http://dx.doi.org/10.7848/ksgpc.2012.30.4.335

# 다단계 정육면체 격자 기반의 가상점 생성을 통한 대용량 3D point cloud 가시화

# Massive 3D Point Cloud Visualization by Generating Artificial Center Points from Multi-Resolution Cube Grid Structure

양승찬[1] · 한수희[2] · 허 준[3]
Yang, Seung-Chan · Han, Soo Hee · Heo, Joon

**Abstract**

3D point cloud is widely used in Architecture, Civil Engineering, Medical, Computer Graphics, and many other fields. Due to the improvement of 3D laser scanner, a massive 3D point cloud whose gigantic file size is bigger than computer's memory requires efficient preprocessing and visualization. We suggest a data structure to solve the problem; a 3D point cloud is gradually subdivided by arbitrary-sized cube grids structure and corresponding point cloud subsets generated by the center of each grid cell are achieved while preprocessing. A massive 3D point cloud file is tested through two algorithms: QSplat and ours. Our algorithm, grid-based, showed slower speed in preprocessing but performed faster rendering speed comparing to QSplat. Also our algorithm is further designed to editing or segmentation using the original coordinates of 3D point cloud.

Keywords : 3D point cloud, Grid, Center sampling, View-dependent rendering, LOD(Level-of-detail)

**초 록**

건축, 토목, 의료, 컴퓨터 그래픽스 분야 등 다양한 분야에서 이용되는 3D point cloud는 최근 레이저 스캐너의 발달로 인해 그 용량이 점점 커지게 되었다. 컴퓨터 메모리의 용량을 넘어서서 모든 데이터를 한 번에 처리할 수 없는 대용량 3D point cloud를 가시화하고 편집하기 위해 여러 전처리 및 가시화 방법들이 소개되었고 본 논문에서 비교한 QSplat의 경우 3D 모델의 형상 확인과 용량 감소를 목적으로 원본 좌표를 손실 압축하여 저장하였다. 본 논문에서 제시하는 방법은 3D point cloud를 정육면체 격자로 분할하고 center sampling을 통해 가상점 집합을 생성하며 가시화 과정에서 격자에 저장된 point 집합 취득을 통한 빠른 렌더링이 가능하다. 홍익대학교 인근 지역을 측정한 약 1억 2천만 개 point의 대용량 3D point cloud를 QSplat과 다단계 정육면체 격자 기반 방법으로 비교한 결과 전처리 과정에서는 QSplat이, 가시화 과정에서는 다단계 정육면체 격자 기반 방법이 빠른 속도를 보여주었다. 또한 다단계 정육면체 격자 기반 방법은 point의 원본 좌표를 저장하기에 추후 가시화 외에 편집, segmentation 등의 작업을 고려하여 고안되었다.

핵심어 : 3D 포인트 클라우드, 격자, 중점 샘플링, 시야각 가시화, 단계별 가시화

## 1. 서 론

컴퓨터 메모리의 용량을 넘어 모든 데이터를 한 번에 로드하지 못하는 3D point cloud를 대용량 3D point cloud라 칭한다. 이를 렌더링하려면 메모리 가용량만큼 데이터를 메모리에 로드하고 해제하는 과정을 반복해야 하는데 이 과정에서 많은 시간이 소요된다. 따라서 대용량 3D point cloud를 신속하고 정확하게 가시화하기 위한 전처리 및 자

1) 정회원 · 연세대학교 토목환경공학과 박사과정(E-mail:yscgrs@gmail.com)
2) 정회원 · 경일대학교 위성정보공학과 조교수(E-mail:scivile@kiu.ac.kr)
3) 교신저자 · 정회원 · 연세대학교 토목환경공학과 부교수(E-mail:jheo@yonsei.ac.kr)

료구조의 연구가 이루어지고 있다.

전처리 과정을 거쳐 3D point cloud를 저장하는 형태는 크게 3가지로 나눌 수 있다. point 들로부터 삼각형(triangle)과 같은 도형(polygon)을 구성하여 저장하는 방법(Delaunay, 1934; Gustav, 1850), point 자체만 저장하는 방법(Levoy and Whitted, 1985), 렌더링 된 2D 이미지에서 차지하는 픽셀(pixel)의 형태(footprint)에 따라 3D 복셀(voxel)의 크기를 결정하는 splat 방법(Westover, 1989)이다.

