외부 메모리를 이용한 메쉬 차트화

최성열[1] 김준호[1] 이혜영[2] 이승용[1]
[1]포항공과대학교 [2]홍익대학교

Out-of-core Mesh Chartification

Sungyul Choe[1] Junho Kim[1] Haeyoung Lee[2] Seungyong Lee[1]
[1]POSTECH [2]Hongik Univ.

요 약

본 논문은 거대한 메쉬를 효율적으로 처리하기 위해 이를 조각화하는 새로운 방법을 제시한다. 메쉬의 특징을 반영하여 데이터를 분할하기 위해서 로이드 양자화 방법을 적용하였으며, 거대한 메쉬 데이터를 다루기 위해 외부 메모리를 이용한 부분적인 차트화 갱신 방법을 제시한다. 차트화 갱신 과정에서 효율적으로 메쉬 데이터를 다루기 위해서 임의접근 가능 메쉬 압축구조를 활용한다. 이러한 방법을 통해서 수행된 외부 메모리를 이용한 차트화 결과가 지금까지 개발된 메쉬 차트화 결과와 크게 다르지 않음을 보여준다. 이와 같은 메쉬 차트화 시스템을 통해 거대한 메쉬에 기존의 파라미터화, 몰핑, 매칭, 재매쉬화 등의 메쉬 처리기법을 적용할 수 있다.

제 1 절 서론

모델링(modeling), 곡면복원(surface reconstruction), 파라미터화(parameterization)등과 같이 메쉬를 처리하기 위한 방법들이 연구되어 왔다. 이와함께 스캐닝(scanning)기술이 발전하면서, 거대하고 복잡한 메쉬를 손쉽게 얻을 수 있게 되었다[7]. 그러나 스캐닝을 통해 얻은 메쉬에 메쉬 처리 도구를 적용하려면 노이즈 제거(noise removal)와 같은 추가작업이 필요하다. 게다가 크기가 큰 메쉬의 경우, 메모리와 처리속도 등의 문제로인하여 메쉬 처리 도구를 바로 적용하기 힘든 경우가 자주 발생한다. 이러한 문제를 해결하기 위해 메쉬를 여러 조각(patch)으로 나누고, 나누어진 각각의 데이터들에 대해서 메쉬처리 도구를 적용하는 방법들이 연구되어 왔다.

메쉬를 여러 조각으로 나누는 연구는 파라미터화[2, 13, 8, 14], 메쉬압축(mesh compression)[3, 4], 메쉬몰핑(mesh morphing)이나 매칭(matching)과 같은 분야에서 주로 연구되어 왔으며, 각각의 분야에 적합한 방법들이 독립적으로 연구되어 왔다. 파라미터화를 위해 메쉬를 조각화 하는 경우 메쉬의 특징을 고려하여 메쉬를 분할한다. 거대한 메쉬를 압축하는 경우에는 거대한 메쉬를 다루기 위한 간단하고 직관적인 방법이 주로 사용되었으며, 이를 위해 메쉬가 위치하는 공간을 분할하는 방법이 주로 사용되었다. 그러므로 거대한 메쉬를 분할하는 경우 메쉬 조각화 결과는 메쉬의 특징이 전혀 반영되지 못한 의미없는 조각들로 구성된다.

이 논문은 거대한 메쉬를 여러 조각으로 나누는 새로운 방법을 제시한다. 이 방법을 통해 거대한 메쉬를 성공적으로 분할하고, 분할된 메쉬의 조각들은 메쉬의 특징을 반영하도록 한다. 이와같이 메쉬의 특징이 반영된 메쉬의 조각들은 각종 메쉬 처리 도구를 적용하기에 적합한 형태가 된다. 이 논문에서 제시하는 메쉬 분할 방법은 기존의 Lloyd 양자화 방법[10]을 기반으로하며, 처리과정의 효율성을 높이기 위해 임의접근 가능 메쉬압축 기법[1]을 이용하여 분할된 메쉬 데이터들을 디스크에 저장한다.

