큐브의 단면을 이용한 기하학적인 물체의 복셀화

권 오 봉†

요 약

최근 볼륨 그래픽스가 의료 영상의 해석 도구로서 주목을 받아 오고 있다. 볼륨 그래픽스에서는 가시화를 위하여 복셀로 정의된 물체가 필요하다. 이 때문에 기하학적인 방법으로 정의한 다각형 및 곡면을 복셀 기반의 물체로 변환시키는데 이를 복셀화(voxelization)라고 한다. 기하학적인 물체를 복셀화하면 기하학적 물체 데이터를 샘플링 데이터와 함께 단일화된 방법으로 볼륨 렌더링할 수 있다. 본 논문에서는 큐브의 단면을 이용한 복셀화의 한 기법을 제안한다. 그리고 개인용 컴퓨터 환경에서 이 기법을 구현한 후에 단순한 기하학적인 데이터를 이용해서 평가하여 논리적인 타당성을 조사한다. 이 기법은 변환된 복셀로부터 정확한 법선 벡터를 계산할 수 있고 복셀간에 구멍(hole)이 발생하지 않고 다해상도(multi-resolution) 표현이 가능한 특성을 가지고 있다.

A Voxelization for Geometrically Defined Objects Using Cutting Surfaces of Cubes

Ou-Bong Gwun†

ABSTRACT

Volume graphics have received a lot of attention as a medical image analysis tool nowadays. In the visualization based on volume graphics, there is a process called voxelization which transforms the geometrically defined objects into the volumetric objects. It enables us to volume render the geometrically defined data with sampling data. This paper suggests a voxelization method using the cutting surfaces of cubes, implements the method on a PC, and evaluates it with simple geometric modeling data to explore propriety of the method. This method features the ability of calculating the exact normal vector from a voxel, having no hole among voxels, having multi-resolution representation.

키워드 : 볼륨모델링(Volume Modeling), 볼륨렌더링(Volume Rendering), 복셀화(Voxelization), 기하학적 데이터(Geometrically Defined Data), 샘플링 데이터(Sampling Data), 법선 벡터(Normal Vector), 큐브(Cube)

1. 서 론

볼륨 그래픽스(volume graphics)는 복셀(voxel)의 볼륨 버퍼 속에 저장된 물체의 모델링, 변환 처리, 렌더링 등을 다루는 컴퓨터 그래픽스의 한 분야이다[2]. 볼륨 그래픽스에서 사용하는 데이터는 두 종류가 있는데 하나는 CT, MR 등의 샘플링 장치에서 생성된 샘플링 데이터이고 다른 하나는 기하학적으로 정의한 물체를 알고리즘을 이용하여 이산화(digitize)시킨 볼륨 기하 데이터이다.

일반적으로 환부 위의 방사선 치료에서와 같이 의료 분야의 응용에서 방사선, 수술 도구 등의 물체는 기하학적으로 정의되고 환부 조직, 기관 등의 인체는 샘플링 데이터로 표시된다[3]. 가상 수술 등에서는 인체의 기관과 수술도구가 함께 표시되어 기하학적으로 정의한 물체와 샘플링 데이터를 함께 표시할 필요가 있다. 기하학적으로 정의한 물체와 샘플링 데이터를 결합시키는 기법은 복셀 기반 기법, 면 기반 기법, 볼륨 렌더링과 면 렌더링(surface rendering)을 독립적으로 실행한 후에 이를 결합시키는 기법 등이 있다[12]. 복셀 기반 기법에서는 복셀화(voxelization)라고 하여 연속적인 수학적 정의로부터 만든 기하학적인 물체를 볼륨 기하 데이터로 변환시키는 과정이 필요하다[5].

복셀 기반의 기법은 다른 기법과 비교하여 많은 장점이 있다. 우선 복셀 기반 기법은 여러 종류의 물체를 복셀이라는 단일 표현으로 나타낸다. 또 복셀 기반 기법은 복셀 블록 트랜스퍼(VOXBLT : Voxel Block Transfer) 연산이 가능하여 여러 개의 임의의 기본물체(primitive)볼륨을 다양한 연산으로 결합시켜 새로운 볼륨을 생성하게 함으로써 CSG (Constuctive Solid Geometry)에서와 같이 다양한 물체를 만들 수 있다. 더구나 볼륨 래스터 고유한 성질인 공간적인 연속성은 계층적인 다해상도(multi-resolution) 표현이 가능하여, 계층 레벨을 증가시켜 3차원 해상도를 감소시키면 원래의 모델을 단순하게 표현할 수 있다[2, 8].

볼륨 기하데이터를 렌더링하기 위해서는 해당 복셀 데이터로부터 기하학적인 기본물체를 찾고 이 기본물체의 법선 벡터를 계산할 수 있어야 하며 각 기본물체들은 이웃의 기

※ 이 논문은 2001년도 전북대학교의 지원연구비에 의하여 연구되었음.
† 종신회원 : 전북대학교 전자전보공학부 교수
논문접수 : 2002년 7월 2일, 심사완료 : 2003년 3월 11일

본물체와 완전히 연결(neighbor connectivity)되어야 한다. 그렇지 않으면 회전, 확대 등의 기하 변환 후에 생성된 영상에 구멍(hole)이 생긴다. 그러나 1세대 볼륨 그래픽스에서는 복셀들이 법선 벡터를 정확히 계산할 수 있는 정보를 갖지 못했다. 예를 들어 이산 광선 추적법(discrete ray tracing)[16]은 물체의 음영과 이차 광선을 계산하기 위해서 정확한 면 법선(surface normal) 벡터를 필요로 하나 이산 복셀 표현은 어떤 기하학적인 정의도 갖고 있지 않았다[20]. 또 쿠베릴르(cuberille) 모델에서는 물체의 모델링 기본물체로서 큐브를 사용하였다[11]. 그러나 큐브의 법선 벡터는 여섯 면으로 한정되어 유연한 면을 갖는 물체를 표현하기가 곤란하여 음영을 계산하기 위해서 복잡한 기법을 도입하였다[15].

