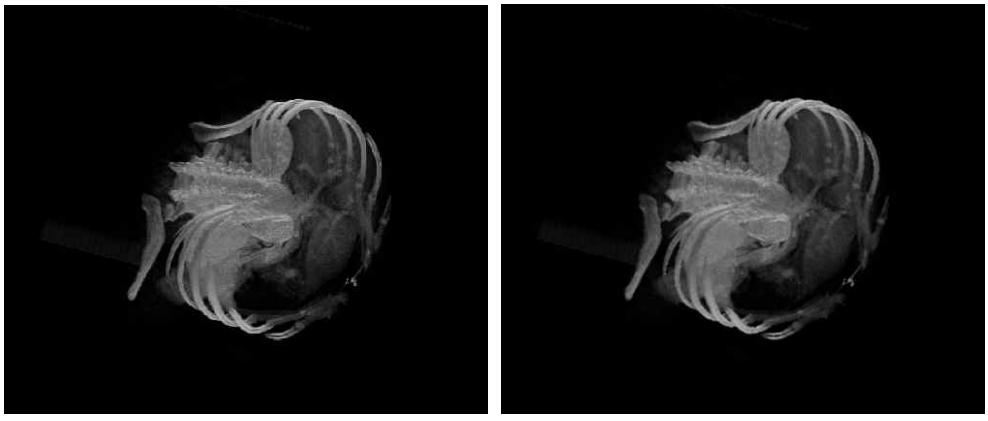
표 2는 CT 데이터 세트에 대한 렌더링 수행 시간의 수행 시간의 최소값, 평균값, 최대값이 나타나 있다. 제안 기법을 적용할 경우 기존 방식에 비해 수행 시간이 최대 35% 감소하였다.

[A] [B]



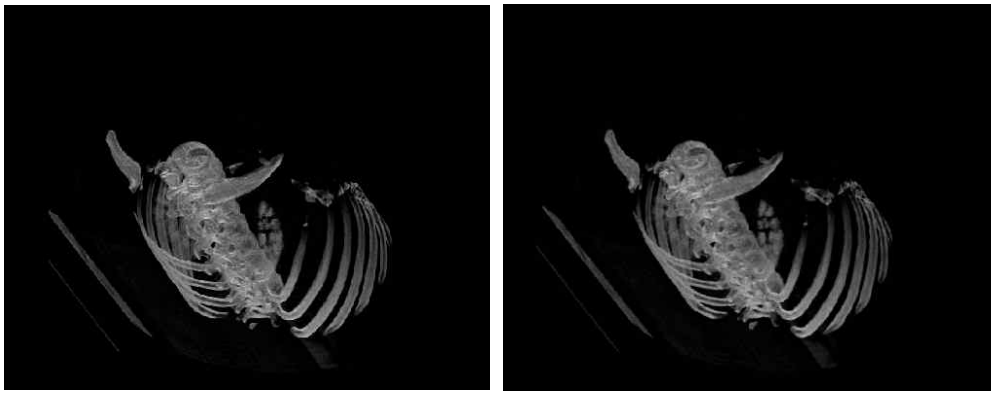
[그림 14] 머리 CT 데이터에 대한 렌더링 결과 이미지 비교

[A] [B]