제 2 절 기존연구

2.1 메쉬 차트화

메쉬를 분할하기 위한 연구는 여러 방면에서 진행되어 왔다. 가장 흔히 사용되는 분야는 메쉬에 대한 파라미터화(parameterization)작업과 파라미터화 결과를 이용하여 텍스쳐매핑(texture mapping)을 수행하는 분야이다[11, 13, 8]. 메쉬를 분할하지 않고 전체 메쉬에 대한 파라미터화 결과를 얻을 수도 있지만, 이 경우 파라미터화의 결과가 왜곡이 심하게 되면 텍스쳐매핑과 같은 작업을 수행하였을 때 어색한 결과를 얻기 쉬워진다. 이러한 문제를 해결하기 위하여 전체 메쉬에 대해서 파라미터화를 수행하지 않고, 메쉬에 조각화를 수행한 후 분할된 각각의 메쉬 조각들에 대해서 파라미터화와 텍스쳐매핑을 수행하는 것이 일반적이다.

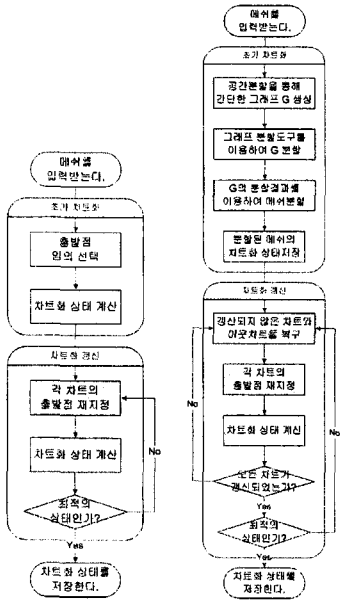
이외에도 메쉬 차트화 기법은 빠른 전체 조도 계산[2]이나, 재메쉬화(remeshing)[14], 메쉬단순화(mesh simplification)[2, 6]등에 널리 사용되고 있다.

2.2 외부 메모리를 이용한 메쉬처리기법

메쉬를 손쉽게 다루기 위해 개발된 대부분의 메쉬 처리기법들은 일반적으로 모든 메쉬 데이터가 메모리에 존재해야만 수행이 가능하다. 이러한 이유로 인하여 거대하고 복잡한 메쉬의 경우 기존의 메쉬 처리 기법들을 적용하기가 쉽지 않다. 이러한 문제를 해결하기 위해서 거대한 메쉬의 간단화[9, 5]와 거대한 메쉬 압축[3, 4]기법 등이 개발되고 있다. 거대한 메쉬를 다루는 경우 전체 메쉬 데이터 중 일부분의 데이터만을 메모리에 올려서 부분적인 작업을 반복적으로 수행하는 기법을 일반적으로 사용한다.

제 3 절 접근방법

Lloyd 양자화 방법(Lloyd's quantization method)[10]은 임의의 데이터를 유사한 종류의 데이터끼리 분류하는 알고리



(a) (b)

그림 1: 메쉬차트화 과정: (a) Lloyd 양자화 방법을 이용한 메쉬 차트화 과정; (b) 본 논문에서 제시한 메쉬 차트화 과정

즘 중 하나이다. Sander는 [14]에서 Lloyd의 방법을 메쉬에 적용하여, 메쉬를 분할하는 방법을 고안하였다. Sander가 제시한 방법을 이용하여 메쉬를 분할하면, 분할된 각 조각들은 메쉬의 특징(feature)을 잘 반영하게 된다. 이와 같이 메쉬의 특징이 반영된 메쉬 조각들에 대해서 파라미터화와 텍스쳐 맵핑과 같은 메쉬 처리 도구들을 적용하면, 이전의 결과들에 비해 나은 결과를 얻을 수 있다. 본 연구에서는 Lloyd 양자화 방법을 기반으로 거대한 메쉬를 분할하고자 한다.

3.1 Lloyd 양자화 방법을 이용한 메쉬 차트화

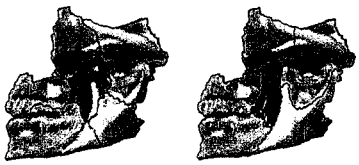
Lloyd 양자화 방법을 이용한 데이터 분할의 기본적인 전략은 다음의 두가지 과정을 반복하여 데이터 분할 결과를 최적에 접근한다:

- 주어진 출발점(seed)에 대한 최적의 분할 결과를 찾는다.
- 각각의 분할 데이터에 대해서 새로운 출발점을 찾는다.

이것을 메쉬에 대해서 설명하면 다음과 같이 표현된다.

