# 모바일 그래픽스를 위한 메쉬 위치정보 압축

## Mesh Geometry Compression for Mobile Graphics

이종석, Jongseok Lee*, 최성열, Sungyul Choe**, 이승용, Seungyong Lee***

**요약** 본 논문은 모바일 그래픽스 응용에 적합한 메쉬 위치정보의 압축 기법을 제시한다. 제시한 기법은 복원 에러를 최소화하기 위한 메쉬 분할 기법과 기존의 방법에서 발생하는 시각적 손상문제를 해결한 지역적 정량화 기법으로 구성된다. 기존 방법에서는 분할된 조각 메쉬들 간의 경계가 벌어지는 시각적 손상문제가 발생하는데, 모든 조각 메쉬의 지역적 양자화 셀이 같은 크기와 정렬된 지역 좌표축을 갖게 하여 이 문제를 해결했다. 제시한 기법은 메쉬를 렌더링할 때 압축된 위치정보를 메모리에서 그래픽스 하드웨어로 전송하여 실시간으로 복원함으로써 모바일 기기의 자원을 절약하는 특징을 갖는다. 압축된 위치정보의 복원을 표준화된 렌더링 파이프라인에 결합이 가능하도록 설계함으로써 조각 메쉬당 한번의 행렬 곱셈으로 복원이 가능하다. 실험에서는 32 비트 부동소수점 수로 표현되는 위치정보를 8 비트 정수로 지역적 정량화하여 70%의 압축률에서 11 비트 전역적 정량화와 대등한 수준의 시각적 품질을 달성했다.

***핵심어:*** *Computer Graphics, Mobile Graphics, Mesh Compression, Object Representation*

## 1. 서론

최근 들어 전 세계의 수많은 사람의 일상생활 속에서 휴대전화, PDA, 휴대용 게임기 등의 모바일 기기들이 사용되고 있다. 최신형 모바일 기기들은 계산력, 저장 공간, 네트워크, 그래픽스 처리 능력 등의 자원이 종전의 기기들에 비해 월등히 향상됐다. 특히, OpenGL ES 와 JSR-184 와 같은 모바일 그래픽스를 위한 API 표준들이 제정되고 모바일 기기용 GPU 의 성능이 향상되면서 모바일 기기의 그래픽스 처리능력이 급격히 향상됐다. 이러한 진보들과 함께 다양한 모바일 응용에 삼차원 그래픽스의 활용이 점점 늘고 있다. 이런 경향으로 말미암아 모바일 기기만의 독특한 특성을 반영한 다양한 그래픽스 기법에 대한 필요성이 증가하고 있다.

최근 모바일 기기의 하드웨어 성능 향상에도 모바일 그래픽스는 데스크톱 PC 에서의 그래픽스에 비해 낮은 계산력, 적은 메모리, 저장공간, 네트워크 대역폭 등으로 말미암아 자원의 제약이 심하다. 특히, 모바일 기기들은 대부분 축전지에 의해 구동되기 때문에 가용 전력이 한정적이고, 배터리 기술의 발전 속도가 느려서 삼차원 그래픽 처리에 필요한 에너지 요구사항을 만족하지 못하고 있다[1].

데이터 압축은 모바일 기기의 자원 제약을 해결하기 위한 효율적인 방법 중의 하나이다. 일반적으로 데이터 압축은 저장공간을 절약하고 네트워크를 통한 데이터 전송 시에 네트워크 대역폭을 절약하는 효과가 있다. 또한, Ström 과 Möller 의 방법[12]에서와 같이 압축된 데이터를 그래픽스 하드웨어에 전송하고 렌더링 직전에 복원하면 응용 프로그램이 사용하는 메모리와 시스템 버스의 대역폭을 절약하여 데이터 전송에 소모되는 전력을 절약할 수 있다. 그러나, 모바일 그래픽스에서 이러한 실시간 복원을 수행하기 위해서는 복원 기술이 그래픽스 처리 성능에 영향을 주지 않을 정도로 간결해야 한다.

본 논문에서는 모바일 그래픽스에서 실질적으로 활용 가능한 삼차원 메쉬의 위치정보 압축 기술을 고안하는 것이 목표이다. 압축 알고리즘의 주요 설계 목표는 그래픽스 하드웨어의 병렬처리 능력을 활용한 위치정보의 실시간 복원이다. 본 논문에서 제시하는 메쉬 위치정보 압축기법은 다음과 같은 특징을 갖기 때문에 모바일 그래픽스에 활용하기 적합하다.