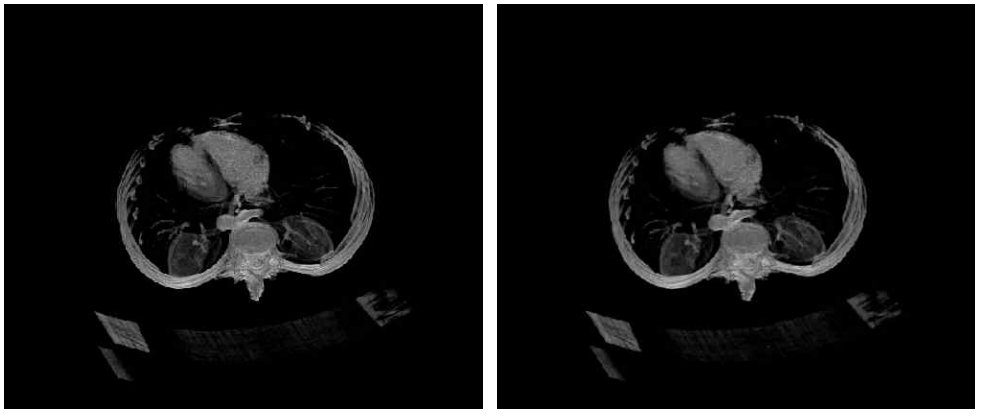
[그림 15] 복부 CT 데이터에 대한 렌더링 결과 이미지 비교

[A] [B]



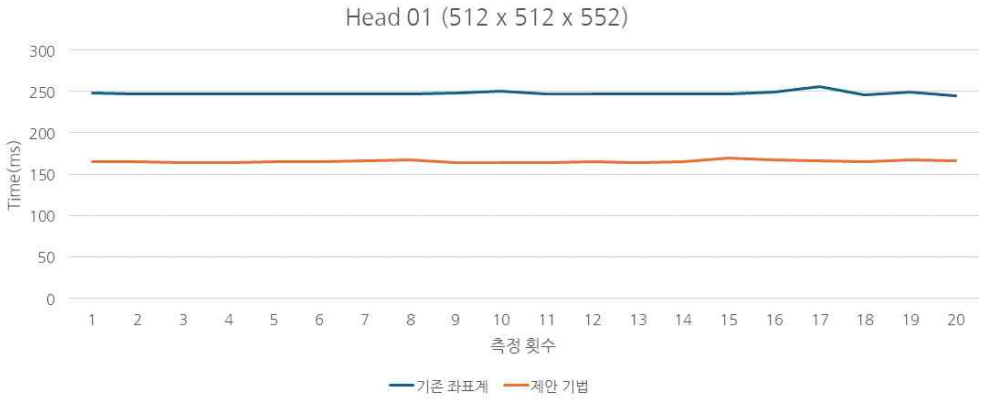
[그림 16] 허리 CT 데이터 1에 대한 렌더링 결과 이미지 비교

[A] [B]