삼각형은 OpenGL에서 가장 빠른 속도로 렌더링이 가능하다(The Khronos Group, 2012). 그러나 대용량 3D point cloud는 삼각형 생성을 위한 추가 연산과 메모리로 인해 point나 splat 방식이 자주 사용된다. 또한 복잡한 형태의 3D 모델은 point를 저장하는데 메모리를 더 사용하지만 삼각형보다 빠른 속도와 나은 렌더링 결과를 보이고(Botsch and Kobbelt, 2003), 삼각형으로 이루어진 메쉬(mesh)는 point 간의 상관관계가 있는 반면에 3D point cloud는 point의 배열로만 이루어져 point 간의 상관관계가 없고 point의 삽입 및 삭제가 간단하다(Dachsbacher et al., 2003). 대신 point만을 다루거나 splat 방법을 이용할 때는 3D point cloud의 일부가 비어보이는 구멍(hole) 현상을 방지해야 한다.

Point는 도형을 구성하기 위한 요소가 아닌 3D 모델을 표현하기 위한 렌더링의 대상으로 여겨지기 시작했다(Levoy and Whitted, 1985). 렌더링의 결과물이 모니터 스크린의 한 픽셀보다 작을 경우, 혹은 point들이 면의 형태와 비슷하게 연속되어 배열되어 있을 경우에는 보다 복합한 정보인 도형 보다는 point가 적은 연산량으로 비슷하거나 동일한 시각적 결과를 내기 때문이다. 대용량 3D point cloud는 통상 분할된 공간에 존재하거나 샘플링 혹은 모델링을 통해 최대한 원본의 형태를 표현하는 간소화된 point cloud로 저장된다.

Splat은 3차원 공간에서 중심점과 영역의 범위를 결정하는 반지름, 그리고 법선 정보를 가지고 있어 하나의 원으로 정의된다(Westover, 1989). 3차원 공간에서 구의 형태로 존재하지만 법선 정보와 2차원 화면에 투영되는 크기를 이용해 splat의 렌더링 크기를 결정하는 이 방법은 볼륨 렌더링의 개념으로서 구멍 현상을 해결하기 좋은 장점이 있지만 시각적 결과가 좋지 않아 QSplat(Rusinkiewicz and Levoy, 2000) 전까지 이론적 개념으로 남아있었다. QSplat은 hierarchical delta coding으로 원본의 3D point cloud를 압축 저장하고 splat을 이용하여 LOD, 가시화 판단(visibility culling), 고정된 렌더링 속도(frame rate)를 구현함으로써 splat 연구를 재조명하고 기준점 역할을 하였다.

이 외에도 그래픽 카드 지향적 자료구조들이 소개되어 가시화 속도를 향상시켰다. Sequential Point Trees는 splat의 반지름으로 point를 정렬함으로써 LOD를 적용할 때 순차적으로 연속된 point들을 렌더링하지만 point들이 그래픽 카드 메모리에 저장되어 있어야 한다(Dachsbacher et al., 2003). 이를 out-of-core 기법으로 개선한 방법이 Layered Point Clouds(Gobbetti and Marton, 2004), XSplat(Pajarola et al., 2005), Instant Points(Wimmer and Scheiblauer, 2006), XGRT(Wand et al., 2007) 등이다. Layered Point Clouds는 대용량 3d point cloud의 검색 속도 개선을 위해 정규 샘플링의 결과를 이진 트리(binary tree)에 저장하였고 XSplat은 그래픽 카드의 캐시(cache) 낭비 방지, Instant Points는 원본 point 렌더링에 중점을 두었다. 또한 Multi-way kd-Trees(Goswami et al., 2010)는 KD 트리의 노드가 다수의 자식을 가지도록 하여 최종 노드에 저장되는 point들의 개수를 비슷한 숫자로 조절하였고 그 결과 그래픽 카드가 사용하는 캐시를 일정 크기로 고정시켰다.

본 연구는 이러한 배경을 바탕으로 전처리 과정에서 point들이 OpenGL의 렌더링에 유리하도록 배치하여 빠른 가시화 속도를 확보하였고 또한 대용량 3D point cloud의 편집을 향후 연구로 고려하여 자료구조를 개발하였다. 본 논문의 구성은 다음과 같다. 2장에서 논문의 이해에 필요한 관련 개념, 3장에서 정육면체 기반 가상점 방법 연구에 대해 소개한다. 4장에서 실험 환경 및 방법과 그 결과를, 5장에서 결론과 더불어 향후 연구 방향을 제시한다.

