한국 CAD/CAM 학회 논문집
제7권 제2호 2002년 6월 pp. 81-88

학술논문

삼차원 메쉬 모델의 압축 및 점진적 전송을 위한 가수부 분할 기법

김덕수*, 정재열**, 김 현***

Mantissa Chunking Algorithm for the Compression and Progressive Transmission of 3D Mesh Models

Kim, D. S.*, Chung, J. Y.** and Kim, H.***

ABSTRACT

Transmission of 3D shape model through Internet has become one of the hottest issues in these days. Presented in this paper is a new approach for the rapid transmission of the geometry data of the shape model. By analyzing the important three factors, the shape fidelity, the file size, and the decompression time, for the compression, we point out the potential problems of previous approaches of using the deltas between consecutive vertices and propose an alternative of directly using the position values of vertices of the model. It turns out that the proposed approach has smaller file size, has lesser distortion in the model, and the decompression is faster.

Key words : 3D Shape, Mesh, Compression, Decompression, Mantissa, Progressive Transmission

1. 서 론

인터넷 이용이 전 세계 대부분의 사람들에게 일상적인 일이 됨과 함께 인터넷을 통한 3차원 형상 모델의 교환 문제 또한 요구되고 있다. 하지만, 지금도 그렇지만 앞으로도 네트워크의 전송속도의 증가율은 전송되어질 모델의 크기의 증가율 보다는 낮을 것이다. 따라서, 네트워크를 통한 데이터의 효율적인 교환은 지속적으로 연구의 초점이 될 것이며, 3차원 형상 모델의 경우도 예외는 아니다. 본 논문에서는 3차원 형상 모델과 메쉬 모델, 그 중 특히 삼각형 메쉬로 이루어진 모델을 주로 다룬다. 그리고 특별히, 메쉬 모델은 방향성을 가진 다면체(orientable manifold)라고 가정한다.

본 논문에서는 모델의 기하정보의 압축 및 점진적 전송을 하기 위해 가수부(mantissa)의 bit chunk에 서로 다른 우선순위를 부여하는 방법을 제시한다. 또한 기하정보의 압축은 대응되는 위상정보와 밀접한 관계가 있기 때문에 기하정보의 압축에 대해서도 다루게 된다.

일반적으로 3차원 형상 모델의 B-rep을 위해 디자인된 Winded-Edge 데이터 구조 같은 표현방법조차도 일정한 정도의 중복되는 정보를 포함하고 있다. 이와 같은 중복된 정보는 여러 종류의 어플리케이션에서 발생할 수 있는 쿼리나 조작에 빠르게 반응할 수 있도록 도와준다. 하지만 이러한 중복된 정보는 모델 저장을 위한 파일의 크기를 증가시키며, 이는 전송의 지연을 불러일으킨다. 파일 크기를 줄이기 위한 압축작업은 이러한 중복된 정보를 제거하는 과정과 관계가 있다. 따라서 이러한 압축작업을 통해 원래의 모델은 중복된 정보 중 원하는 만큼만의 정보를 포함하게 되고, 특정한 어플리케이션에서 좀 더 빠르게 전송되어 복원될 수 있다. 압축을 하는데 중요한 세가지 기준은 형상의 정확도, 압축률, 그리고 압축을 푸는데 걸리는 시간이다. 압축은 일반적으로 오프라인 작업이므로 그에 걸리는 시간은 대개 고려대상에서 제외된다.

2. 관련 연구

메쉬 모델은 일반적으로 위상정보, 기하정보, 법선벡터, 컬러, 텍스쳐 좌표, 텍스쳐 파일 등으로 이루어져 있다. 본 논문에서 기하정보는 메쉬 모델의 정점의 좌

*종신회원, 한양대학교 산업공학과
**학생회원, 한빛전자통신(주) 중앙연구소
***종신회원, ETRI
- 논문투고일: 2001. 9. 4
- 심사완료일: 2001. 11. 5

표값을 의미한다. 이러한 요소들의 실제 크기는 모델을 얼마만큼 정밀하게 표현하느냐에 따라 달라지지만 최악의 경우의 한계는 계산할 수 있다. 예를 들어 V를 메쉬 모델의 정점의 개수라고 할 때, 위상정보와 기하정보 부분은 각각 2×3×V×4바이트(일반적으로 위상정보는 기하정보의 두 배 정도라고 알려져 있기 때문에)와 3×V×4바이트 정도를 필요로 한다.

본 논문에서 우리는 형상 모델의 기하정보와 위상정보를 압축하는 방법에 대해 논할 것이다. 위상정보의 압축에 대해서는 지금까지 많은 학자들이 연구해 왔고, 좋은 결과를 얻어내었기 때문에 위상정보를 압축하는 기술은 현재 잠정적인 이론상의 한계에 도달해 있다고 생각한다. 현재 모델 전체의 위상정보를 표현하기 위해 삼각형 하나 당 2비트를 필요로 하는 알고리즘이 나와 있으며 이는 모델을 구성하는 삼각형의 패턴을 고려하여 더 줄일 수도 있다[13].