- *간결성* 정량화된 정점 위치는 정점 위치 벡터와 행렬의 곱셈에 의해 복원이 가능하기 때문에, 모델 위상 변환과의 결합이 가능하다. 따라서, 복원에 소모되는

*주저자 : 포항공과대학교 컴퓨터공학과 석사과정 e-mail: thirdeye@postech.ac.kr

**공동저자 : 포항공과대학교 컴퓨터공학과 박사과정 e-mail: ggalssam@postech.ac.kr

***교신저자 : 포항공과대학교 컴퓨터공학과 교수 e-mail: leesy@postech.ac.kr

비용은 조각 메쉬당 한번의 행렬 곱셈에 불과하다.

- *실시간 복원* 정량화된 정점 위치는 렌더링 시에 모델 위상 변환에 의해 복원된다. 따라서, 압축된 형태의 정점 데이터가 메모리에서 그래픽스 하드웨어로 전송되기 때문에, 메모리와 시스템 버스 대역폭의 절약이 가능하다.
- *압축 성능 향상* 시각적 손상을 방지하는 지역적 정량화를 사용함으로써 전역적 정량화를 사용하는 기존 방법[1] 보다 더욱 뛰어난 압축률과 품질을 달성했다.
- *고정된 데이터 크기* 정량화된 정점 위치는 좌표축당 8 비트의 고정된 크기를 갖기 때문에, 임의 접근이 용이하고 복원에 추가적인 연산이 필요치 않다.
- *호환성* 복원 연산이 표준화된 그래픽스 파이프라인에 호환되기 때문에, 모바일 기기는 물론이고 PC 나 게임기 등에도 활용 가능하다.
- *융통성* 조각 메쉬의 수를 조절함으로써, 압축률과 복원 에러간의 조작이 가능하다.

실험에서는 각 축당 32 비트 부동소수점 수로 표현되는 정점의 위치정보를 모바일 그래픽스 라이브러리에서 지원하는 최소 데이터 크기인 8 비트 정수로 정량화한다. 지역적 정량화는 경계 입방체 정보와 중복 저장되는 경계 정점들로 인해 실제 저장하는 데이터는 8 비트보다 조금 많다. 그러나 실험 결과에서는 본 연구에서 제시한 압축 기법이 11 비트 전역적 정량화와 비교할만한 영상 품질과 데이터 감소 비율 70%을 달성했음을 보여준다. 이러한 결과는 8 비트 사용으로 11 비트 전역적 정량화의 정확도를 달성한 것을 의미한다. 압축된 메쉬의 복원을 위한 부가 정보를 고려하면 전체 데이터 크기가 각 축당 9 비트로 전역적 정량화를 수행했을 때와 비슷하지만 메모리에서 그래픽스 하드웨어로 전송되는 데이터는 여전히 각 축당 8 비트이기 때문에 더욱 큰 의미가 있다.

## 2. 관련 연구

메쉬 압축은 그래픽스에서 활발한 연구가 수행되어온 분야이다. 본 논문에서는 메쉬 압축 기술들의 개괄적인 내용은 다루지 않으며 관련 내용은 개관 논문들[1, 7]에서 살펴볼 수 있다.

이전의 메쉬 압축 기술들은 압축률에서 훌륭한 성능을 제공하지만 일반적으로 부호화/복호화 알고리즘이 복잡하고 속도가 느리며, 많은 컴퓨팅 파워와 메모리가 필요하다. 따라서 기존의 방법들을 제한된 자원을 갖는 모바일 기기에 적용하면 오히려 자원의 낭비를 초래할 수 있다. 특히 모든 방법들이 압축된 메쉬 데이터를 복원하기 위해서 데이터가 순차적으로 처리 되야 하기 때문에, GPU 를 이용한 병렬 처리가 근본적으로 불가능하다.

Calver 는 위치정보, 법선 벡터, 색상 등의 정점 속성 데이터들을 정량화하여 프로그램 가능한 정점 세이더(programable vertex shader)를 이용해 메쉬의 렌더링 시에 복원하는 구조를 갖는 다양한 메쉬 압축/복원 기술들[2]의 기본 개념을 제시했다.

Purnomo 등은 Calver 의 기본 아이디어를 구체화하여 메쉬 단순화(mesh simplification)와 정점 속성의 정량화를 사용한 위치정보 압축 기술[10]을 고안했다. 이 방법은 정점의 위치정보, 법선 벡터, 텍스처 좌표에 렌더링 되는 이미지의 에러가 최소화 되도록 비트를 할당하여 각각을 정량화하여 정점 속성들을 압축한다. 이렇게 정량화된 속성 데이터들을 정점 당 96 비트로 압축하여 정점 셰이더에 전송한 후 정점 셰이더에서 비트 분해를 수행한 후 각 속성 별로 복원을 수행한다.

