적응형 정점 군집화를 이용한 메쉬 분할

김대영[O] 김종원 이혜영

홍익대학교 컴퓨터공학과

{dykim99, breezjw}@gmail.com, leeh@hongik.ac.kr

A Mesh Partitioning Using Adaptive Vertex Clustering

Daeyoung Kim[O] Jongwon Kim Haeyoung Lee

School of Computer Engineering. The Hongik University of Korea

요 약

본 논문에서는 분할 축과 평면의 위치를 동적으로 결정하는 적응형 KD 트리 구조를 이용한 정점 군집화(Adaptive Vertex Clustering) 알고리즘과 이를 이용한 새로운 메쉬 분할 방법을 소개하고자 한다. 정점 군집화는 주로 한 개의 거대한 3차원 메쉬를 여러 개의 파티션(Partition)으로 분할하여 효율적으로 처리하고자 할 때 사용되는 기법으로, 옥트리 구조를 이용한 공간 분할 기법과 K-평균 군집화(K-Means Clustering) 방법 등이 있다. 그러나 옥트리 방식은 공간 분할 축과 이에 따른 분할된 공간의 크기가 고정되어 있어서 파티션 메쉬 면의 정렬 상태가 고르지 못하고 포함된 정점의 개수가 균등하지 못한 단점이 있다. 또한, K-평균군집화는 균등한 파티션을 얻을 수 있는 반면 반복처리와 최적화를 위해 많은 시간이 소요된다는 단점이 있다. 본 논문에서는 적응형 정점 군집화를 통해 빠른 시간에 균등한 메쉬 분할을 생성하는 알고리즘을 제안하고자 한다. 본 적응형 KD 트리는 메쉬가 포함된 경계상자(Bounding Box) 공간을 정점의 개수와 분할 축의 크기를 기준으로 계층적으로 분할한다. 그 결과 각 파티션 메쉬는 컴팩트성(compactness)의 특성을 유지하며 균등한 수의 정점을 포함하게 되어 각 파티션의 균등한 처리시간 및 메모리 소요량 등의 장점을 살려 향후 메쉬 간소화 및 압축 등의 다양한 메쉬 처리에 활용될 수 있기를 기대한다. 본 방법을 적용한 3차원 모델의 실험 통계와 분할된 파티션 메쉬의 시각적인 결과도 함께 제시하였다.

Abstract

In this paper, a new adaptive vertex clustering using a KD-tree is presented for 3D mesh partitioning. A vertex clustering is used to divide a huge 3D mesh into several partitions for various mesh processing. An octree-based clustering and K-means clustering are currently leading techniques. However, the octree-based methods practice uniform space divisions and so each partitioned mesh has non-uniformly distributed number of vertices and the difference in its size. The K-means clustering produces uniformly partitioned meshes but takes much time due to many repetitions and optimizations. Therefore, we propose to use a KD-tree to efficiently partition meshes with uniform number of vertices. The bounding box region of the given mesh is adaptively subdivided according to the number of vertices included and dynamically determined axis. As a result, the partitioned meshes have a property of compactness with uniformly distributed vertices.

키워드: 정점 군집화, 메쉬 분할, KD 트리
Keywords: Vertex Clustering, Mesh Partitioning, KD Tree

1. 서 론

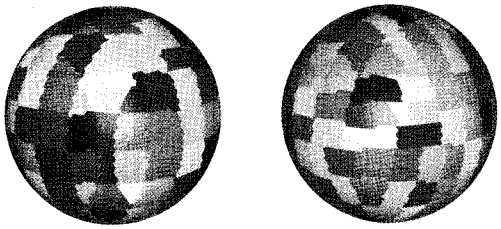
최근 3D 스캐닝 기술의 발달로 인해 거대한 3차원 메쉬 모델이 비교적 쉽게 생성되고 있다. 하지만 이러한 메쉬를 처리하기에는 메모리나 CPU 등의 일반 컴퓨터 자원이 제한적이다. 이에, 대용량 메쉬를 효율적으로 처리하기 위해 간소화(Simplification), 압축(Compression), 외부메모리(Out-of-Core)를 이용한 처리 등의 다양한 메쉬 처리 알고리즘이 발표되고 있다. 이러한 알고리즘들은 대부분 원본 메쉬를 여러 개의 파티션 메쉬로 분할하는 것이 특징이다.