반면에 기하정보의 압축에 대해서는 아직 많은 연구가 이루어지지 않았으며 따라서 우리는 개선의 여지가 있다고 믿는다. 본 논문의 주된 논점이 기하정보이기는 하지만 압축된 기하정보가 위상정보와 밀접한 연관이 있기 때문에 우리는 두 가지를 병행하여 설명할 것이다.

2.1 위상 정보의 압축

일반적으로 위상 정보를 압축하기 위해서는 비손실 압축방법이 사용된다. 일반적으로 위상정보는 기하정보의 약 두 배이므로 위상정보를 효율적으로 압축하는 알고리즘을 고안하는 것이 전체 모델의 압축률을 더 좋게 할 수 있다[8]. 그러므로 지금까지 모델의 압축에 대해 논할 때 전 세계 학자들의 관심사는 주로 위상정보의 압축에 있었다.

1995년에 Deering은 OpenGL등과 같은 그래픽 라이브러리에 사용되고 있는 Generalized Triangle Strip의 확장인 Generalized Triangle Mesh(GTM)를 사용한 선구적인 이론을 제안하였다[3]. Deering의 알고리즘은 정점 하나당 약 11비트를 필요로 한다. Taubin과 Rossignac이 제안한 Topological Surgery에서는 위상정보를 나타내기 위해 Vertex Spanning Tree와 이것의 듀얼(dual)인 Triangle Spanning Tree를 구축하였다. 이 알고리즘은 평균적으로 정점 하나당 4비트를 필요로 한다[15]. 1997년에 Chow는 임의의 메쉬를 Deering의 GTM 구조로 바꾸는 Meshfying Algirithm을 제안했다[2]. Touma와 Gotsman은 메쉬에서 삼각형의 에지를 따라 생기는 정점의 주기적인 순서인 Vertex Cycle을 사용한 알고리즘을 발표하였으며[16], Gumhold와 Strasser는 Cut-Border 데이터 구조를 사용한 알고리즘을 제안하였다[6]. 그 후 1999년에 Rossignac이 메쉬를 구성하는 삼각형들을 C, L, E, R, S의 심벌로 표현하고 이 각각의 심벌을 엔트로피 코딩을 하여 파일 크기를 줄이는 Edgebreaker라는 알고리즘을 제안하였다[12]. 반면, Kim *et al.*은 기존의 연구와는 완전히 다른 방식의 알고리즘을 제안하였다[10]. 그 논문에서는 메쉬의 위상정보가 주어진 정점집합의 Delaunay Triangulation과 모델의 주어진 메쉬 구조와의 차이로써 정의된다. 그러므로 그의 알고리즘은 모델이 2차원적으로 가시적일 경우에만 적용될 수 있다.

2.2 기하 정보의 압축

일반적으로 그래픽과 관련된 기법에서 실수는 연속되는 4바이트(즉, 32비트)로 정의되는 Float자료형으로 표현이 되며, 모델의 각 정점의 x, y, z 성분이 모두 이 실수로 표현이 된다. 첫 번째의 부호 비트 다음에 나타나는 8비트는 지수부분을 나타내고 나머지 23비트는 가수부분을 나타낸다[1]. 일반적인 IEEE 32비트 부동소수점 표기법은 150억 광년에서부터 원자의 반지름에 이르는 크기를 표현할 수 있다고 한다[3].

위상 정보의 압축이 비손실 압축인 반면 기하정보는 일반적으로 손실 압축이다. 기하정보의 압축이 가능한 이유는 우선, 대상 모델을 표현하는데 중복되는 정보가 존재한다는 점이다. 그리고 일반적인 32-비트 IEEE 부동 소수점 표기법은 모델을 표현하기에 지나치게 정밀하다는 점이다. IEEE 32-비트 부동소수점 표기법으로 표현이 되는 실수 축 상의 420억 개의 서로 다른 수들은 일반적인 모델을 표현하기에는 너무 많다. 정점들은 Float 자료형으로 표현한다면 $(420억)^3$개의 유일한 점들이 생성될 수 있으며 이들 거의 대부분은 모델의 기하정보를 나타내는데 불필요하다.

이러한 이유로 기하정보의 압축 문제는 발전할 수 있는 이슈가 되었다. 정점의 좌표값들 또는 정점간의 차이 벡터들 사이의 중복된 정보의 일부분을 제거하여 시각적으로 거의 변화를 주지 않을 정도로 모델을 변화시킴으로써 압축을 할 수 있다.

일단 중복된 정보가 발생하면 그러한 정보는 고정적이거나 가변적인 길이를 가진 코드로 코딩이 될 것이다. 일반적으로 중복된 정보의 빈도수를 고려한 허프만 코딩을 사용함으로써 좋은 압축률을 얻을 수 있다.