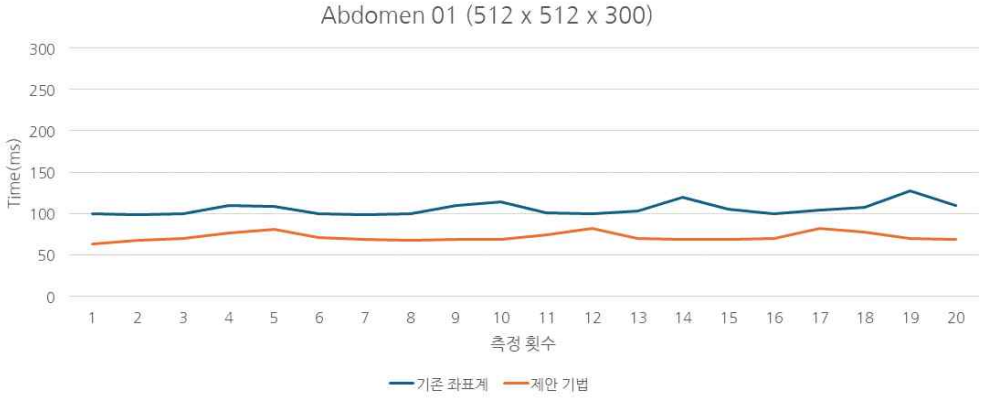


[그림 17] 허리 CT 데이터 2에 대한 렌더링 결과 이미지 비교

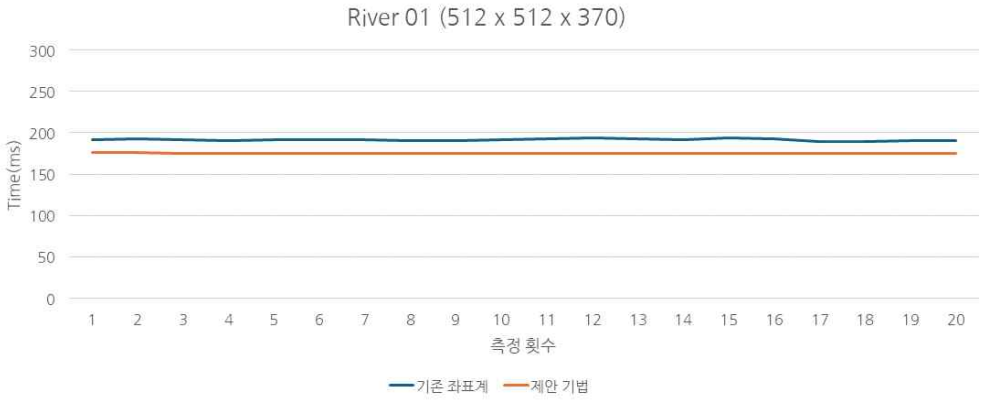
그림 14, 그림 15, 그림 16, 그림 17에서 A은 기존의 볼륨 렌더링 방식을, 그림 14-B는 본 논문의 제안 기법을 적용한 경우의 렌더링 결과를 나타내며, 두 경우의 시각적 차이는 나타나지 않았다.



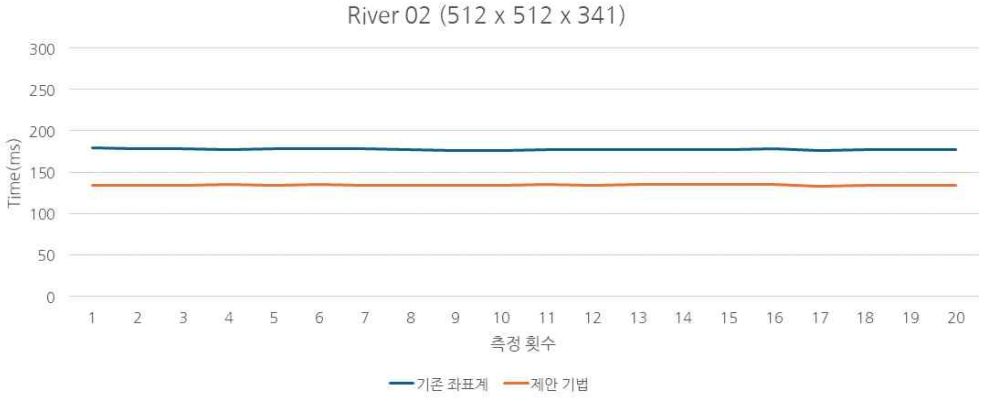
[표 3] 머리 CT 데이터에 대한 렌더링 속도 측정 결과



[표 4] 복부 CT 데이터에 대한 렌더링 속도 측정 결과



[표 5] 허리 CT 데이터 1에 대한 렌더링 속도 측정 결과



[표 6] 허리 CT 데이터 2에 대한 렌더링 속도 측정 결과

표 3, 표 4, 표 5, 표 6은 각각의 CT 데이터에 대해 20회 측정한 모든 결과를 보여주고 있다. 이는 제안된 방식이 볼륨 렌더링 성능 향상에 효과적임을 입증한다.

제 6 장 결론

본 연구에서는 고화질 볼륨 렌더링을 가속화하기 위해 비효율적인 메모리 접근 패턴을 최소화하는 스레드 동렬 투사 기법을 제안하였다. B-Spline 보간과 같은 고차 보간을 적용한 볼륨 렌더링은 빈번한 메모리 참조로 인한 성능 문제로 인해 대화적 속도를 달성하기 어렵다는 문제가 있고, 이를 해결하기 위해 다양한 가속화 알고리즘이나 GPU를 활용한 병렬 연산을 이용하는 연구가 이루어졌다. 그러나 샘플링 과정에서 스레드가 메모리 영역을 non-coalesced access 상태로 접근하는 비효율적인 메모리 참조 패턴은 여전히 존재했으므로, 이에 본 연구는 관찰 방향에 상관없이 스레드 인덱스와 데이터 인덱스를 적절하게 대응시키는 방법을 제시한다.