- 주어진 각 출발면(seed face)에 대해서 영역확장 방법(chart growing)을 통해 최적의 차트를 찾는다.
- 각각의 차트들에 대해서 새로운 출발면을 찾는다.

이러한 과정은 그림 1(a)와 같이 표현된다. 위의 두가지 과정을 수행하면 메쉬에 대해서 한번의 차트화 갱신이 수행된다. 이러한 갱신과정을 반복하면 최적의 차트화에 가깝게 다가갈 수 있으며, 일정한 기준을 만족하여 최적의 차트화라고 판단되면 갱신 과정의 반복을 멈추게 된다.



(a) 갱신 이전 (b) 갱신 이후

그림 2: 부분적인 차트화 갱신: (a) 차트화 갱신을 위해 목표가 되는 차트와 그 이웃 차트들을 복구한 모습; (b) 부분적인 차트화 갱신을 수행한 이후의 모습

그림 1(a)에서 볼 수 있듯이 차트화 갱신 과정을 시작하기 위해서 초기 차트화(initial chartification)가 필요하다. 기존의 방법들의 경우 갱신과정에서 최적의 차트화가 찾아진다는 가정하에 초기 차트화는 임의로 선택하였다. [14]의 경우에도 초기의 출발면을 임의로 선택하는 방법을 통해서 초기 차트화를 정하였다.

3.2 기본 아이디어

Lloyd 양자화 방법을 기반으로 거대한 메쉬를 분할하려면, 한정된 메모리를 사용해야하기 때문에 발생하는 여러 문제를 해결해야만 한다. 이전의 방법은 차트화를 갱신하기 위해서 모든 데이터를 메모리에 올려두고, 메쉬 데이터 전체에 대해서 최적의 차트화에 접근하였다. 그러나 거대한 메쉬의 경우 모든 데이터를 메모리에 올려두는 것이 불가능 하기 때문에, 본 연구에서는 메쉬 일부분을 메모리에 올려두고, 선택된 일부분에 대해서 부분적으로 최적의 차트화를 찾는 방법으로 메쉬를 분할 한다. 본 연구에서 제시된 거대한 메쉬에 대한 차트화 과정은 그림 1(b)와 같다.

위에서 제시한 바와 같이 부분적인 데이터에 대해서 차트화를 갱신하기 위해서, 메쉬 데이터를 부분적으로 접근할 수 있는 구조가 필요하다. 거대한 메쉬를 다루는 연구에서 메쉬 데이터를 부분적으로 접근하는 일반적인 방법은 [3, 4]에서 수행했던 것과 같이 메쉬가 차지하는 공간을 분할하는 방법이다. 분할된 공간에 속한 메쉬 데이터들은 독립적으로 저장되고 접근된다. 그러나 이와같은 간단하고 직관적인 방법은 본 연구에서 제시하는 차트화 갱신을 원활하게 수행하기에는 적합하지 않다. 왜냐하면 차트화 갱신을 통해 차트들의 외곽선이 변하며, 이러한 외곽선의 변화를 계속해서 유지하기에는 이전의 간단하고 직관적인 메쉬 분할 방법이 적합하지 않기 때문이다. 이러한 문제들을 고려하여 본 연구에서는 전체 데이터에서 일부분의 데이터를 접근하기 위한 도구로 임의접근 가능 메쉬압축 구조 (*random accessible mesh compression framework*)[*1*]를 적용한다.

3.3 부분적인 메쉬조각화 차트화 갱신

거대한 메쉬에 대해서 차트화 갱신을 수행하기 위해서 메쉬의 일부분에 대해서 갱신을 수행하게 된다. 본 논문에서 제시된 방법은 한 번에 하나의 차트에 대해서 메쉬 차트화를 갱신하는 것을 원칙으로 한다. 하나의 차트에 대해서 메쉬 차트화를 갱신하기 위해서는, 목표가 되는 하나의 차트와 그 차트에 이웃한 차트들을 메모리에 올려두고 작업을 수행하여야만 목표가 되는 차트의가 올바르게 갱신된다. 이러한 이유에서 부분적인 차트화 갱신을 수행하기 위해서는 그림 2와 같이 목표가 되는 차트와 그 이웃 차트들이 메모리에 복구된 이후에 갱신 작업을 수행한다.