복셀화에는 포인트 샘플링(point sampling) 기법[5, 7, 15], 필터링(filtering) 기법[19, 13], 거리 필드(distance field) 기법[4, 14] 세 가지가 있다. 포인트 샘플링 기법은 연속적인 물체의 존재 여부를 복셀 중심에서 판정하여 물체가 존재하면 1, 존재하지 않으면 0을 할당한다. 이 기법은 복셀화 과정에서 법선 정보를 잃어버려 렌더링 할 때 정확한 법선 벡터를 계산할 수 없어 많은 에일리어싱(aliasing)이 발생한다. 필터링 기법에서는 에일리어싱 문제를 저역 필터를 이용하여 해결하였다. 우선 전처리 과정에서 저역 필터를 이용하여 연속적인 물체를 필터링 한 후에 샘플링 과정에서 각 복셀 중심에서 샘플링하였다. 물체 내부는 물체 내부의 밀도가 할당되었고 물체 외부는 배경 밀도가 할당되었다. 물체의 표면 즉, 천이 영역은 배경의 밀도와 내부의 밀도를 보간하여 표시하였다. 이 기법은 비교적 정확한 법선 벡터를 구할 수 있었으나 물체의 표면을 표시하는데 몇 개의 복셀 층을 요구하기 때문에 면은 두께를 갖지 않는다는 개념을 파괴하였다. 거리 필드 기법은 물체 내부의 각 점에 표면으로부터 거리를 할당하여 물체를 모델링 하였다. 이 기법은 물체가 물체 공간 내부에 서로 독립하여 지역적으로 위치하고 있다는 기존의 개념을 파괴하는 단점이 있다[13].

본 논문에서는 다각형을 기반으로 하는 위의 방법들과 다른 복셀화 기법을 제안한다. 본 기법은 복셀이 기하학적인 정의를 가지고 있기 때문에 법선 벡터의 계산은 물론, 면을 한 개의 복셀 층으로 표시하여 기존의 면에 대한 개념을 유지하는 한편, 물체는 물체 공간에 독립하여 지역적으로 존재한다는 개념을 유지한다. 이외에 원래의 물체가 가지고 있는 표현법을 그대로 유지하고 쿠베릴르 모델처럼 에일리어싱이 발생하지 않고 다해상도 표현이 가능한 특징이 있다. 그러나 제안된 기법은 물체를 다각형으로 분할한 후 각 정점(vertex)을 부호화하였기 때문에 기존의 방법보다 메모리가 많이 필요하고 복셀화 시간이 물체의 복잡도 및 해상도와 함께 증가하는 단점도 있다.

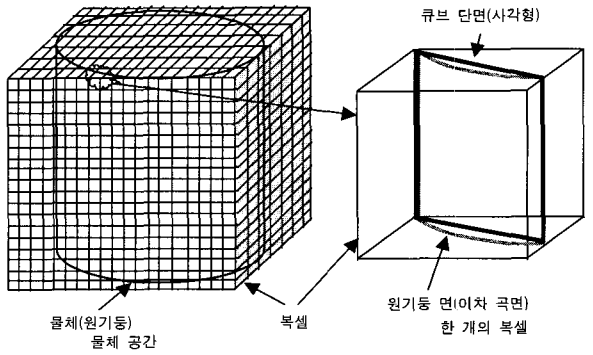
본 논문의 구성은 다음과 같다. 2장에서는 큐브 단면의 개념을 설명하고 이것을 이용하여 복셀화를 위한 몇 가지 가정을 한다. 3장에서는 큐브 단면의 기하학적인 정보를 복셀로 부호화하기 위한 기법을 고안한다. 4장에서는 기하학적으로 정의한 물체를 복셀화하고 이것을 렌더링 하는 방법에 대해서 설명한다. 5장에서는 제안된 기법의 구현 결과를 보이고 평가한다. 6장에서는 제안된 기법의 장단점에 대한 언급과 함께 결론을 맺는다.

2. 큐브 단면의 개념

볼륨 표현법은 흔히 산업 디자인, 애니메이션, 모의 수술 등의 응용에 필요한 복잡한 연산 및 다양한 조작을 위하여 기하학적 물체 표현법(geometric modeling)과 함께 사용되고 있다. 일반적으로 원래의 물체 표현법을 그대로 유지하면서 물체를 조작하는 것이 편리하고 정확하며 경제적인 것으로 알려져 있다[18]. 일반적인 복셀화 알고리즘은 기하학적인 좌표 데이터를 볼륨 밀도(volumetric density) 데이터로 변환시키는 데 이 과정에서 물체 표현법이 변하기 때문에 물체 모양이 변할 수 있다. 기하학적으로 정의한 물체 모양을 유지하기 위해서는 표현법을 변화시키는 것보다는 동일한 기하학적인 표현법을 그대로 사용하는 것이 바람직하다.

본 논문에서는 기하학적으로 정의된 물체가 면 그래픽스에서와 같이 다각형의 집합으로 표현되며, 복셀은 쿠베릴르 모델에서와 같이 큐브를 기반으로 하나 내부가 임의의 면을 가질 수 있다고 가정한다. 즉, 다각형은 (그림 1)에서와 같이 큐브의 단면으로 분할되며 부호화(encode) 과정에서 부호화하여 복셀 데이터로 저장된다. 그리고 렌더링 과정에서 복셀 데이터를 해독(decode)하여 이 데이터로부터 큐브 단면의 법선 벡터와 이웃 단면과의 연결 정점을 계산한다. 본 논문은 복셀화를 취급한 다른 문헌과 유사하게 다음과 같은 가정을 한다[2].

① 복셀(큐브)은 물체 공간에 있는 물체에 비하여 아주 작다.
② 한 개의 복셀은 단지 큐브의 한 단면만을 가지고 있다.

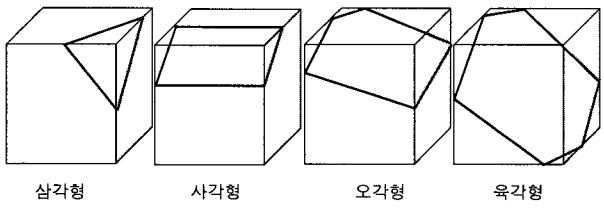


(그림 1) 큐브 단면을 이용한 곡면의 분할

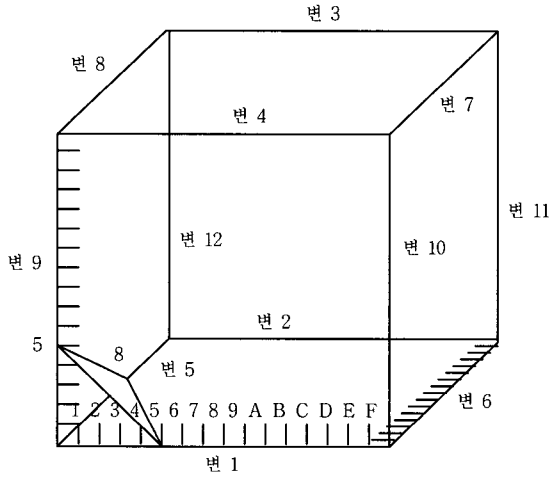
본 논문에는 원래의 데이터 특성에 따라 크게 두 종류의 복셀이 존재한다. 하나는 볼륨 기하 데이터 복셀이고 다른 하나는 샘플링 데이터 복셀이다. 볼륨 기하 데이터 복셀의 법선 벡터는 면 그래픽스에서와 같이 다각형 법선 벡터를 이용하는데 이 데이터는 해당 데이터 한 개만을 메모리에

