J. Biomed. Eng. Res.
Vol 21, No 5, 519-526, 2000

TMS320C6201에 적합하게 보정된 쉬어-웝 볼륨렌더링 구현

최석원 · 권민정 · 박현욱

한국과학기술원 전기 및 전자공학과

(2000년 2월 10일 접수, 2000년 9월 4일 채택)

Implementation of Modified Shear-warp Volume Rendering in TMS320C6201

S.W. Choi, M.J. Kwon, and H.W. Park

Department of Electrical Engineering, Korea Advanced Institute of Science and Technology

(Received February 10, 2000. Accepted September 4, 2000)

요 약 : 볼륨 렌더링은 3D 의료영상 데이터를 가시화하는 중요한 기법 중 하나이다. 그러나 볼륨 렌더링을 실시간으로 이를 때, 많은 계산량을 필요로 하는 것이 볼륨 렌더링을 사용하는 데 걸림돌이 되고 있다. 이 논문에서는 Superscalar와 VLIW(Very Long Instruction Word)의 구조를 가지고 있어 동시에 8개의 명령어 수행이 가능한 TI사의 TMS320C6201 DSP를 이용하여 3D 초음파 영상의 쉬어-웝 볼륨 렌더링을 구현하였다. 쉬어-웝 방법을 DSP 상에서 최적으로 구현하기 위하여 ray map 방법, one-to-four ray casting, voxel skipping 방법을 제안하였다. 제안한 방법들을 이용한 볼륨 렌더링과 적용하지 않은 기존의 알고리즘을 DSP에 구현하여 PSNR과 렌더링 시간의 비교 평가를 통해 만족할 만한 영상 화질에 빠른 렌더링 성능을 얻을 수 있음을 보여주었다.

Abstract : Volume rendering is a key technique in visualization of 3D medical data. However we need a heavy and complex computational power for volume rendering, and it results in having difficulty in achieving real-time volume rendering.

In this thesis, the shear-warp volume rendering for 3D ultrasound image is implemented in the Texas Instruments TMS320C6201, which has a superscalar and VLIW(Very Long Instruction Word) architecture capable of issuing eight operations in parallel. We propose ray map method, one-to-four ray casting, and voxel skipping method to implement shear-warp factorization optimized to DSP. We compared the proposed method implemented on DSP with the conventional through PSNR and rendering time, and we showed the allowable image quality and good rendering performance.

Key words : 3차원 가시화(visualization), 볼륨 렌더링(volume rendering), TMS320C6201, ray map, one-to-four ray casting, voxel skipping

서 론

볼륨 렌더링(volume rendering)은 전산화 단층촬영기(CT), 핵자기 공명 영상장치(MRI)와 같은 진단 장치에서 얻어진 여러 장의 2차원 단면 영상들을 이용하여 입체감 있는 영상을 표현하는 가시화 기법이다[1]. 이를 통해 인체를 비침습적인 방법으로 관찰하고 그 정보를 다른 사람들에게 보다 손쉽게 전달할 수 있다. 그러나 볼륨 렌더링은 많은 계산량을 필요로 하기 때문에 실시간으로 영상을 표현하는 데 제한이 있다. 이를 극복하기 위해 여러 가지 시도가 이루어져 왔다. 그 중 하나가 알고리즘의 개선을 통해 계산량을 줄이고 보다 빠른 렌더링을 가능하게 하려는 시도이고, 다른 하나는 하드웨어를 이용하여 볼륨 렌더링을 가속시키고자 하는 시도였다.

많은 알고리즘들이 볼륨 렌더링을 가속시키기 위해 제안되어 왔다. 3차원 데이터로부터 ray가 진행하여 최종 영상을 구성하는 object-order 방식[2]과 최종 영상의 각 pixel로부터 ray가 진행하여 진행하는 모든 볼륨 데이터에 대해서 고려해 주는 image-order 방식[3]이 대표적이다. object-order 방식은 volume에 대해 효율적인 memory access가 가능하고 각 slice들이 점차적으로 영상을 성장시키므로 data resampling 과정이 불필요하다는 장점이 있지만, early ray termination이 불

통신저자 : 박현욱, (305-701) 대전광역시 유성구 구성동 373-1 한국과학기술원 전자전산학과
Tel. +82-42-869-8066, Fax. +82-42-869-8066
E-mail. hwpark@athena.kaist.ac.kr

가능하고 적절한 filter 계산이 요구되어 계산량이 많은 단점이 있다. 반면 image-order 방식은 효율적이고 고화질의 영상을 제공하며 early-ray termination을 통해 계산량의 부담을 줄일 수 있는 장점이 있다. 그러나 ray가 진행할 때 얻어지는 sampling point가 원래 주어진 3차원 데이터와 일치하지 않아 3차원 보간을 통해 sampling 값을 구해야 하므로 계산량의 부담이 크다. 이 두 알고리즘 외에 sampling 과정에서 생기는 계산량을 줄이기 위해 도착할 평면의 각 pixel에 해당하는 시선의 간격을 조절하여 각 시선과 각 슬라이스가 만나는 위치를 규칙적으로 만들어 2차원 보간만으로 중간 영상을 만들고, 시선의 간격을 조절함으로써 발생하는 영상의 왜곡을 워핑을 통해 실제의 입체감 있는 영상으로 만들어 주는 쉬어-웜(shear-warp) 알고리즘이 있다[4]

이 쉬어-웜 알고리즘은 보통 그래픽스 워크스테이션에서 화질의 저하 없이 적절한 크기의 볼륨 데이터를 실시간 렌더링을 가능하게 하여 많이 이용되고 있다. Lacroute와 Levoy는 쉬어-웜 알고리즘에 early-ray termination과 run-length coding을 도입하여 단일 프로세서 시스템을 구성하였고, 이는 여러 개의 프로세서를 사용하여 볼륨 렌더링을 병렬 처리하는 접근 방식과 비슷한 성능을 얻었다. 그러나 최적화된 쉬어-웜 알고리즘이 사용되었다 하더라도 큰 3차원 데이터에 대해서는 여전히 렌더링 시간이 많이 걸려 실시간으로 처리하는 것이 힘들다는 단점이 있다[5].