## 2. 관련 개념

### 2.1 축 정렬 경계 박스

3차원 공간에서 경계 박스(bounding box, 이후 BB)는 3D 모델을 표현하는 최소한의 육면체로서 연산 과정에서 평면 혹은 직선 방정식을 이용해야 하는 반면에 축 정렬 경

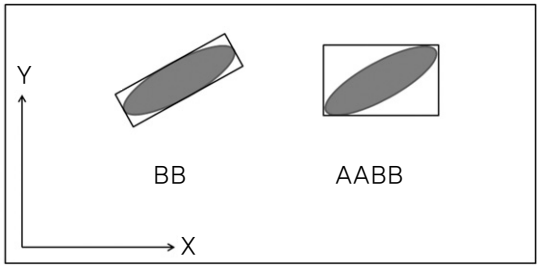


**그림 1. BB와 AABB 개념도**

계 박스(axis-aligned bounding box, 이후 AABB)는 3D 모델 외의 공간을 더욱 차지하지만 축과 평행한 좌표 값을 가지고 있어 좌표 값의 단순 비교를 통한 빠른 연산이 가능하다. 그림 1은 2차원 좌표계에서 BB와 AABB의 개념을 비교한 것이다.

## 2.2 3D 공간 분할 기법

3D 공간을 분할하는 방법은 크게 정규(uniform) 분할과 비정규(uon-uniform) 분할로 나눌 수 있다. 잘 알려진 정규 분할 방법에는 팔진트리(octree), 격자(grid) 등이 있고 비정규 분할 방법에는 BSP(Binary Space Partitioning), KD 트리, 경계 구(bounding sphere) 등이 있다. 3D point cloud를 공간 분할할 때 자주 사용되는 방법은 격자, 팔진트리, KD 트리이다(Gobbetti et al., 2008).

### 2.2.1 팔진트리

팔진트리는 3D 모델을 가로, 세로, 높이별로 2등분 하여 연속적으로 공간을 8등분한다. 8개의 하위 노드 중 3D 모델이 존재하는 노드를 다시 8등분하여 계층(hierarchical) 구조를 가진다. 색인 및 검색 작업이 빠르지만 3D 모델 정보가 비대칭적으로 존재할수록 트리의 레벨이 깊어지고 트리를 구성하기 위한 용량이 증가한다.

### 2.2.2 다단계 정육면체 격자

3차원 격자(grid) 구조는 가로, 세로, 높이 크기 변화가 자유롭고 계층적 구조를 가지지 않는다. 정육면체 격자는 가로, 세로, 높이를 같게 해 모든 격자가 정육면체이다. 다단계 정육면체 격자는 격자 크기를 바꿔가며 여러 개의 정육면체 격자를 생성하는 것이다. 그림 2는 하나의 AABB에서 점점 작은 정육면체 격자들로 형성되는 다단계 정육면체 격자를 나타낸다.

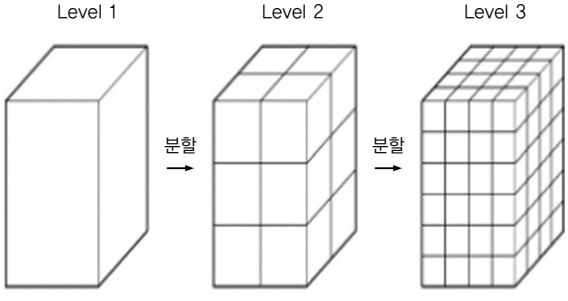


**그림 2. 다단계 정육면체 격자의 점진적 분할**

### 2.2.3 KD 트리

KD 트리는 하나의 공간을 연속해서 2개로 분할하는 BSP(Binary Space Partitioning) 트리의 일종이다. BSP 트리는 좌표계 축에 평행의 제약 없이 공간을 분할하지만 KD 트리는 축과 평행하게 분할함으로써 BSP 트리의 단점인 트리 생성 및 처리 연산량을 줄인다.

### 2.2.4 Splat

Splat(Westover, 1989)은 3차원 복셀이 2차원 영상에 투영되는 형상으로 splat이 2차원 영상에서 차지하는 픽셀의 형태나 크기를 분석하여 3차원 복셀의 렌더링 방법을 결정한다. 그림 3은 3차원 공간에서 eye가 Q 방향으로 구를 바라볼 때 2차원 화면에 투영되는 구의 형상을 판단하는 것으로 이 경우 3D 공간에서는 구이지만 2차원으로 투영될 때는 타원을 렌더링한다. 타원의 크기는 2차원 영상에 채워질 픽셀의 개수에 따라 달라진다.

