메쉬의 계층 및 연산코드 기반 3차원 메쉬 압축

[0]이민정[*,**] 권용무[*] 김창헌[**]
한국과학기술연구원 영상미디어연구센터[*], 고려대학교 컴퓨터학과[**]
[0]mjlee[*,**]@kist.re.kr, ymk[*]@cherry.kist.re.kr, chkim[**]@korea.ac.kr

3D Mesh Compression Based on Layer of Mesh and Operation Code

[0]Min-Jeong Lee[*,**] , Yong-Moo Kwon[*] , Chang-Hun Kim[**]
Imaging Media Research Center, Korea Institute of Science and Technology[*]
Dept. of Computer Science, Korea University[**]

요 약

날로 커져가는 3D 모델을 효율적으로 사용하기 위한 노력으로 압축처리 방법들이 연구되고 있다. 본 논문에서는 3D 모델의 메쉬를 Layer로 분할하여 Vertex Layer와 Triangle Layer를 생성 후, 삼각형들을 몇 가지 연산코드로 분류하여 압축(compression)하는 방법을 제안한다. Triangle Layer는 기본 정점으로부터 연결된 선분의 정점들로 이루어진 Vertex Layer의 쌍을 이용하여 만들어진다. 이 Triangle Layer에 해당되는 삼각형들의 연결 정보를 제안한 연산코드로 분류하고, 이것을 엔트로피 코딩하여 3D 모델을 압축한다. 이 기법은 삼각형의 형태를 기준으로 한 개나 두 개의 삼각형을 하나의 연산코드로 분류하거나 삼각형의 연결 상황에 따라 하나의 연산코드로 분류하여 연결정보를 표현한다. 복원(decompression)시에는 연산 코드를 이용하여 삼각형의 연결정보를 뽑아내면 원 상태의 3D 모델을 획득할 수 있다. 이 방법은 연결정보를 무손실 압축하는 방법으로, 지금까지 제안된 압축기법과 비교할 때, 간단하면서도 월등한 압축 효과를 볼 수 있다.

1. 서 론

오늘날 멀티미디어 산업에서 필수 불가결한 것은 데이터의 저장과 전송의 최적화라 할 수 있다. 인터넷, 게임, CAD, 원격 의료, 3D 애니메이션 등의 산업으로 인해 3D 모델의 거대한 용량이나 실시간 스트리밍 서비스에 관련한 데이터 압축에 관한 연구는 현재 활발히 이루어지고 있다.

3D 모델의 메쉬는 기본적으로 연결(connectivity) 정보와 위치(position) 정보로 나눌 수 있다. 본 논문에서는 3D 모델의 연결정보에 중점을 두어 이를 무손실로 압축하는 기법을 제안하고, 위치 정보는 제안한 연산코드에 따라 적응적으로 예측한다. 이 기법은 기존의 기법들에 비해 간단하면서도 더 우수한 압축률을 가진다.

본 논문의 2장에서는 기존에 나와있는 3D 모델의 압축 기법에 대해 설명하고, 3장에서는 제안하는 연결정보 압축기법과 이에 해당하는 위치정보를 예측하기 위한 방법에 대해 설명한다. 3장의 실험결과를 4장에서 보여주고, 5장에서는 결론 및 향후 계획에 대해 설명한다.

2. 기존의 기법들

3D 모델의 압축을 위하여 우선 메쉬를 삼각형화 하게 된다. 이 방법은 OpenGL graphics library[1]와 같은 3D 그래픽에서 지원하는 triangle strip[2]을 사용할 수 있어서 메쉬를 손쉽게 다룰 수 있게 한다. 이에 따라 본 논문에서도 이 같은 방법으로 메쉬를 전처리 할 것이다.

이렇게 삼각형화 된 메쉬는 몇몇 기법들에 의해 압축하여 사용되어왔다. M. Deering[3]은 스왑 연산자를 사용한 triangle strip을 generalized triangle strip으로 표현하고, 이것을 버퍼를 사용하여 중복되는 정점(vertex)을 처리하여 generalized triangle mesh로 나타내는 기법을 제안하였다.