하드웨어를 이용한 시도는 크게 전용 가속기를 제작하여 빠른 렌더링 시간을 얻고자 하는 전략과 볼륨 렌더링 알고리즘이 갖는 병렬성을 이용하여 그에 적합한 병렬 컴퓨터를 이용하고자 하는 전략으로 나눌 수 있다. Gelder[6] 등은 texture mapping 기법을 위해 전용 하드웨어를 제작하여 거의 실시간에 가까운 속도의 고화질 렌더링 영상을 얻는 방법을 제안했고, Pfister[7] 등은 1024^3의 볼륨 데이터에 대해 초당 30 프레임의 렌더링 능력을 갖는 전용 가속기 Cube-4를 개발하였다. 이 전용 가속기 시스템은 고화질의 실시간 볼륨 렌더링을 구성할 수 있지만, 하드웨어 가격이 비싸고 최신 알고리즘에 대한 하드웨어 유연성도 부족하다는 단점을 가지고 있다. 반면 상용적으로 이용 가능한 다중프로세서 시스템을 이용하면 상대적으로 낮은 가격으로 좋은 성능의 볼륨 렌더링을 구현하는 것이 가능하고 공유메모리에 여러 프로세서들이 동시에 접근하여 렌더링을 수행하는 공유메모리 방식이나 볼륨 데이터를 각 프로세서에 할당하고 각 프로세서에서 처리된 데이터를 메시지를 통해 모아 최종 영상을 얻는 메시지 전달 방식과 같은 알고리즘을 유연성있게 수행할 수 있다.

이 논문에서는 앞에서 언급된 방식들의 장점, 즉 하드웨어의 스피드와 소프트웨어의 유연성을 결합함으로써, DSP와 PC사이에 데이터를 주고 받으며 빠른 볼륨 렌더링을 수행하는 시스템을 구현하였다. 많은 계산량을 효과적으로 처리하기 위해 선택된 8개의 명령어를 동시에 수행할 수 있는 TMS320C6201 DSP에 적합한 렌더링 알고리즘을 제안하여 최적화하였다.

이 논문의 구성은 다음과 같다. 우선, 이 논문에서 사용하는 TMS320C6201 DSP 구조에 대해 설명하고, DSP에 적합하도록 제안된 알고리즘에 대한 설명에 이어서 이 논문에서 제안된 알고리즘과 기존의 알고리즘을 DSP에 구현한 결과를 비교·평가하고, 마지막으로 결론을 맺는다.

TMS320C6201 DSP의 구조

TMS320C6201 DSP는 superscalar 방식의 VLIW 구조를 가지고 있기 때문에 명령어의 병렬 연산이 가능하다. TMS320C6201 DSP는 CPU, 주변장치, 저장장치 세 부분으로 구성되어 있다. 8개의 연산장치는 각각 4개씩 2개의 데이터경로(datapath)로 나뉘어져 있다. 논리를 수행하는 L, shift와 같은 연산을 수행하는 S, 곱셈을 수행하는 M, 데이터를 저장하거나 부를 수 있는 D의 4개의 연산장치는 독립적으로 데이터 의존성이 없을 때 한 cycle 내에 동시 수행이 가능하다. 2개의 데이터경로는 각각 16개의 32-bit register를 가지고 있고, cross path를 통해 서로 데이터를 주고 받을 수 있다.

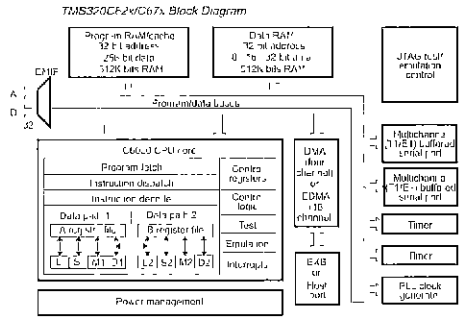


그림 1. TMS320C6201 DSP구조
Fig. 1 Architectural block diagram of TMS320C6201

제안된 알고리즘

TMS320C6201 DSP의 내부 메모리는 64KB의 프로그램 메모리와 64KB의 데이터 메모리로 구성되어 있다. 이 중 데이터 메모리는 크기가 작기 때문에 stack이나 변수들의 임시 저장으로 사용되고 볼륨 데이터는 외부 메모리에 저장된다. 그러나 내부 메모리와 외부 메모리간의 데이터 저장과 데이터 불러오기는 DSP의 clock보다 느린 메모리 clock과 외부 메모리의 접근을 담당하는 memory controller의 제어기능 때문에 부가적인 cycle을 요구한다. 그렇기 때문에 빈번하게 발생하는 볼륨 데이터에 대한 접근을 최소화시킬 수 있는 알고리즘이 적용되어야 한다. 또한 하나의 8bit 볼륨 데이터를 처리하기 위해 32bit register와 32bit bus 전부를 사용해야 하는 낭비적인 요소들을 적절히 고려하고, DSP 프로그램 내의 loop를 줄일 수 있는 알고리즘을 생각해야 한다. 이를 위해 볼륨을 쉬어하면서 생기는 부가적인 공간을 고려하지 않도록 제안된 Ray

map과 8bit 볼륨 데이터들을 한꺼번에 4개씩 불러와 32bit bus와 register를 효율적으로 이용하도록 제안된 one-to-four ray casting, 그리고 중간 영상을 얻기 위해 필요한 단면 영상의 수를 고려하여 DSP 프로그램의 loop를 줄이는 voxel-skipping 방법 등을 제안한다.