서 액세스한 후 해독(decode)하여 각 정점의 좌표를 이용하여 벡터를 만든 후 이것의 역수를 취하면 된다. 예를 들면 삼각형의 각 정점의 값이 (3, 0, 0), (0, 4, 0), (0, 0, 5)라고 하면 이를 이용하여 (3, 4, 5)의 벡터를 만든 후 각 성분의 역수 (1/3, 1/4, 1/5)를 취하면 간단히 그 삼각형의 법선 벡터가 구해진다. 그러나 샘플링 데이터 복셀 법선은 주위 복셀을 함께 액세스하여 각 복셀 간의 값의 차이를 이용하여 경사 벡터(gradient vector)를 구한다. 따라서 볼륨 기하 데이터의 법선을 계산하는데 필요한 연산 시간은 샘플링 데이터 법선을 연산하는데 필요한 시간과 비슷하나 볼륨 기하 데이터의 메모리 접근 시간은 샘플링 데이터 메모리 접근 시간의 수분의 일이면 된다.

3. 큐브 단면의 부호화



(그림 2) 큐브 단면으로 생성되는 네 종류의 다각형

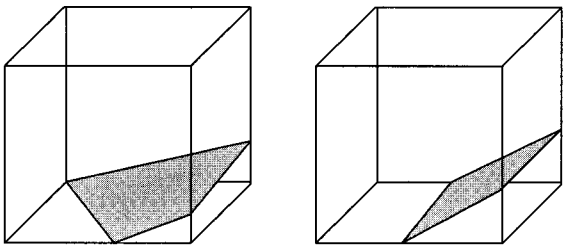


(그림 3) 변의 넘버링 및 분할

(그림 2)는 큐브에 존재하는 네 종류의 다각형 즉, 삼각형, 사각형, 오각형, 육각형을 표시하는데 본 논문에서는 이 다각형들을 이용하여 기하학적인 물체를 복셀화 한다. 각 다각형을 될 수 있는 한 좁은 데이터 폭(data width)의 복셀 데이터로 부호화하려고 시도하였다. 이를 위해서 우리는 (그림 3)과 같이 큐브상의 모든 변(edge)을 16눈금으로 나누어 큐브에 존재하는 모든 단면을 이 눈금 위의 정점을 연결하여 표시하였다. 예를 들면 (그림 3)에서는 큐브의 변 1, 5, 9 위에 존재하는 눈금 번호 5, 8, 5를 연결하면 하나의 삼각형이 된다. 비록 같은 종류의 다각형은 위상(topology)이 동일하지만 다른 변과 다른 법선 벡터를 갖는다. 예를 들어 종류가 삼각형인 경우에 큐브의 각 정점 밑에는 삼각형이 하나씩 존재하여 모두 8 경우가 된다. 큐브에 존재하는 모든 다각형을 종류 별로 분류하면 <표 1>과 같이 삼각형이 8 경우, 사각형이 15 경우, 오각형이 24 경우, 육각형이 4 경우, 총계가 51 경우가 된다.

〈표 1〉 다각형의 경우 수

다각형의 종류	경우 수	다각형의 한 예(변 번호)
삼각형	8	9, 5, 1
사각형	15	4, 3, 2, 1
오각형	24	9, 8, 7, 6, 1
육각형	4	1, 6, 11, 3, 8, 9
총 계	51	



(a) 특수한 경우 (b) 보통의 경우

(그림 4) 특수한 다각형의 처리

큐브의 단면을 분류하는 과정에서 (그림 4)(a)와 같이 특별한 경우가 발생하지만 이것은 <표 1>의 경우로 통합할 수가 있다. 즉 (그림 4)(a)와 (그림 4)(b)는 동일한 것으로 취급할 수 있다. 렌더링 단계에서 큐브의 단면을 빨리 해독하기 위해서 <표 2>와 같이 단순한 방법으로 부호화한다.

〈표 2〉 큐브 단면의 부호화

복셀 종류	다각형 종류	예 약	면의 방향	*경우 수	*변 번호
비트 31~30	비트 29~28	비트 27	비트 26	비트 25~24	비트 23~0
00 : 데이터 없음 01 : 샘플링 데이터 10 : 기하데이터 표면 11 : 기하 데이터 내부	00 : 삼각형 01 : 사각형 10 : 오각형 11 : 육각형		0 : 시계 방향 1 : 시계 반대 방향	00 : 경우 1 01 : 경우 2 10 : 경우 3 11 : 경우 4	비트3~0 : 변9 비트7~4 : 변8 비트11~8 : 변3 비트15~12 : 변11 비트19~16 : 변6 비트23~20 : 변1

* 현재 육각형인 경우를 표시한다. 다른 다각형의 경우 이 칸의 내용은 다르다.

또 복셀 데이터의 폭을 효율적으로 이용하기 위해서 다각형별로 나누어 처리한다. 예를 들어 육각형은 4 경우가 존재하고 오각형은 24 경우가 존재한다. 한 정점을 표시하는데 4비트를 사용하기 때문에 오각형은 20비트가 필요하며 육각형은 24비트가 필요하다. 다각형별로 나누어 처리하면 육각형에서 정점을 표현하기 위해서 사용하던 4비트가 오각형에서는 경우 수를 표시하는데 사용할 수가 있다. 이러한 점을 고려하여 큐브의 단면을 32비트 5필드 즉 복셀 종류, 다각형 종류, 면의 방향, 경우 수, 변 번호 필드로 표현한다.