G. Taubin[4]은 IBM 알고리즘이라고도 불리는 Topological Surgery기법을 제안하였는데, 3D 메쉬를 오렌지의 껍질을 벗기는 형상으로 잘라서 연결된 폴리건(connected polygon)으로 만든 후 이것을 기본으로 vertex spanning tree와 triangle spanning tree를 생성하여 압축한다. 이 기법은 MPEG-4 (ISO/IEC 14496-2 :1999) part2: Visual[5]에서 SNHC (Synthetic Natural Hybrid Coding)에 필요한 3D 모델을 위해 채택되어 사용되고 있다.

Tauma & Gotsman[6]은 3개의 연산코드(add, split, merge)를 사용하여 연결 정보를 표현한다.

J. Rossignac[7]은 edgebreaker라는 이름으로, 연결 정보를 5가지의 연산코드(C, L, E, R, S)로서 표현하여 압축하고, 이론적으로 2bit/T의 압축률을 보장한다.

Bajaj[8]는 triangle layer개념을 이용하여 메쉬를 분할하여 압축하는 기법을 제안하는 등 3D model을 압축하기 위한 활발한 연구가 이루어지고 있다.

3. 제안하는 기법

기존의 기법들은 원 3D 모델의 용량을 줄이긴 하였지만, 복잡한 인코딩 기법들이 사용되었고, 몇 가지 예외적인 처리들이 부족한 것들도 있었다. 본 논문은 간단한 연산코드만으로 삼각형의 연결 정보를 표시하고, 예외적인 사항에 대해서도 처리할 수 있는 연산코드를 할당하여 압축한다.

기존의 기법들과 마찬가지로 메쉬는 삼각형화하여 사용하고, 3D 모델은 심플 메쉬(triangulated connected oriented manifold[9] without boundary having Euler Characteristic 2)[4]를 사용한다. 여기에 제안하는 기법을 적용 후, 허프만 코딩을 통해 다시 한번 압축시킨다.

3.1. Layer생성

3.1.1. Vertex Layer(VL)생성

임의의 정점을 VL[0]으로 두고, 이 정점을 중심으로 연결된 선분들을 선택한다. 선택된 선분들의 끝 정점들을 순서대로 연결한 것이 VL[1]이다. 다시 VL[1]의 정점들과 연결된 선분들을 선택하고, 그 선분들에 연결된 정점들을 순서대로 VL[2]에 정렬한다. 이 순서를 반복하여, VL[i-1](i>1)에 연결된 선분들을 선택하고, 이 선분들에 연결된 정점들을 순서대로 연결하여 VL[i]라 이름 붙인다. 3D 모델의 정점들이 다 선택되면 n개의 VL집합이 생기고, VL의 생성을 마치게 된다.

3.1.2. Triangle Layer(TL)생성

TL은 VL[i-1](윗 레이어)과 VL[i](아랫 레이어)의 쌍으로 둘러 싸인 삼각형들로 구성된다. 이때의 TL을 TL[i]라 하고, TL[i]의 삼각형은 다음과 같이 두 가지로 구분한다.

(1) TB(Triangle made by Both layer)

TB는 VL[i-1]에 정점을 한 개 이상 두고, VL[i]에도 정점을 한 개 이상 두는 삼각형을 일컫는다.

-**LT**(Link Triangle) : 두 정점을 VL[i-1]에 두고, VL[i]에 한 정점을 둔 TB중에서 VL[i-1]의 정점 사이의 간격이 2 이상일 때, LT라 한다.

-**LE**(Link Edge) : LT에서 VL[i-1]의 두 정점으로 연결된 선분

-**PU**(Partial of Upper Layer) : LE로 분리되는 VL[i-1]의 집합

(2) TU(Triangle made by Upper layer)

TU는 VL[i-1]에 세 정점을 둔 삼각형을 일컫는다. TU들로 이어진 삼각형들의 순서가 있는 집합을 **TUL**(TU Layer)라 칭한다.