1. Ray map

쉬어-웝 알고리즘에 있어서 계산량의 대부분을 차지하는 것은 쉬어 과정을 통해 중간 영상을 만들어내는 부분이다. 이 쉬어 과정을 구현하는 데 있어 가장 손쉽게 생각할 수 있는 것은 관찰자가 보는 각도에 따라 단면 영상을 전부 쉬이시킨 뒤 그 쉬어된 볼륨 데이터를 전부 메모리에 담아 ray를 중간 영상에 수직하게 진행시키면서 color 값을 축적해 나가는 방법이다. 이는 초기의 쉬어를 위한 단면 영상의 수평이동 계산량을 제외하면 효율적으로 메모리에 접근하고 단순한 덧셈 한번으로 ray를 진행시키면서 color 값을 축적할 수 있다는 큰 장점이 있다. 그러나 관찰자가 보는 각도가 커짐에 따라 2배 정도까지 볼륨 데이터가 커질 수 있기 때문에 DSP에 적용시키기에는 부적합하다. 이를 개선하여 각도에 따라 볼륨 데이터를 새로 생성하지 말고, 중간 영상을 구성하기 위해 각 단면 영상들을 위치에 따라 이동시킨 뒤 중간 영상에 덮어 씌우는 방식을 생각할 수 있겠다. 이는 볼륨 데이터를 직접 쉬어시켜 object-order 방식과 유사한 방법으로 단면 영상 단위로 중간 영상으로 footprint하는 volume-shear method 방식이다. 이는 볼륨 데이터를 단순히 순차적으로 읽어와 임의의 메모리 영역에 계속 갱신시키기 때문에 중간 영상에 해당되는 메모리 영역만이 요구되고 전체 볼륨 데이터에 해당되는 부분만 읽어오기 때문에 보는 각도와는 상관없이 항상 일정한 프레임을 유지할 수 있다는 장점이 있다. 그렇지만 이 방식은 early-ray termination의 장점을 최대한 살리지 못하고 모든 볼륨 데이터들에 대해서 불투명도를 계산하고 compositing 과정을 수행해야 된다는 단점이 있다. 이와는 달리 image-order 방식과 유사하게 중간 영상을 얻는 ray-shear 방식이 있는데, 이는 메모리에 담겨 있는 볼륨 데이터를 ray가 쉬어하면서 access하여 축적해 나가는 방식이어서 early-ray termination이 가능하여 보다 빠른 랜더링이 가능하다. Early-ray termination의 장점을 살리고 ray-shear로 인한 ray의 위치 계산은 DSP의 강력한 pipelining 성능을 통해 시간을 줄일 수 있으므로 이 방식을 채택하였다. 그리고 이 방식의 최대 단점이라 할 수 있는 보는 각도에 따라 늘어나는 연산을 줄이기 위해 ray map을 제안하였다. ray map은 ray가 제일 처음 만나는 단면 영상의 인덱스를 저장하고 있는 메모리 영역으로 DSP에서 쉬어 과정을 수행하기 전에 미리 계산된다. Ray는 이 메모리 영역을 참조하여 그 범위만큼만 진행한다. 이 ray map을 도입함으로써 DSP 내에서 가장 pipelining을 힘들게 하는 분기 과정을 없애 효율적인 연산을 수행토록 하고 보는 각도에 따라 생기는 필요 없는 메모리 영역에 대해서는 compositing 과정을 생략하는 효과를 가져온다. 이 다음 그림 2의 예제를 살펴보자. 이 예제는 5장의 단면 영상을 가지고 있고 $1\times N$ 크기를 갖는 볼륨 데이터에 대해서 생각해보자. 이 때 그림 2처럼 볼륨 데이터가 임의의 각으로 쉬어됐을 때 ray가 시작하는 단면 영상을 나타내는 ray map은 R_{start}로 나타내어진다. 이 때 ray가 끝나는 단면 영상 인덱스 R_{end}는 R_{start}로부터 구해질 수 있으며 그 식은 그림 2에 나타나 있다.

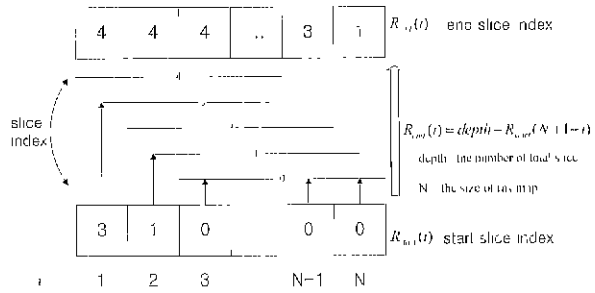


그림 2. Ray map
Fig. 2. Ray map

R_{end}는 따로 호스트에서 계산되어 DSP 메모리에 저장시킬 수 있지만 이는 부가적인 메모리를 필요로 하기 때문에, 한 ray를 진행시킬 때 R_{start}를 참조하여 R_{end}의 메모리 영역을 계산하도록 하였다. 즉 ray map에 R_{start} 값들이 저장되어 있어서 각 ray마다 시작되는 slice와 끝나는 slice를 쉽게 알 수 있게 한다. 이는 다음과 같은 shear 성질에 착안하여 제안하였다

