Journal of Korea Multimedia Society Vol. 12, No. 4, April 2009(pp. 512-520)

단일 명령 복수 데이터 연산과 순차적 메모리 참조를 이용한 효율적인 최대 휘소 투영 볼륨 가시화

계 희 원†

요 약

최대 휘소 투영(MIP) 볼륨 가시화는 의료기기 등에서 생성된 삼차원 영상 데이터로부터 관찰자가 바라보는 방향으로 최대값을 추출하여 영상을 생성하는 가시화 기법이다. MIP는 조영된 혈관 같은 높은 밀도의 구조를 가려짐 없이 드러내어 의료 영상 등에서 많이 사용된다. 본 연구는 두 단계의 가속화 방법을 제안하여 상용 CPU에서 고속으로 MIP를 수행할 수 있도록 한다. 먼저, 기존 MIP 알고리즘이 다수의 조건 분기 명령으로 구성된다는 것에 착안하여, 상용 CPU에서 제공하는 단일 명령 복수 데이터(single instruction multiple data: SIMD) 연산으로 조건 분기 명령을 제거한다. 많은 시간이 소요되는 조건 분기 명령을 제거하여 가시화 속도가 향상된다. 또한 본 연구는 메모리 참조가 순차적으로 발생하도록 알고리즘을 구성한다. 기존 가시화 방법에서 영상과 객체의 메모리 참조가 무작위로 발생하여 발생하던 속도 저하 문제를 완화시킨다. 두 가지 제안 방법을 통해 기존의 쉬어-왑 볼륨 가시화 기법에 비해 7배 이상의 성능 향상을 얻는다.

Efficient Maximum Intensity Projection using SIMD Instruction and Streaming Memory Transfer

Heewon Kye†

ABSTRACT

Maximum intensity projection (MIP) is a volume rendering method which extracts maximum values along the viewing direction through volume data. It visualizes high-density structures, such as angiographic datasets so that it is frequently used in medical imaging systems. We have proposed an efficient two-step MIP acceleration method that uses the recent CPUs. First, we exploited SIMD instructions to reduce conditional branch instructions which take up a considerable part of whole rendering process, so that we improved rendering speed. Second, we proposed a new method, which accesses volume and image data successively by modifying the shear-warp rendering. This method improves memory access patterns so that cache misses are reduced. Using the current CPUs, our method improved the rendering speed by a factor of 7 than that of the shear-warp rendering.

Key words: MIP volume rendering(최대 휘소 투영 렌더링), cache-efficient rendering(캐쉬 효율적 렌더링), SIMD instruction(SIMD 명령어), branch removal(분기 제거)

1. 서 론

최대 휘소 투영 볼륨 가시화(maximum intensity projection; MIP)는 조영된 혈관이나 골격계 구조와 같이 높은 밀도의 인체 조직을 관찰하기에 적합한 의료 영상 가시화 기법이다. 일반적으로, 의료기기 등에서 측정한 밀도 값의 공간 분포는 볼륨 데이터(volume data)라고 부르는 삼차원 배열 상에 저장되

※ 교신저자(Corresponding Author) : 계희원, 주소 : 서울 성북구 삼선동3가 389(136-792), 전화 : 02)760-8014, FAX : 02)760-4442, E-mail : kuei@hansung.ac.kr

접수일 : 2008년 11월 5일, 완료일 : 2009년 2월 13일
† 정회원, 한성대학교 정보시스템공학과 전임강사
※ 본 연구는 2008년도 한성대학교 교내연구비 지원과제 임

며, MIP는 주어진 관찰자의 시선 방향으로 볼륨 데이터의 밀도 값을 비교하고 최대 밀도 값을 추출해 내는 기법이다. MIP 결과 영상은 직접 볼륨 가시화(direct volume rendering)와는 달리, 깊이 정보가 손실되는 특징이 있기 때문에, 사용자는 관찰 방향을 수시로 변경하여 깊이를 추정하게 된다. 잦은 관찰 방향 변화에 신속하게 대응하기 위해, 빠른 속도로 MIP 영상을 생성하려는 많은 관심과 연구가 있었다.

MIP의 가속화 방법을 몇 가지 영역으로 나누어 보면, 우선, 가속에 도움이 되는 보조 자료를 전처리(pre-processing) 단계에서 생성하고 가시화할 때 그 추가 정보를 사용하는 방법이 있다. 삼차원 볼륨 데이터는 복셀(voxel)이라고 부르는 배열 원소로 구성되는데, 여덟 개의 인접 복셀로 입방체를 구성하거나 더 많은 인접 복셀로 고정된 크기의 블록을 구성하여 해당 영역의 최대값을 미리 계산해 둘 수 있다. 이 최대값으로 공간적 계층 구조(octree)를 만들면, 계층 구조에서 상위 영역(tree node)을 생략했을 때 화질에 영향이 있는지 판단하여 소속 하위 영역을 가시화 연산에서 건너뛸 수 있다[1,2].