첫 번째로 스레드 동렬 투사 기법이 관찰 방향에 상관없이 최적의 효율을 낼 수 있도록 사전 데이터 정렬을 도입하였다. x, y, z축을 기준으로 정렬된 데이터를 사전에 생성하고, 이후 볼륨 렌더링 과정에서 주된 관찰 방향에 따라 적절한 데이터를 선택하여 최적의 메모리 접근 패턴을 유도한다.

두 번째로 관찰자 좌표계의 화면을 구성하는 벡터가 스레드 인덱스의 축과 대응한다는 점에서 착안해, 관찰자 좌표계의 변형을 통해 스레드 인덱스와 데이터 인덱스를 적절하게 유지한다. 관찰 방향 벡터는 유지한 채 화면을 이루는 두 벡터만 회전으로 변환하여 화면의 왜곡을 방지했다. 볼륨-광선 교차가 적용된 사례와 적용되지 않은 사례를 모두 제시하여, 볼륨-광선 교차 알고리즘의 적용 여부와 관계없이 렌더링 속도를 최적으로 가속화했다.

본 논문의 연구 결과는 메모리 참조 횟수가 많은 3차 B-Spline 보간법이 적용된 볼륨 렌더링을 수행하는 사례를 선보여, 스레드 동렬 투사 기법이 고화질 볼륨 렌더링의 가속화에 효과적임을 보여준다. 이 기법은 B-spline 보간법에만 국한되지 않고, 메모리 참조 횟수가 많은 다양한 볼륨 렌더링 알고리즘에 적용 가능한 확장성을 가진다. 의료 영상 분야에서 고품질의 볼륨 렌더

링을 실시간으로 수행하는 기술에 대한 수요를 고려했을 때, 본 연구 결과는 볼륨 렌더링 기술의 실용성과 효용성 증가에 기여한다. 본 논문에서는 3차 B-Spline 보간법이 적용된 볼륨 렌더링에 한해 연구하였으나, 향후에는 제시하는 스레드 동렬 투사 기법을 다양한 상황에 적용하여 연구를 진행한다면 실시간 고화질 볼륨 렌더링을 실현하는 데 기여할 것이다.

참 고 문 헌

1. 국내문헌

신용하, 계희원. (2016). B 스플라인 보간을 이용한 고화질 볼륨 가시화.『한국 컴퓨터 그래픽스 학회 논문지』,, 22(3), 1-9

2. 국외문헌

Aquino, J., & Russel, D. W. (1996). Employment status, social support, and life satisfaction among the elderly, Journal of counseling Psychology, 43(4), 480-490.

Drebin, Robert A., Loren Carpenter, and Pat Hanrahan. "Volume rendering." ACM Siggraph Computer Graphics 22.4 (1988): 65-74.

Engel, K., Hadwiger, M., Kniss, J. M., Rezk-Salama, C., & Weiskopf, D. (2004). Real-time volume graphics. ACM SIGGRAPH 2004 Course Notes.

Hadwiger, M., & Ljung, P., Rezk-Salama, C. and Ropinski, T. (2011). Advanced illumination techniques for GPU volume raycasting. ACM SIGGRAPH ASIA 2008 courses. 1, 1-166.

Han, J., & Sharma, B. (2011). CUDA optimization: Memory access patterns and texture fetching.

K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf, (2006). Real-Time Volume Graphics,

Lee, B., Yun, J., Seo, J., Shim, B., Shin, Y. G., & Kim, B. (2010). Fast high-quality volume ray casting with virtual samplings. IEEE Transactions on Visualization and Computer Graphics, 16(6), 1525-1532.

Levoy, M. (1990). Efficient ray tracing of volume data. ACM Transactions on Graphics (TOG), 9(3), 245-261.

Levoy, M. (1988). Display of surfaces from volume data. IEEE Computer graphics and Applications 8.3, 29-37.

Mihajlovic, Z., Budin, L., & Radej, J. (2004). Gradient of B-splines in volume rendering. In Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean, 1, 239-242.