메쉬를 분할하는 기준은 메쉬 처리 용도나 방식에 따라 달라질 수 있다. 메쉬 처리는 주로 삼각형의 면을 구성하는 정점 처리가 기본이다. 그러므로, 균등한 수의 정점을 포함하고 분할된 영역의 컴팩트성(Compactness, 식 (1) 참조)을 유지한다면 균등한 처리시간과 메모리가 소요되므로 캐쉬나 외부메모리 I/O 사용시 보다 효율적으로 처리할 수 있으며 각 파티션 메쉬의 대표 정점을 좀 더 정확하게 산출할 수 있다. 또한 메쉬가 컴팩트성을 띌수록 압축 효율을 높일 수 있다. 기존의 메쉬 분할 방식 중 공간 분할 기법의 하나인 옥트리 방식은 정확하게 균등한 파티션 메쉬를 얻을 수 없으며, K-평균군집화 방법은 일정 오차 범위내의 결과를 얻기 위해서 복잡한 비용 함수를 사용하며 최적화를 위한 반복 수행으로 상대적으로 긴 처리 시간이 요구된다. 본 논문에서는 KD 트리를 이용하여 공간 분할 기법의 장점인 빠른 분할 속도와 K-평균군집화의 장점인 균등한 메쉬 분할 속성을 갖춘 새로운 정점 군집화 알고리즘을 소개하고자 한다. 본 알고리즘은 메쉬 정점의 개수와 분할 축의 크기에 기반한 적응형 정점 군집화(Adaptive Vertex Clustering)를 수행하고 그 결과를 활용하여 메쉬를 여러 개의 파티션으로 분할하는 알고리즘이다. 이 알고리즘을 활용하여 대용량의 메쉬를 분할 압축하거나 간소화하는 데 응용할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서는 메쉬 분할에 대한 관련 연구를 살펴보고 3절에서 본 논문에서 제시한 적응형 정점 군집화를 통한 메쉬 분할 알고리즘을 소개한다. 마지막으로 4절에서 결론 및 향후 연구과제에 대해 언급한다.

2. 관련 연구

메쉬 분할은 크게 계층적인 공간 분할 또는 주위의 점 또는 면을 검색하여 추가하는 군집화를 통해 이루어진다. 3차원 공간을 계층적으로 분할하는 기법은 간소화, 재구성(Reconstruction), LOD(Level of Detail) 및 가시성(Visibility) 등을 활용한 렌더링, 충돌처리 등의 다양한 메쉬 처리에 사용된다. 이 기법은 메쉬를 포함하는 경계 상자(Bounding Box)를 만든 후에 사용자가 정해준 기준에 도달할 때까지 재귀적으로 공간을 계층적으로 분할한다. 주로 사용되는 방식으로 옥트리(Octree), kD-트리, BSP 트리[1]등이 존재하며 이들 데이터 구조를 이용하여 메쉬를 분할하는 많은 연구가 진행되어 왔다. 군집화 중 가장 기본적인 방법은 유니폼 군집화로 메쉬를 포함하는 유니폼 격자를 구성하고 한 격자 셀에 포함되는 점들을 하나의 군집으로 묶은 후에 대표점으로 표현하는 방법이다. [4,5]는 메쉬 간소화에 이 기법을 사용하였다. [6]에서는 유니폼 격자에 BSP 트리 구조를 추가하여 군집화를 수행하였다. 옥트리를 이용한 알고리즘으로는 제한된 메모리를 가진 시스템에서 대용량의 메쉬를 처리할 수 있도록 하는 OEMM(Octree Based External Memory Mesh)[2]이 있다. 최근에는 3차원 공간을 분할하는 옥트리와 2차원 공간을 분할하는 쿼드 트리(Quad-Tree)를 혼합한 VS-트리[3]도 소개되었다. [7]에서는 유니폼 격자 대신 정해진 개수의 노드를 갖는 적응형 옥트리를 사용하여 군집화를 수행하였고 [8]에서는 이중 그래프(Dual Graph)를 사용하여 점 대신 면을 계층적으로 군집화하는 기법을 소개하였다. 이외에도 메쉬를 분할하는 데 많이 사용되는 기법으로 K-평균 군집화(K-Means Clustering)라고 불리는 Lloyd[9] 알고리즘이 있다. 이 알고리즘은 k개의 임의의 시작점/면(Seed)을 정하는 것으로부터 시작한다. 주어진 비용 함수(Cost Function)를 사용하여 시작점/면에 근접한 점 또는 면을 추가시켜 나간다. 모든 점 또는 면이 추가되어 k개의 분할 부분이 만들어지면 각 분할 부분 내에서 시작점/면을 재 선택하고 위의 과정을 반복한다. 메쉬 간소화를 위해 Lloyd 알고리즘을 이용한 논문으로는 [10,11]등이 있다. [12]에서는 임의 접근 가능한 메쉬 압축을 위해 메쉬를 분할하는데 이 알고리즘을 사용하였다.



(가) 적응형 옥트리 (나) 적응형 KD 트리

그림 1. 적응형 옥트리와 정점-군집 트리의 분할 결과 비교

대용량 메쉬 처리는 거대한 크기의 메쉬를 제한된 메모리와 디스크 I/O 대역폭에 적합한 크기를 갖는 파티션 메쉬로 나누는 작업이 필수적이다. 현재까지 많은 연

구가 진행되어 왔다. 특히, [12,13]에서도 언급했듯이 균등한 크기의 파티션으로 분할하는 것은 메쉬 압축 등의 다른 처리들을 위해 중요한 쟁점이다.