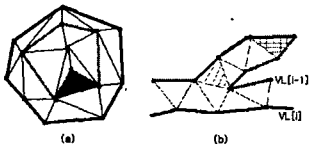

그림 1. Triangle Layer (a)TL에 따른 TB(흰색)와 TU(회색) (b)TUL

3.1.2.1. PU로 이루어지는 삼각형의 처리

TL안에서 PU로 이루어지는 삼각형은 앞서 말한 TU와 TB 둘다 생성이 될 수 있다. TU가 생성될 때는 TUL이 생성됨을 의미하는 것이고, TB가 생성될 때는 예외적인 처리가 필요하다.

(1) PU로 이루어지는 TU 처리

TL에서 TUL이 시작되는 삼각형 TU는 **FT**(First triangle of TUL, 그림 1-(b). 의 빗살 무늬 삼각형)라 정의한다. 삼각형의 세 정점이 VL[i-1]에 있는 순서대로 연결된 세 정점으로 이루어져 있을 때 이를 **LT**(Last Triangle of TUL, 그림 1.의 체크 무늬 삼각형)이라 하고, TUL생성도중 TB가 나타나 LT를 정할 수 없을 때는 TB가 출현하기 직전의 삼각형을 LT로 간주한다.

TUL은 LT의 마지막 정점을 제외하고, VL[i-1]에서의 정점으로부터 순방향으로 만나는 정점까지를 윗 레이어, 역방향으로 만나는 정점까지를 아랫 레이어로 재정의 하여 처리한다.

TL[i]의 삼각형 구분에 따라 TUL은 또 다른 TUL을 가질 수 있게 된다. (그림 2-(a). 참조) 이때는 TL상에서 TUL을 처리하듯이 한다.

(2) PU로 이루어지는 TB 처리

PU를 윗 레이어(VL[i-1])처럼 취급하는 메쉬의 일부분이 나타나는 경우가 있다. 이때는 TB의 예외로 보고 따로 처리한다. 우선 TB가 출현하는 PU부분을 새로운 윗 레이어로 정의하고, 초기의 메쉬를 처리할 때와 같이 한다. (그림 2-b. 회색 부분) 이 같은 삼각형은 실린더나 절구통모양의 모델(queen, bunny등)에서 초기 정점이 곡면의 중간부분에서 선택될 때 나타난다.

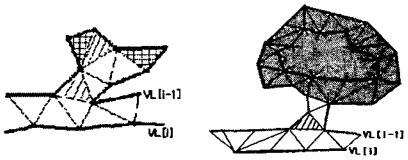

그림 2. (a) TUL안의 TUL (b) PU로 이루어지는 TB

3.2. 제안하는 연산코드들

제안하는 연산코드는 TL[i]의 삼각형을 표현하기 위한 것과 예외적인 연결성 정보를 표현하기 위한 것으로 나눌 수 있다.

3.2.1. 연산코드 I

TL로 만들어진 삼각형들은 몇 가지 경우로 나눌 수 있다. 본 논문에서는 아래 그림 3.과 같은 6가지의 연산코드로 삼각형의 연결 형태를 표현하기 위한 방법을 제안한다.

그림 3. 제안하는 연산코드 I

그림 3.에서 윗 꼭지점들은 VL[i-1](윗 레이어)에 속하는 것이고, 아래 꼭지점들은 VL[i](아랫 레이어)에 속하는 것이다. 그림과 같이 e 나 f의 삼각형을 조합하여 a~d와 같은 연산 코드를 사용한다. 이 방법은 두 개의 삼각형의 연결 형태를 하나의 연산 코드로 표현하여 더 좋은 압축률이 나오게 한다. 연산 코드들은 그림에 따라 a, b, c, d, e, f로 정의한다.

3.2.2. 연산 코드 II

3.2.1.에서 표현된 삼각형 외의 예외적인 정보의 표현을 위해 또 다른 몇 가지 연산 코드를 사용한다.