또한 볼륨 데이터의 자료구조를 고속 가시화에 유리한 방법으로 재배치하는 방법도 많이 사용되었다. Mroz 등은 전처리 단계에서 가시화를 가속하기 쉬운 방법으로 데이터를 재배치하는 방법을 제안하였다[3]. 전체 데이터를 밀도 값에 대한 내림차순으로 정렬하고, 높은 값부터 가시화하면 영상에 반영되지 않는 낮은 값을 제거할 수 있다. 다만, 이 방법은 추가 메모리가 과도하게 필요하고 전처리에 많은 시간이 소요된다는 단점이 있다. 데이터 재배치는 캐시 메모리(cache memory)의 효율을 높이는 방향으로 할 수도 있다[4]. 이 또한 전처리 시간이 오래 걸리며, 경우에 따라 별도의 저장 공간이 필요하다는 단점이 있다.

마지막으로 3차원 그래픽스 가속기를 이용하여 고속 가시화를 수행하는 방법이 있다. 초기에는 각 복셀을 정점(vertex)으로 변환하고, 정점 기반 가시화를 수행하는 방식이 제안되었으나[5], 최근에는 OpenGL이나 DirectX 등에서 제공하는 텍스처 매핑 기법을 일반적으로 사용한다[6,7]. 또한 VolumePro™와 같은 볼륨 가시화 전용의 특수 장비를 사용한 연구도 있었다[8]. 상용 그래픽스 장치를 이용한 고속 가시화 방법이 현재 주요한 방법이나, 별도의 장비가 필요하다는 단점이 있다. 예를 들어, 노트북 컴퓨터나 내장형 그래픽스 하드웨어를 사용하는 컴퓨터의 경우, 그래픽스 장치는 볼륨 가시화를 수행하기 충분한 능력을 가지고 있지 않다. 본 논문에서 제안하는 방법은, 상용의 CPU에서 효율적인 메모리 참조와 연산을 통해 고속으로 MIP를 수행하는 것이다.

또한 본 연구는 캐시 메모리에서 필요한 데이터를 찾지 못하는 경우 시간 손실이 매우 크다는 것[9]에 착안한다. 예를 들어 한 번의 캐시 참조 실패는 수백 번의 연산에 해당하는 시간을 낭비하게 된다[10]. 본 논문은 볼륨 데이터를 순차적으로 읽어, 캐시 메모리의 공간 연관성(spatial coherency)을 증대시키는 방법을 제안한다. 본 논문은 데이터를 재배치하거나 블록 자료구조를 생성하는 등의 전처리가 전혀 없이, 알고리즘의 변경을 통해서 효율적인 가시화를 수행할 수 있다.

이후 논문의 순서는 다음과 같다. 먼저 2장에서 SIMD 연산을 이용한 분기 제거에 대해 설명한다. 3장에서는 쉬어-왑 분해 볼륨 가시화를 기반으로 하여 순차적 메모리 참조로 가시화 효율을 높이는 방법을 설명한다. 4장에서는 실험 결과를 보이고 5장에서 결론을 맺는다.

2. SIMD 연산을 이용한 분기 제거

일반적인 MIP 알고리즘은 샘플 위치를 찾고, 재표본화(resampling)를 수행한 다음, 현재까지 누적된 영상의 값과 재표본화 값을 비교하여 큰 값을 저장하는 순서로 구성되어있다. 최대값을 얻으려면 여러 샘플 값을 비교하여 가장 큰 값을 찾는 과정을 반복해야 한다. 볼륨 데이터는 삼차원 데이터이기 때문에 $O(n^3)$의 비교가 발생한다. 이 비교는 CPU상에서 조건 분기(conditional branch) 명령으로 해석되며, CPU의 분기 예측(branch prediction)이 빗나갈 경우 CPU의 파이프라인이 잠시 멈추게(stall) 되어 성능이 저하된다[9]. 게다가 최근의 CPU와 같이 파이프라인의 단계가 많을수록 성능 손실이 커진다.

한편, 최근에 사용하는 상용 CPU는 SIMD 명령어(instruction)를 기본으로 제공하고 있으며[10], 이 명령어는 다수(일반적인 CPU에서는 4개)의 데이터에 동일한 연산을 병렬적으로 수행시킨다 (그림 1(a)). 또한 이 명령어 집합에는 조건 분기 없이 최대값을 얻는 연산이 포함되어 있다. 그림 1(b)에서 볼 수 있

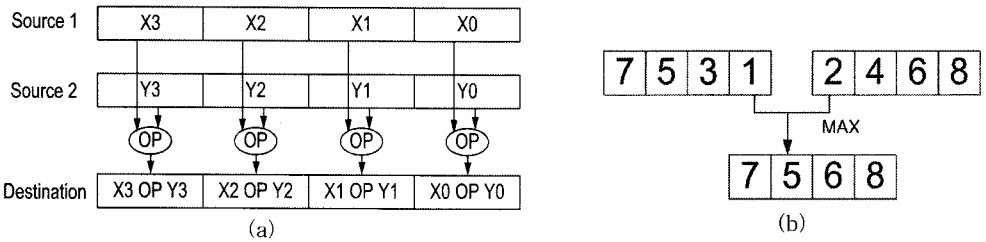

그림 1. SIMD 명령은 여러 데이터에 단일한 연산을 병렬적으로 수행할 수 있다. 특히 두 개의 값 중 큰 값을 병렬적으로 계산하는 명령이 포함되어 있어, MIP에 적용할 수 있다.

듯이 두 벡터의 성분 별로 더 큰 값을 단일 연산으로 단일 수행시간에 얻을 수 있다.