복셀 종류 필드는 샘플링 데이터와 볼륨 기하 데이터를 함께 취급하기 위해서 설정하였다. 볼륨 기하 데이터는 특별한 경우를 제외하고 표면만 렌더링 된다. 다각형 종류 필드는 다각형의 종류 즉 삼각형, 사각형, 오각형, 육각형을 표시한다. 면의 방향 필드는 큐브 단면의 방향을 표시한다. 일반적으로 면 그래픽스에서 면의 방향 지정은 다각형 정점을 지정하는 순서(시계 방향, 시계 반대 방향)에 의해서 결정되기 때문에 필요 없으나 여기에서는 각 다각형에 대해서 면의 필드 할당이 하나뿐이기 때문에 이 필드가 필요하다. 경우 수 필드는 폴리곤의 종류에 따른 경우 수를 표시하며 각 폴리곤은 다른 비트 수를 갖는다. 즉 삼각형은 3비트, 사각형은 4비트, 오각형은 5비트, 육각형은 2비트가 필요하다. 변 번호 필드는 다각형의 각 정점을 표현하기 위해서 설치했는데 다각형의 종류에 따라 비트 수가 다르다. 즉, 삼각형의 3정점, 사각형의 4정점, 오각형의 5정점, 육각형의 6정점을 표시하기 위해서는 한 개의 정점 당 4비트가 필요하기 때문에 각각 삼각형이 12비트, 사각형이 16비트, 오각형이 20비트, 육각형이 24비트 필요하다.

4. 복셀화 및 렌더링

물체를 복셀화하는 알고리즘으로서는 초기에 나온 것으로 분류(classification)를 이용한 공간 열거법(spatial enumeration)[23]이 있으나 계산 시간이 너무 많이 걸리는 단점이 있고 2차원 다각형 스캔라인 알고리즘을 3차원 다각형 스캔 컨버전(scan conversion)[1] 알고리즘으로 확장시킨 것이 있으나 복잡하다. 그리고 위의 알고리즘은 본 논문에서 제안하는 복셀을 여러 비트로 표현하는 것과는 다르게 1비트로 표현하는 것이어서 여기에서는 그래픽스 젬(graphics gems)에 나와있는 큐브와 물체와의 교차계산을 하는 방법을 이용하여 복셀화 하였다. 제안하는 방법은 기본적으로 물체의 바운딩 볼륨(bounding volume)을 구한 다음에 이 바운딩 볼륨에 속해있는 각각의 복셀과 물체와의 교차 계산을 하여 복셀의 각 변과 물체와 만나는 교차점을 구하여 이 교차점을 부호화하여 복셀에 표시한다. 삼각형과 복셀의 교차 판정은 Douglas Voorhies의 방법을 사용하고[6] 음함수와 복셀의 교차판정은 James Arvo의 방법을 사용하였다[10]. 물체를 복셀화하는 알고리즘은 다음과 같다.

① 물체공간(object space)에 있는 각 기본 물체에 대하여 바운딩볼륨(bounding volume)을 구한다. 바운딩 볼륨 내부의 복셀이 해당 물체를 포인팅한다.
② 바운딩 볼륨의 계층 트리(hierarchy tree)를 구성한다. 계층 트리의 잎 노드(leaf node)는 기본 물체로 구성되고 내부 노드(inner node)는 유니온(union) 연산자로 구성된다.
③ 잎 노드의 각각에 대하여 바운딩 볼륨 내부에 있는 물체와 바운딩 볼륨을 구성하는 복셀과 교차 계산을 한다. 이때 물체가 이차 곡면으로 이루어져 있으면 James arvo 알고리즘을 이용하고 삼각형으로 이루어져 있으면 Douglas Voorhies 알고리즘을 이용한다. 물체와 교차하는 복셀에 대해서는 3장의 방법을 이용하여 부호화한다.
④ 각각의 잎 노드를 이용하여 유니온 연산을 하여 복셀화된 기본 물체를 통합한다. 이때 하나의 복셀 내부에 2개의 면을 만나면 이 2개의 면의 방향을 보간하여 하나의 면으로 만든다. 따라서 면과 면이 교차하는 곳에서의 법선 벡터의 방향은 두 면의 중앙을 향하게 된다.

James arvo 알고리즘과 Douglas Voorhies 알고리즘은 물체와 큐브를 구성하는 성분과 모두 교차계산을 하지 않고 교차 판정을 할 수 있어 매우 고속이다. James arvo 알고리즘은 타원 등의 이차 곡면으로 확장할 수 있으나 구와 복셀의 교차계산에 대하여만 고찰한다. 복셀 B가 반경 r의 3차원 구 C와 교차한다고 하고 구의 중심을 c_1, c_2, c_3라고 하자. 복셀 B는 3 방향의 범위 $[b_1^{\min}, b_1^{\max}]\cdots,[b_3^{\min}, b_3^{\max}]$로 표시된다. 구의 중심에서 가장 가까운 복셀 위의 점을 P라고 하면 이 점이 r 보다 크지 않을 때 복셀은 구와 교차한다. 이를 C 언어 형식의 의사 코드(pseudo code)로 기술하면 다음과 같다.

```
int SphereVoxelIntersect (C, r, B)
{
        d = 0 ;
        for (x, y, z를 갖는 각각의 i에 대하여)
        {
                if ( c_i < b_i^min )
                        d = d + (c_i - b_i^min)^2 ;
                else if (c_i > b_i^max) ;
                        d = d + (c_i - b_i^max)^2 ;
        }
        if (d > r^2) 교차하지 않음을 되돌린다 ;
        교차함을 되돌린다 ;
}
```

복셀과 삼각형 교차 판정의 단순한 방법은 복셀의 모든 변과 삼각형 또는 삼각형의 모든 변과 복셀을 구성하는 면과 교차계산을 하는 것이나 계산량이 너무 커 Douglas Voorhies는 대다수의 경우에 면과 교차계산 없이 교차 판정을 할 수 있는 방법을 제안했는데 이 알고리즘은 다음의 3 단계로 구성된다.

① 삼각형의 정점이 복셀을 이루는 6면의 내부에 있는가 확인하여 정점이 하나라도 내부에 있으면 교차로 판정한다. 삼각형의 모든 정점이 다음과 같이 만들어진 면의 외부에 있으면 교차하지 않음으로 판정한다.
  ①-1 복셀을 이루는 원래의 6면
  ①-2 복셀의 12개의 변과 45도로 교차하는 면
  ①-3 복셀의 중심에서 출발하여 꼭지점으로 나오는 선이 꼭지점에서 수직으로 교차하는 8개의 면
② 삼각형의 3변에 대하여 하나씩 복셀과 교차계산을 하여 하나라도 교차하는 변이 있으면 교차로 판정한다.
③ 복셀안에 4개의 대각선에 대하여 하나씩 삼각형과 교차계산을 하여 하나라도 교차하면 교차로 판정한다.