그림 3: 부분적인 차트화 갱신과정: (a) 초기 차트화; (b)-(d) 하나의 차트와 그 이웃차트들을 포함하는 영역에 대하여 메쉬 차트화 갱신

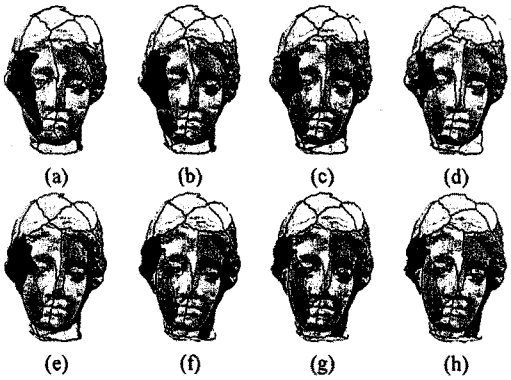


그림 4: 차트화갱신 비교. 동일한 초기 차트화에 대해서 수행시작; (a)-(d) 전체적인 차트화 갱신[14]을 적용한 경우. 차례대로 1회갱신, 2회갱신, 3회갱신, 4회갱신; (e)-(h) 부분적인 차트화 갱신을 적용한 경우. 차례대로 1회갱신, 2회갱신, 3회갱신, 4회갱신

위와 같이 부분적인 차트화 갱신을 모든 차트에 대해서 수행하면, 모든 차트들의 차트화가 최적에 가깝게 접근하게 된다. 그림 3은 부분적인 차트화 갱신을 수행하는 모습을 차례대로 보여주고 있다. 그림에서 볼 수 있듯이 갱신작업이 수행된 부분은 이전에 비해서 메쉬의 특징(feature)를 더 잘 반영하고 있음을 확인할 수 있다.

기존의 [14]에서와 같이 전체 데이터에 대해서 차트화 갱신을 수행하는 경우 데이터 전체에 대해서 최적의 차트화에 접근하는 것에 비해서, 부분적인 차트화 갱신의 경우 목표가 되는 하나의 차트에 대한 최적의 차트화에 접근한다는 차이가 있다. 이러한 차이점에 의해서 차트화 결과가 얼마나 나빠질 수 있는가에 대해서 고려해야만 한다. 전체적인 차트화 갱신과 부분적인 차트화 갱신의 결과를 비교하기 위해서 동일한 모델에 대해서 차트화 갱신을 수행하여 비교하였으며, 그 결과는 그림 4와 같다. 그림 4에서 윗쪽 열은 전체적인 차트화 갱신을 수행한 결과이며, 아랫 열은 부분적인 차트화 갱신을 수행한 결과이다. 그림 4에서 확인할 수 있듯이 차트화갱신 결과는 거의 유사하다는 것을 확인 할 수 있으며, 이러한 결과를 통해 부분적인 차트화 갱신을 수행하더라도 메쉬 분할 결과는 나빠지지 않음을 확인할 수 있다.

제 4 절 초기차트화와 차트화 저장

4.1 초기 차트화 과정

Lloyd 양자화 방법을 이용한 데이터 분할 방법은 기본적으로 초기 분할 데이터가 존재하여야만 차트화 갱신을 통해 최적의 분할 결과를 찾을 수 있다. 기존의 메쉬 분할 방법의 경우 임의로 출발면을 선택하여 초기 차트화를 결정한다. 하지만 거대한 메쉬의 경우 모든 데이터가 메모리에 존재하는 것이 아니기 때문에 메쉬 전체에 대해서 무작위로 출발면을 선택하기가 쉽지 않다. 또한 출발면을 임의로 선택하는 경우, 메쉬 차트화를 갱신하는 과정에서 부분 최적(local minimum)에 빠지는 경우가 생길 수 있기 때문에 메쉬 전반적으로 균일하게 분포된 출발면을 선택하는 것이 더 좋다.

이와 같은 문제를 해결하여 양질의 초기 차트화를 얻기 위해서 이 논문에서는 간단하고 직관적인 방법으로 메쉬를 균일하게 쪼개고, 그 결과를 이용하여 메쉬 차트화를 갱신하였다. 이를 위해 기존 방법들 중 Ho의 외부 메모리 사용 메쉬 차트화 방법[3]을 이용하였다.