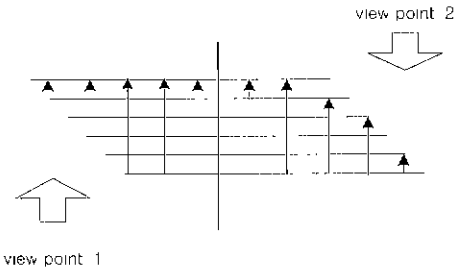


그림 3. Ray map으로부터 end slice index를 구하는 원리
Fig. 3. The principle which we can acquire the end slice index from Ray map

Shear시 단면 영상간에 일정한 규칙을 가지기 때문에 그림에서 view point 1으로부터 보이는 Ray map은 view point 2로부터 보이는 Ray map과 동일하다. 단면 영상이 $1\times N$이 아닌 2차원 단면일 때도 해당이 된다. 이 성질을 이용하면 사용자가 보는 각도에 따라 정해지는 Ray map으로부터 그 반대편의 ray map을 대칭성에 의해 depth와 volume data 개수로부터 구할 수 있다.

2. One-to-four ray casting

볼륨 렌더링에 있어 가장 중요한 문제는 보는 각도가 달라짐에 따라 모든 볼륨 데이터에 대해서 compositing 과정을 수행해 주어야 하기 때문에 계산량이 많다는 것이다. 이는 DSP 내에서 compositing 과정을 수행하는 핵심 loop 과정을 모든 voxel에 대해서 전부 수행해 주어야 한다는 말과 같다. 그렇기 때문에 DSP가 효율적으로 연산을 수행하고 계산량을 줄이기 위해서는 핵심 loop 과정의 명령어 수를 줄이는 것도 중요하지만 loop 수를 줄이는 것이 가장 관건이 된다. 또한 볼륨 데이터는 8bit이고 DSP 내에서 bus와 register는 32bit 크기이기 때문에 8bit 볼륨 데이터 하나를 처리하기 위해 32bit 전부를 사용해야 한다는 것은 낭비이다. 이를 위해 loop unrolling 과정을 수행한다. 즉 ray를 2×2로 4개씩 진행시킨다면 loop 회수는 전체의 1/4이 되고 4개의 볼륨 데이터를 한꺼번에 외부 메모리에서 불러오기 때문에 bus나 register를 효율적으로 사용하는 효과를 가져온다. 핵심 core 부분의 명령어 개수는 늘어나겠지만 pipelining을 통해 명령어 수는 크게 줄일 수 있다. 이에 기존에 사용되고 있는 불투명도 값이 실험적으로 구한 threshold 값 이하이면 그 voxel에 대해서는 composting 과정을 수행하지 않는 opacity threshold 방식과 결합하여 one-to-four ray casting 방법을 제안한다.

이 방법에서는 하나의 기준 ray를 따라서 voxel 값들을 opacity threshold 값과 비교하다가 voxel 값이 opacity threshold를 초과한다면 이 voxel 주위에는 유효한 값들을 가질 가능성이 높으므로 ray가 4개(2×2)의 ray로 갈라져서 각자 compositing 과정이 수행된다. 이 개념을 그림 3에서 2차원적으로 나타내었다. 이 때 고려되어야 할 것은 4개의 ray가 한 묶음으로 움직이기 때문에 early-ray termination 과정을 잘 해주어야 한다. 그런데 early-ray termination 과정을 기준 ray에 대해서만 수행하게 되면 기준 ray가 termination 조건을 만족하여 4개의 ray가 진행을 멈추게 될 때 나머지 3 ray의 불투명도 값에 상관없이 termination 과정이 일어나게 되어 모서리 부근에서 artifact가 생길 가능성이 높아진다. 이를 막기위해 early-ray termination 조건은 4 ray의 불투명도 평균값으로 비교를 하도록 한다.

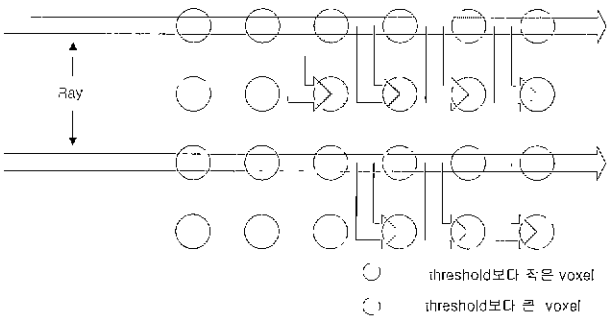
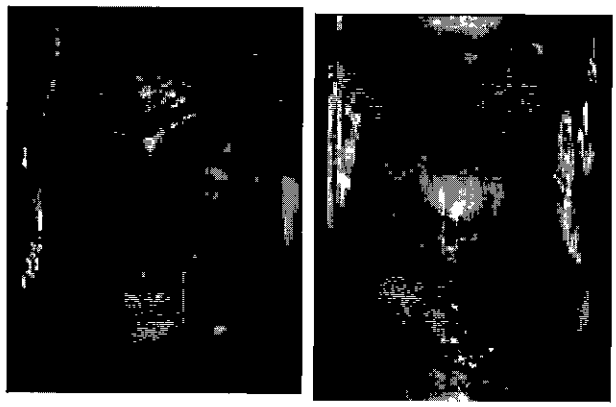
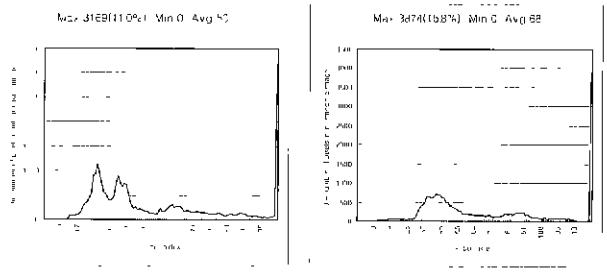