본 논문에서 제시하는 적응형 정점 군집화 알고리즘은 대용량 메쉬 처리, 빠른 속도, 계층성 제공, 파티션의 균등성과 컴팩트성 유지 등의 특성을 고려하도록 설계되었다. 기존의 메쉬 분할 알고리즘들은 이러한 특성들을 동시에 만족시켜주지 못한다. Lloyd 알고리즘은 비용함수와 반복 회수에 따라 해당 메쉬 처리 목적에 맞는 양질의 분할 결과를 얻을 수 있는 반면 대용량 메쉬를 분할하는 데에는 부적합하고 좋은 결과를 얻기 위해 여러 번 반복해야 하기 때문에 많은 시간이 소요된다는 단점이 있다 (표2 참조). 또한 분할되는 과정에 시작점/면을 정해준다든가 분할 후에 최적화하는 일련의 작업이 필요하다. 파티션 메쉬 간의 계층성을 제공해주지 못하는 것도 단점 중에 하나이다.

[7][14]에서는 각각 대용량의 메쉬에 대해 간소화(Simplification), 재구성(Reconstruction)을 하기 위해 적응형 옥트리(Adaptive Octree) 방식을 사용하였다. 옥트리를 사용할 때의 장점은 빠른 분할 시간과 계층성의 제공이다. 하지만 옥트리를 사용한 분할 방식의 단점은 비어있는 노드가 많이 발생하고 분할된 파티션 메쉬 간의 크기나 형태에 차이가 많다는 것이다 (그림 1 참조).

본 논문에서는 위에서 제시한 특성들을 만족시키기 위해 KD 트리 구조를 3차원 메쉬 분할에 도입하였다. 또한 정점을 군집화하는 것이 면을 군집화는 것에 비해 계산 비용과 데이터 표현량이 적기 때문에 제한된 메모리 가진 시스템에서 대용량의 메쉬를 처리하는 인터랙티브한 애플리케이션에 더 적합하다는 판단 하에 점의 개수를 균등하게 분할하는 알고리즘을 연구하였다.

3. KD 트리를 이용한 적응형 정점 군집화

본 논문에서 제시한 KD 트리는 다음과 같은 특징을 갖는다.

① 크기가 균등한 파티션으로 분할

제한된 메모리 크기에 맞게 대용량의 메쉬를 점의 개수가 동일한 균등한 파티션으로 분할한다. 파티션 메쉬들을 외부 메모리에 저장하였다가 인코어(In-Core) 알고리즘을 사용하는 다른 메쉬 처리에서 사용할 수 있다. 이 때 파티션들의 데이터 크기가 일정하므로 디스크 I/O의 효율성을 높일 수 있다.

② 계층성

기본적으로 KD 트리 구조가 계층성을 가지므로 점진적인 전송(Progressive Transmission)이나 LOD를 제공하는 메쉬 처리에 적합하다. 점진적인 압축이 대표적인 예이다.

③ 빠른 속도

다른 알고리즘에 비해 상대적으로 빠른 시간 안에 메쉬를 분할한다(표 1, 2 참조).

④ 분할된 파티션 메쉬가 컴팩트성의 속성을 유지

[15]에서 언급한 컴팩트성(Compactness)의 정의에 따르면 두 물체의 부피가 동일할 때 표면적이 더 작은 물체가 컴팩트성이 더 높다.

3D 컴팩트성 = (면적^3)/(부피^2) ---------- (식 1)

따라서 긴 직육면체 보다 정육면체가 컴팩트성이 더 높기 때문에 고정된 축을 사용하지 않고 메쉬를 둘러싸는 경계 상자의 각 축의 길이를 이용하여 동적으로 축을 선택하여 메쉬를 분할하도록 하였다. 결과적으로 생성되는 두 파티션 메쉬는 선택된 분할 축 외에 다른 분할 축을 사용했을 때보다 정육면체 모양에 더 근접하기 때문에 컴팩트성의 속성을 유지하게 된다. 나눠진 부분의 파티션 메쉬는 주로 한 개의 연결된 패치(Surface Patch)로 구성되나, 복잡한 메쉬의 특성상 나눠진 영역 내의 파티션 메쉬가 두 개 이상의 패치로 구성될 수 도 있다.

3.1 적응형 KD 트리 구조

적응형 KD 트리는 [16]에서 제시한 것처럼 SAH(Surface Area Heuristic)를 이용하여 분할 축과 분할 평면을 동적으로 결정하는 알고리즘을 적용시켰다. 즉, 정점들을 포함하는 경계상자를 고려하여 분할 축을 정하고 축의 양쪽 노드에 균등한 개수의 점이 존재하도록 분할 평면을 정하였다(그림 3 참조).