단, 이 연산은 벡터의 네 성분 중 가장 큰 성분을 얻는 연산이 아니고 각 성분 별 병렬 연산이라는 제한이 있다. 따라서 본 연구는 두 개의 병렬적인 데이터를 구성하고 두 병렬 데이터 사이의 누적에 SIMD 연산을 사용한다. 이 경우, 볼륨 데이터와 영상 데이터 공간을 동시에 움직이면서 가시화를 수행하는 쉬어-왑 볼륨 가시화 기법[11]이 적합하다(3장 참고). 이를 이용한 MIP 과정은 다음과 같이 설명할 수 있다.

먼저, 네 개의 인접한 볼륨 데이터 원소에 대해 대응되는 영상 좌표를 찾는다. 해당 볼륨 데이터 값과 영상 좌표의 화소(pixel) 밝기 값을 입력으로 SIMD 명령을 수행한다. 네 개의 화소와 볼륨 데이터 원소에 대해 더 큰 값들의 배열을 출력으로 얻게 된다. 이 출력을 영상 화소에 저장(갱신)하면, 네 개 볼륨 데이터 원소가 영상에 반영(투영)된 셈이 된다. 이 과정을 전체 볼륨 데이터에 대해 반복하면 MIP 처리가 종료된다.

SIMD를 이용한 방법은 각 볼륨 데이터 원소와 화소 값의 비교 연산이 제거될 뿐만 아니라, 연산이 병렬적으로 수행되므로 큰 성능 향상을 얻을 수 있다. 또한, 투영 영상의 좌표를 찾는 것은 이전에 처리한 좌표와의 상대적 거리를 이용하여 고속으로 처리된다.

SIMD 연산의 또 다른 제한은, 데이터가 물리 메모리 공간에서 연속적이어야 한다는 것이다. 즉, SIMD 레지스터(register) 단위로 명령을 수행할 수 있도록, 각 자료는 인접해서 존재해야 한다. 만약 메모리의 연속적이지 않은 자료들에 대해 SIMD 연산을 수행하려 한다면 SIMD 레지스터를 재정렬하는 비용을 감수해야 한다. 3장에서 이 재정렬 비용을 줄이는 방법에 대해서 자세히 설명한다 (3.2절 참고).

3. 순차적으로 메모리를 참조하는 MIP 볼륨 가시화

본 연구는 객체 기반 순서 방법으로, 소프트웨어 기반의 기법 중에 가장 빠른 방법으로 알려져 있는 쉬어-왑 분해 볼륨 가시화[11]를 기반으로 한다. 쉬어-왑 볼륨 가시화는 시각행렬을 전치 행렬, 기울임(shear) 행렬, 와핑(warping) 행렬로 분해하여 가속화 하는 방법이다. 전치 행렬과 기울임 행렬을 반영하면 투영연산의 좌표 변환이 간단해져서 표본 위치(sampling position)를 고속으로 계산할 수 있다. 그 결과로 빠르게 얻은 중간 영상 (intermediate image)[11]을 이차원 와핑하여 최종 영상을 생성하게 된다 (그림 2).

또한 본 연구는 재표본화 시간을 줄이기 위해 최근접 보간(nearest neighbor interpolation)을 사용한다. 본 연구의 목적은 빠른 시간에 적당한 화질의 영상을 생성하는 것으로서 최근접 보간으로 인한 화질 저하는 감수한다. 비록 최근접 보간이 영상에서 계단현상(aliasing)을 유발하지만, MIP 볼륨 가시화는 최대값을 추출하는 기법이므로, 선형 보간(linear interpolation)에 의한 평활화(blurring) 문제가 없기 때

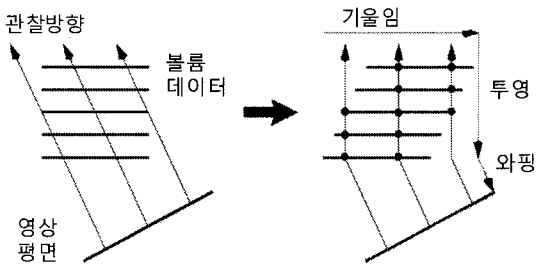

그림 2. 쉬어-왑 가시화[11]는 광선의 진행을 기울임(shear) 단계와 와핑(warping) 단계로 나누어 메모리 참조 연관성을 향상시키는 방법이다.

문에 구조물을 명확히 드러내는 장점도 있다. 선형 보간은 큰 값을 보간 과정에서 작게 만들어 최대값 추출에서 제외하는 문제가 일부 발생하기 때문이다. 그리고, 최근의 볼륨 데이터는 객체의 정밀도가 높아 최근접 보간으로 인한 화질 저하가 완화되는 효과가 있다. 따라서 최근접 보간 방법은, 잦은 시점 변화를 통해 물체의 개략적인 정보를 영상으로 생성하는 본 연구에 적합하다.

3.1 순차적인 메모리 접근을 이용하는 MIP

쉬어-왑 분해 볼륨 가시화는 그림 3(a), 그림 3(b), 그림 3(c)와 같이 관찰방향에 따라 세 가지 경우로 나누어 설명할 수 있다. 그림 3의 x, y, z축은 물체 좌표계를 의미한다. 한편, 사용자의 관찰 방향이 x, y, z축 중 어느 축과 가장 나란한지 계산할 수 있으며, 이 축을 주시각축(principal axis)이라고 한다. 그림 3(a), 그림 3(b), 그림 3(c)는 각각 주시각축이 z, y, x축인 경우를 보이고 있다.