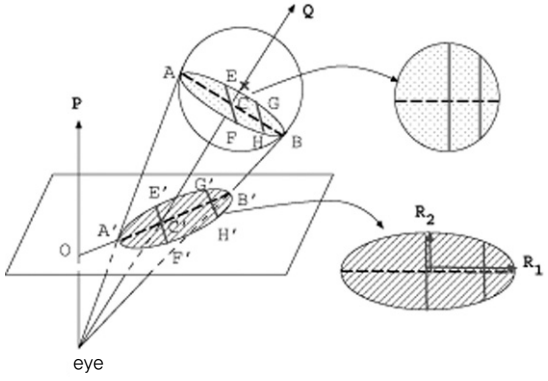


**그림 3. Splat 개념도** (Yuan et al., 2003)

### 2.2.5 QSplat

QSplat은 대용량 3D point cloud의 실시간 렌더링을 위해 제안되었다(Rusinkiewicz and Levoy, 2000). QSplat의 목적은 시점에 따라 3D 모델의 형상을 유지할 수 있을 만큼 판단하여 정해진 시간 내에서 점진적 렌더링을 하는 것이다. QSplat은 전처리 과정을 통해 3D 모델을 2 ~ 4개의 하위 경계 구로 연결된 변형된 KD 트리로 구성하고, 렌더링 과정에서는 경계 구들을 검색하며 현재 시야에 존재하는 노드에 대해 하위 노드를 계속 검색할지 또는 해당 노드를 렌더링할지의 여부를 2차원 영상에 투영된 경계 구의 면적과 미리 설정된 렌더링 속도에 기반하여 결정한다. 경계 구의 3차원 좌표는 부모 노드로부터 떨어진 정도를 13단계로 나누고 근사치의 단계로 압축하기 때문에 원본

보다 작은 용량이 되지만 원본 데이터는 손실된다. 그림 4는 QSplat에서 경계 구들이 연결된 형상으로, 3차원 공간에서 경계 구들에 속하지 않는 공간이 생기지 않도록 경계 구들의 크기를 조절하여 구멍 현상을 방지한다. 그림 5는 QSplat의 점진적 렌더링을 나타낸다.

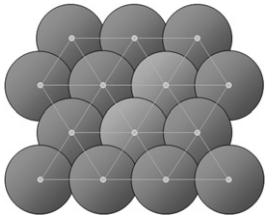

**그림 4. 경계 구들이 연결된 개념도** (Rusinkiewicz, Levoy, 2000)

**그림 5. QSplat의 점진적 렌더링 (좌->우)**

### 2.3 샘플링 및 가상점

공간 분할된 3D point cloud의 샘플링 기법에는 random sampling, average sampling, center sampling, closest-center sampling 등이 있다(Wand, 2004). 공간에 포함된 point들을 기준으로 random sampling은 임의의 한 point를, average sampling은 모든 point들의 평균 좌표 값을, center sampling은 3차원 공간의 중간 값을, closest-center sampling은 center에 가장 가까운 point를 대푯값으로 취한다. 본 논문에서는 격자 내에서 center sampling을 통해 획득한 point를 가상점으로 정의한다.

## 3. 다단계 정육면체 격자 기반 가상점 방법

다단계 정육면체 격자 기반 가상점 방법(이후 가상점 방법)은 전처리 과정에서 점진적으로 격자 크기를 바꿔가며 3D point cloud의 AABB를 분할하고 단계별로 가상점을 생성하여 고정 격자에 가상점과 원본 point를 배치한다. 가시화 과정은 전처리 결과물에 저장된 정육면체 격자 구조에서 격자의 중점을 추려낸 후 카메라 절두체와 비교하며 가시화 판단과 렌더링할 가상점 집합의 레벨을 선택한다.

### 3.1 전처리 과정

그림 6은 전처리 과정의 순서도이다.

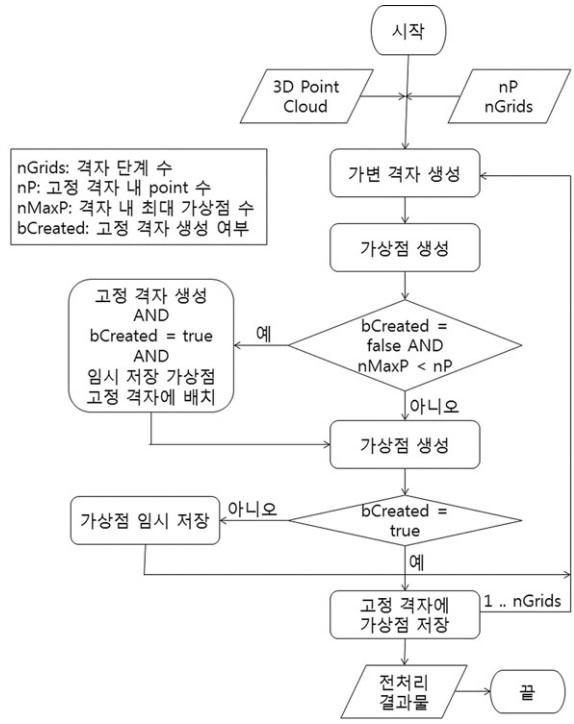


**그림 6. 가변 격자에 의해 생성되는 가상점들**

3D point cloud는 격자 단계 수와 고정 격자 내 point 수의 값에 따라 AABB를 분할해가며 가변 격자에서 가상점을 생성한다. 격자의 크기가 다른 가변 격자에서 생성된 격자 단계 수 만큼의 가상점 집합과 원본 3D point cloud는 고정 격자에 재배치된다.