그림 3. One-to-four ray casting 방법
Fig. 3. One-to-four ray casting method



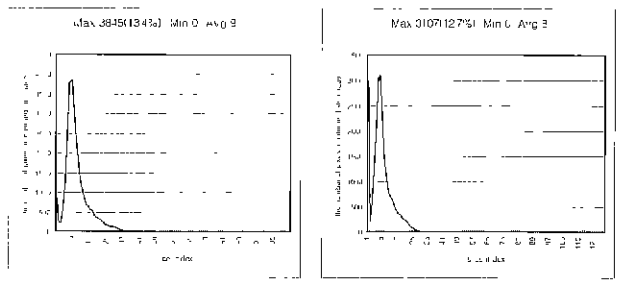
(a) 148×194×112 3D face data (b) 132x186x126 3D back data

그림 4. 3D 초음파 데이터로부터 얻은 영상
Fig 4 Volume rendered image from 3D ultrasound data



(a) 148×194×112 3D face data (b) 132×186×126 3D back data

그림 5. Ray termination이 일어난 단면 영상 인덱스에 대한 histogram
Fig. 5. Histogram for the slice-number of early-termination



(a) 148×194×112 3D face data (b) 132x186x126 3D back data

그림 6. Pixel 값을 결정하는 단면 영상들의 수에 대한 histogram
Fig. 6. Histogram for the number of slices determining the value of pixel

3. Voxel skipping 방법

보통 볼륨 데이터 분포도를 살펴본다면 중심부에 우리가 원하는 물체가 몰려 있고 나머지 부분들은 opacity threshold 값보다 작은 값들로 채워지는 경향이 있음을 알 수 있다. 우리가 원하는 중심의 물체를 계산하기 위해서는 필요 없는 voxel들에 대해서도 compositing 과정을 수행해야 된다. 이는 분명

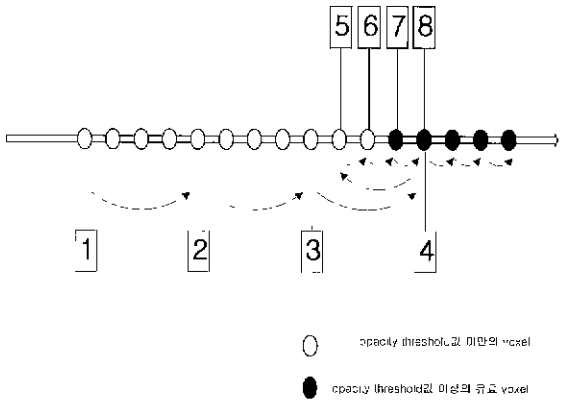


그림 7. Voxel skipping 방법
Fig. 7. Voxel skipping method

많은 계산량의 원인이 되고 있다. 다음의 예를 살펴보자. 그림 4(a)는 148×194×112 초음파 태아 얼굴 데이터를, 그림 4(b)는 132×186×126 초음파 태아 등 데이터를 정면에서 볼륨 렌더링시킨 결과이다. 이 데이터들에 대해서 볼륨 렌더링을 수행시키되 early-ray termination이 몇 번째 단면 영상에서 일어나는지에 대한 histogram과 volume rendering 중간 영상의 각 pixel 값을 결정하기 위해 필요했던 유효한 voxel의 수를 구해 histogram으로 그림 5와 그림 6에 그려보았다. 그림 5에서 보듯이 early-ray termination은 거의 전체 단면 영상 중 중간위치정도에서 일어난다. 그렇지만 실제로 중간 영상을 만들어 내기 위해 필요한 단면 영상 수는 전체 중 10% 미만으로 compositing을 위한 대부분의 계산은 opacity threshold 값 미만의 값을 가지는 voxel들이라는 것을 알 수 있다. 이 낭비를 막기 위해 voxel-skipping 방법을 제안하였다.

Voxel-skipping 방법은 그림 7에서 설명되어 있다. 그림에 나타나 있는 번호는 ray의 진행 순서이다. Ray를 따라서 compositing을 진행하는데, Opacity threshold보다 낮은 값들을 가진 voxel들을 만나면 ray는 사용자가 지정한 voxel만큼 건너뛰면서 compositing을 수행하게 된다. (그림에서는 4개씩 건너뛰도록 그려져 있다.) 진행하다 threshold보다 큰 값을 만나게 되면 그 이전의 위치와 현재의 위치 사이에 유효한 voxel들이 있을 수 있으므로 그 이전의 voxel 이후부터 한 voxel씩 ray를 진행시킨다

이 방법은 DSP 내에서 compositing 과정을 위한 핵심 계산부분에 들어가기 전에 미리 전처리 과정으로 compositing을 시작하는 voxel 위치를 지정하는 방법으로서 DSP 내의 looping 회수를 줄이는데 효과적이다