그림 4. 제안하는 연산 코드 II
(a)Change Layer (b)TU Layer (c)Null (d)JV

그림 4-(a).의 CL(Change Layer)는 TL[i-1]의 삼각형의 코드화를 끝낸 후 TL[i]의 삼각형을 코드화하는 것을 알려주기 위하여 사용한다. 복원시에 CL을 취하면, VL[i-1]의 첫 번째와 두 번째 정점과 VL[i]가 되는 정점으로 이루어진 삼각형 하나가 자연적으로 생성이 되도록 하고, CL다음의 코드가 TUL, Null이 오게 되면 삼각형을 자연 생성하지 않도록 한다. 또, TUL 처리중 CL이 오게 되면 PU로 이루어지는 TB가 시작됨을 알리는 역할을 하게 함으

로써, 후에 엔트로피 코딩시에 bit수를 줄일 수 있게 한다.

(b)는 TUL이 시작됨을 알리기 위함이고, <TUL, length>로 표현한다. 여기서의 length는 VL[i-1]에서 TUL이 시작하여 끝나는 정점간의 길이를 나타낸 것이다. 원 메쉬의 TL의 연산코드를 추출하고 난 후, 연산코드안의 TUL을 처리하여 준다. 이는 3.1.2.1.(1)에서 설명한 윗 레이어와 아랫 레이어를 이용하여 FT부터 LT까지의 TUL의 삼각형의 연결정보를 연산 코드로 표현한다.

(c) Null 는 그림 5.에서 보이는 VL[i-1]의 정점에서 VL[i]와 연결되는 정점이 없을 때를 표현하기 위하여 사용된다.

(d) JV(Joint Vertex)는 TUL에서 발생하는 것을 처리하기 위한 것으로, TUL의 윗레이어와 아랫 레이어가 만나서 하나의 정점을 이룬다. 이 정점으로부터 중간에 남아 있는 정점들은 다른 메쉬를 생성하거나, TUL을 생성하게 된다.

여기까지 설명된 연산 코드에 대한 예는 3.4.에서 간단한 메쉬로 선보인다.

3.4. 3D model의 연산 코드로의 변환의 예

연산 코드의 모든 예를 들 수 있는 간단한 메쉬(심플 메쉬는 아니지만, 이해를 돕기 위한 메쉬)로써 제안한 연산 코드로의 변환을 예를 들어 설명하면 다음과 같다.

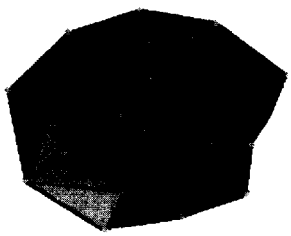

그림 5. 간단한 메쉬에 제안한 연산 코드를 표현한 형태

그림 5.의 굵은 선은 VL을 나타낸 것이고, 삼각형은 연산 코드에 따라 표현하였다. 빗살무늬의 삼각형은 CL연산 후 생긴 삼각형이다. 이와 같은 메쉬를 변환한 코드는 <b, b, b, CL ,a, a, a, c, a, b, c, c, CL ,a ,a ,c ,d ,e, Null, TUL, 2, f, b, a, a, a > 이 된다.

3.4. 위치(position) 정보의 예측

위치 정보를 그대로 인코딩 하는 것보다는 그 정보를 줄여서 표현하면 압축률이 향상됨은 이미 널리 알려져 있다. 이를 위해 본 논문에서의 위치 정보는 제안한 연산 코드에 따라 예측한다.

먼저 위치정보는 정점 당 10bit 의 양자화의 전 단계를 거친다. 그 후 그림 3-(a), (b), (e)는 VL[i-1]의 정점 중에서 구하려는 정점과 제일 가까이 연결되어 있는 정점 두 개를 택하여 두 정점 벡터의 차를 구한다. 그리고 나서 구해진 VL[i]의 마지막 정점에서 구해놓은 값을 더하여 구하려는 값을 예측하고, 예측한 값과 원본의 값의 차이를 인코딩 한다.