그림 7은 격자의 크기가 변하는 가변 격자에서 가상점을 생성하는 과정이다. 좌측 점선 사각형은 가변 격자, 우측 실선 사각형은 고정 격자, 속이 채워진 point는 원본, 속이 빈 point는 생성된 가상점이다.

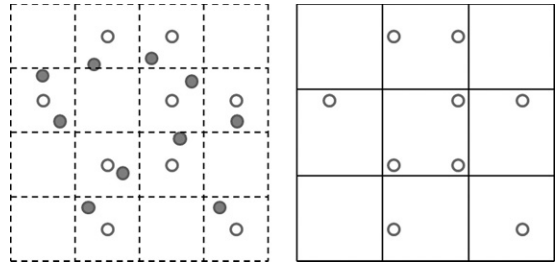

**그림 7. 가변 격자에 의해 생성되는 가상점들**

그림 8은 가상점 집합과 원본 point가 고정 격자에 중첩되어 저장되는 모습이다. 가상점 개수가 많아질수록 점점 원본 point의 형상과 비슷해진다.

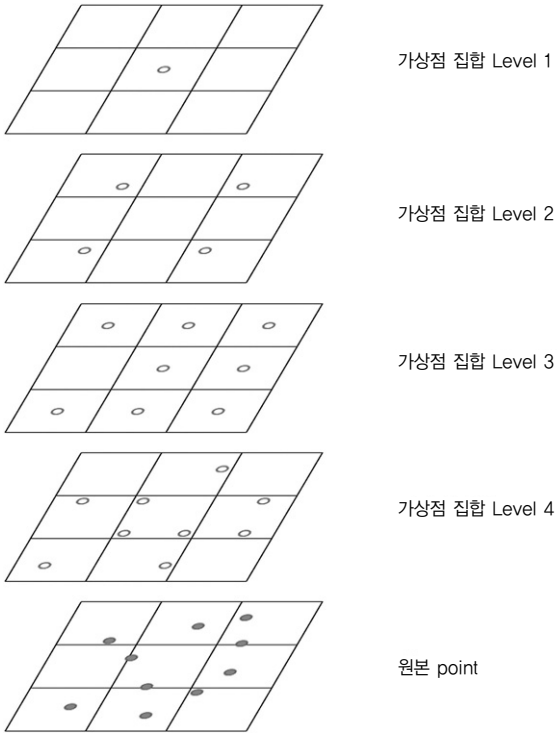


**그림 8. 가상점들과 원본 point의 고정 격자 배치**

고정 격자의 크기와 수는 가시화 과정에서 가시화 판단을 거쳐 렌더링 속도와 품질에 영향을 미칠 수 있는데, 화면 가장자리에 일부가 걸친 격자의 경우 해당 격자 내의 point들을 렌더링하면 렌더링 시간이 더욱 걸리고 렌더링하지 않으면 화면 가장자리의 일부가 비어보일 수 있다. 따라서 고정 격자에서 격자의 크기는 작고 수가 많을수록 렌더링 단계에서 유리하다. 반면에 격자의 수가 많을수록 전처리 과정의 시간이 증가하기 때문에 격자의 수를 결정할 때 trade-off가 발생한다.

### 3.2 가시화 과정

가시화 방법은 3D point cloud를 바라보는 시점에 따라 렌더링 여부와 정도가 달라진다. 전처리 결과물에 저장된 정육면체 격자 구조에서 각 격자의 꼭짓점을 추려내고 카메라 절두체와 좌표를 비교하여 가시화 판단을 수행한다. 이후 카메라의 가시거리 대비 카메라와 격자간의 거리에 따라 해당 수준의 가상점 집합을 선택하여 렌더링한다. 즉 view-dependent LOD를 구현하기 위해 카메라의 가시거리, 카메라와 각 격자간의 거리, 가상점 집합의 수를 이용하였다. 그림 9는 가시화 과정의 순서도이다.