시스템 구현

기존 알고리즘은 image-order ray casting 방식을 쓰되 early-ray termination을 적용시켰고 미리 계산된 template을 이용하여 ray를 진행시킴으로써 ray의 다음 위치 계산을 단순한 메모리 참조만으로 가능하게 하였다. 이 때 viewing angle에 따른 template data, ray가 shearing된 volume과 처음 만나는 slice number를 저장하는 ray map, 그리고 opacity table과 같은 데이터들은 부동소수점 계산이 요구되어 CPU에서 전처리 과정을 수행토록 하여 DSP의 부담을 줄였다. CPU의 제어 하에 parameter를 DSP에 넘겨준 뒤 DSP는 shearing 과정을 수행하고 그 동안 CPU는 이전 frame에서 DSP에서 처리된 intermediate image를 warping시키고, 다음 frame에 필요한 pre-processing 과정을 수행한다.

즉, 메인 CPU는 DSP 보드를 control하는 controller로서의 역할과 DSP에 필요한 여러 preprocessing data들을 계산해 주고 warping 과정을 pipeline으로 처리하는 역할을 수행한다

표 1. 초음파 볼륨 데이터를 DSP에서 렌더링하는데 걸린 시간(초)
Table 1. Rendering time on DSP for ultrasound volume data

	148×194×112 태아 얼굴 데이터			132×186×126 태아 등 데이터		
	기존 알고리즘	제안된 알고리즘 (c프로그램)	제안된 알고리즘 (어셈블리)	기존 알고리즘	제안된 알고리즘 (c프로그램)	제안된 알고리즘 (어셈블리)
정면	0.672초	0.157초	0.110초	0.718초	0.125초	0.079초
30°	0.965초	0.172초	0.125초	0.703초	0.140초	0.110초

표 2. 기존 알고리즘에 대한 제안한 알고리즘을 렌더링한 결과 영상의 PSNR(dB)
Table 2. PSNR between the conventional and proposed results

	148×194×112 태아 얼굴 데이터		132×186×126 태아 등 데이터	
	제안된 알고리즘 (c프로그램)	제안된 알고리즘 (어셈블리)	제안된 알고리즘 (c프로그램)	제안된 알고리즘 (어셈블리)
정면	37.91dB	38.54dB	27.40dB	27.67dB
30°	33.08dB	33.26dB	28.26dB	28.01dB

제안된 알고리즘은 C프로그램으로 짜서 TI에서 제공되는 컴파일러를 사용하여 실행하였고, 이것을 직접 어셈블리 코딩을 통해 software pipelining, 작업분배, 비순차실행 방법 등을 사용함으로써 다시 최적화시켰다.

결과 및 분석

3차원 초음파 볼륨 데이터에 대해서 기존 알고리즘과 본 연구에서 제안한 알고리즘을 구현하였을 때 DSP 내에서의 렌더링 시간을 측정한 결과는 다음과 같다. 렌더링되는 볼륨 데이터는 모두 8bit 데이터이다.

표 1은 148×194×112의 볼륨 사이즈를 가지는 태아의 얼굴 데이터와 132×186×126의 볼륨 사이즈를 가지는 태아의 등 데이터에 대한 렌더링 결과이고, 표 2는 태아의 얼굴 데이터와 태아의 등 데이터를 기존 알고리즘으로 볼륨 렌더링한 결과와 제안한 알고리즘과의 PSNR을 비교한 것이다. 이 결과는 Pentumum III 450MHz dual CPU 환경에서 TMS320 C6201을 부착하여 나온 결과이다. TMS320C6201을 사용하지 않을 때는 process 성능에 따라 차이가 나는데 여기에서는 고려하지 않는다.

표 1에서 보는 바와 같이 기존의 알고리즘에 비해 제안된 알고리즘을 사용하면 크게 시간을 줄일 수 있음을 알 수 있다. 어셈블리 코딩 시 DSP 프로그램 내부의 명령어를 직접 최적화시킴으로써 수를 줄였으므로 C프로그램을 컴파일시킨 결과와 비교하여 렌더링 시간에 있어 30% 이상의 향상이 있음을 볼 수 있다. 기존 알고리즘에서는 보는 각도에 따라 20~30% 정도의 렌더링 시간이 더 걸림을 알 수 있는데 이는 보는 각도가 더 커짐에 따라 점점 선형적으로 증가한다. 그렇지만 제안된 알고리즘의 ray map에서는 범위를 미리 계산하여 그만큼만 ray를 진행시켜 전체적으로 봤을 때 ray는 볼륨 사이즈만큼만 진행하기 때문에 각도에 상관없이 범위를 계산하는 전처리 시간만이 더 걸릴 뿐이다. 그 시간은 단면 영상의 크기에 비례하고 대략 전체 처리 시간의 10% 정도 더 걸린다. 제안된 알고리즘 중 voxel skipping 방법은 깊이 방향으로 불필요한 voxel들을 건너뛰는 방법이고 one-to-four ray casting 방법은 단번에 쏘아지는 ray 수를 1/4로 줄이는 방법이므로 단면 영상의 크기가 작고 깊이 방향의 단면 영상 개수가 많을 때 더 효과적으로 렌더링 시간이 줄어든다. 이는 148×194×112의 face 데이터와 132×186×126의 back 데이터를 볼륨 렌더링한 표1을 보면 쉽게 알 수 있다. 각 과정은 유기적으로 연결되어 있어 따로 결과를 측정하였을 경우 DSP에서 오히려 더 나쁜 결과를 초래할 수 있어 각각 적용 시 결과는 측정하지 않았다.