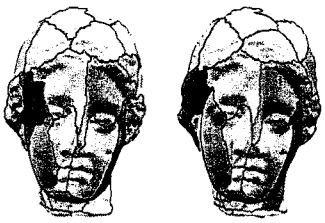
Ho의 방법은 먼저 메쉬가 차지하는 공간을 균일하게 나눈다. 이와같이 공간을 나누면, 나뉘어진 공간을 이용하여 메쉬를 단순한 그래프(simplified graph) G로 표현하는 것이 가능하다. 이 때, 단순한 그래프의 각 노드(node)들은 나뉘어진 공간의 각 셀(cell)에 대응되며, 각 셀들은 이웃한 셀들과 연결관계를 갖는다. 나뉘어진 각 셀들 중에서 메쉬의 정점이 하나도 존재하지 않는 셀들을 G에서 제외하게 되면, 메쉬를 대표하는 단순한 그래프 G를 손쉽게 얻을 수 있다. 단순 그래프 G를 얻게 되면, METIS[12]와 같은 그래프 분할 도구(graph partitioning tool)을 이용하여 단순 그래프 G를 분할할 수 있다. 이 때, G는 주어진 메쉬를 대표하기 때문에 G를 분할하는 작업은 결국 주어진 메쉬를 분할하는 것과 같은 작업이 된다. 이와같이 Ho의 방법을 이용하면 거대한 메쉬를 균일하게 분할 할 수 있다. 하지만 Ho의 방법으로 얻은 결과는 단순히 공간을 기준으로 균일하게 나눈것 이상은 아니기 때문에 메쉬 차트화 결과는 메쉬의 특징을 전혀 반영하지 못한다.

4.2 차트화 저장구조

위와같이 간단히 분할된 초기 차트화를 얻게 되면, 이를 이용하여 부분적인 차트화 갱신을 수행하여 메쉬의 특징을 잘 반영한 메쉬 차트화 결과를 얻을 수 있다. 그러나, 부분적인 차트화 갱신을 통해서 메쉬의 조각들이 동적으로 변하기 때문에 분할된 메쉬 데이터를 다룰수 있는 구조가 필요하게 된다. 이를 위해서 이 논문에서는 임의접근 가능 메쉬압축[1] 구조를 활용하여 분할된 데이터를 저장하고, 저장된 데이터들에 접근할 수 있도록 한다.

임의접근 가능 메쉬압축 구조를 이용하여 메쉬 차트화를 저장하면 다음과 같은 장점을 가진다.

- 분할된 각 차트들은 압축된 형태로 디스크에 저장되기 때문에 디스크에 존재하는 데이터의 크기를 줄일 수 있을뿐만 아니라, 디스크와 메모리 사이의 데이터 교환을 용이하게 해 준다.
- 임의접근 가능 메쉬압축 구조에서 제공하는 데이터 구조를 이용하여 차트들간의 인접관계를 손쉽게 알 수 있을 뿐만 아니라 차트들간의 인접관계가 변화할 경우 이를 손쉽게 반영할 수 있다.



(a) (b)

그림 5: Igea 모델에 대한 차트화 결과 비교 (a) 기존의 [14]을 적용하여 얻은 메쉬 차트화 결과; (b) 부분적인 갱신방법을 통해 얻은 메쉬 차트화 결과

제 5 절 외부 메모리 사용 차트 갱신

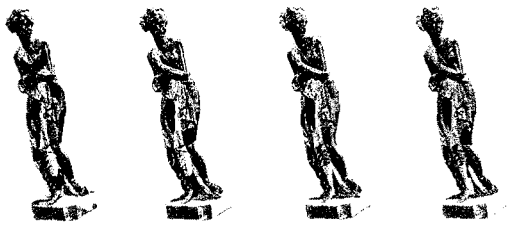
앞에서 본것과 같이 초기 차트화를 얻게 되면 부분적인 차트화 갱신과정을 반복하여 최적의 차트화에 접근한다. 부분적인 차트화 갱신 과정은 앞에서 보았듯이 크게 두부분으로 나뉘며, 각각은 다음과 같은 방법을 통해서 수행된다.