정점들의 좌표값의 차이(이후 델타라 한다)의 절대값은 일반적으로 정점들 그 자신보다 작은 값을 가지기 때문에 대부분의 학자들은 델타를 사용해 왔다. 이러한 관찰에 기초하여 1995년에 Deering은 델타의 좌

표값의 가수부분의 16비트를 잘라내어 허프만 코딩을 함으로써 압축을 하는 방법을 제시하였다[3]. 그리고 1996년에 Taubin과 Rossignac은 Deering의 연구를 일반화시켜, 앞선 k개의 연관된 정점들을 사용하여 현재의 정점의 위치를 예측하는 방법을 제안하였다[Taubin 1996]. 한편, Touma와 Gotsman은 평행사변형 법칙을 사용한 방법을 제안하였다[16]. 그의 연구에서는 앞선 삼각형의 세 정점과 예측하고자 하는 새로운 정점이 평행사변형을 이룬다는 가정하에 새로운 정점을 예측하여 실제의 정점의 위치와의 차이를 인코딩한다. 2000년에 Lee와 Ko는 다른 모델링 공간으로의 매핑 사이의 차이를 사용하여 Touma와 Gotsman의 방법과 유사한 알고리즘을 제시하였다[11].

하지만 위와 같은 기법들은 기하정보의 점진적인 전송문제까지 동시에 고려하지 않았다는 문제가 있었다. 그리고 32비트의 일부분을 무시함으로써 발생하는 에러들을 보완하는 방법에 대한 고려가 없었다. 그러므로 우리는 개념면에서나 구현면에서 쉽고 또 앞서 언급한 두 가지 문제를 해결할 수 있는 방법을 제안한다.

3. 델타를 사용할 경우 에러와 압축률 사이의 Trade-off

모델의 기하정보를 압축하기 위해 Deering은 GTM에 의해 순서가 결정된 연속된 정점들 사이의 델타를 계산하였다[3]. 그 후 각각의 델타의 좌표값들은 가수부분을 앞에서부터 16비트만큼 잘라내어 허프만 코딩을 하였다. 여기서 Deering은 통계적인 분석을 통해 만약 델타의 가장 큰 성분을 나타내기 위해 n비트를 사용하였다면 다른 두 성분의 표현을 위해서는 평균 n-1.4비트가 필요하다고 하였다. 그것은 앞서 말한 16비트 심벌들을 위해 단지 하나의 지수값을 사용한다는 것을 의미하는 것 같다.

델타들은 분포를 하고 있고 그러한 분포에는 반드시 변동이 있다. 게다가 실수집합을 전송하기 위해서는 가수부와 지수부가 동시에 전송이 되어야 한다. 이를 압축하기 위해서는 가능한 가수부나 지수부를 제거하는 것이 좋다. 일반적으로 지수의 분포가 가수의 분포보다 변동이 적기 때문에 실수의 지수부분을 동일하게 만들고 몇몇 수들의 가수부를 지수값의 차이만큼 쉬프팅 시키게 된다. 이때 델타를 표현하기 위해 사용된 비트수가 쉬프팅시킨 비트 수에 가까울수록 잃어버리는 비트가 많아져서 나쁜 결과를 초래한다. Deering의 경우 델타의 지수들의 최대 변동의 크기가 16 이상이면 쉬프팅 되는 비트 수가 16 이상이므로 일부 델타가 실제 값에 상관없이 0이 되는 문제가 발생한다.

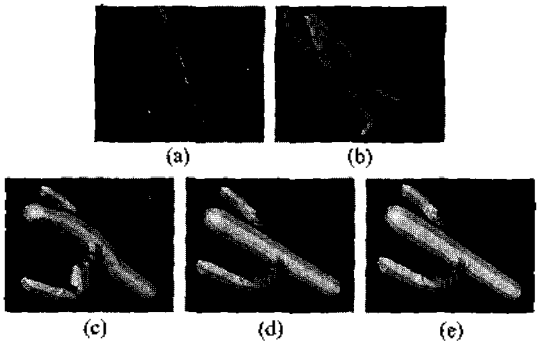


Fig. 1. The distortion of cactus model by compressing deltas with one exponent value; (a) 4-bit, (b) 8-bit, (c) 12-bit, (d) 16-bit precision, (e) the original model.

만약 메쉬 모델의 에지의 길이가 일정하다면 GTM은 바로 다음 순서의 정점을 '인접' 정점으로 선택하기 때문에 Deering의 기하정보 압축 방법은 별 문제 없이 수행될 수 있다. 하지만 만약 정점들 사이의 거리의 분산이 크도록 정점의 순서를 생성해 내는 다른 위상정보 압축 알고리즘과 함께 사용된다면 심각한 결과를 초래하게 된다. 게다가 정밀도를 낮추면 이 문제는 더 심각해진다. 심지어 GTM을 사용한다 해도 메쉬를 구성하는 삼각형들의 크기가 다양할 경우 이러한 문제는 발생할 수 있다. 그 이유는 메쉬 모델은 네트워크 상에서 전송되기 전에 종종 단순화되며, 파일 크기를 줄이기 위한 이 과정에서 메쉬는 불규칙적인 크기로 변화되기 때문이다.

Fig. 1에 보이는 선인장 모델은 이러한 문제를 보여주고 있다. 우리의 실험에서는 정점의 순서를 생성해 내기 위해 Edgebreaker를 사용하였고, 생성된 정점들 사이의 델타를 압축하기 위해 Deering의 방법을 사용하였다.