3.2 적응형 KD 트리 생성 및 정점 군집화

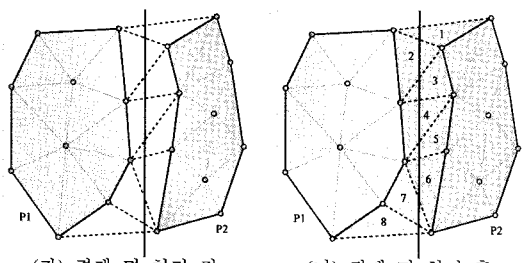
다음은 주어진 메쉬에 대해서 적응형 KD 트리를 만드는 과정에 대한 수도(Pseudo) 코드이다.

Vcell : 노드 내부의 정점
Ncell : 노드 내부의 정점의 개수
Nuser : 사용자가 입력한 각 노드당 정점의 개수
B(Vcell) : 노드 내부의 정점들의 경계 상자
RMin, RMin : 정점의 범위
AxisSplit : 분할하고자 하는 기준 축

1) 적응형 KD 트리 생성
① 만약 RMin * Nuser ≤ Ncell < RMax * Nuser 을 만족한다면 모듈을 종료한다.
② B(Vcell) 의 x,y,z 축의 길이를 계산하여 가장 긴 축을 AxisSplit로 선택한다.
③ AxisSplit을 기준으로 노드의 정점을 정렬한다.
④ 정렬된 정점 중 (Ncell * (1/2))와 (Ncell * (1/2)+1)의 중간에 해당하는 AxisSplit의 좌표를 계산한다.
⑤ 위에서 계산된 AxisSplit의 좌표를 가지고 하위 노드를 생성한다.
⑥ 하위 노드에 대해서 1)을 재귀적으로 수행한다.

2) 적응형 KD 트리를 이용한 메쉬 분할
① 메쉬의 각 면에 대해 1) 모듈에서 생성된 KD 트리의 리프 노드에 포함이 되는지 판단한다.
② 포함이 된다면 면을 해당 리프 노드로 복사한다.

2)에서 두개의 리프 노드에 걸쳐져 있는 즉, 경계에 해당되는 면들이 존재한다. 이 면들을 양쪽 노드에 포함된 파티션 메쉬의 면적을 고려하여 처리한다. 선택된 면들이 노드의 경계에 걸쳐있을 경우 양쪽 노드에 속한 파티션 메쉬의 면적을 계산하여 더 적은 쪽에 포함시킨다. 결과적으로 분할된 메쉬는 비슷한 면적을 갖게 되므로 균등성을 지향하는 본 알고리즘에 부합하다. 그림 2는 각 노드에 속한 파티션 메쉬의 면적을 고려하여 경계에 걸친 면을 처리하는 과정을 보여준다. 그림에서 볼 수 있듯이 양쪽 파티션의 점의 개수는 균등하지만 왼쪽 파티션 P1의 메쉬보다 오른쪽 파티션 P2의 메쉬 면적이 더 적다. 따라서 경계에 걸친 면들 중에서 P2에는 면 1~6을, P1에는 면7,8을 포함시켜서 P1과 P2의 최종 면적이 비슷하도록 분배한다. 결과적으로, 점의 개수 뿐만 아니라 각 파티션 메쉬의 면적도 비슷한 결과를 얻을 수 있다.



(가) 경계 면 처리 전 (나) 경계 면 처리 후

그림 2. 면적을 고려한 경계 면 처리

3.3 적응형 군집화의 특성

① 적응형 옥트리에 비해 파티션 메쉬의 개수 조절 용이

옥트리를 통해 메쉬를 분할하는 경우에는 파티션 메쉬의 개수가 빈 노드를 제외하고 대략 8의 제곱배가 된다. 즉 레벨 1의 경우 리프 노드의 수는 8개, 레벨 2의 경우 64, 레벨 3은 512가 된다. 따라서 레벨이 하나씩 증가할 때마다 파티션 메쉬의 수가 급격히 증가하기 때문에 사용자가 원하는 정점의 분포도를 가지는 레벨을 정하기가 어렵다. 적응형 KD 트리의 경우에는 빈 노드 없이 파티션 메쉬의 수가 2의 제곱배가 된다. 즉 레벨이 증가함에 따라 2, 4, 8, 16, ... 개의 파티션 메쉬를 가지게 된다. 이러한 특성으로 인해 사용자가 원하는 파티션의 개수에 보다 근접하게 나눌 수 있는 이점이 있다.

② 한 노드내의 정점의 개수가 동일

사용자가 한 노드에 포함될 정점의 개수를 정하면, 모든 노드의 정점의 개수가 그에 근접한 범위 안에 들어가도록 분할을 수행하게 된다. 적응형 옥트리의 경우에는 각 노드 내부의 정점의 개수를 균등하게 맞출 수 없다(그림 6 참조). 정점-군집 트리는 파티션 메쉬 데이터의 크기가 비슷하게 결정되기 때문에 3D 모델을 부분적으로 처리하는 경우 비슷한 대역폭과 처리 시간을 보장할 수 있다.