- 새로운 출발점 찾기: 주어진 차트화에서 새로운 출발점을 찾는 과정은 연결관계 측면에서 차트의 중점(topological center)을 찾는 과정과 같다. 차트의 외곽선(chart boundary)에서부터 차트의 내부(chart interior)를 향하여 영역을 넓혀간다. 이때 연결관계 측면에서 차트의 외곽선에 가장 가까운 면부터 선택하여 영역에 포함시킨다. 이런 방법으로 영역을 넓혀가며, 이 과정에서 가장 마지막으로 영역에 포함되는 면이 바로 새로운 출발점으로 선택된다.

- 영역확장을 통한 차트화 계산: 새로이 선택된 출발면에 대해서 최적의 차트화를 찾아내는 과정이다. 주어진 출발면들을 각각 하나의 차트로 만든다. 그리고 메모리에 존재하는 모든 면 중에서 일정한 기준에 가장 부합하는 하나의 면을 선택하고, 그 면을 이웃한 차트에 포함시킨다. 이 때 면을 선택하는 기준은 여러가지가 존재할 수 있지만, 이 논문에서의 시스템에서는 메쉬의 특징을 잡아내기 위해서 각 차트의 평균 법선벡터와 면의 법선벡터와의 차이가 가장 작은 면을 선택한다. 이와같은 기준으로 메모리상에 존재하는 모든 삼각형들이 어떤 차트에 속할때 까지 면을 차트에 포함시키는 영역확장(region growing) 작업을 계속한다. 영역 확장이 끝나면 메모리상의 모든 삼각형들은 어떤 차트에 속하게 되고, 결국 메쉬는 일정한 차트들로 분할 된다.

위와같은 과정을 통해 부분적인 차트화를 갱신한 후, 갱신된 차트화를 반영하여 디스크에 저장된 데이터를 갱신한다. 이러한 작업을 전체 차트들에 대해서 수행하며, 차트들의 차트화가 모두 갱신되면 주어진 메쉬에 대해 한 번의 차트화 갱신 작업을 수행한 것으로 본다. 이와 같은 메쉬에 대한 차트화 갱신 작업을 반복하여 주어진 기준에 의해 최적의 차트화에 도달했다고 판단되면 메쉬의 차트화 갱신 과정을 멈춘다.

그림 5은 메쉬 차트화 결과를 비교해주고 있다. (a)는 기존의 방법을 이용하여 얻은 메쉬 차트화 결과이고, (b)는 부분적인 갱신과정을 통해서 얻은 메쉬 차트화 결과이다. 그림에서 알 수 있듯이 메쉬 차트화 결과는 서로 유사하다는 것을 알 수 있으며, 부분적인 갱신과정을 통해 얻은 메쉬 차트화 결과가 최적에 가깝다는 것을 알 수 있다.



(a) (b) (c) (d)

그림 6: Iphigene 모델에 대한 외부 메모리를 이용한 메쉬 차트화 결과; (a) 초기차트화; (b) 첫 번째 갱신 된 차트화; (c) 두 번째 갱신 된 차트화; (d) 세 번째 갱신 된 차트화

제 6 절 실험 결과

그림 6은 Iphigene모델에 대해서 외부 메모리를 이용한 메쉬 차트화 과정을 단계적으로 보여주는 그림이다. 그림 6에서 보는 바와 같이 갱신과정을 거칠수록 차트들이 메쉬의 특징을 잘 반영하며, 갱신 횟수가 증가할 수록 최적의 차트화에 접근한다는 사실을 확인할 수 있다.

표 1는 본 논문에서 제안하는 메쉬 차트화 기법의 수행시간과 기존의 [14]의 수행시간을 보여주고 있다. 외부메모리를 이용하는 작업의 경우 일반적으로 외부 메모리와 메인 메모리 간의 입출력 과정에서 발생하는 병목현상으로 인하여 수행시간이 기존의 방법에 비해서 상당히 많이 걸린다. 그러나 본 논문에서는 임의접근 가능 메쉬압축 구조를 이용하여 차트 데이터를 관리하기 때문에, 각각의 차트 데이터들은 압축된 형태로 외부 메모리와 메인 메모리 사이에서 전송된다. 이러한 이유로 차트 데이터 전송시 발생할 수 있는 병목현상을 현저히 줄일 수 있었다.