Fig. 1의 (e)는 원래의 선인장 모델이며, (a), (b), (c)와 (d)는 각각 델타를 4, 8, 12, 16비트의 정밀도로 압축한 후 하나의 지수값과 함께 전송하여 압축을 푼 것이다. (d)에서 볼 수 있듯이 Deering이 제시한 대로 16비트로 압축할 경우 원래 모델과 시각적으로 큰 차이를 느낄 수 없지만 4, 8, 그리고 12비트로 압축한 모델은 눈에 거슬리게 뒤틀려 있다. 다른 예제로 고양이 모델을 실험한 결과가 Fig. 2에 있다.

우리의 실험에서, 그러한 모델의 왜곡은 주로 Edgebreaker를 사용할 경우 나타나는 몇몇 S 노드에 기인하는데, 그 이유는 S 노드에 의해 생성되는 새 정점과 바로 앞 순서에 해당하는 정점 사이의 델타는 대

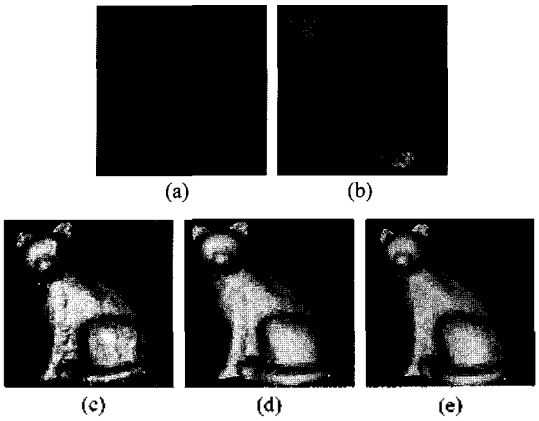


Fig. 2. The distortion of cat model by compressing deltas with one exponent value; (a) 4-bit, (b) 8-bit, (c) 12-bit, (d) 16-bit precision, (e) the original model.

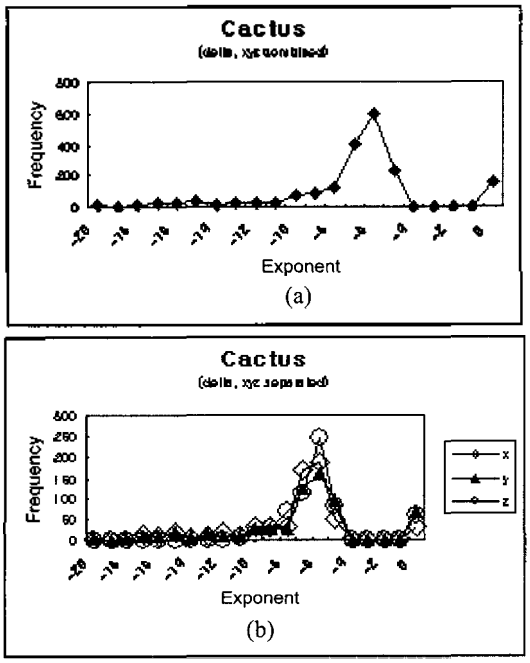


Fig. 3. The distribution of the values of the exponents of deltas for the cactus model; (a) all coordinates together, (b) X, Y, and Z coordinates separated.

개는 C 노드에 의한 델타보다 크다는 데 있다. 그러므로 모든 델타의 지수를 강제로 가장 큰 것으로 고정시킬 경우 에러가 발생하는 것은 필연적이다. 일단 에러가 발생하게 되면 이 에러는 모델의 나머지 부분의 모든 정점에 누적이 되며, 추후 발생할 수도 있는 에러와

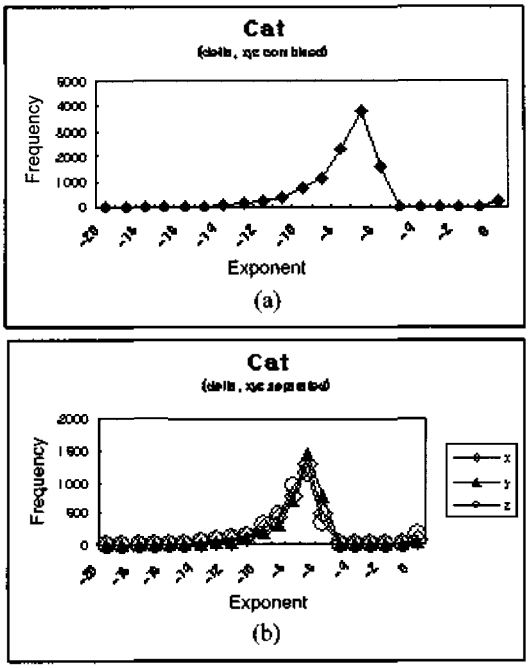


Fig. 4. The distribution of the values of the exponents of deltas for the cat model; (a) all coordinates together, (b) X, Y, and Z coordinates separated.

누적이 된다.