Mihajlovic, Z., Budin, L., & Radej, J. (2004). Gradient of B-splines in

volume rendering. In Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean, 1, 239-242.

Mitchell, D. P., & Netravali, A. N. (1988). Reconstruction filters in computer-graphics. ACM Siggraph Computer Graphics, 22(4), 221-228.

Yuki Sugimoto, Fumihiko Ino, Kenichi Hagihara. (2014). Improving cache locality for GPU-based volume rendering. Parallel Computing 40. 5-6. 59-69.

Rössl, C., Zeilfelder, F., & Nürnberger, G. (2004). Spline approximation of general volumetric data. In Proceedings of the Symposium on Solid Modeling and Applications. 277-282.

Ruijters, D., & Thévenaz, P. (2010). GPU prefilter for accurate cubic B-spline interpolation. The Computer Journal, 55(1), 15-20.

Ruijters, Daniel, Bart M. ter Haar Romeny, and Paul Suetens. (2008). Efficient GPU-based texture interpolation using uniform B-splines. Journal of Graphics Tools 13.4 61-69.

Schoenberg, I. J. (1946). Contributions to the problem of approximation of equidistant data by analytic functions. Quarterly of Applied Mathematics, 4(1), 45-99.

Sigg, C. & Hadwiger, M., (2005)., Fast third-order texture filtering. GPU gems, 2(22), 313-329

Thévenaz, P., Blu, T., & Unser, M. (2000). Interpolation revisited. IEEE Transactions on Medical Imaging, 19(7), 739-758.

Shi, Z., Yi, C. J., & Hua, X. C. (2014). Algorithm of ray casting volume rendering based on CUDA. In The 2nd International Conference on Industrial Application Engineering.

Sugimoto, Y., Ino, F., & Hagihara, K. (2014). Improving cache locality for GPU-based volume rendering. Parallel Computing, 40(1), 28-40.

Unser, M. (2002). Splines: a perfect fit for medical imaging. Progress in Biomedical Optics and Imaging, 3(22), 225-236.

Wang, J., Yang, F., & Cao, Y. (2017). A cache-friendly sampling strategy for texture-based volume rendering on GPU. Visual Informatics, 1(1), 26-34.
Zhang, C., Xi, P., & Zhang, C. (2011). CUDA-based volume ray-casting using cubic B-spline. In 2011 International Conference on Virtual Reality and Visualization. 84-88.

3. 기타 자료

CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/ (accessed May 20, 2024)

ABSTRACT

GPU Thread Alignment Projection Techniques for Accelerating High-Definition Volume Visualization

Kim, Na-Yeon

Major in Computer Engineering

Dept. of Computer Engineering

The Graduate School

Hansung University

Volume rendering in medical image analysis should be performed in high definition to observe detailed structures such as microscopic blood vessels and small tumors. The main technique for this is to improve the interpolation method through higher-order interpolation. However, volume rendering with higher-order interpolation method is limited in performance to perform at a conversational speed due to the large number of access to memory. To overcome this, various optimization methods such as using GPUs have been studied, but performance degradation problems due to inefficient memory access still exist. This problem is more pronounced in higher-order interpolation volume rendering with a high number of memory accesses.

In this paper, we propose a thread isomorphism projection technique

to solve the inefficient memory access problem. The thread isomorphism projection technique improves the performance of volume rendering by aligning volume data and sequentially responding to threads and data. The technique presented in this paper generates three aligned volume data in advance, taking into account all observation directions, and induces the reference of appropriate data according to the main observation direction. In addition, by rotating and transforming the observer coordinate system while maintaining the observation position and direction, threads and data are sequentially matched, resulting in acceleration without distortion of the screen or deterioration of image quality. As a result, it shows that volume rendering with third-order B-Spline interpolation can be performed at a conversational speed through the proposed technique. This enables more accurate and faster diagnosis in the field of medical imaging, increasing the practicality and effectiveness of volume rendering technology.

【Key words】 Volume Rendering, High-Definition Rendering, Cubic B-Spline Interpolation, GPU Memory Access Pattern, Viewing Coordinate System