본 논문에서 제시하는 삼차원 메쉬의 위치정보 압축 기술은 기존의 방법들 보다 더욱 뛰어난 영상 품질와 압축 성능을 달성했다. 위치정보 데이터에 24 비트(좌표축당 8 비트)의 고정된 비트를 할당함으로써 비트 분해가 필요 없기 때문에 실행시간 비용이 매우 적으며, 정점 셰이더를 활용하지 않아도 복원이 가능하기 때문에 보다 높은 호환성을 갖는다. 또한, 이전 연구들보다 더욱 뛰어난 영상 품질을 달성했다. 하나의 메쉬를 여러 개의 영역으로 분할하여 각각의 영역을 지역적으로 정량화하는 다중 압축 변환의 기본 아이디어는 Calver 에 의해 제시되었다. 그러나, 기본적인 지역적 정량화는 인접한 메쉬 조각들 사이에 균열이 생기는 문제가 있다. 본 논문에서는 이러한 문제점을 해결하는 방법을 해결함으로써 지역적 정량화를 활용하여 전역적 정량화를 사용하던 기존 방법들보다 더욱 뛰어난 영상 품질을 달성했다.

## 3. 기본 아이디어

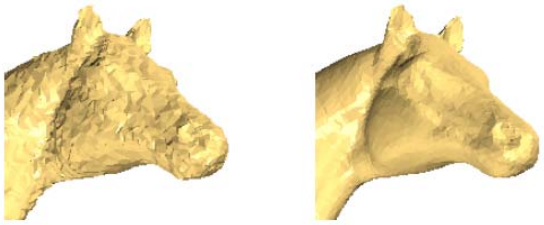
본 연구에서는 메쉬의 위치정보와 연결관계 중에서 위치정보의 압축만 고려한다. 연결관계는 삼각형 스트립과 같은 적절한 형태로 표현됨을 가정한다. 삼각형 스트립은 대부분의 그래픽스 하드웨어서 기본적으로 지원되기 때문에 JSR-184[9]와 같은 모바일 그래픽스 표준에서도 연결관계 표현의 기본 형태로 사용될 정도로 효율적인 구조이다.

본 연구에서 목표로 하는 삼차원 메쉬 위치정보의 실시간 복원을 위해서는 압축 알고리즘이 다음과 같은 조건들을 만족해야 한다.

- 복호화 과정이 실시간 수행에 충분할 정도로 간결해야 한다.
- 그래픽스 하드웨어의 병렬처리 능력을 활용하려면 각 정점 간의 위치정보 복호화 연산에 종속성이 없어야 한다.
- 부호화된 데이터가 중앙처리장치와 그래픽스 하드웨어 간의 데이터 채널에 맞는 고정 크기 형식으로 표현돼야 한다.

정점 위치정보의 정량화는 위와 같은 요구사항들을 만족하는 가장 간단한 형태의 해결책이다. 32 비트 부동소수점 수로 표현되는 정점 위치정보는 수치의 표현

범위를 축소하고 소수점 이하를 버림으로써 더 작은 범위를 갖는 정수로 변환할 수 있다. 이렇게 변환된 정수는



(a) 전역적 정량화 (b) 지역적 정량화

**그림 1. 8 비트 전역적 정량화와 지역적 정량화의 품질 비교.**

간단한 덧셈과 곱셈으로 복원할 수 있다. 또한, 압축과 복원과정에서 정점 간의 종속성이 없어서 병렬처리가 가능한 장점이 있다.

많은 메쉬 압축관련 연구[1,7]들에 의하면 비정형 메쉬의 압축 시에 전역적 정량화를 사용하면 좌표축 당 12 비트 이상일 때 시각적으로 원본과 거의 유사한 품질을 얻을 수 있다. 그러나 12 비트 전역적 정량화는 대부분의 그래픽스 API 가 고정된 크기의 데이터 채널(e.g., 8, 16, 32 비트)을 제공하기 때문에 비트 별 연산이나 비트의 낭비를 초래할 수 있다. 반면에, 고정된 8 비트로 전역적 정량화를 수행할 경우 수치 표현의 정확도가 부족하여 그림 1(a)와 같이 심각한 시각적 손상이 발생한다.