볼륨 데이터가 삼차원 배열로 저장되어 있기 때문에, 일반성을 잃지 않고, 물리 메모리는 x축 방향으로 나열되어 있다고 가정할 수 있다. 볼륨 데이터의 물리 메모리 배열 방향을 그림에서 굵은 화살표로 나타내었다. 또한 i, j축은 중간 영상 좌표계를 의미하며, 이차원 배열의 경우, 물리 메모리는 i축 방향으로 나열되어 있다고 가정한다.

그림 3(a), 그림 3(b), 그림 3(c)에서 물체와 중간 영상의 물리적 메모리의 배열은 그림 3(b)와 그림 3(c)에서 평행하지 않다. 즉, 그림 3(b)과 그림 3(c)에서 x축과 i축이 평행하지 않은 것이다. 이때, 객체를 기준으로 데이터를 읽으면 중간 영상의 물리 메모리 참조가 순차적이지 못하며, 반대로 중간 영상을 기준으로 데이터를 읽으면 객체의 메모리 참조가 순차적이 되지 않는다. 메모리 참조가 순차적이 되지 않으면, 캐시 메모리의 효율이 저하되며, 2장에서 설명한 SIMD 연산을 사용할 수 없게 되는 문제가 있다. 비록, 전처리 과정에서 메모리 순서를 변경한 볼륨 데이터를 세 벌 만들어 놓으면 이 문제를 피할 수 있으나, 대용량 데이터를 처리하기 어렵다.

본 연구는 이 문제를 가시화 순서를 변경하여 해결한다. 먼저 주시각축이 z축인 그림 3(a)의 경우 특별한 처리 없이도 물체와 중간 영상 모두에서 순차적 메모리 접근을 수행한다. 따라서, 변경 없이 그림 3(d)과 같이 수행한다. 다음, 주시각축이 y축인 그림 3(b)의 경우 중간 영상의 원점을 중심으로 90°의 회전 변환을 수행한다[12,13]. 이렇게 되면 그림 3(e)와 같이 x축과 i축이 평행하게 되므로 순차적 메모리 접근이 가능하다. 마지막으로 주시각축이 x축인 그림 3(c)과 같은 경우, 본 연구는 새로운 방법을 제안

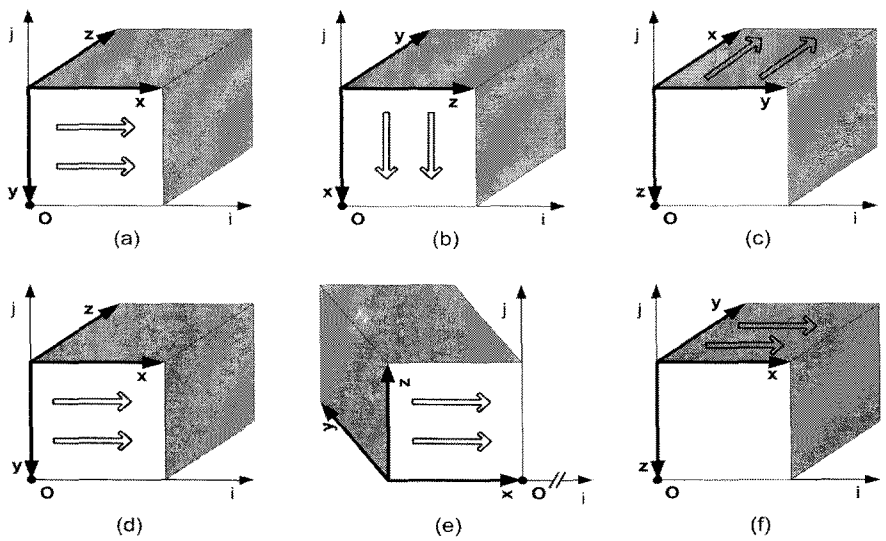

그림 3. 제안 방법은 관찰자의 위치에 따라 (a), (b), (c)와 같은 3가지의 상태로 구분하고, 각 상태에 대해 i축과 x축이 평행하도록 좌표변환을 선택적으로 수행한다. 그 결과로 3가지 상태는 순서대로 (d), (e), (f)의 상태로 변환된다. 변환된 상태에서 i축과 x축이 평행한 것을 확인할 수 있다.

한다.