위의 교차 판정 처리는 복셀의 좌표를 좌표계의 중심으로 이동시켜 -0.5~+0.5 범위에 위치시켜 처리하면 ①과정의 처리를 보다 간단히 할 수 있다. ①-1에서 삼각형의 모든 정점이 복셀을 이루는 6면의 외부에 존재하는가?의 판정은 정점이 x, y, z값이 -0.5 이하에 있고 0.5 이상에 있는 것만 확인하면 된다. ①-2에서 삼각형의 정점이 변과 교차하는 면의 외부에 존재하는가? 의 판정은 예를 들어 $x+y>1.0$과 같이 삼각형 정점의 좌표 성분을 더하여 1과 비교하면 된다. ①-3 처리도 $x-y+z>1.5$와 같이 간단히 할 수 있다. ②의 계산은 삼각형의 각 변의 두 정점 P_1, P_2에 대하여 매개변수 방정식을 $\alpha P_1+(1-\alpha)P_2$를 세워 복셀의 각 면과 교차 계산을 하여 만일 결과 값이 ±0.5 이내에 있으면 교차로 판정한다. ③의 계산은 삼각형의 법선 벡터 A, B, C를 이용하여 평면의 방정식 $AX+BY+CZ+D=0$를 세운다. 그리고 복셀의 대각선의 점들은 $X=\pm Y=\pm Z$이라는 조건을 이용하여 위의 방정식을 풀어 X, Y, Z가 ±0.5 이내에 있으면 교차로 판정한다.

복셀의 변과 물체와 교차 계산(edge-object intersection)은 광선 추적(ray tracing)알고리즘에서 광선과 물체의 교차 계산과 유사하다. 즉 광선추적 알고리즘에서 광선과 다각형 또는 광선과 이차 곡면과의 교차 계산을 하는데 광선 대신에 큐브의 변을 이루는 직선을 사용하면 동일한 알고리즘으로 복셀의 변과 물체와의 교차점을 계산할 수 있다. 광선과 삼각형, 광선과 구와의 교차 계산은 각각 Moller의 방법과 Haines의 방법을 사용하였다[21]. 교차점은 언제나 복셀의 변 위에 존재하며 이 값은 소수가 되는데 16진수의 정수로 표시하기 위해서 이 소수에 16을 곱하여 3장에서 설명한 방법을 이용하여 정수만을 취하여 부호화한다.

일단 물체 공간에 있는 모든 물체들이 큐브 단면의 볼륨 데이터로 표현되면 이것을 볼륨 데이터 또는 기하 데이터로도 취급할 수 있어 이산 광선 추적법, 레이 캐스팅(ray casting), Z 버퍼 기반의 면 기반 렌더링(Z-buffer based surface rendering)중 어느 것이나 이용하여 렌더링 할 수 있다. 광선 추적법은 광선과 물체와의 교차점에서 이차 광선과 그림자 광선(shadow ray)을 재귀적으로 생성하며 이것을 이용하여 광역 조명(global illumination)을 계산한다. 이때 모든 기본물체는 서로 연결되어 있어야 한다. 만일 추적하고 있는 광선이 기본물체가 서로 완전히 연결되어 있지 않은 곳으로 통과하게 되면 구멍이 생긴다. 다각형들은 정점을 매개로 이웃과 완전히 연결되어야 한다. 또 정확한 반사 광선을 계산하기 위해서는 복셀의 법선 벡터를 계산할 수 있어야 한다. 필터링 기법으로 생성된 물체는 밀도 데이터로 면에 대한 기하학적인 정의가 없기 때문에 광선과 면과의 교차 계산은 쉽지가 않았다[20]. 그러나 본 논문에서의 광선과 면과의 교차 계산은 아주 쉽다. 왜냐하면 각 복셀이 큐브 단면의 기하학적인 정보를 가지고 있고 큐브 단면들은 서로 완전히 연결되어 있기 때문이다.

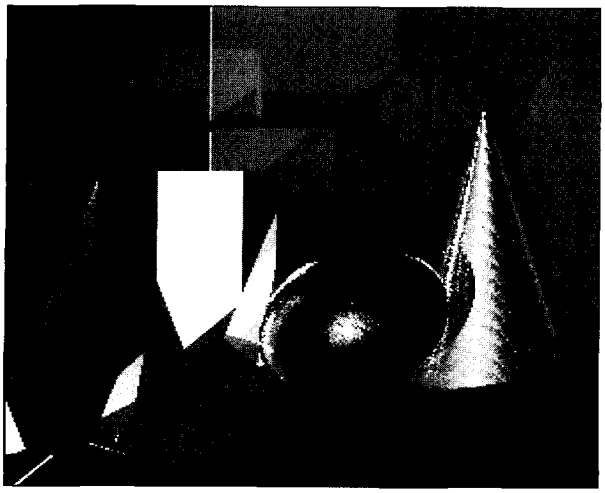
5. 구현 결과 및 평가

본 기법의 타당성을 검토하기 위해서, 구현 목적을 면 법선 벡터의 정확성 검증, 큐브상의 다각형의 정점이 서로 연결되는가의 확인, 복셀화 및 렌더링 시간 측정, 다해상도 표현의 가능성 확인으로 정했다. 그리고 윈도 98 환경에서 Visual C++를 이용하여 복셀라이저(voxelizer) 및 광선 추적기(ray tracer)를 구현하였다. 플랫폼이 되는 개인용 컴퓨터는 450MHz Pentium Ⅲ CPU 및 192MB RAM의 주기억장치를 가지고 있다.

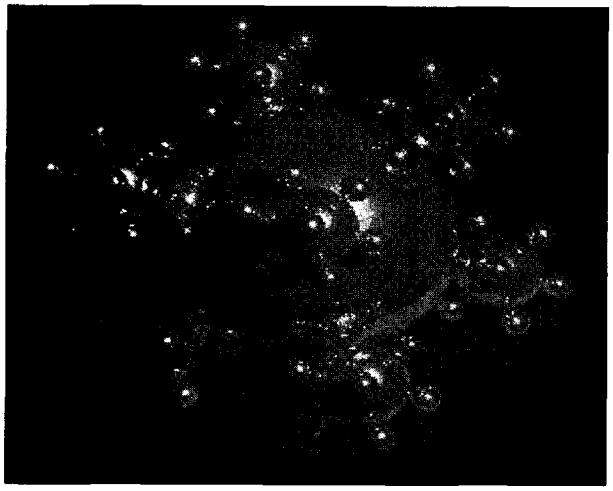
시험판의 광선추적기를 고속 광선추적 알고리즘(fast voxel traversal algorithm)[9]을 기반으로 구현하였다. 이 알고리즘은 복셀 추적 과정(voxel traverse)에서 구멍이 발생하지 않기 때문에 본 복셀화 기법의 평가용 첫 버전으로서 이것을 선택하였다. 이것 이외에 본 광선추적기는 투시 투상 기능, 그림자 광선의 생성 등의 기능을 가지고 있다. 그림자 광선은 교차점에서 광원을 향하는 광선인데 이것을 이용하면 복셀 추적을 이용하여 그림자를 생성할 수 있다.