Fig. 3은 앞서 보여준 선인장 모델의 델타의 지수값의 분포를 나타낸 것으로, 이 사실을 명백히 보여주고 있다. Fig. 3의 (a)는 델타의 모든 좌표값에 대한 지수 분포를 나타낸 것이고 (b)는 델타의 X, Y, Z좌표를 각각 분리한 것의 지수값의 분포를 나타낸 것이다. 위 두 그림에서도 볼 수 있듯이 지수값의 가장 큰 값과 작은 값의 차이는 16보다 크다. Fig. 4는 고양이 모델에 대해서 Fig. 3과 같은 정보를 보여주고 있다. 주목할 것은 정점의 순서는 모두 Edgebreaker에 의해 생성된 것이라는 점이다.

지수값이 동일하여 가수부에서 쉬프팅이 발생하지 않는 경우에도 모든 델타에서 나머지 7비트는 잃어버리게 되며, 점의 개수가 증가할수록 에러는 누적이 된다. 16비트의 정밀도를 사용한다면 V를 모델의 정점의 개수라 할 때 최악의 경우의 에러(Worst-case Error)는 $(V-1)/2^{16}$이 된다.

4. 정점의 가수부의 직접 분할방법

앞서 설명한 우리의 실험에서 평균적인 압축률은 원

래 모델의 10% 정도인 것으로 나타났다. 우리는 이 압축률이 Edgebreaker를 이용한 위상정보의 압축과정에 기인했다고 믿는다. Deering의 알고리즘은 트리 정보의 저장을 위해 별도의 저장공간을 필요로 하는 허프만 코딩을 사용하고, 허프만 코딩은 데이터 집합으로 생성되는 트리 정보를 저장하기 위해 별도의 저장공간을 필요로 하기 때문에 만약 중복되는 데이터들이 적을 경우 파일 크기는 오히려 원래 파일보다 커질 수도 있다. 하지만, 델타들 사이에 얼마만큼의 중복이 있어야적절한 압축률을 얻을 수 있는가는 명확치 않으며, 델타들 사이에 많은 중복을 발생시키는 유일한 방법은 낮은 정밀도를 사용하는 것이다. 하지만 낮은 정밀도를 사용하면 모델의 왜곡이 일어날 가능성이 커진다.

특히, 미리 정한 정밀도에 따라 가수부분을 잘라내는 것에 의해 에러가 발생하며 이런 에러들은 누적된다. 따라서 본 논문에서는 정점의 좌표값의 델타를 사용하는 대신에 그 정점들을 직접 다루고자 한다. 이는 좌표값을 직접 전송한다면 발생되는 에러가 누적되지 않고 전체적으로 균일하게 분포할 것이기 때문이다. 게다가 데이터의 가수부분을 다루기가 쉬운 만큼 형상의 왜곡된 정도를 조절하기가 훨씬 수월할 것이다. Fig. 5와 Fig. 6에서 앞서 소개한 메쉬 모델의 정점의 좌표값의 지수값의 분포를 보여주고 있다. 주의할 것은 Fig. 5와 Fig. 6에서 나타내는 지수값은 델타들의 지수값의 분포가 아니라는 점이다.

선인장과 고양이 모델은 각각 620개와 3,545개의 정점들로 이루어져 있다. 하지만 Fig. 5와 Fig. 6에서도 볼 수 있듯이 각각의 모델 모두 4개의 서로 다른 지수

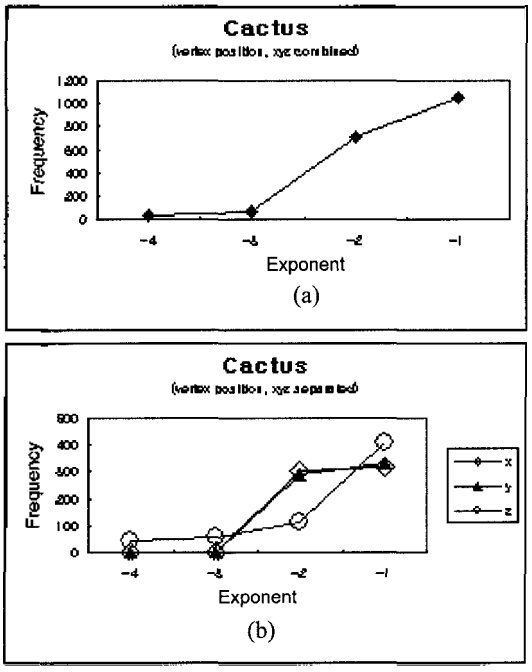


Fig. 5. The distribution of exponents of vertices for the cactus model; (a) all coordinates together, (b) X, Y, and Z coordinates separated.