전역적 정량화의 품질 문제를 해결하는 방법으로 Calver[2]는 그래픽스 하드웨어의 장점을 최대한 활용하는 다양한 정량화 기법들을 소개했다. 그 중 다중 압축 변환 기술은 메쉬를 여러 개의 부분으로 분할한 후 각각의 분할된 메쉬를 독립적으로 정량화하는 지역적 정량화 기술이다. 메쉬를 여러 부분으로 분할하면 경계 입방체의 크기가 작아지므로 왜곡이 줄게 된다. 따라서, 지역적 정량화를 사용하면 좌표축 당 8 비트만 사용해도 그림 1(b)와 같이 충분한 영상 품질을 달성할 수 있다. 그러나 이러한 지역적 정량화는 근본적으로 분할된 메쉬들 사이에 그림 2 와 같은 균열이 생기는 심각한 문제가 발생한다. Calver 는 지역적 정량화의 개념과 균열 문제가 발생할 수 있음을 언급했고 구체적인 방법이나 해결책은 제시하지 않았다.

균열 문제는 메쉬 파티션(mesh partition)들 사이에 공유되는 경계 정점이 두 번 이상 부호화되기 때문에 발생한다. 메쉬를 여러 개의 부분으로 분할하면 분할 경계의 정점들은 인접한 메쉬 파티션 각각에 중복하여 포함된다. 따라서 각각의 메쉬 파티션을 지역적 정량화를

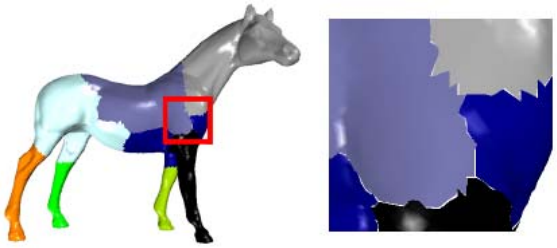
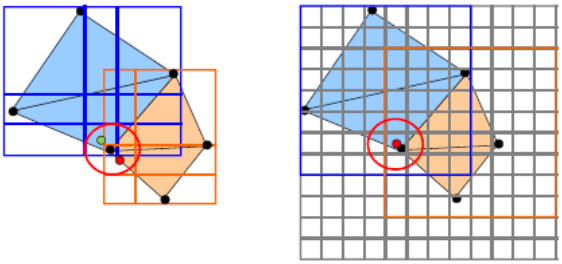


**그림 2. 기본적인 지역적 정량화의 렌더링 결과.**

사용하여 독립적으로 부호화하면 원래 같은 위치에 있던 경계 정점이 각 메쉬 파티션에서 서로 다른 값으로



(a) 분석 (b) 해결책

**그림 3. 지역적 정량화의 문제점 분석과 해결책.**

부호화된다. 서로 다른 값으로 부호화된 경계 정점들이 복원 시에 같은 위치로 복원되어야 하지만 각 메쉬 파티션들의 지역 좌표계의 원점과 경계 입방체의 크기가 다르므로 복원 에러가 달라지고 결과적으로 서로 다른 위치로 복원되게 된다. 그림 3(a)는 이러한 상황을 나타내고 있다.

지역적 정량화의 균열 문제를 해결하기 위한 해결책은 메쉬 파티션들의 경계 입방체의 크기와 지역적 정량화 셀을 하나의 좌표계에 정렬하는 것이다. 그림. 3(b)와 같이 지역적 정량화 셀이 정렬되어 있으면 공유 경계 정점의 복원 위치가 같아지게 된다. 따라서, 모든 지역적 정량화 셀의 크기를 같게 하고 지역 좌표계를 공통 그리드(grid)에 정렬하면 공유 정점이 서로 다른 위치로 복원되는 문제를 해결할 수 있다. 이러한 제약 조건을 만족하고자, 본 논문에서는 모든 메쉬 파티션의 경계 입방체의 x, y, z 각 축에 대해서 가장 긴 길이를 찾아서 하나의 경계 입방체의 크기를 구한다. 이 경계 입방체의 크기를 모든 분할 메쉬에 적용하여 지역적 정량화를 수행함으로써 메쉬 파티션 경계의 균열 문제를 해결했다.

## 4. 부호화 과정

정점 위치정보를 정량화할 때, 원본 메쉬와 압축 후 복원된 메쉬의 왜곡은 정량화 셀들의 크기에 의해 결정된다. 3 절에서 설명한 바와 같이, 본 연구에서 사용한 방법에서는 지역적 정량화 셀들의 크기가 메쉬 파티션들 중 가장 큰 경계 입방체의 크기에 의해 결정된다. 따라서 왜곡을 최소화하기 위해서는 가장 큰 경계 입방체를 가능한 한 작게 만들어야 한다.