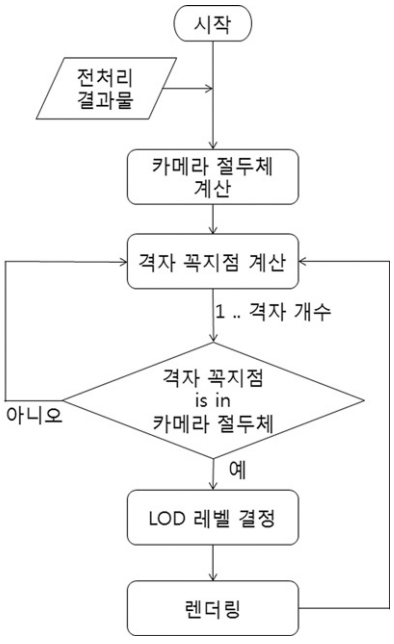


**그림 9. 가시화 과정 순서도**

## 4. 실험 및 결과 분석

### 4.1 실험 데이터 및 환경

실험 데이터는 홍익대학교 인근 지역을 지상 LiDAR 장비로 측량한 44개 건물을 취합한 것으로 약 1억 2000만 개 point가 x, y, z의 3차원 좌표 값으로 이루어져 있다. 목표 point 간격은 10m 거리에서 5mm 단위이며 하나의 건물은 190만 ~ 400만 point로 구성된다(연세대학교, 2009). 하나의 point를 12바이트로 사용하는 바이너리 파일로 저장하여 약 1.3GB의 용량을 차지한다. 표 1은 실험을 수행한 컴퓨

**표 1. 실험에 사용된 컴퓨터 사양**

| 항 목 | 사 양 |
|---|---|
| CPU | Intel(R) Core(TM) i5-2300 |
| Ram | 16 GB |
| 운영체제 | 64 bit Windows 7 |
| 그래픽카드 | ATI Radeon HD 5700 Series / 1 GB Memory |

터의 사양이다. 16 GB 메모리를 사용한 이유는 QSplat의 전처리 과정에 15 GB 용량이 필요해서이다.

## 4.2 실험 과정

실험은 가상점 방법과 QSplat을 전처리 및 렌더링 과정으로 나누어 수행하였다.

팔진트리 방법은 트리 구조의 용량이 크고 가시화 과정에서는 트리 구조의 특성상 point를 하나씩 취득하여 속도의 저하를 불러오기 때문에 제외하였다.

가상점 방법은 전처리 과정에서 경험적 수치로 8개 가상점 집합을 생성하였다. 격자 내 최대 point 수가 2,097,152 개인 이유는 하드디스크에서 빠른 시간 내에 읽어올 수 있는 양이며 동시에 추후 연구에서 고정 격자에서 각 격자를 팔진트리로 생성하여 사용할 것을 고려한 것이다. 결과 파일은 C++과 OpenGL로 구현된 소프트웨어에서 렌더링하였다.

QSplat의 전처리 과정은 법선 정보를 사용하기 때문에 삼각형으로 구성된 메쉬 혹은 법선 정보를 포함한 3D point cloud를 입력받는다. 그러나 실험 데이터는 법선 정보를 가지고 있지 않아 trimesh2 라이브러리(Rusinkiewicz, 2000)를 이용하여 법선 정보를 생성한 후에 QSplat 전처리 과정을 수행하였다. 렌더링 과정에서는 splat 크기를 고정시켜 point 렌더링을 구현하였다.

## 4.3 실험 결과 및 분석

표 2는 실험 결과로서 전처리 과정에서 QSplat이 가상점 방법보다 빠르고 point의 정보를 압축해 원본보다 용량이 줄은 반면 가상점 방법은 생성된 가상점만큼 용량이 늘었다.

그림 10과 11은 가상점 방법과 QSplat의 렌더링 결과를, 그림 12와 13은 그림 10과 11의 빨간 부분을 확대한 것이다.

렌더링의 품질은 주관적 선호가 작용되는 부분이지만 가상점 방법은 샘플링 특성에 따른 격자의 패턴이 보이고, QSplat은 일정 렌더링 속도를 보장하기 위해 렌더링 결과가 고르지 않을 수 있다.

**표 2. 실험 결과**

| | QSplat | 가상점 |
|---|---|---|
| 전처리 시간 (분) | 8 | 37 |
| 결과물 용량 | 745 MB | 1.33 GB |
| 렌더링 시간 (초) | 0.68 | 0.06 |
| 렌더링 point (개) | 6890859 | 1326519 |

**그림 10. 가상점 방법 렌더링 결과 (가상점 레벨 8/9)**

**그림 11. QSplat 렌더링 결과**

**그림 12. 가상점 방법 렌더링 결과 확대**

**그림 13. QSplat 렌더링 결과 확대**

렌더링 과정에서는 두 방법이 같은 장면을 렌더링 하지만 렌더링 point 개수, 방법, 시간이 다르다. 두 방법의 OpenGL 렌더링 속도 차이는 point 개수보다도 QSplat은 point를 하나씩 렌더링하고 가상점 방법은 VBO(Vertex Buffer Ojbect)를 이용하는 근본적인 렌더링 방법에서 기인한다. VBO는 point 배열을 그래픽 카드의 메모리에 저장한 후 한 번에 렌더링을 하지만 QSplat의 경우 이번 실험에서 1,326,519 번의 렌더링을 한다. 따라서 트리 구조를 검색하며 point를 1개씩 취득하는 QSplat이 렌더링 속도에서 불리하다.