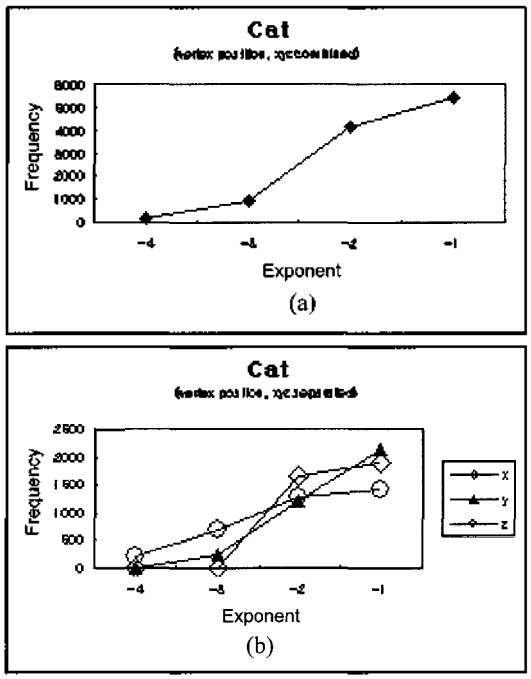


Fig. 6. The distribution of exponents of vertices for the cat model; (a) all coordinates together, (b) X, Y, and Z coordinates separated.

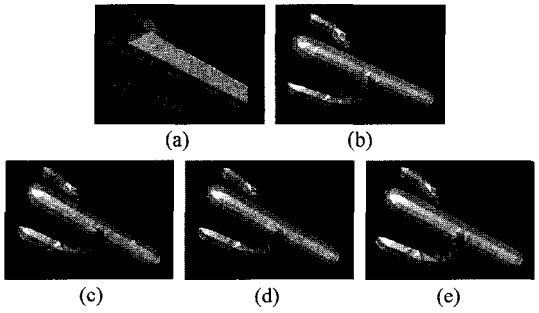


Fig. 7. Cactus model via mantissa chopping; (a) 4-bit, (b) 8-bit, (c) 12-bit, (d) 16-bit precision, (e) original model with full 23-bit precision.

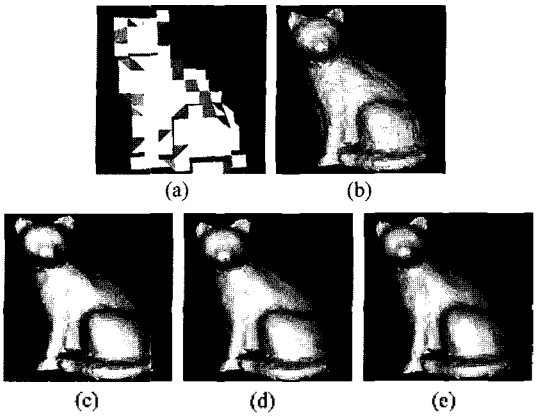


Fig. 8. Cat model via mantissa chopping; (a) 4-bit, (b) 8-bit, (c) 12-bit, (d) 16-bit, (e) original model with full 23-bit precision.

값만을 가지고 있으며 두 모델 모두 가장 큰 지수값과 가장 작은 지수값의 차이는 겨우 3이다.

만약 모델의 좌표값의 지수값이 동일할 경우 원하는 정밀도만 결정되면 요구되는 비트 크기를 알 수 있다. 예를 들어 1/1000의 정밀도로 압축하기를 원한다면, 가수부분의 10비트만을 잘라내어 압축하면 된다(본 논문에서는 논의상의 편리를 위해 숨겨진 비트(hidden bit)가 존재하지 않는다고 가정한다).

Fig. 7과 Fig. 8에서 그림 (a), (b), (c)는 모든 정점의 좌표값을 각각 4, 8, 12비트로 압축한 결과를 보여준다. 주목할 것은 4비트로 압축하였을 때 모델의 전반적인 형상이 보여지며, 모델에 따라 다르지만 대부분 8비트 정도면 원래 모델과 거의 유사한 결과를 얻을 수 있다.

만약 초기 모델에 4비트의 정밀도를 적용한다고 하면 16개(2^4)의 표현 가능한 좌표값이 나오므로 결국 표현할 수 있는 모든 정점의 수는 $16^3 = 4096$개가 된다. 따라서, 초기 모델은 원래의 모델에 비해 상대적으로 거칠게 보일 수 있다. 이 문제는 초기 모델이 좀 더 높은 정밀도를 가지도록 함으로써 해결할 수 있으나 일반적으로 초기모델로서는 원래 모델의 정점의 수와 무관하게 적절한 숫자인 것으로 사료된다.

5. Mantissa Chunking 기법을 이용한 데이터의 점진적 전송

앞 장에서, 우리는 정점들 사이의 델타를 전송하는 것 보다 정점의 좌표값의 가수부분을 일부를 잘라 내어 전송하는 방법이 더 낫다는 것을 보였다. 나아가 우리는 잘라낸 가수부분의 첫 4비트와 다음 4비트는 독립적이라는 것에 착안하여 잘라낸 가수부분에 서로 다른 우선순위를 부여하여 전송함으로써 모델을 점진적으로 보여주는 방법을 고안해낼 수 있었다.