### 4.1 메쉬 분할

첫 단계인 메쉬 분할에는 여러 메쉬 분할[3,11]에서 성공적으로 활용된 *로이드의 알고리즘*[6]에 기초한 메쉬 분할 프레임웍을 적용했다. 메쉬 분할은 먼저 입력된 개수만큼 초기 분할을 수행한 후, 초기 분할의 결과에 *시드 재계산*과 *영역 확장*의 두 단계를 반복하여 최종 결과를 얻는다. 시드 재계산 단계에서는 각 메쉬 파티션의 경계 입방체의 중심에서 가장 가까운 삼각면을 새로운 업데이트 단계를 위한 시드 페이스로 설정한다. 영역 확장 단계에서는 각 삼각면들이 가장 가까운 시드 페이스를 갖는 파티션에 속하도록 조정한다.

초기 분할을 수행하기 위해서 *계층적 페이스 군집 병합(hierarchical face cluster merging)*[5]방법을 사용했다. 알고리즘은 최초에 하나의 삼각면이 하나의 군집을 형성하는 상태에서 시작하여 입력 개수 만큼의 군집이 남을 때 까지 인접한 군집들을 병합하는 과정을 반복적으로 수행한다. 각 반복 단계에서 병합할 메쉬 파티션의 쌍을 선택하기 위해서 다음과 같은 비용 함수를 사용했다.

$$F(C_i, C_j) = \max\{x_{ij}, y_{ij}, z_{ij}\},$$

$B_{ij} = x_{ij}, y_{ij}, z_{ij}$ 는 두 개의 메쉬 조각 $C_i$ 와 $C_j$ 를 포함하는 좌표 축에 정렬된 경계 입방체의 크기이다. $F(C_i, C_j)$의 크기를 최소화 함으로써 결과 메쉬 파티션들의 경계 입방체 크기를 줄일 수 있다.

시드 재계산과 영역 확장에서는 거리 함수를 정의하기 위해서 $L_\infty$ 메트릭을 사용했다. 시드 재계산에서는 각 파티션에서 새로운 시드 페이스를 정의하기 위해서 파티션의 $L_\infty$ 중심으로 간주할 수 있는 파티션의 경계 입방체의 중심을 계산한다. 여기서 파티션의 중심을 새로운 시드 페이스로 선택할 수도 있지만, 영역 확장의 효율성을 위해 페이스들의 인접성을 활용하기 때문에 파티션의 중심에서 가장 가까운 페이스를 새로운 시드 페이스로 선택한다. 영역 확장에서는 파티션 $C$의 시드 페이스로부터 하나의 페이스 $f$ 까지 거리는 다음과 같은 수식에 의해 정의된다.

$$F(f, C) = \max\{\delta_x, \delta_y, \delta_z\},$$

$\delta_x$ 는 $C$ 의 시드 페이스의 중심으로부터 $f$ 의 세 개의 정점 중에서 가장 먼 x 축의 거리이다. $\delta_y$ 와 $\delta_z$ 도 이와 유사하게 정의된다.

메쉬 분할 기법들[3,11]에서 로이드의 알고리즘이 주어진 메쉬를 거의 같은 크기의 메쉬 조각들로 나눠준다는 사실이 입증된 바 있다. 이와 유사하게, 삼차원에서 $L_\infty$ 구체는 정육면체와 같기 때문에 $L_\infty$ 메트릭에 의한 메쉬 분할은 로이드의 알고리즘에 의해 만들어진 메쉬 조각들이 거의 같은 크기의 경계 입방체를 갖는 결과를 기대할 수 있다. 따라서 본 연구에서 사용한 메쉬 분할 기법은 주어진 개수의 메쉬 파티션들을 만들면서 메쉬 파티션들의 가장 큰 경계 입방체가 가능한 한 작아지는 결과를 기대할 수 있다.

### 4.2 지역적 정량화

본 논문에서는 좌표축당 32 비트 부동소수점 수로 표현되는 위치정보를 8 비트 정수로 정량화한다. 지역적 정량화를 사용하여 위치정보를 부호화하는 과정은 다음과 같다.

- 각 메쉬 파티션의 경계 입방체 크기를 계산한 후 경계 입방체들 중에서 *x, y, z* 각 축 별로 가장 긴 길이를 찾아서 하나의 공통 경계 입방체 크기를 계산한다.
- 위에서 계산한 경계 입방체의 크기를 *x, y, z* 각 축 별로 $(2^8 - 1)$로 나눠서 지역적 정량화 셀의 크기를 계산한다.
- 원래의 위치정보를 지역적 정량화 셀 크기 $(C_x, C_y, C_z)$로 나누고 소수점 이하를 버림으로써 모든 위치정보를 정량화한다.
- 각 파티션에서 *x, y, z* 각 축 별로 가장 작은 정량화된 값을 찾아 오프셋 $(O_x, O_y, O_z)$을 저장한다. 그 다음 정량화된 값들을 $(O_x, O_y, O_z)$로 빼서 최종적으로 정량화된 [0, 255] 사이의 값인 $(P_x, P_y, P_z)$를 구한다.
- 지역적 정량화 셀의 크기와 파티션들의 오프셋 값, 정량화된 위치정보들을 저장한다.