③ 정점의 경계 상자를 고려한 동적인 분할 축 결정

기본적인 KD 트리 알고리즘의 경우 고정된 분할 축을 사용하기 때문에 분할된 각 파티션 메쉬의 형태나 크기가 다양하게 될 가능성이 높다 (그림 6 참조). 이러한 문제점을 해결하기 위해 본 논문에서는 고정된 축이 아닌 노드에 포함된 메쉬의 형태를 고려하여 동적으로 분할 축을 결정하였다. 노드에 포함된 모든 정점을 둘러싸는 경계 상자를 구하고 이 경계 상자에서 가장 긴 축을 분할 축으로 선택하였다. 이를 통하여 최종적으로 만들어진 리프 노드의 모양이 정육각형에 가까워짐으로써 파티션 메쉬의 형태가 좀 더 균일해지도록 하였다.

④ 계층성을 지닌 균형 트리

모든 리프 노드로부터 루트 노드까지의 깊이가 동일한 균형 트리이다. 또한 동일 레벨의 모든 노드들은 거의 동일한 개수의 정점을 가진다. 따라서 점진적인 메쉬 처리나 LOD(Level of Detail)을 제공할 수 있다.

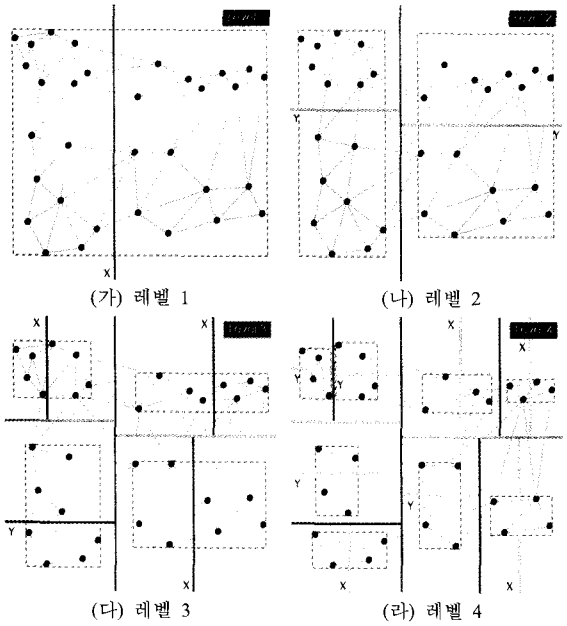


그림 3. 메쉬 분할 과정

그림 3은 적응형 KD 트리를 사용하여 임의의 메쉬를 분할하는 과정을 보여준다. 그림에서 빨간선은 1레벨, 파란선은 2레벨, 녹색선은 3레벨, 노란선은 4레벨의 노드를 분할하는 데 사용된 직선이다. 그림의 예제는 2차원이기 때문에 직선이 되고 3차원의 경우는 평면이 된다. 그림 3 가)에서는 메쉬를 둘러싸는 경계 박스의 축 중에서 x축이 가장 길기 때문에 x축에 수직이 되면서 정점을 양분하는 빨간선을 기준으로 두 개의 파티션으로 분할된다. 나)에서는 두 파티션의 경계 상자가 모두 y축이 길기 때문에 그 축에 수직인 파란선을 기준으로 분할된 것을 볼 수 있다. 다)에서는 4개의 파티션이 각 경계 상자에 따라 서로 다른 축으로 분할된 것을 볼 수 있다.

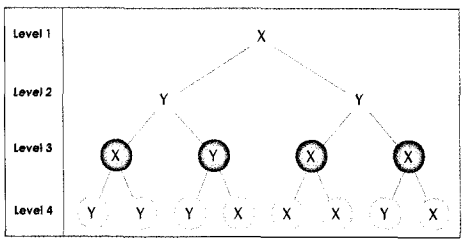


그림 4. 적응형 KD 트리 예

그림 4는 그림 3에 해당하는 적응형 KD 트리 구조를 보여준다. 앞서 그림 3을 통해 설명한 내용을 그림 4의 트리 구조에서도 확인할 수 있다. 레벨 2의 노드는 동일한 축(Y, Y)으로 분할되었지만 레벨 3에서는 서로 다른 축(X, Y, X, X)로 분할된 것을 볼 수 있다. 또한 리프 노드까지 분할될 때 기본적인 KD 트리처럼 라운드 로빈(round-robin) 방식이 아니라 동일한 축이 연속으로 중복되어 사용되는 것을 확인할 수 있다. 마지막으로 모든 리프 노드들이 동일한 레벨이기 때문에 정점-군집 트리는 균형 트리임을 알 수 있다.