Fig. 7과 Fig. 8을 다시 보자. 가수부분의 첫 4비트를 사용하여 초기 모델을 전송한 후에 나머지 19비트는 어플리케이션의 요구사항에 맞추어 여러 단계로 나누어 전송할 수 있다. 이러한 결정은 어플리케이션의 요구사항에 전적으로 의존하며 압축을 풀 때는 단지 정밀도에 따라 비트-쉬프팅 연산만을 수행하면 되기 때문에 Mantissa Chunking은 매우 간단하고 빠른 방법

Table 1. File size comparison between Deerings and the proposed algorithms

file size unit : Byte

Model	A	B	C	D	E	F			J	
	# of Triangle	# of Vertices	Original File Size (VRML, ASCII)	Zip File Size (VRML, ASCII)	Topology File Size (compressed)	Geometry File Size (compressed)			Total File Size (Compressed)	
						G	H	I	K	L
						Bit Precision	Deering	Mantissa Chopping (Incremental)	Deering	Mantissa Chopping
Cactus	1,236	620	43,192	14,626	309	4 bit	374	946	683	1,255
						8 bit	1,045	930	1,354	2,185
						12 bit	2,284	930	2,593	3,115
						16 bit	4,900	930	5,209	4,045
Cat	6,988	3,545	267,509	89,850	1,760	4 bit	2,302	5,334	4,062	7,094
						8 bit	7,347	5,318	9,107	12,412
						12 bit	13,862	5,318	15,622	17,730
						16 bit	31,224	5,318	32,984	23,048

Table 2. Decompression time comparison between Deerings and the proposed algorithms

time unit: millisecond

Model	M	N			R	
	Decompression Time (Topology)	Decompression Time (Geometry)			Total Decompression Time (milliseconds)	
		O	P	Q	S	T
		Bit Precision	Deering	Mantissa Chopping (Incremental)	Deering	Mantissa Chopping
Cactus	26.9444	4 bit	6.3	4.5	33.3	31.4
		8 bit	11.1	3.0	37.9	34.4
		12 bit	23.8	3.0	50.7	37.4
		16 bit	47.4	2.0	74.3	39.4
Cat	685.3889	4 bit	29.6	20.0	714.9	705.4
		8 bit	53.3	18.0	738.7	723.4
		12 bit	88.5	18.55	773.9	741.9
		16 bit	259.7	21.0	945.1	762.9

이다. 일단 초기모델의 기하정보의 압축이 풀려지고 모니터에 보여지는 동안에 m-비트의 추가의 가수 부분들이 전송이 된다. 그리고 이 추가 정보는 해당되는 좌표값에 단순히 더하기만 하면 되는 것이다. 본 논문에서 제안하는 방법은 어떤 면에서는 주어진 모델의 전체 공간을 균일한 크기의 bucket으로 분할하여 모든 정점들을 해당 bucket에 할당하는 것과 같으며, 각 4 bit 단위에 들어있는 정보는 각 bucket에 대한 암묵적인 포인터(implicit index) 와 같은 역할을 한다.

Table 1과 2에 Fig. 7과 Fig. 8에서 보여주었던 모델의 전송에 대한 통계자료가 있다. 자바로 작성된 소스코드는 아직 최적화가 되지 않았으며 128MB 메모리와 CPU Celeron 333인 컴퓨터에서 실행하였다.

F열 아래에 있는 H열은 Deering의 방법으로 G열에 나타난 정밀도로 압축한 기하 정보의 파일 크기를 나타낸다. I열은 우리가 제시한 방법으로, 각 단계마다 전송될 압축된 기하정보의 파일크기이다. 예를 들어 8bit라는 인덱스의 930이라는 숫자는 모델의 각각의 정점의 모든 좌표값에 대하여 쉬프팅된 가수부분의 5, 6, 7, 8번째 비트의 크기를 나타낸다. 예상했던 바이지만 추가되는 파일의 크기는 거의 상수이다.

압축된 전체 파일의 크기는 J열에 나타나 있다. 결과를 보면 제안한 알고리즘이 Deering의 알고리즘보다 나쁜 것처럼 보인다. 하지만 주목할 것은 제안된 알고리즘에서 선인장 모델의 경우 사용자는 첫 1,255 바이트를 전송 받는 즉시 모델을 볼 수 있다. 그 후 I열에 있는 크기의 파일이 전송되는 대로 모델의 좌표값이 수정되며 화면이 새롭게 갱신된다. 또한, 클러스터링 기법을 이용하여 지수값을 클러스터링한 후 허프만 코딩을 함으로써 압축률을 더 높일 수 있을 것이라 믿는다.

Table 2에서는 압축을 푸는데 걸리는 시간을 보여주고 있다. P열에서 볼 수 있는 것처럼 정밀도가 높아질수록 Deering의 알고리즘이 압축을 푸는데 걸리는 시간이 증가하는 반면, Q열에서 보여지는 것처럼 제안된 알고리즘은 걸리는 시간이 상수이다. 제안된 알고리즘이 압축을 푸는데 소요되는 총 시간은 Deering의 알고리즘의 수행시간을 넘지 않는다. 하지만 Deering의 알고리즘은 74.3 millisecond가 지난 후에야 모델을 볼 수 있는데 비해 제안된 알고리즘은 31.4 millisecond가 지난 후 바로 초기 모델을 볼 수 있다. 전송에 필요한 시간은 이 통계자료에 포함되어 있지 않다.