여기서 주의할 점은 정량화 셀의 크기를 계산할 때 $2^8$ 이 아닌 $2^8 - 1$ 로 경계 입방체를 분할하는 점이다. 이는 오프셋팅 이후에 정량화된 위치정보 값이 [0, 255]의 범위에 항상 위치하도록 보장하기 위한 것이다.

### 4.3 파일 구조

정점의 위치정보를 지역적 정량화로 부호화한 후 위치정보를 복원하기 위해서는 정량화 셀 크기, 각 파티션의 오프셋, 정량화된 위치정보가 필요하다. 정량화 셀 크기는 x, y, z 각 32 비트 부동소수점 수로 하나의 메쉬에서 하나의 값만 저장하면 된다. 하나의 메쉬 파티션의 오프셋 값은 해당 파티션의 경계 입방체의 시작점이 전역 좌표계의 원점으로부터 얼마나 떨어져 있는지를 나타낸다. 각 파티션의 오프셋 값들은 메쉬의 크기에 따라 x, y, z 각 16 또는 32 비트 정수로 저장한다. 위치정보는 8 비트 정수로 정량화 하기 때문에 정점 하나당 24 비트가 필요하다. 이외에도 메쉬를 복원하기 위해서는 메쉬의 총 파티션의 개수와 각 파티션당 정점의 수등의 정보를 추가적으로 저장해야 한다. 이러한 개수 정보는 메쉬 크기에 따라 16 또는 32 비트 정수로 저장할 수 있다.

지역적 정량화는 두 가지 추가적인 저장공간 부담을 초래한다. 하나는 그림 5 에서 볼 수 있듯이 메쉬와 파티션의 헤더 정보를 저장하는 공간이 필요하다. 다른 하나는 정점 중복이다. 3 절에서 볼 수 있듯이 파티션들 사이에 공유되는 경계 정점들은 중복하여 저장해야 한다. 따라서 부호화된 메쉬 위치정보에 포함되는 정점의 수는 원래 정점의 수보다 증가하게 된다.

## 5. 복호화 과정

복호화는 메쉬를 렌더링할 때 매우 적은 연산을 통해 이뤄진다. 복호화는 그래픽스 하드웨어상에서 기하 변환과 동시에 이뤄지기 때문에 실질적인 복호화 비용은 렌더링 이전에 각 파티션의 기하 변환 행렬을 구하는 연산에 불과하다.

### 5.1 위치정보 복호화

부호화된 위치정보를 갖는 메쉬를 렌더링할 때, 먼저 메쉬 헤더에서 정량화 셀 사이즈 $(C_x, C_y, C_z)$를 읽어 들인다. 그 다음, 메쉬의 각 파티션의 위치정보를 다음과 같은 절차로 복호화한다.

그림 4 실험에 사용한 모델들. 좌 상단부터 시계방향으로 아이들, 무용수, 필라인, 장식물, 말, 다람쥐 모델.

- 파티션 헤더에서 오프셋 값 ($O_x$, $O_y$, $O_z$)를 읽어 들인다.
- 다음과 같은 4x4 복원 행렬을 계산한다.

$$\mathbf{M_d} = \begin{pmatrix} C_x & 0 & 0 & C_x \cdot O_x \\ 0 & C_y & 0 & C_y \cdot O_y \\ 0 & 0 & C_z & C_z \cdot O_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

- 복원 행렬 $M_d$ 에 기하 변환 행렬을 곱한다. (e.g., OpenGL 의 모델-뷰 행렬)
- 파티션의 다각형을 그린다.

복호화 과정에서 정점의 위치정보 데이터는 좌표축당 8 비트 정수의 압축된 형태 그대로 주 기억장치에서 그래픽스 하드웨어로 전송된다. 다각형을 렌더링할 때 위치정보들은 복원 행렬 $M_d$ 에 의해 수정된 기하 변환 행렬에 의해 렌더링 파이프라인의 기하 변환 단계에서 자동적으로 복원된다. 여기서 $M_d$ 는 [0, 255]의 범위로 오프셋팅 된 파티션의 위치정보 들을 모델 좌표계 상의 원래 위치로 복원하는 이동 변환 행렬과 제 위치를 찾은 정량화된 값들을 정량화 셀 크기를 곱하여 원래의 32 비트 부동소수점 수로 복원하는 크기 변환 행렬의 결합 행렬이다.