만약, x축이 i축과 평행한 조건만 만족시키려면, 중간 영상의 원점을 중심으로 적당한 각도의 회전을 수행하면 된다[14]. 그러나 회전으로는 길이가 달라지게 되는 문제가 생긴다. 예를 들어 두 점(0, 0, 0)과 (1, 0, 0) 사이의 거리는 물체 좌표계에서 1이다. 그러나 쉬어 연산과 투영을 수행한 후의 거리는, 기울임 비율인 s_x와 s_y ($0 \leq s_x, s_y \leq 1$)에 의존하는, $\sqrt{s_x^2+s_y^2}$이 되어 일반적으로 길이가 달라진다[11]. 즉, 물체에서 얻은 데이터의 크기를 늘이거나 줄이는 재표본화를 통해 영상에 누적해야 하는데, 그림 3(a)와 같이 물체 데이터와 영상 데이터의 좌표 변환이 1 대 1 대응이 성립하지 않으면 SIMD 연산을 적용 할 수 없게 된다. 따라서 본 연구는, 주시각축이 x축인 경우 xy평면에 평행한 각각의 이차원 영상에 전치(transpose) 연산을 적용하여 x축과 y축을 서로 바꾼다. 그러면 그림 3(f)와 같이 x축과 i축이 평행하게 된다. 또한 그 길이도 동일하게 보존되는 효과가 있어 SIMD 연산을 적용할 수 있다. 이렇게 각 주시각축에 대해 정리된 알고리즘을 그림 4에 나타내었다.

3.2 효율적인 전치연산 사용

대용량 볼륨을 가시화할 경우 3.1절에서 설명한 전치 연산은 성능을 크게 떨어뜨린다. 고해상도 이차원 영상에 데이터를 쓰고 읽는 것은 많은 캐시 참조 실패(cache miss)를 유발하기 때문이다. 이런 단점을 보완하기 위해 본 연구는 전치연산을 작은 단위로 나누어 개별적으로 전치연산을 수행하는 방법을 사용한다. 이러한 방식은 반복 블록화(loop blocking)이라는 방식으로 잘 알려져 있는 기법이다[10]. 일반적인 반복 블록화는 작업 집합(working set)을 캐시 메모리 내부로 제한하는 역할을 한다. 그러나 제안 방법은 전치연산을 가시화 알고리즘 속으로 포함시켜 레지스터 단위로 전치연산이 수행되도록 한다. 즉, 오랜 시간이 걸리는 전치연산을, 캐시 메모리보다 더욱 고속인 레지스터에서 수행되도록 한다.

제안 방법은 4개의 볼륨 주사선(scanline)을 읽어 4×4 복셀 크기의 조각(tile)을 생성한다. 그리고 각 조각에 대한 전치연산을 수행한다(그림 5). 최근의 CPU는 최소 8개의 SIMD 레지스터를 가지고 있기 때문에 그림 5의 프로그램은 레지스터 내부에서 동작할 수 있다. 따라서 주 메모리에 쓰고 읽는 시간 낭비가 사라지고 전체 성능이 향상된다. 주시각축이 x축인 경우 수정된 알고리즘을 그림 6에 보였다.

그림 6에서 좌표계 변환을 수행하는 *Get image buffer* 문장은 매 주사선마다 한 번만 좌표 변환을 계산하면 그 결과를 재사용할 수 있어 계산 증가로

주시각축	알고리즘

z축	for each z
	  for each y
	  volume_stream ← Get slice buffer (y, z)
	  image_stream ← Get image buffer (y, z)
	  image_stream ← Max (image_stream, volume_stream)

y축	for each y
	  for each z
	  volume_stream ← Get slice buffer (y, z)
	  image_stream ← Get image buffer (y, z)
	  image_stream ← Max (image_stream, volume_stream)

x축	for each z
	  t_stream ← Transpose slice (z) { transposed stream }
	  for each y
	  volume_stream ← Get slice buffer (t_stream, y)
	  image_stream ← Get image buffer (y, z)
	  image_stream ← Max (image_stream, volume_stream)

그림 4. 관찰 방향에 따른 수정 알고리즘. 볼륨과 영상을 순차적으로 읽어 SIMD 연산을 적용할 수 있다. 각 알고리즘에서 Max 연산은 그림 2의 SIMD 명령을 이용하여 가속화 할 수 있다. 단, 주시각축이 x일 때, Transpose slice(z) 명령의 수행은 오랜 시간이 걸린다.

```
Transpose(row0,row1,row2,row3)
{
 __m64 tmp3, tmp2, tmp1, tmp0;
 tmp0=_mm_unpackhi_pi16(row0,row1);
 tmp2=_mm_unpacklo_pi16(row0,row1);
 tmp1=_mm_unpackhi_pi16(row2,row3);
 tmp3=_mm_unpacklo_pi16(row2,row3);
 row0=_mm_unpacklo_pi32(tmp2,tmp3);
 row1=_mm_unpackhi_pi32(tmp2,tmp3);
 row2=_mm_unpacklo_pi32(tmp0,tmp1);
 row3=_mm_unpackhi_pi32(tmp0,tmp1);
}
```

그림 5. SIMD 레지스터를 사용하는 4 × 4 원소의 전치연산을 내재 함수(intrinsic function)를 이용한 구현. 전치연산은 8개의 SIMD 연산으로 이루어지며 레지스터 상에서 작동하므로 메모리 참조 비용 없이 고속으로 동작할 수 있다.

```
for each z
 for each y mod 4 = 0
   volume_stream[y..y+3] ← Get slice buffer(y..y+3, z)
   for each x mod 4 = 0
     sample_stream ← Transpose(volume_stream
                                [y..y+3][x..x+3])
   image_stream ← Get image buffer(y, z)
   image_stream ← Max
                  (image_stream, sample_stream)
```

그림 6. 수정된 알고리즘. 가시화 코드가 전치연산을 포함하고 있다.

인한 속도 저하를 최소화할 수 있다.