기존의 알고리즘과 제안된 알고리즘으로 얻은 결과 영상에 대한 PSNR은 표 2에서 보는 바와 같다. Face data에 대해서는 만족할 만한 PSNR이 얻어지지만, back data는 다소 image quality가 떨어짐을 알 수 있다. 이는 알고리즘이 volume data의 형태에 따라 영향을 받는 것을 뜻하는데, 그림 5, 6을 참조한다면 face data인 경우 전체 볼륨 중 ray termination이 back data때보다 일찍 일어남을 알 수 있고, back data는 pixel값을 결정하는 단면영상수가 face data과는 달리 1~2장으로 결정되는 경우가 압도적으로 많다. 이 때문에 제안된 알고리즘은 속도면에서는 back data가 유리하지만, 화질면에서는 back data가 pixel 값을 결정하는 단면 영상을 제대로 sam-

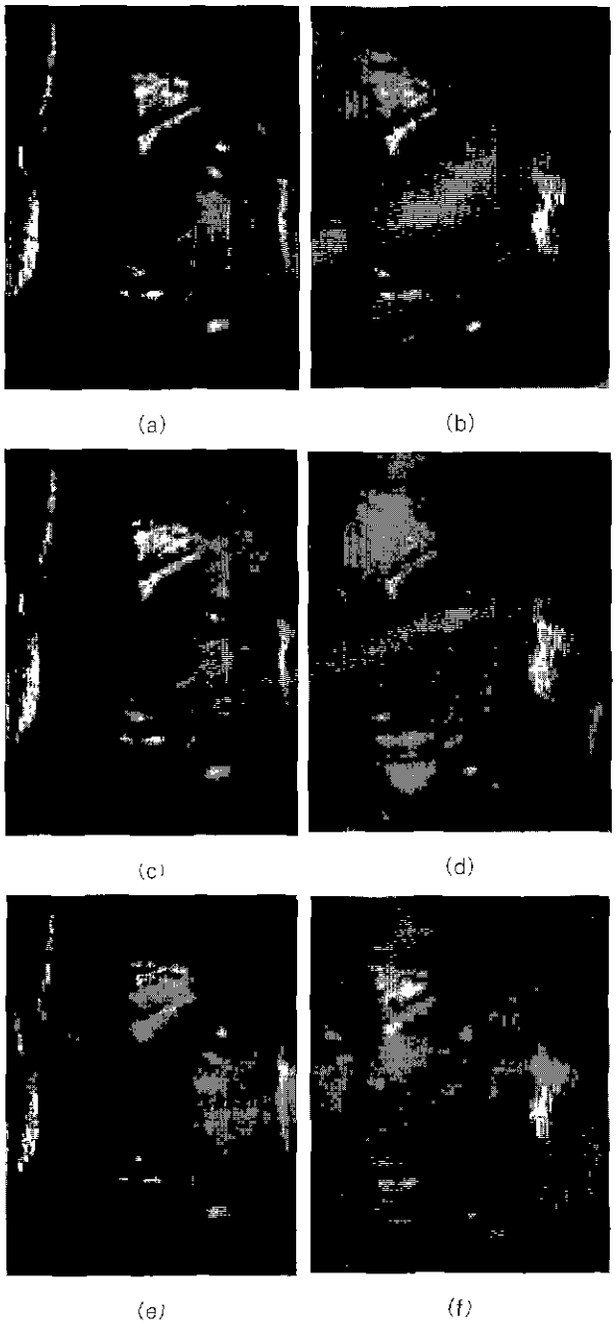


그림 8 (a)(b) 기존 알고리즘을 적용한 렌더링 결과 (c)(d) 제안한 알고리즘을 C로 구현한 렌더링 결과 (e)(f) 제안한 알고리즘을 어셈블리로 구현한 148×194×112 태아 얼굴 데이터 렌더링 결과
Fig. 8. Results of rendering 148×194×112 petus face data in (a)(b) Conventional algorithm (c)(d) Proposed algorithm in C (e)(f) Proposed algorithm in assembly