그림 5는 적응형 KD트리를 사용하여 Feline 모델을 64개의 파티션 메쉬로 분할했을 때의 결과이다.

그림 5. 정점-군집 트리로 분할된 Feline

적응형 KD 트리에서 메쉬를 분할하는 축은 여러 가지 의미를 함축적으로 지니고 있다.

① 균등 분할

선택된 축에 수직인 평면을 기준으로 양쪽에 비슷한 개수의 정점이 존재한다.

② 메쉬의 형태

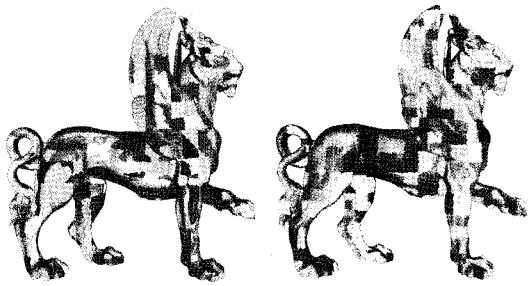
노드를 분할할 때 그 메쉬를 둘러싸는 경계 상자에서 가장 긴 축을 선택하여 그 축에 수직인 평면으로 분할하기 때문에 선택된 축을 통해서 메쉬의 형태를 대략 짐작할 수 있다.

③ 파티션 사이의 연결 정보

선택된 축에 수직인 평면을 기준으로 두 개로 분할되기 때문에 두 파티션 간의 연결 정보를 알 수 있다.

4. 실험 결과 및 분석

본 실험에 사용된 시스템의 사양은 Intel Xeon 2GHz CPU, 3G RAM이다. 다양한 분할 알고리즘 중에서 여기서는 적응형 옥트리, 기본 KD 트리, Lloyd의 알고리즘과 비교한 결과를 제시하고자 한다.



(가) 기본 KD 트리 (나) 적응형 KD 트리

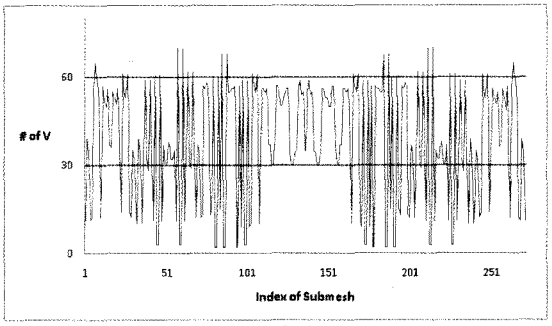
그림 6. 기본 KD 트리와 적응형 KD 트리의 분할 결과 비교

그림 6은 Feline 모델을 각각 기본 KD 트리와 적응형 KD 트리 방식을 사용하여 512개의 파티션으로 분할했을 때의 결과이다. 고정된 축으로 분할하는 기본 KD 트리 방식에 비해 각 노드에 속한 파티션의 형태에 따라 분할 축과 분할 평면을 가변적으로 결정해주기 때문에 더 균등한 크기와 모양의 분할 결과를 보여주고 있다.

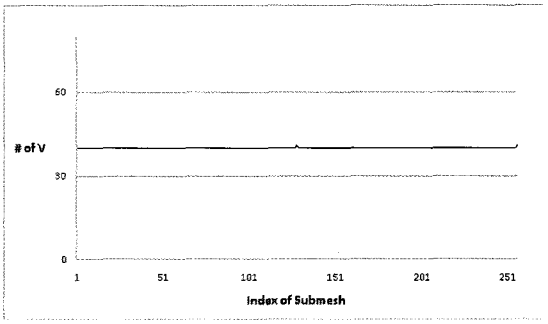
표 1은 메쉬를 분할하기 위해 트리를 생성하는 데 걸린 시간을 비교한 결과이다. 적응형 KD트리는 적응형 옥트리 방식에 비해 더 좋은 분할 결과를 얻는 것에 비해서 트리 생성하는 데 걸린 시간도 크게 뒤지지 않음을 확인할 수 있다.

표 1. 트리 생성 시간 비교

모델 (# of V)	적응형 KD 트리			적응형 옥트리		
	# of V	# of Part.	시간 (sec)	레벨	# of Part.	시간 (sec)
Feline (49,864)	1,558	32	0.64	2	46	0.594
	779	64	0.765			
	390	128	0.828	3	194	0.859
Iphigenie (100,023)	1,562	64	2.156	2	44	1.203
	781	128	2.328			
	391	256	2.64	3	207	1.75
Dragon (399,332)	6,240	64	7.125	2	53	4.859
	1,560	256	9.547	3	271	6.906
	390	1,024	12.813	4	1,253	9.203
Buddha (541,366)	2,115	256	11.968	3	294	9.375
	529	1,024	15.531	4	1,469	12.141
David_Head (1,931,797)	7,546	256	50.969	3	283	33.141
	1,886	1,024	58.297	4	1,469	43.109