제 7 절 결론

본 논문에서는 부분적인 차트화 갱신과정을 이용하여 한정된 메모리를 가지고도 기존의 연구와 비슷한 양질의 메쉬 차트화 결과를 얻을 수 있는 시스템을 제안하였다. 이 시스템을 통해서 메모리에 올릴 수 없는 거대한 메쉬에 대해서 메쉬의 특징을 반영한 메쉬 차트화 결과를 얻을 수 있다. 부분적인 차트화 갱신과정을 통한 메쉬의 차트화 갱신 결과가 기존의 차트화 갱신 결과와 다르지 않다는 사실을 실험을 통해 알 수 있었으며 이러한 사실을 통해 이 논문에서 제시된 외부메모리 사용 메쉬 차트화 시스템을 통해 메쉬의 특징이 잘 반영된 최적의 결과를 얻을 수 있음을 알 수 있다.

분할 된 각 차트 데이터들은 임의접근 가능 메쉬압축 구조를 이용하여 디스크에 저장된다. 이러한 구조를 사용하기 때문에 각 차트 데이터들은 압축된 형태로 접근이 가능하며, 이를 통해 디스크와 메모리 사이에서 데이터를 전송하면서 발생하는 병목현상으로 인한 지연을 현저히 감소시킬수 있다. 또한 임의접근 가능 메쉬압축 구조에서 제공되는 정보를 통해 차트들간의 인접관계를 손쉽게 알 수 있으며, 갱신된 차트화로 인하여 변화된 인접관계를 손쉽게 반영할 수도 있다.

model	#charts	initial of [14] (min' sec")	update once by [14] (min' sec")	OOCC initial (min' sec")	OOCC update once (min' sec")
Igea	50	0'30"	0'20"	2' 16"	0' 32"
	70	0'30"	0'18"	4' 52"	0' 33"
	100	0'30"	0'16"	5' 51"	0' 33"
Iphigenie	70	N/A	N/A	18' 08"	3' 37"
	100	N/A	N/A	19' 48"	3' 51"

표 1: 메쉬 차트화 수행시간의 비교

감사의 글

본 논문에서 사용된 Igea모델과 Iphigenie모델은 Cyberware와 RWTH-Aachen에서 구하였습니다. 본 연구는 BK21 사업을 통하여 포항공과대학교 전자.컴퓨터공학부에 주어진 교육부의 재정 지원, 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성 및 지원사업과 한국과학재단 국제공동연구(KOSEF-042010702, F01-2005-000-10377-0)의 지원을 얻어 수행되었습니다.

참고 문헌

[1] S. Choe, J. Kim, H. Lee, S. Lee, and H.-P. Seidel. Mesh compression with random accessibility. *The 5th Korea-Israel Bi-National Conference on Geometric Modeling and Computer Graphics 2004*, pages 81–86, 2004.

[2] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proc. 2001 ACM Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

[3] J. Ho, K.-C. Lee, and D. Kriegman. Compressing large polygonal models. In *Proceedings of the IEEE Visualization 2001*, pages 357–362, 2001.

[4] M. Isenburg and S. Gumhold. Out-of-core compression for gigantic polygon meshes. *Siggraph 2003 Conference Proceedings*, pages 935–942, 2003.

[5] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink. Large mesh simplification using processing sequences. In *Proceedings of the IEEE Visualization 2003*, pages 465–472, 2003.

[6] A. D. Kalvin and R. H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, 1996.

[7] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. *ACM Computer Graphics (Proc. SIGGRAPH 2000)*, pages 131–144, 2000.

[8] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Computer Graphics (Proc. SIGGRAPH 2002)*, pages 362–371, 2002.

[9] P. Lindstrom. Out-of-core simplification of large polygonal models. *Siggraph 2000 Conference Proceedings*, pages 259–262, 2000.

[10] S. Lloyd. Least square quantization in PCM. *IEEE Transaction on Information Theory*, 28:129–137, 1982.

[11] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. *ACM Computer Graphics (Proc. SIGGRAPH '93)*, pages 27–34, 1993.

[12] METIS. http://www-users.cs.umn.edu/~karypis/metis/ by George Karypis and Vipin Kumar.

[13] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. *ACM Computer Graphics (Proc. SIGGRAPH 2001)*, pages 409–416, 2001.

[14] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Proc. 2003 Eurographics Symposium on Geometry Processing*, pages 146–155, 2003.