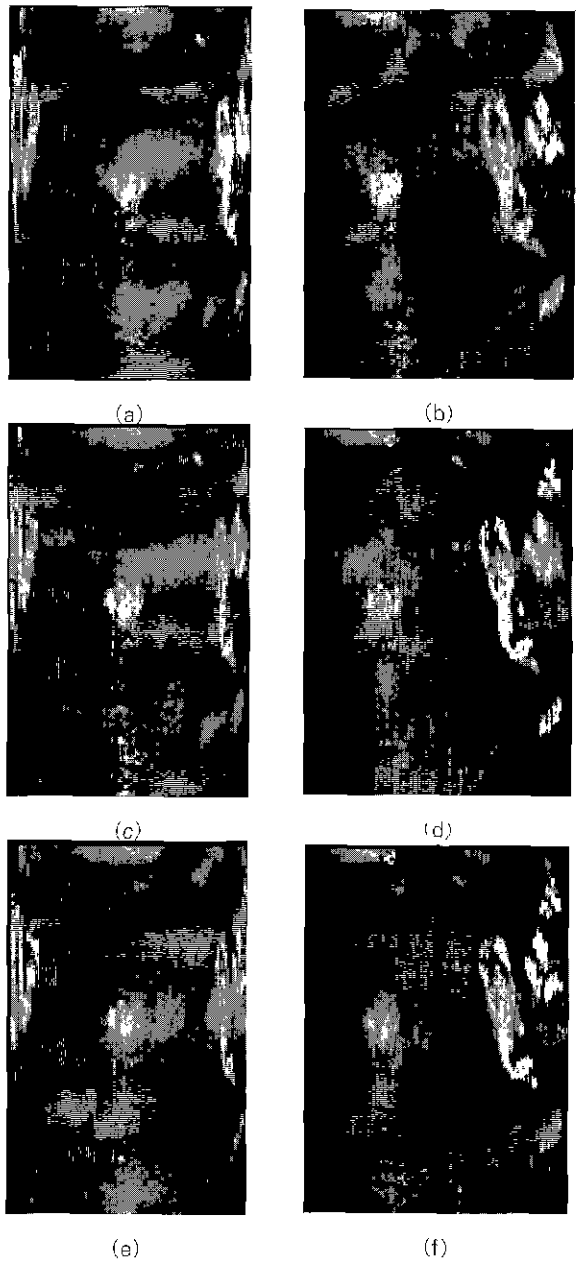


그림 9 (a)(b) 기존 알고리즘을 적용한 렌더링 결과 (c)(d) 제안한 알고리즘을 C로 구현한 렌더링 결과 (e)(f) 제안한 알고리즘을 어셈블리로 구현한 132×186×126 태아 등 데이터 렌더링 결과
Fig. 9. Results of rendering 132×186×126 petus back data in (a)(b) Conventional algorithm (c)(d) Proposed algorithm in C (e)(f) Proposed algorithm in assembly

pling하지 못하는 위험성 때문에 face data에 비해서 PSNR이 떨어짐을 알 수 있다

고 찰

One-to-four ray casting 방법은 ray가 4개씩 한꺼번에 진행하고 동시에 termination이 일어나므로 볼륨 데이터에 따라 모서리 부근에서 계단 무늬 현상이 일어나는 경우를 관찰할 수 있었다.

Voxel skipping 방법에 의해 얇은 막이나 조직들이 고려되지 않는 경우가 발생할 위험성을 내포한다. 이는 사용자가 voxel을 건너 뛰는 정도를 직접 제어 할 수 있기 때문에 어느 정도 위험성을 제거할 수 있지만 렌더링 시간이 늘어남을 감수해야 한다 렌더링 시간과 화질을 고려하여 상황에 따라 적절히 제어하는 것이 필요하다

결 론

이 논문에서는 볼륨 렌더링을 TMS320C6201 DSP에 구현할 때 적합한 몇 가지 알고리즘을 제안하여 효과적으로 렌더링 시간을 줄일 수 있었다.

먼저 볼륨 렌더링이 가지는 병렬성(parallelism)의 효과적 처리를 위해 선택한 TMS320C6201의 구조를 살펴보고, 알고리즘을 적용시키는 데 있어 고려해야 될 문제점들을 살펴보았다. 기존의 early-ray termination과 template의 개념이 도입되어 성능이 향상된 쉬어-웜 알고리즘을 DSP에 적용시켰다. 물체를 보는 각도에 따라 쉬어-웜 알고리즘에서 생기는 부가적인 계산량을 줄이기 위해 *ray map* 개념을 도입하여 ray가 시작되고 끝나는 범위를 미리 계산하여 DSP에서 이용하였고, 이로 인해 branch operation을 줄여 보는 각도에 따라 부가되는 계산량을 약 10% 정도로 줄일 수 있었다 하나의 ray로 진행하나 유효한 값을 가지는 voxel을 만나면 4개의 원래 ray로 나뉘어져 voxel 값을 계산하는 *one-to-four ray casting* 방법을 제안하여 loop 수를 줄이고 볼륨과 DSP 내 버스나 레지스터의 데이터 사이즈를 정합시켜 성능 향상을 꾀하였고, 유효하지 않은 voxel에 대해서는 사용자가 지정한 수만큼 건너뛰면서 ray를 진행하다가 유효한 voxel 값을 만나면 그때부터 하나씩 진행하도록 하여 계산량을 줄이는 voxel skipping 방법 등을 제안하였다.

이 제안된 알고리즘을 DSP에 적용시켰을 때 기존의 알고리즘보다 빠른 렌더링 시간을 얻을 수 있었고, 기존의 알고리즘에 상응하는 화질을 얻을 수 있었다.

References

1. A Kaufman (Eds.), Volume Visualization, IEEE Computer Society Press, 1991
2. L A Westover, *"Footprint evaluation for volume rendering,"* Computer Graphics, Vol. 24, No. 4, August 1990
3. M. Levoy, *"Display of surfaces from volume data,"* IEEE Computer Graphics and Applications, May 1988.
4. p Lacroute and M. Levoy, *"Fast volume rendering using a shear-warp factorization of the viewing*

transformation," Proceedings of Computer Graphics, Annual Conference Series (SIGGRAPH "94), July 1994

5. T.S. Yoo, U. Neumann, H Fuchs, S.M Pizer, T.Cullip, J. Rhoades, and R Whitaker, "*Direct visualization of volume data,*" IEEE Computer Graphics and Applications, pp.63-71, 1992
6. B. Cabral, N.Cam, and J.Foran, "*Accelerated volume rendering and tomographic reconstruction using texture mapping hardware,*" Proceeding of the 1994 symposium on volume visualization, pp.91-98, 1994
7. A.V Gelder and K. Kim, "*Direct volume rendering with shading via three-dimensional textures,*" Proceeding of the 1996 symposium on the volume visualization, pp 23-30, 1996