두 개의 위치 정보를 예측해야하는 그림 3-(c)는 첫 번째 정점은 전과 같이 하고, 두 번째 정점은 앞에서 구한 첫 번째 정점과 VL[i]에서의 그 전의 정점간의 차이벡터를 구한 후 첫 번째 정점에 더한 값을 예측 값으로 사용한다.

(d)와 (f)는 이미 알고 있는 VL[i-1]의 정보를 사용하므로 따로 예측할 필요가 없다.

4. 실험 및 평가

본 논문에서 제안한 방법을 사용하여 3D 모델을 실험하였다. IBM에서 제안한 알고리즘과 함께 비교 평가한 표는 다음과 같다. 두 기법은 위치 정보에 대해 10bit/vertex로 양자화의 전처리 과정을 거쳤다.

표 1. 제안한 방법과 IBM 알고리즘의 압축률 비교

	VRML	IBM 알고리즘	VRML/IBM 압축률	제안법	VRML/제안법 압축률
sphere	3668B	360B	10.2	278B	13.2
bunny	48619B	4837B	10.1	3084B	15.8

표 1.에서는 몇 가지 모델에 대한 실험 결과를 보인다. 보는 바와 같이 원본의 데이터 용량보다 평균 14.5배 정도의 압축률을 보였다. 기존의 IBM 알고리즘과 비교해 보아도 약 1.4배 정도의 더 좋은 압축률을 보이는 것을 알 수 있다.

5. 결론 및 향후 계획

본 논문에서는 3D 모델을 이용하여 Layer를 구하고, Triangle Layer와 제안한 10가지의 연산 코드를 통해 무손실 방법으로 연결 정보를 압축하는 연구를 수행하였다. 제안한 연산 코드는 두 가지의 삼각형 형태를 하나의 연산 코드로 해결함으로써 결과에 보이듯이 기존의 방법에 비해 간단하면서도 더 좋은 압축률이 나타난다.

향후 기존의 기법들과의 정확한 비교 분석이 필요하겠고, 3D 모델의 압축을 위한 속성 정보 등과 관계된 지속적인 연구와 더 나아가서는 심플 메쉬외의 적용이나 제안한 기법을 점진적인 (progressive) 전송에 적용하는 등의 연구 또한 필요하겠다.

참고 문헌

[1] M. Woo, J. Neider, T. Davis, D. Shreiner. OpenGL Programming Guide 3rd Edition. The Official Guide to Learning OpenGL, Version 1.2. OpenGL Architecture Review Board.

[2] F. Evans, S. Skiena, and A. Varshney. Optimizing triangle strips for fast rendering. In IEEE Visualization '96 Proceedings, pages 319-326, Oct. 1996.

[3] M. Deering. Geometry compression. In SIG- GRAPH'95 Conference Proceedings, pages 13-20, 1995.

[4] G. Taubin and J. Rossignac. Geometric compression through topological surgery. In ACM Transactions on Graphics, pages 84-115, April 1998.

[5] ISO/IEC 14496-2. Information technology - Coding of audio-visual objects - Part 2: Visual - Amendment 1: Visual extensions.

[6] C. Touma and C. Gotsman. Triangle mesh compression. In Proceedings of the 24th Conference on Graphics Interface (GI-98), pages 26-34, 1998.

[7] J. Rossignac. Edgebreaker: Connectivity Compression for Triangle Meshes. In IEEE Transactions on Visualization and Computer Graphics, 5(1):47-61, January-March 1998.

[8] C. Bajaj, V. Pascucci, and G. Zhuang. Single resolution compression of arbitrary triangular meshes with properties. In Data Compression Conference'99 Conference Proceedings, pages 247-256, 1999.

[9] N. Aspert, T. Ebarahimi. Photo-Realistic 3D Model Coding in MPEG-4. In IEEE International Conference on Multimedia and Expo (ICME), pages 1111-1114, 2000.