### 4.4 알고리즘 비교

팔진트리는 모든 point를 색인화하여 저장하기 때문에 빠른 검색 속도를 보장한다. 하지만 팔진트리를 구성하기 위한 추가 메모리가 필요하고 대용량 3D point cloud의 경우 그 양이 많기 때문에 실제 사용에 적절치 못하다. 또한 렌더링시 point를 하나씩 획득하기 때문에 다수의 렌더링 명령을 내리거나 point들을 모아서 렌더링하게 되어 렌더링 속도를 지연시킨다.

반면에 격자는 point의 집합을 하나의 셀(cell)에 저장함으로써 렌더링 명령에는 유리하지만 셀에 포함된 일부의 point를 렌더링하기 위해 셀 전체를 렌더링하여 과부하를 일으킬 때가 있다. 또한 셀 내에서 point들이 상관관계 없이 불규칙하게 저장되어 있어 검색 등의 작업에서 모든 point를 처리해야하는 단점이 있다.

다단계 정육면체 격자를 기반으로 한 가상점 방법은 전처리 과정에서 point들 외에 추가적으로 사용하는 메모리 용량이 적어 대용량 3D point cloud 처리에 용이하다. 다만 생성하는 가상점 집합의 수에 따라 전처리 과정의 소요 시간이 변동되고 적절한 수의 가상점 집합이 생성되지 않으면 렌더링 품질에 악영향을 미친다. 또한 고정 격자의 크기와 수에 따라 가시화 판단 이후 화면 가장자리의 일부가 비어보일 수 있다.

KD 트리는 3차원 공간을 균일하게 분할하는 팔진트리와 격자와는 달리 point의 배치에 따라 불규칙하게 공간을 분할하기 때문에 기본적으로 3D point cloud의 전체 정보를 알고 있어야 생성이 가능하다. 그러나 대용량 3D point cloud는 모든 정보를 동시에 메모리에 올리지 못하기 때문에 한 차례의 전처리 과정을 통해 전체 정보를 구성한 후에 KD 트리 구성이 가능하다. 또한 KD 트리 구성 작업에도 파일을 여러 차례 나누어 읽으며 처리해야 하기 때문에 분산된 자료들 간을 연결해주는 추가 작업이 필요하다.

변형된 KD 트리를 기반으로 한 QSplat은 3D point cloud를 손실 압축해 원본보다 적은 용량으로 빠르고 점진적인 형상의 확인이 가능하지만 100% 정확한 형상이 보장되지 않고 가시화 외의 작업에 사용하기 어렵다. 또한 전처리 과정에서 자료구조를 구성하기 위해 대용량의 메모리를 사용한다. 한 예로 본 연구의 실험에 사용한 약 1억 2천만 여 개 3D point cloud를 처리하기 위해 약 15 GB의 메모리가 사용되었다.

표 3은 가상점 방법과 QSplat의 성능을 비교한 것이다.

**표 3. 가상점 방법과 QSplat 비교**

| 항 목 | QSplat | 가상점 |
|---|---|---|
| 전처리 속도 | 빠름 | 느림 |
| 메모리 사용 | 많음 | 적음 |
| 결과물 용량 | 적음 | 많음 |
| 데이터 손실 | 항상 | 일부 |
| 렌더링 속도 | 느림 | 빠름 |
| 렌더링 품질 | 주관적 | 주관적 |
| 검색 속도 | 빠름 | 느림 |

## 5. 결론 및 고찰

본 연구에서는 다단계 정육면체 격자와 point의 좌표를 이용하여 가상점을 생성하고 최종 단계에는 원본 point를 저장하는 대용량 3D point cloud 처리 방법을 제시하였다. 계층적 구조를 가지지 못하는 격자의 단점을 보완해 다단계 정육면체 격자를 고안하고 샘플링의 결과물인 가상점만 연속적으로 파일에 저장하여 OpenGL 렌더링에 유리함을 보였다. 경계 구와 변형된 KD 트리에 기반한 QSplat과 비교한 결과 느린 전처리 속도와 빠른 가시화 속도를 보였고 전처리 과정에서 많은 수의 가상점 집합이 생성되지 않으면 가시화 과정에서 가상점 집합이 교체되기 전 조약한 렌더링 품질을 보였다. 반면에 QSplat은 손실 압축 방법으로 인해 원본 3D point cloud를 저장하지 않지만 점진적 LOD 렌더링이 이루어졌다.