6. 결 론

본 논문에서 우리는 삼각형 메쉬 모델의 기하정보를 압축하고 점진적으로 전송하는 방법에 대해 소개하였다. 기존 연구의 에러 분석을 통해 우리는 정점들 사이의 델타를 압축하여 전송하는 것보다, 직접 정점의 좌표값을 그대로 압축하여 전송하는 것이 모델의 에러를 제어하기에 더 유리하다는 것을 알 수 있다. 제안된 알고리즘에 엔트로피 코딩 기법을 적용할 경우 압축률이 더 좋아질 것이다.

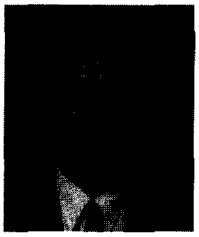
그리고 현재는 하나의 지수값이 사용되지만 지수값의 클러스터링 작업이 이루어진다면 더 많은 비트 수를 줄일 수 있을 것이다.

감사의 글

본 연구는 한양대학교 2001년도 교내연구비의 지원으로 이루어졌습니다.

참고문헌

1. Cheney, W. and Kincaid, D., *Numerical Mathematics and Computing*, Brooks/Cole Publishing Company, 1994.
2. Chow, M. M., "Optimized Geometry Compression for Real-time Rendering," *In Proc. IEEE Visualization '97*, pp. 347-354, October, 1997.
3. Deering, M., "Geometry Compression," *In Proc. SIGGRAPH' 95*, pp. 13-20, August 1995.
4. Forst, G. and Thorup, A., "Minimal Huffman Trees," *Acta Informatica*, Vol. 36, pp. 721-734, 2000.
5. Goodrich, M. T. and Tamassia, R., *Data Structures and Algorithms in JAVA, Second Edition*, John Wiley & Sons, Inc., 1998.
6. Gumhold, S. and Strasser, W., "Real Time Compression of Triangle Mesh Connectivity," *In Proc. SIGGRAPH '98*, pp. 133-140, 1998.
7. Hoppe, H., "Progressive Meshes," *In Proc. SIGGRAPH '96*, pp. 99-108, August 1996.
8. Hoppe, H., "View-Dependent Refinement of Progressive Meshes," *In Proc. ACM SIGGRAPH '97*, August 1997.
9. Huffman, D. A., "A method for the construction of minimum redundancy codes," *In Proc. Symposium on Theory of Computing*, pp. 703-712, 1995.
10. Kim, Y.-S., Park, D.-G., Jung, H.-Y. and Cho, H.-G., "An Improved TIN Compression Using Delaunay Triangulation," *In Proc. Pacific Graphics '99*, pp. 118-125, 1999.
11. Lee, E.-S. and Ko, H.-S., "Vertex Data Compression For Triangle Meshes," *Eurographics 2000*, Vol. 19, No. 3, pp. 1-10, 2000.
12. Rossignac, J., "Edgebreaker: Compressing the incidence graph of triangle meshes," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 1, pp. 47-61, January-March, 1999.
13. Rossignac, J. and Szymczak, A., "Wrap & Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker," *Computational Geometry*, Vol. 14, pp. 119-135, 1999.
14. Sayood, K., *Introduction to data compression, Second Edition*, Morgan Kaufmann Publishers, San Francisco, California, 2000.
15. Taubin, G. and Rossignac, J., "Geometric Compression Through Topological Surgery," *ACM Transactions on Graphics*, Vol. 17, No. 2, pp. 84-115, April 1998.
16. Touma, C. and Gotsman, C., "Triangle Mesh Compression," *In Proc. Graphics Interface '98*, pp. 26-34, 1998.
17. Kim, D.-S., Chung, J., Cho, Y, Jang, T. and Kim, H., "Compression and Progressive Transmission of 3D Shape on Internet," *In Proc. Korea/US Joint Workshop on Information Technology for Product Development*, pp. 39-44, July, 2001.

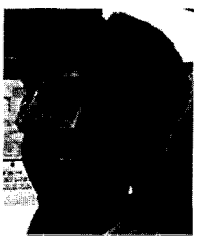


김 덕 수

1982년 한양대학교 산업공학과 학사
1985년 New Jersey Institute of Technology 산업공학과 석사
1990년 The University of Michigan 산업공학과 박사
1989년~1991년 Schlumberger Technology CAD/CAM Co. Senior Software Engineer
1991년~1995년 삼성종합기술원 선임 연구원
1995년~현재 한양대학교 산업공학과 부교수
관심분야: Geometric Modeling, Computational Geometry, STEP and Internet Applications

정 재 열

2000년 한양대학교 산업공학과 학사
2002년 한양대학교 산업공학과 석사
2002~현재 한빛전자통신(주) 중앙연구소 연구원
관심분야: Computational Geometry, 3D Data Compression, Web Application, Computer Graphics



김 현

1984년 한양대학교 기계설계학과 학사
1987년 한양대학교 기계설계학과 석사
1997년 한양대학교 기계설계학과 박사
1998년~1999년 한양대학교 산업대학교 겸임교수
1990년~현재 한국전자통신연구원 동시공학연구팀장, 책임연구원
관심분야: Concurrent Engineering, Virtual Engineering, CAD/CAM/CAE/PDM