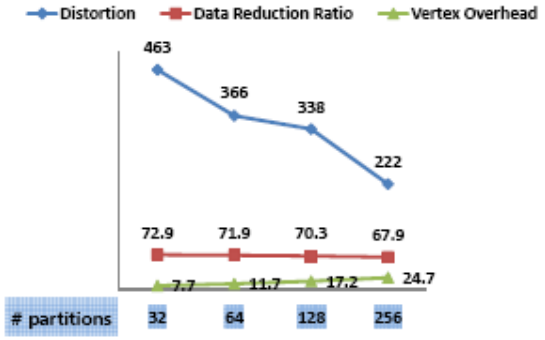



그림 5. 무용수 모델의 파티션 수에 따른 결과 비교. 왜곡은 $10^{-6}$으로 표시.

### 5.2 복호화 비용

지금까지 살펴봤듯이 복호화 과정은 표준 렌더링 파이프라인에 훌륭하게 결합된다. 따라서 메쉬를 렌더링할 때 부호화된 위치정보를 복호화하기 위해 필요한 유일한 연산 부담은 파티션 당 한번의 행렬 곱셈에 불과하다. 이것은 전체 렌더링 파이프라인에서 처리하는 연산량에 비교하면 거의 무시할만한 수준이라고 할 수 있다. 비록 경계 정점들이 파티션들 간에 중복 저장되지만, 렌더링하는 다각형의 수는 변함이 없기 때문에 실질적으로 렌더링 파이프라인 상에서 처리되는 정점의 수는 복원 과정의 유무와 상관없이 동일하다.

## 6. 실험 결과

실험은 인텔 펜티엄 4 3.6GHz CPU, 2GB RAM, NVIDIA GeForce 7800 GTX 그래픽 카드를 장착한 데스크톱 PC 에서 수행했으며 실험에 사용한 모델들은 그림 6 과 같다. 실험에서는 정점 위치정보 데이터의 감소 비율과 메쉬 파티션들의 경계를 따라 생기는 중복 정점의 수를 측정하여 정점 부담을 계산했다. 데이터 감소 비율과 정점 부담은 다음과 같은 공식에 의해 계산했다.

$$Data\ Reduction\ Ratio = \left(1 - \frac{compressed\ data}{uncompressed\ data}\right) \times 100$$

$$Vertex\ Overhead = \frac{\#\ of\ duplicated\ vertices}{\#\ of\ vertices} \times 100$$

부호화된 위치정보에서 복원된 메쉬의 비율 왜곡(rate-distortion)은 일반적으로 메쉬 압축에서 활용하는 메트로 툴[4]을 사용하여 원본 메쉬와의 평균 $L^2$ 놈을 측정했다.

그림 5 는 무용수 모델에서 파티션의 수를 증가했을 때 왜곡, 데이터 감소 비율, 정점 부담의 변화를 보여준다. 파티션의 수가 많아지면 저장해야 하는 파티션 헤더 데이터가 증가하고 중복 저장되는 경계 정점이 많아지므로 데이터 감소 비율은 감소한다. 반면에 파티션의 수가 많아지면 파티션들의 경계 입방체의 크기가 작아지기 때문에 정량화 정확도가 높아져서 복원 메쉬의 왜곡은 감소하게 된다. 파티션 수에 따른 데이터량과 왜곡의 상관관계는 실험에 사용한 모든 모델에서 동일하게 나타난다. 이후의 모든 실험 결과에서는 다른 기법과의 비교를 위해 각 모델의 데이터 감소율이 70%(+0.1)일 때를 기준으로 결과를 측정했다.

(a) 8 비트 전역적 (b) 8 비트 지역적

그림 6. 8 비트 전역적 정량화와 8 비트 지역적 정량화의 영상 품질 비교.

| 모델 | 정점의 수 | 11-전역적 | 8-전역적 | 8-지역적 |
|---|---|---|---|---|
| 아이들 | 100,000 | 0.000436 | 0.003429 | 0.000409 |
| 무용수 | 24,998 | 0.000449 | 0.003622 | 0.000338 |
| 필라인 | 49,864 | 0.000442 | 0.003583 | 0.000398 |
| 장식물 | 514,300 | 0.000452 | 0.003642 | 0.000156 |
| 말 | 19,851 | 0.000430 | 0.003375 | 0.000462 |
| 다람쥐 | 9,995 | 0.000416 | 0.003360 | 0.000984 |