면 법선 벡터의 정확성 검증과 모든 정점이 과연 정확히 연결되어 있는가? 하는 점을 확인하기 위해서 두 가지 평가용 모델을 만들었다. (그림 5)는 240×240×240의 복셀 해상도로 구성된 장면인데 이것은 50×50×50의 구, 70×160×70의 직각기둥, 80×120×80의 피라미드, 48×190×48의 원추, 1×240×240의 평면을 3, 4장의 방법을 이용하여 복셀화하고 복셀 블록 트랜스퍼 연산을 이용하여 합성하였다. (그림 6), (그림 7), (그림 8)는 73개의 구와 1개의 평면으로 이루어진 장면인데 복셀 해상도는 각각 140×140×140, 240×240×240, 340×340×340이다. 이것들은 본 복셀화 방법을 이용하여 한번에 생성하였다. 스크린의 해상도는 600×600으로 하였다. 본 논문에서 이상으로 하는 복셀 크기는 픽셀 정도로 작게 하는 것이나 PC 주 기억의 한계 등 현실적인 문제 때문에 복셀의 크기를 픽셀 크기보다 크게 하였다.

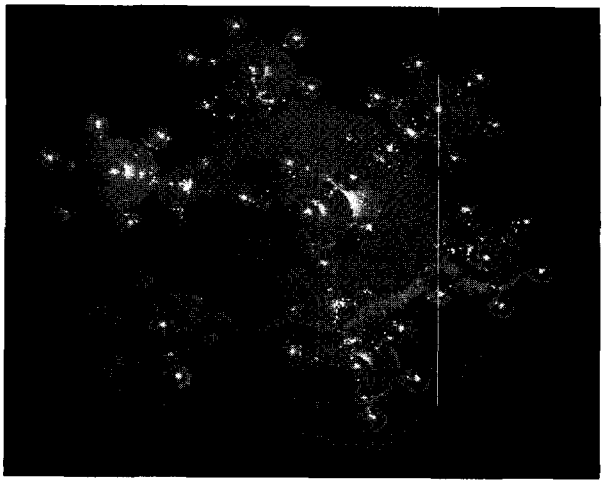
(그림 5)를 관찰하면 직각 기둥, 피라미드, 평면 등의 다각형으로 구성된 물체의 법선 벡터는 비교적 정확하나 구, 원추 등의 이차 음 함수 면으로 이루어진 물체의 법선 벡



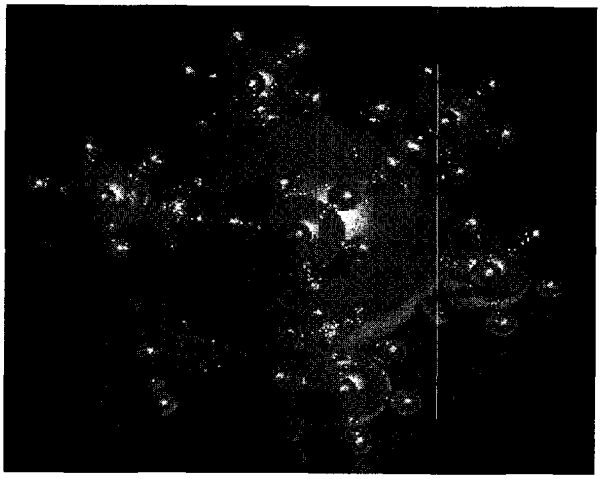
(그림 5) 다각형 복셀 및 이차 곡면 복셀로 구성, 복셀 해상도 : 240^3



(그림 6) 73개의 구 복셀 및 1개의평면 복셀로 구성, 복셀 해상도 : 140^3



(그림 7) 73개의 구 복셀 및 1개의 평면 복셀로 구성, 복셀 해상도 : 240^3



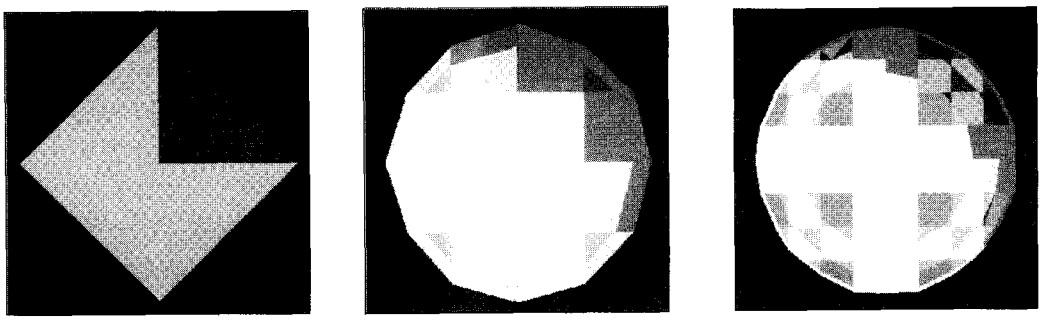
(그림 8) 73개의 구 복셀 및 1개의 평면 복셀로 구성, 복셀 해상도 : 340^3

터는 오차가 있음을 알 수 있다. 이것은 곡면을 한정된 크기를 갖는 다각형으로 근사하였고 복셀의 해상도가 픽셀의 해상도보다 낮기 때문이라 생각된다. 복셀 수를 증가시키면 이러한 문제는 해소되리라 생각된다. 일반적으로 필터링 기법을 이용한 복셀화에서는 다각형의 변과 정점을 명확하고 예리하게 표현할 수 없는 단점이 있으나 (그림 5)의 직각기둥과 피라미드에서 알 수 있듯이 본 논문의 기법은 변과 정점을 명확하고 예리하게 표현하였다. 투시 투상을 하고 측면에서 보았음에도 (그림 5)는 어떤 구멍도 발생하지 않음을 보이고 있다. 이것은 큐브상의 다각형의 정점이 모두 이웃 큐브의 정점과 연결되어있음을 의미한다.