향후 연구 방향으로 다단계 정육면체 격자 기반 방법의 전처리 속도와 렌더링 품질의 개선 및 검색과 편집을 위한 자료구조를 제시한다. 현재 전처리 과정에서는 격자 구조가 변경될 때마다 원본 3D point cloud 파일을 새로 읽어 들이는 방법을 사용하고 있으나 이는 미리 여러 개의 격자 구조를 생성해두고 3D point cloud를 한 번만 읽어 처리함으로써 파일 입출력 횟수를 줄이고 속도 개선이 가능

하다. 또한 렌더링 품질의 개선을 위해 가상점 집합의 수가 충분하지 않을 경우 가시화 과정에서 현재 렌더링 할 격자의 가상점 집합과 다음 단계의 가상점 집합을 실시간으로 합성하여 중간 단계의 가상점 집합을 생성하는 방법을 고려할 수 있다. 편집 및 검색을 위해서는 point의 색인화가 필요하다. 따라서 다단계 정육면체 격자 구조에서 작은 크기의 격자로 격자 수를 늘리고 격자 내 point를 적게 하여 모든 격자를 낮은 레벨의 팔진트리로 관리하는 다단계 정육면체 격자-팔진트리 혼합 구조를 연구할 계획이다.

## 감사의 글

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2011-0005758).

## 참고문헌

연세대학교 (2009), LiDAR 자료를 이용한 빌딩모형 자동화 추출 알고리즘 개발, 최종 보고서, SK C&C

Botsch M. and Kobbelt L. (2003), High-Quality Point-Based Rendering on Modern GPUs, *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Canmore, p. 335.

Dachsbacher C., Vogelgsang C. and Stamminger M. (2003), Sequential Point Trees, *ACM Transactions on Graphics*, ACM, Vol. 22, No. 3, pp. 657-662.

Gobbetti E., Kasik D. and Yoon S. E. (2008), Technical Strategies for Massive Model Visualization, *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, ACM, Stony Brook, pp. 405-415.

Gobbetti E. and Marton F. (2004), Layered Point Clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models, *Computers & Graphics*, Pergamon Press, Vol. 28, No. 6, pp. 815-826.

Goswami P., Zhang Y., Pajarola R. and Gobbetti E. (2010), High Quality Interactive Rendering of Massive Point Models using Multi-way kd-Trees, *Proceedings of the 18th Pacific Conference on Computer Graphics and Applications 2010*, IEEE Computer Society, pp. 93-100.

Khronos Group (2012), OpenGL FAQ / 22 Performance, The Khronos Group, Oregon, USA, http://www.opengl.org/archives/resources/faq/technical/performance.htm

Levoy M. and Whitted T. (1985), The Use of Points as a Display Primitive, *Technical Report 85-022, Computer Science Department*, University of North Carolina at Chapel Hill.

Pajarola R., Sainz M. and Lario R. (2005), XSplat: External Memory Multiresolution Point Visualization, *International Conference on Visualization, Imaging and Image Processing*, IEEE Computer Society, Benidorm, pp. 628-633.

Rusinkiewicz S. (2000), trimesh2, *Princeton University*, New Jersey, http://gfx.cs.princeton.edu/proj/trimesh2

Rusinkiewicz S. and Levoy M. (2000), QSplat: A Multiresolution Point Rendering System for Large Meshes, *Proceedings of ACM SIGGRAPH 2000*, ACM, New Orleans, pp. 343-352.

Wand M. (2004), *Point-Based Multi-Resolution Rendering*, PhD dissertation, Universität Tübingen, Germany.

Wand M., Berner A., Bokeloh M., Fleck A., Hoffmann M., Jenke P., Maier B., Staneker D. and Schilling A. (2007), Interactive Editing of Large Point Clouds, *Proceedings of Symposium on Point-Based Graphics 2007*, Eurographics Association, Prague, pp. 37-46.

Westover L. (1989), Interactive Volume Rendering, *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, ACM, New York, pp. 9-18.

Wimmer M. and Scheiblauer C. (2006), Instant Points, *Proceedings Symposium on Point-Based Graphics 2006*, Eurographics Association, Boston, pp. 129-136.

Yuan X., Nguyen M. X., Xu H. and Chen B. (2003), Hybrid Forward Resampling and Volume Rendering, *IEEE TVCG Workshop on Volume graphics 2003*, ACM, New York, pp. 119-127.

Delaunay (1934), Sur la sphere vide, *Izvestia Akademii Nauk SSSR*, Otdelenie Matematicheskikh i Estestvennykh Nauk, Vol. 7, pp. 793-800.

Gustav K. (1850), Uber die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal fur die Reine und Angewandte Mathematik*, Vol. 40, pp. 209-227.

(접수일 2012. 04. 06, 심사일 2012. 06. 28, 심사완료일 2012. 08. 31)