정리하면, 제안 알고리즘은 전치연산을 통해 볼륨 주사선과 중간 영상 주사선 간에 1 대 1 대응을 만들게 된다. 따라서 제안 방법은 SIMD 연산을 사용하여 조건 분기 명령을 제거하고, 메모리 공간을 순차적으로 검색하여 성능을 향상시킨다. 효율적 전치 연산이 추가된 전체 알고리즘 구성을 그림 7에 보이고 있다. 관찰자의 관찰 방향에 따라 주시각축을 그림 3(a), 그림 3(b), 그림 3(c)와 같이 3가지 경우로 나눈다. 이후, 주시각축이 y축 또는 z축인 경우 회전 연산을 통해 변환 행렬을 구하며, x축인 경우 3.2절의 고속 전치연산을 이용한다. 이를 통해 그림 3(d), 그림 3(e), 그림 3(f)와 같이 객체 좌표와 영상좌표가 평행하게 되어, 투영 과정에서 고속의 SIMD 연산을 사용할 수 있게 된다. 이후 이차원 와핑 변환을 수행하면 최종 출력 영상이 만들어진다.

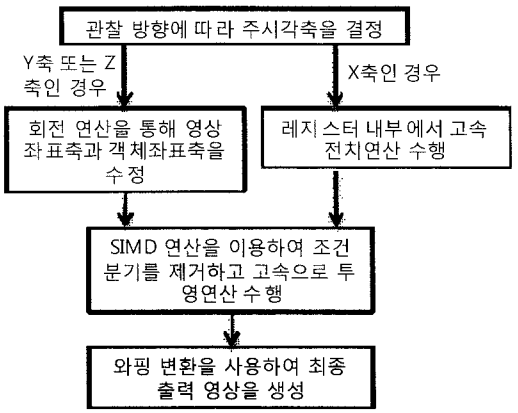

그림 7. 본 논문에서 제안한 알고리즘의 흐름도

4. 실 험

이번 장에서는 제안 방법을 사용하여 가시화 속도의 향상 정도를 확인하고, 결과 영상을 보인다. 모든 실험은 Intel Core2Duo™ 2.13GHz의 상용 CPU를 장착한 개인용 컴퓨터에서 수행하였다. CPU의 캐시 메모리 크기는 4MB이고 주 메모리의 크기는 2GB이다. 구현은 Microsoft의 Visual studio 2005에서 C++을 이용하였으며 다중 코어를 사용하기 위한 다중 쓰레드 프로그래밍은 적용되지 않았다. 실험에 사용한 데이터를 표 1에 정리하였다. 각 데이터는 인체의 컴퓨터 단층 촬영 데이터이며 각 복셀은 2바이트를 차지한다.

제안 알고리즘의 성능을 비교하기 위해 가시화에 걸린 시간을 측정하여 표 2에 보였다. 실험은 세 가지 알고리즘을 비교하였는데, 먼저 기존의 쉬어-왑 분해 볼륨 가시화[11]의 가시화 속도를 측정하였고, 다음으로 본 논문에서 제안한 SIMD 연산과 순차적 메모리 접근을 사용한 방식(표 2의 SIMD)의 가시화 속도를 측정하였다. 마지막으로 3.2절에서 제안한 최적화도 포함된 최종 알고리즘(표 2의 모든 최적화)의 가시화 속도를 측정하여 비교하였다. 실험

표 1. 실험에 사용한 볼륨 데이터

데이터 이름	크기	용량(MB)
Head	256 × 256 × 225	28.1
Brain	512 × 512 × 185	92.5
Aorta	512 × 512 × 310	155
Body	512 × 512 × 512	256

표 2. 각 실험 데이터에 대한 제안 방법의 성능 향상 결과 (시간단위: ms)

	쉬어-왑	SIMD	모든 최적화
Head	174.96	36.62	23.91
Brain	577.44	114.91	72.99
Aorta	961.26	192.01	121.15
Body	1587.33	318.51	205.48

은 각 데이터를 회전시키면서 측정한 가시화 시간의 평균값이다.

제안 방법에 의해 가시화 속도가 크게 향상된 것을 확인할 수 있다. 기존의 쉬어-왑 알고리즘은 Head 데이터의 경우 174.96ms의 시간이 걸리고 Aorta나 Body와 같이 300장을 초과하는 큰 데이터의 경우 오랜 시간이 소요되어 실용적이지 않다. 반면, SIMD 연산을 이용한 순차적 가시화에 의해 속도가 5배 정도로 크게 향상된 것을 확인할 수 있다. 그리고 성능 향상 폭도 데이터의 종류에 무관하게 일정한 것을 알 수 있다. 제안 알고리즘은 데이터 종속적인 가속 알고리즘 없이 SIMD 연산과 메모리 참조의 개선만으로 속도 향상을 이루었기 때문이다. 다만, 크기가 작은 Head 데이터의 경우 성능 향상이 4.78배 (174.96ms / 36.62ms)로서 다른 데이터에 비해 향상 폭이 약간 낮다. 작은 데이터의 경우 메모리 접근 방법이 나빠도 작업 집합이 작아지는 효과가 있어 캐시 메모리의 효율이 약간 향상되기 때문이다.