복셀화에 걸리는 시간은 (그림 5)의 50×50×50 구가 1초, 48×190×48 원추가 6초, 80×120×80 피리미드가 7초, 70×160×70 직각기둥이 35초 (그림 6), (그림 7), (그림 8)의 73개의 구 및 한 개의 평면으로 된 복셀 해상도 140×140×140, 240×240×240, 340×340×340 구가 각각 2분 1초, 9분 38초, 24분 27초가 걸렸다. 복셀화 시간이 광선 추적법에서와 같이 물체의 개수 및 해상도가 증가함에 따라 빠르게 증가하고 있는데 이것은 전처리 단계에서 이루어지기 때문에 큰 문제는 없으리라 생각되나 시간을 단축하기 위해서 보다 더 연구가 필요하다.

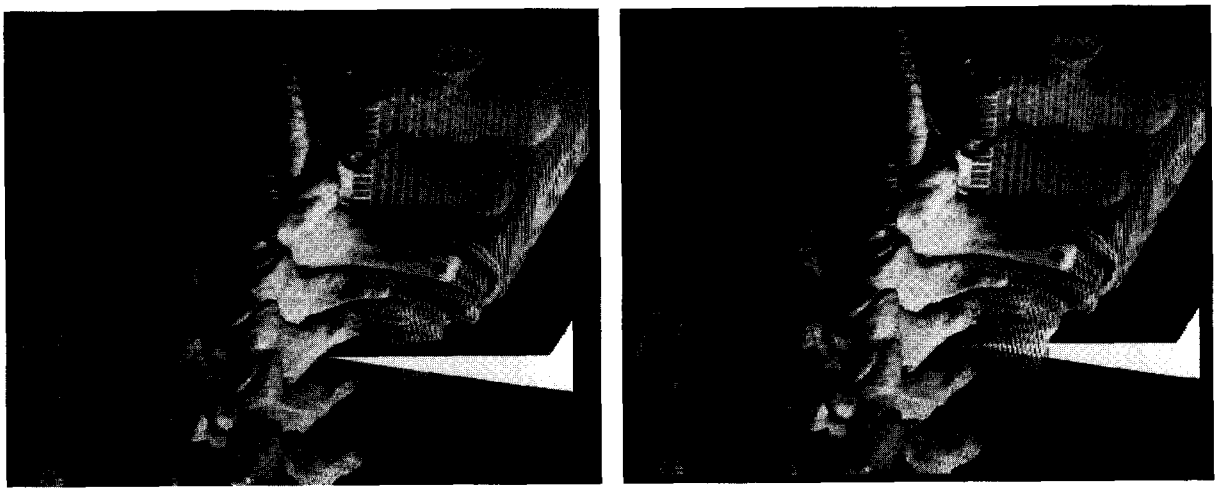
(그림 5)의 광선 추적 시간은 600×600 스크린 해상도에서 7분 5초가 소요되었고 (그림 6), (그림 7), (그림 8)의 광선 추적 시간은 동일한 스크린 해상도에서 각각 2분 14초, 2분 36초, 8분 47초가 소요되었다. 여기서 광선 추적법 코드를 최적화하기 위한 어떤 시도도 하지 않았다. 또 결과는 광선 추적 시간이 복셀의 해상도가 증가함에 따라서 빠르게 증가함을 보였다. 그러나 복셀 수가 어느 한계 이상에서는 전체적인 영상의 질은 그다지 변하지 않았다.

다해상도 표현의 가능성을 조사하기 위해서 구를 낮은 복셀 해상도를 갖는 큐브 단면으로 모델링 하였다. (그림 9)(a), (그림 9)(b), (그림 9)(c)는 각각 복셀 해상도 2×2×



(a) 복셀 해상도 : 2^3 (b) 복셀 해상도 : 4^3 (c) 복셀 해상도 : 8^3

〈그림 9〉 구의 다해상도 표현



(a) 기하데이터가 샘플링 데이터 앞에 위치 (b) 기하데이터가 샘플링 데이터 뒤에 위치

〈그림 10〉 기하데이터와 샘플링 데이터를 함께 표시

2, 4×4×4, 8×8×8로 구를 근사화 한 것인데 본 기법이 다해상도 표현의 성질을 갖는 것을 보인다. (그림 9)에서 삼각형은 적색, 사각형은 녹색, 오각형은 청색, 육각형은 황색으로 표시하였다. (그림 9)(b)는 구와 유사하게 보이기 위해서 최소한 4×4×4의 복셀이 필요한 것을 보인다. 또 이 그림은 복셀 해상도가 증가할수록 구에 접근함을 보인다. 마지막으로 본 논문에서 제안된 기법을 레이캐스팅(ray casting)코드에 혼합하여 기하데이터와 샘플링 데이터를 함께 가시화 하였다. 샘플링 데이터는 스탠포드 대학의 SPINE으로 복셀당 8비트인데 여기에서는 이것을 32비트로 확대하여 본 논문에서 제안한 기법으로 이산화시킨 삼각뿔 기하데이터와 결합시켰다. 샘플링 데이터는 중앙 차분(central difference)을 기반으로 기하데이터는 다각형의 정점을 기반으로 법선을 구하였다. (그림 10)(a)는 기하 복셀을 샘플링 복셀의 외부에, (그림 10)(b)는 기하 복셀을 샘플링 복셀의 내부에 놓은 결과를 보인다.

<표 3>은 본 논문의 기법을 다른 대표적인 복셀화 기법과 개념적으로 비교한 것이다. 본 기법의 자료형은 폴리곤 정점으로 복셀 법선을 다각형 법선을 이용하여 계산하기 때문에 중앙 차분으로 계산하는 다른 기법들 보다 정확한 법선을 갖는다. 이웃 기본물체와의 연결 관계도 필터링 등의 기법에서는 연결이 보장되지 않아 확대나 회전 변환을 할 때 구멍이 생길 수 있다. 그러나 본 기법은 이러한 변환이 다각형 정점의 어파인(affine) 변환으로 면 그래픽스와 동일하게 구멍이 생기지 않는다. 단 다른 기법보다 많은 데이터를 갖는 단점이 있다.