(가) 적응형 옥트리



(나) 적응형 KD 트리

그림 7. 노드에 포함된 정점의 개수

그림 7은 그림 1의 Sphere 모델을 분할했을 때 각 파티션에 포함되어 있는 정점의 개수를 차트로 그린 것이다. 옥트리 방식은 메쉬의 형태나 정점의 분포를 고려하지 않고 공간을 균등하게 분할하는 방식이기 때문에 리프 노드에 속하는 정점의 개수가 천차만별인 반면, 적응형 KD 트리는 메쉬의 형태나 정점의 분포를 고려하여 균등하게 나누는 방식이므로 리프 노드에 속하는 정점의 개수가 거의 균일한 것을 확인할 수 있다.

표 2는 Dragon과 Buddha 모델에 대해서 적응형 KD 트리 방식으로 메쉬를 분할했을 때와 [12]에서 Lloyd 방식으로 메쉬를 분할했을 때 걸리는 시간을 비교한 것이다. [14]에서 메쉬를 분할하는 데 사용된 비용함수는 평면성과 컴팩트성, 그리고 면의 개수에 기반한 균등성을 고려하였다. 정점-군집 트리에 비해 훨씬 더 많은 시간이 걸리는 것을 확인할 수 있다. 참고로 [12]의 테스트 환경은 Pentium Core2 Duo 2.4GHz CPU와 2GB RAM이다.

표 2. 메쉬 분할 시간 비교

분할 방식	모델 (# of V)	Dragon (399,332)			Buddha (541,366)		
적응형 KD 트리	# of Part.	64	256	1024	256	512	1024
	트리 생성 시간 (sec)	7.1	9.6	12.8	12.0	13.0	15.5
	메쉬 분할 시간 (sec)	14.4	57.8	230.5	78.3	157.4	312.1
	합계 (sec)	21.5	67.4	243.3	90.3	170.4	327.6
Lloyd [12]	# of Part.	-	273	-	-	584	-
	초기 분할 시간 (sec)	-	309	-	-	410	-
	업데이트 시간 (sec)	-	30	-	-	390	-
	합계 (sec)	-	339	-	-	800	-

그림 8은 적응형 KD 트리를 사용하여 다양한 모델을 분할한 결과이다. 모델 이름 옆에 숫자는 분할된 파티션의 개수이다.

5. 결론 및 향후 과제

본 논문에서 제시한 적응형 정점 군집화는 적응형 KD트리를 활용하여 대용량의 메쉬를 빠른 속도로 균등하게 분할하는 계층적인 구조를 가진 알고리즘이다. 기존의 K-평균군집화(Lloyd) 알고리즘과 옥트리 방식의 장점들을 갖춘 알고리즘이라 할 수 있다. 특히, 분할된 파티션 메쉬가 컴팩트성을 유지하도록 하기 위해 점의 분포를 고려하여 분할 축을 동적으로 결정하였다. 실험 결과를 통해 비교적 빠른 시간에 균등한 정점의 개수를 지닌 메쉬 분할 결과를 얻었음을 확인할 수 있었다. 또한 파티션의 개수나 한 파티션에 속하는 정점의 개수를 사용자가 원하는 수준에 가깝게 조절할 수 있으므로 다양한 메쉬 처리 알고리즘, 응용 프로그램, 시스템의 자원에 보다 적합하고 효율적인 처리를 제공해 줄 수 있다. 이 트리 구조를 이용하여 [7,14]과 같이 메쉬 간소화나 재구성에 활용할 수 있으며 대용량 포인트 기하(Point Geometry) 데이터 처리에도 적용할 수 있을 것으로 기대된다.

K-평균군집화(Lloyd) 알고리즘의 경우 면 단위 연산을 하므로 한 개의 파티션 영역에 한 개의 패치 (Surface Patch)가 생성되나, 본 방법은 컴팩트성을 유지하기 위한 정점 단위 연산으로, 한 개의 영역에 한 개 이상의 분리된 패치(Surface Patch)가 생성될 수 있다. 향후에 본 방법과 기존 방법 간의 질적 비교를 위한 측정 방법을 연구할 계획이다. 또한 Lloyd 알고리즘과 같이 분할하고자 하는 파티션의 개수를 사용자가 원하는 대로 지정할 수 있도록 트리 생성 방식을 확장하는 연구도 진행할 것이다. 아울러 본 논문에서 제시한 트리 구조를 활용하여 대용량의 메쉬를 가지고 임의 접근이 가능한 점진적 압축을 수행하는 알고리즘도 개발하고자 한다.