한편, 3.2절에서 제안한 전치연산의 가속화를 포함하면, 제안 방법은 기존 가시화 방법에 비해 7배 이상의 성능 향상을 얻는다. 특히 512^3의 큰 용량을 가진 Body데이터도 5fps (1000ms / 205.48ms)의 대화적 속도로 가시화 할 수 있으며 310장의 크기의 데이터도 8fps (1000ms / 121.15ms)의 속도를 보인다.

다음은 2장에서 설명한 것과 같이, SIMD 연산을 이용하여 조건 분기 명령을 제거하였을 때의 효과를 알아보기 위해 실험하였다(표 3 참고). 데이터는 Aorta를 이용하였고, 메모리의 참조의 효과를 무시하기 위해, 주시각축이 z축일 때로 한정하여 가시화

표 3. 분기 제거로 인한 가속화 효과 (시간단위: ms)

방법	시간
기존 쉬어-왑	353.53
분기 제거	109.25

표 4. 효율적 전치연산을 이용한 성능 향상 (시간단위: ms)

알고리즘	주시각축 y축	주시각축 x축
SIMD	113.32	426.65
모든 최적화	113.32	146.94

시간을 측정하였다. 주시각축이 z축인 경우, 기존 쉬어-왑 가시화도 메모리 참조가 효율적이 되기 때문이다. 실험 결과는 3배 이상(353.53ms / 109.25ms) 빨라진 것으로 측정되었으며, 분기명령을 제거하여 큰 성능 향상이 있었음을 확인할 수 있다.

3.2절에서 제안한 최적화의 효과를 보이기 위한 실험 결과를 표 4에서 보이고 있다. 데이터는 Aorta를 이용하였고, 주시각축이 y축인 경우의 속도와 x축인 경우의 속도를 비교하였다. 전치연산은 주시각축이 x축일 때만 적용되므로 주시각축이 y축일 때 가시화 시간은 113.32ms로 변함이 없다. 주시각축이 x축인 경우 전치연산의 가속화가 없으면 가시화 시간이 426.65ms로 3배 이상 속도 저하가 생기는 것을 알 수 있다. 그리고 효율적인 전치연산을 통해 그 시간이 146.97ms로 감소하였다. 비록 주시각축이 y축인 경우의 113.32ms에 비해서 30% 정도의 가시화 시간이 증가하지만 기존 방식에 비해 대폭 향상된 성능을 나타낸다.

마지막으로 그림 8에 가시화 결과 영상을 Head, Brain, Aorta, Body 순서대로 나타내었다. 각 영상은 조영된 혈관과 골격계를 가려짐 없이 드러내고 있어서 MIP의 결과를 충실하게 표현하고 있다. 참고로, Head를 제외한 데이터는 CT의 영상 간격이 화소 간격보다 길기 때문에 이차원 와핑 연산에서 실제 인체 비율에 맞춰 상하 길이가 보정된다.

5. 결 론

본 연구는 두 단계의 가속화 기법을 제안하여, 이를 통해 최대 휘소 투영(MIP)의 가속화를 수행하였다. 먼저 MIP의 빈번한 연산인 조건 분기 명령을 SIMD 연산을 이용하여 제거하였다. 이 과정에서 SIMD 연산을 사용하기 위한 조건인 순차적 메모리 접근, 볼륨 데이터와 영상 데이터간의 1 대 1 대응 조건을 만족시키기 위해, 가시화 순서를 변경하는 기법을 제안하였다. 조건 분기 명령의 수행 속도가 느리고 사용 횟수가 많기 때문에, 제안 방법을 통해 기