〈표 3〉 다른 복셀라이제이션 기법과 비교

복셀라이제이션 기법	자료형	법선계산방법	법선의 정확도	이웃 프리미브와 연결 관계	복셀당 데이터 길이
큐브 단면 이용	폴리곤 정점	다각형 법선	매우 좋음	연 결	2~4 바이트
필터링	밀 도	중앙 차분(Central Difference)	좋 음	연결되지 않음	1~2 바이트
거리 필드	표면으로 부터 거리	중앙 차분(Central Difference)	좋 음	연결되지 않음	1~2 바이트
포인트 샘플링	이진 데이터	이웃 복셀로부터 추적	양 호	연결되지 않음	1 비트

6. 결 론

본 논문에서는 큐브의 단면을 이용한 복셀화의 한 기법을 제안하고 이 기법을 이용하여 구, 육각기둥, 원추, 평면을 복셀화하고 이 기본물체 볼륨들을 복셀 트랜스퍼 연산을 이용하여 결합시켜 볼륨으로된 모델링 데이터를 생성하였다. 다음에 이 모델링 데이터를 고속 광선 추적법을 이용하여 렌더링 하였다. 복셀 해상도 140×140×140~340×340×340과 스크린 해상도 600×600에서 복셀화시간은 1초~24분 27초가 걸렸다. 그리고 생성된 영상이 비교적 정확한 물체의 법선을 유지하고 구멍이 발생하지 않으며 다해상도 표현이 가능한 것을 확인하였다.

본 기법의 자료 구조는 다각형을 기반으로 하기 때문에 원래의 물체가 가지고 있는 표현법을 유지하고 물체의 법선을 정확하게 계산할 수 있게 하고 인접하는 물체와 서로 완전히 결합하며 다해상도 표현이 가능한 장점이 있다. 그러나 복셀 당 데이터의 길이가 길고 복셀화 시간이 긴 것이 단점이다. 이 문제는 현재 컴퓨터의 메모리가 대용량화, 고속화되고 있어 시간이 지나면 어느 정도 해결되리라 기대된다.

향후 연구 목표로 마칭큐브 알고리즘[22]에 의해서 생성된 다각형을 복셀화 하는 문제에 대하여 검토하고 있다.

참 고 문 헌

[1] A. Kaufman, Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces and Volumes, Computer Graphics, 21(4), pp.171-179, 1987.

[2] A. Kaufman, D. Cohen and R. Yagel, Volume Graphics, *IEEE Computer*, 26(7), pp.51-64, 1993.

[3] A. Kaufman, R. Yagel and D. Cohen, Intermixing Surface and Volume Rendering, *3D Imaging in Medicine* NATO ASI series, Vol.F60, pp.217-227, 1990.

[4] B. A. Payne and A. W. Toga, Distance Field Manipulation of Surface Models, *IEEE Computer Graphics and Applications*, 12(1), pp.65-71, 1992.

[5] D. Cohen-Or and A. Kaufman, Fundamentals of Surface Voxelization, *Graphical Models and Image Processing*, 57(6), pp.453-461, 1995.

[6] D. Voorhies, Triangle-Cube Intersection, *Graphics Gems III Edited by D. Kirk*, Morgan Kaufman, 1992.

[7] E-A. Karabassi, G. P. Papaioannou and T. Theharis, A Fast Depth-Buffer-Based Voxelization Algorithm, *journal of graphics tools*, 4(4), pp.5-10, 1999.

[8] F. D. IX and A. Kaufman, Incremental Triangle Voxelization, *Graphics Interface 2000*, pp.205-212, May, 2000.

[9] J. Amanatides and A, Woo, A Fast Voxel Traversal Algorithm for Ray Tracing, *EUROGRAPHICS87*, Elsevier Science Publishers B. V., pp.3-9, 1987.

[10] J. Arvo, A Simple Method for Box-Sphere Intersection Testing, *Graphics Gems edited by A. S. Glassner*, Morgankaman, 1998.

[11] L. S. Chen, G. T. Herman, R. A. Reynolds and J. K. Udupa, Surface Shading in the Cuberille Environment, *IEEE Computer Graphics and Applications*, 5(12), pp.33-43, 1985.

[12] M. Frhauf, Combining Volume Rendering with Line and Surface Rendering, *EUROGRAPHICS 91*, Elsevier Science Publishers B. V., pp.21-31, 1991.

[13] M. Sramek and A. E. Kaufman, Alias-Free Voxelization of Geometric Objects, *IEEE, Transactions on Visualization and Computer Graphics*, 5(3), pp.251-267, 1999.

[14] M. W. Jones, The Production of Volume Data from Triangular Meshes Using Voxelisation, *Computer Graphics Forum*, 15(5), pp.311-318, 1996.

[15] R. E. Webber, Shading Voxel Data via Local Curved-Surface Interpolation, *Volume Visualization*, IEEE Computer Society Press Tutorial, pp.229-239, 1991.

[16] R. Yagel, D. Cohen and A. Kaufman, Discrete Ray Tracing, *IEEE Computer Graphics and Application*, 12(5), pp.19-28, 1992.

[17] R. Yagel, D. Cohen and A. Kaufman, Normal Estimation in 3D Discrete Space, *The Visual Computer*, 8(5-6), pp. 278-291, 1992.

[18] S. Fang and H. Chen, Hardware Accelerated Voxelization, *Computers & Graphics*, 24(3), pp.433-442, 2000.

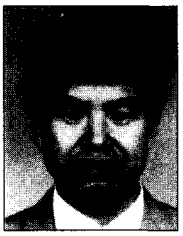
[19] S. Oomes, P. Snoeren and T. Dijkstra, 3D Shape Representation : Transforming Polygons into Voxels, *Scale-Space Theory in Computer Vision,* Springer Verlag, 1997.

[20] S. W. Wang, A. E. Kaufman, Volume-Sampled 3D Modeling, *IEEE Computer Graphics and Applications*, 14(5), pp.26-32, 1994.

[21] T. Moller and Eric Hains, Real Time Rendering, A. K. Peters, 1999.

[22] W. E. Loresen and H. E. Cline, Marching Cubes : A High Resolution 3D Surface Construction Algorithm, *SIGGRA PH87*, pp.163-169, 1987.

[23] Y. T. Lee and A. A. G. Requicha, Algorithms for Computing the Volume and Other Integral Properties of Solids. II. A Family of Algorithms Based on Representation Conversion and Cellular Approximation, *Communications of the ACM*, 25(9), pp.642-650, 1982.



권 오 봉

email : obgwun@moak.chonbuk.ac.kr

1980년 고려대학교 전기공학과(석사)

1983년 고려대학교 전기공학과(공학석사)

1992년 일본구주대학 총합이공학연구과 (공학박사)

1992년~1993년 일본구주대학 정보공학과 조수

1994년~1995년 전북대학교 컴퓨터과학과 전임강사

1996년~1999년 전북대학교 컴퓨터과학과 조교수

2000년~현재 전북대학교 전자정보공학부 부교수

관심분야 : Visualization, Computer Graphics, Parallel Processing