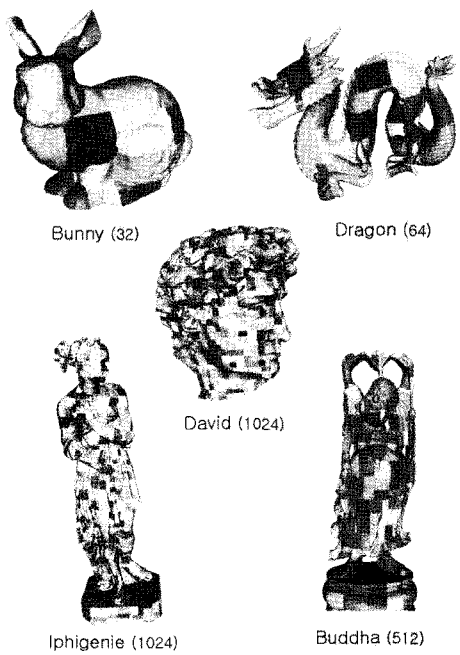


그림 8. 적응형 KD 트리로 분할된 모델들

감사의 글

이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임
(R01-2008-000-20544-0)

참고 문헌

[1] H. Samet, "Spatial Data Structures: Quadtrees, octrees, and other hierarchical methods," *Addison-Wesley*, Redding, Mass., 1989.

[2] P. Cignoni, C. Montani, C. Rocchini, and R. Scopigno, "External memory management and simplification of huge meshes," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 4, pp. 525-537, 2003.

[3] T. Boubekeur, W. Heidrich, X. Granier, and C. Schlick, "Volume-surface trees," *Computer Graphics Forum (In Proc. of Eurographics 2006)*, vol. 25, no.

3, pp. 399-406, 2006.

[4] J. Rossignac and P. Borrell, "Multi-resolution 3D approximation for rendering complex scenes," *Modeling in Computer Graphics*, pp. 455-465, 1993.

[5] Kok-Lim Low and Tiow-Seng Tan, "Model simplification using vertex-clustering," *Siggraph: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, pp. 75-82, 1997.

[6] E. Shaffer and M. Garland, "Efficient adaptive simplification of massive meshes," *In Proc. of IEEE Visualization*, pp. 127-134, 2001.

[7] Scott Schaefer and Joe Warren, "Adaptive vertex clustering using octrees," *SIAM Geometric Design and Computing*, 2003.

[8] M. Garland, A. Willmott, and P. Heckbert, "Hierarchical face clustering on polygonal surfaces," *In Proc. of ACM Symposium on Interactive 3D Graphics*, pp. 49-58, 2001.

[9] S. Lloyd, "Least square quantization in PCM," *Information Theory, IEEE Transactions*, vol. 28, no. 2, pp. 129-137, 1982.

[10] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe, "Multi-chart geometry images," *In Proc. of the 2003 Eurographics*, pp. 146-155, 2003.

[11] D. Cohen-Stiner, P. Alliez and M. Desbrun, "Variational shape approximation," *ACM Siggraph 2004*, pp. 905-914, 2004.

[12] Sungyul Choe, Junho Kim, Haeyoung Lee, and Seungyong Lee, "Random accessible mesh compression using mesh chartification," *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 1, pp. 160-173, 2009.

[13] Jeffrey Ho, Kuang Chih Lee, David Kriegman, "Compressing large polygonal models," *In Proc. of the conference on Visualization*, pp. 357-362, 2001.

[14] T. K. Dey, J. Giesen, and J. Hudson, "A Delaunay based shape reconstruction from large data," *In Proc. of IEEE Symposium on Parallel and Large Data Visualization and Graphics*, pp. 19-27, 2001.

[15] Ernesto and Bribiesca, "An easy measure of compactness for 2D and 3D shapes," *Pattern Recognition*, vol. 41, no. 2, pp. 543-554, 2008.

[16] J. D. MacDonald and K. S. Booth, "Heuristics for ray tracing using space subdivision," *Visual Computer*, vol. 6, no. 3, pp. 153-166, 1990.

〈저자소개〉

김대영
- 1998년 2월 홍익대학교 컴퓨터공학과 (학사)
- 2001년 2월 홍익대학교 전자계산학과 (석사)
- 2001년 3월 ~ 현재 홍익대학교 컴퓨터공학과 박사 과정
- 〈관심분야〉 3차원 메쉬 처리 알고리즘

김종원
- 2008년 2월 홍익대학교 컴퓨터공학과 (학사)
- 2008년 3월 ~ 현재 홍익대학교 컴퓨터공학과 석사 과정

이혜영
- 1986년 2월 연세대학교 수학과 (학사)
- 1992년 5월 San Jose State University 전산학과 (석사)
- 2003년 8월 University of Southern California 전산학과 (박사)
- 1993년 3월 ~ 2003년 8월 KT 연구개발 본부 연구원
- 2003년 9월 ~ 현재 홍익대학교 컴퓨터공학과 조교수
- 〈관심분야〉 3차원 메쉬 처리 알고리즘, 3차원 게임 엔진, 이러닝 콘텐츠 및 솔루션 등