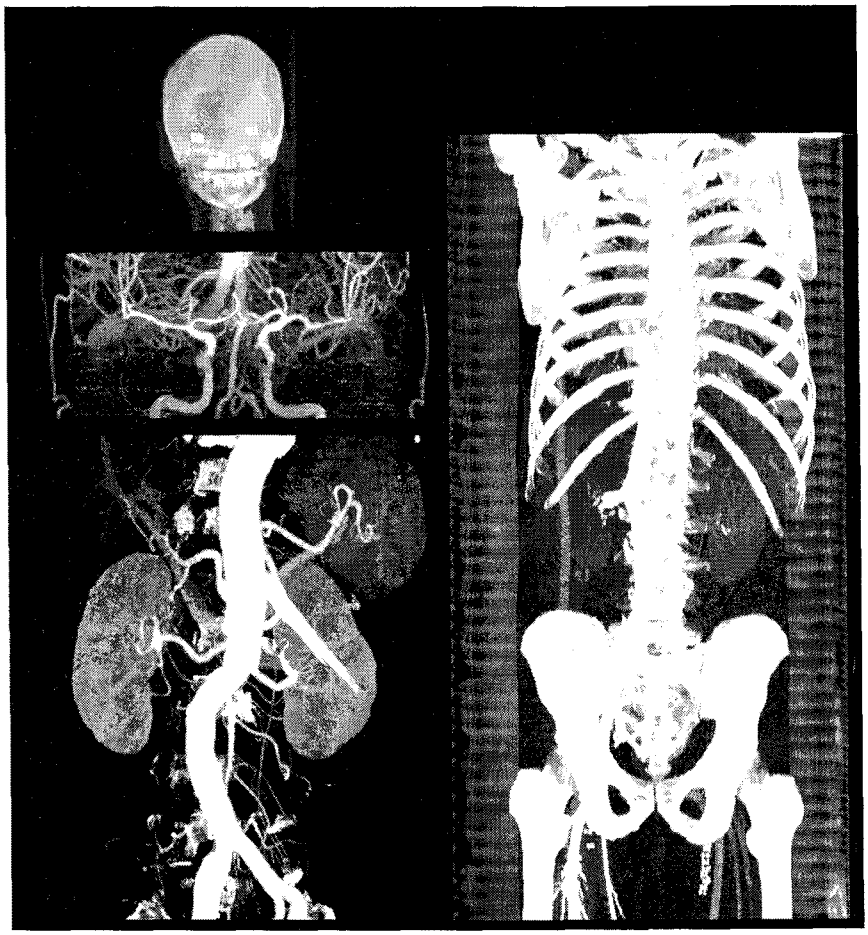

그림 8. 가시화 결과 영상. 왼쪽 위부터 아래 방향으로 각각 Head, Brain, Aorta, Body 데이터이다.

존의 방법에 비해 5배 정도의 속도향상을 얻을 수 있었다. 또한 제안 알고리즘에서 발생하는 전치 연산의 메모리 읽고 쓰기 오버헤드를 줄이기 위해, 작업 집합을 줄여 전치연산을 수행하는 방법도 제안하였다. 두 가지 방법을 모두 포함한 전체적 속도는 기존 방법에 비해 7배 이상의 향상을 얻었다. 그 결과로 일반적으로 사용하는 512 × 512 × 200 정도의 CT 데이터는 실시간으로 영상을 관찰할 수 있었으며 512^3 크기의 대형 볼륨 데이터에 대해서도 5fps의 대화적인 속도로 가시화를 수행할 수 있었다. 본 알고리즘은 일체의 전처리 알고리즘을 요구하지 않을 뿐만 아니라 빈 공간 탐색이나 조기광선종료와 같은 데이터 종속적인 소프트웨어적 가속 기법 없이도 고속으로 영상을 생성할 수 있다는 장점이 있어서 의료 영상 프로그램에 폭넓게 적용 가능하다.

참 고 문 헌

[1] G. Sakas, M. Grimm, and A. Savopoulos, "Optimized Maximum Intensity Projection (MIP)," *Proceedings 6th Eurographics Workshop on Rendering*, pp. 51-63, 1995.

[2] V. Pekar, D. Hempel, G. Kiefer, M. Busch, and J. Weese, "Efficient Visualization of Large Medical Image Datasets on Standard PC Hardware," *Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization*, pp. 135-140, 2003.

[3] L. Mroz, H. Hauser, and E. Groller, "Interactive High-Quality Maximum Intensity Projection," *Computer Graphics*

Forum, Vol.19, pp. 341-350, 2000.

[4] G. Knittel, "The Ultravis System," *IEEE/ ACM SIGGRAPH Volume visualization and graphics symposium*, pp. 71-78, 2000.

[5] W. Heidrich, M. McCool, and J. Stevens, "Interactive maximum projection volume rendering," *Proceedings of Visualization*, pp. 11-18, 1995.

[6] B. Cabral, N. Cam, and J. Foran, "Accelerated volume rendering and tomographic reconstruction using texture mapping hardware," *Symp. on Volume Visualization'94*, pp. 91-8, 1994.

[7] M.-Y. Chan, H. Qu, K.-K. Chung, W.-H. Mak, and Y. Wu, "Relation-Aware Volume Exploration Pipeline," *IEEE Transactions on Visualization and Computer Graphics* , Vol.14, No.6, Nov.-Dec. 2008.

[8] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The volume pro real-time ray-casting system," *Proceedings of ACM SIGGRAPH'99*, pp. 251-260, 1999.

[9] JL. Hennessy and DA. Patterson, "Computer Architecture: A Quantitative Approach," 3rd ed. San Mateo, CA: Morgan Kaufmann; 2003

[10] Intel, "Intel64 and IA-32 Architectures Software Developer's Manuals," Intel Corporation 2008. Available at http://www.intel.com/products/processor/manuals/

[11] P. Lacroute and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," *Proceedings of SIGGRAPH94*, pp. 451-458, 1994.

[12] CH. Lee, YM. Koo, and YG. Shin, "Template-Based Rendering of Run-Length Encoded Volumes," *J. of Visualization and Computer Animation*, Vol.9, pp. 145-161, 1998.

[13] J. Sweeney and K. Mueller, "Shear-Warp deluxe: the Shear-Warp algorithm revisited," *Proceedings of the symposium on Data Visualization*, Vol.22, pp. 95-104, 2002.

[14] H. Kye, B. Shin, Y. Shin, and H. Hong, "Shear- Rotation-Warp Volume Rendering," *Computer Animation and Virtual Worlds*, Vol.6, pp. 547-557, 2005.

계 회 원

1999년 2월 서울대학교 전산과학과 학사

2001년 2월 서울대학교 전기컴퓨터공학부 석사

2005년 8월 서울대학교 전기컴퓨터공학부 박사

2006년 1월~2007년 3월 서울대학교 컴퓨터연구소 연구원

2007년 9월~현재 한성대학교 정보시스템공학과 전임강사

관심분야 : 볼륨 가시화, 실시간 렌더링, 대용량 영상처리