**표 1. 11 비트 전역적, 8 비트 전역적, 8 비트 지역적 정량화의 복원 메쉬의 왜곡 비교**

표 1 은 정량화 방법에 따른 복원 메쉬의 왜곡을 보여준다. 모든 모델에서 8 비트 지역적 정량화가 8 비트 전역적 정량화에 비해 왜곡이 훨씬 적은 것을 알 수 있다. 또한 대부분의 모델에서 8 비트 지역적 정량화가 11 비트 전역적 정량화와 비슷하거나 보다 나은 결과를 보여준다. 다람쥐 모델에서만 더 나쁜 결과를 보이는데 이것은 다람쥐 모델이 매우 작은 크기의 모델이기 때문에 파티션의 수를 충분히 만들 수 없기 때문이다.

그림 6 은 8 비트 전역적 정량화와 지역적 정량화의 복원 메쉬의 영상 품질을 비교한 것이다. 그림 6(a)의 전역적 정량화 결과에서는 시각적인 손상이 확연히 드러난다. 반면에 그림 6(b)의 지역적 정량화 결과에서는 원본 메쉬와 시각적으로 거의 구분이 불가능할 정도의 뛰어난 영상 품질을 보인다. 이것은 8 비트 지역적 정량화가 시각적 손상이 거의 없이 정점의 위치정보를 표현하기에 충분한 것을 나타낸다.

## 7. 향후 연구

본 논문은 모바일 그래픽스에 적합한 삼차원 메쉬의 정점 위치정보 압축 방법을 제시하였다. 본 논문에서 제시한 메쉬 분할과 지역적 정량화를 통해 위치정보 이외에도 정점의 법선 벡터, 색상, 텍스처 좌표 등을 압축하는 것이 가능하다. 그러나 위치정보 외의 정점 속성 데이터들은 각각의 속성이 갖는 특징을 고려하면 더욱 뛰어난 압축률을 달성할 가능성이 있다. 향후 연구에서는 기타 정점 속성들을 각각에 특성에 맞게 압축하는 방법을 연구할 필요가 있다.

## 감사의 글

본 논문에서 사용한 실험 모델 중 무용수, 아이들, 장식물, 필라인 모델은 각각 IMATI, IMATI-GE, SensAble technologies, Caltech 의 Multi-Res Modeling Group 의 제공 받은 것이며, 모든 모델은 AIM@SHAPE Shape Repository 에서 구한 것입니다.

## 참고문헌

[1] P. Alliez and C. Gotsman, "Recent advances in com-pression of 3D meshes," in Advances in Multiresolution for Geometric Modelling, N. A. Dodgson, M. S. Floater, and M. A. Sabin, Eds. Springer-Verlag, 2005, pp. 3 26.

[2] D. Calver, "Vertex decompression in a shader," in Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks, W. F. Engel, Ed. Wordware, 2002, pp. 172 187.

[3] S. Choe, M. Ahn, and S. Lee, "Feature sensitive out-of-core chartification of large polygonal meshes," in Proc. of Computer Graphics International 2006, 2006, pp. 518 529.

[4] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," Computer Graphics Forum, vol. 17, no. 2, pp. 167 174, 1998.

[5] M. Garland, A. Willmott, and P. S. Heckbert, "Hierarchical face clustering on polygonal surfaces," in Proc. 2001 ACM Symposium on Interactive 3D Graphics, 2001, pp. 49 58.

[6] S. Lloyd, "Least square quantization in pcm," IEEE Transaction on Information Theory, vol. 28, pp. 129 137, 1982.

[7] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: a survey," Journal of Visual Communication and Image Representation, vol. 16, no. 6, pp. 688 733, 2005.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, 2nd ed. Cambridge University Press, 1992, pp. 456 495.

[9] K. Pulli, T. Aarnio, K. Roimela, and J. Vaarala, "Designing graphics programming interfaces for mobile devices," IEEE Computer Graphics and Applications, vol. 25, no. 8, pp. 66 75, 2005.

[10] B. Purnomo, J. Bilodeau, J. D. Cohen, and S. Kumar, "Hardware-compatible vertex compression using quantization and simplification," in Proc. Graphics Hardware 2005, 2005, pp. 53 61.

[11] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe, "Multi-chart geometry images," in Proc. Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2003, pp. 146 155.

[12] J. Ström and T. Akenine-Möller, "iPACKMAN: High-quality, low-complexity texture compression for mobile phones," in Proc. Graphics Hardware 2005, 2005, pp. 63 70