한국측량학회지, 제 29권 제 1호, 2011. 2 : pp. 39~46

대용량 3차원 지상 레이저 스캐닝 포인트 클라우드의 탐색을 위한 3D R-tree와 옥트리의 비교

A Comparison of 3D R-tree and Octree to Index Large Point Clouds from a 3D Terrestrial Laser Scanner

한수희[1] · 이성주[2] · 김상필[3] · 김창재[4] · 허준[5] · 이희범[6]
Han, Soohee · Lee, Seongjoo · Kim, Sang Pil · Kim, Changjae · Heo, Joon · Lee, Heebum

Abstract

The present study introduces a comparison between 3D R-tree and octree which are noticeable candidates to index large point clouds gathered from a 3D terrestrial laser scanner. A query method, which is to find neighboring points within given distances, was devised for the comparison, and time lapses for the query along with memory usages were checked. From tests conducted on point clouds scanned from a building and a stone pagoda, it was shown that octree has the advantage of fast generation and query while 3D R-tree is more memory-efficient. Both index and leaf capacity were revealed to be ruling factors to get the best performance of 3D R-tree, while the number of level was of octree.

Keywords : 3D R-tree, Octree, Terrestrial Laser Scanner, Point Cloud, Query

초 록

본 연구에서는 3차원 지상 레이저 스캐너로부터 취득된 대용량 포인트 클라우드로부터 효과적인 포인트 탐색을 수행하기 위한 인덱싱 방법으로서 3D R-tree와 옥트리를 비교하였다. 포인트 클라우드의 각 포인트로부터 일정 거리 이내의 포인트를 조회하는 방식으로 탐색을 수행하였으며, 탐색 시간 및 메모리 사용량을 측정하였다. 실제 건물과 석탑을 대상으로 취득된 포인트 클라우드에 적용한 결과, 옥트리는 3D R-tree에 비하여 생성 및 탐색 속도가 우수하며 3D R-tree는 보다 메모리 효율적임을 확인할 수 있었다. 3D R-tree는 인덱스 용량과 리프 용량이, 옥트리는 계층 수가 탐색 성능을 좌우함을 확인하였으며, 주어진 자료에 대한 최적의 수치를 도출할 수 있었다.

핵심어 : 3차원 R-tree, 옥트리, 지상 레이저 스캐너, 포인트 클라우드, 탐색

1. 서 론

3차원 지상 레이저 스캐너로부터 취득되는 포인트 클라우드(point cloud)의 크기가 급격히 증가하고 있다. 이는 스캐너 성능의 향상과 스캐닝 대상 영역의 확대에 의한 것으로 일반적인 포인트 클라우드에 속하는 포인트의 개수는 수백만으로부터 수십억에 이른다. 이와 같은 방대한 크기의 포인트 클라우드로부터 특정 위치의 포인트를 신속하게 탐색(query)하기 위해서는 효율적인 자료구조 또는 인덱싱(indexing) 방식이 필요하다. 2.5차원 형태의 항공 레이저 스캐너로부터 취득된 포인트 클라우드는 pseudo grid(Cho, 2004) 또는 가상격자(Han, 2007), 쿼드트리(quadtree), kd-tree, R-tree 등의 인덱싱 방식이 적용 가능하다. 그 중에서 쿼드트리와 R-tree 등은 이미 여

1) 연세대학교 사회환경시스템공학부 박사후과정(E-mail:scivile@yonsei.ac.kr)
2) 연세대학교 컴퓨터과학과 박사수료(E-mail:seongjoo@csai.yonsei.ac.kr)
3) 연세대학교 사회환경시스템공학부 박사과정(E-mail:spmrmm@yonsei.ac.kr)
4) 연세대학교 사회환경시스템공학부 박사후과정(E-mail:earth2moon@gmail.com)
5) 교신저자 · 연세대학교 사회환경시스템공학부 부교수(E-mail:jheo@yonsei.ac.kr)
6) 육군3사관학교 토목건축공학과 부교수(E-mail:topkma41@hanmail.net)

러 상용 공간데이터베이스 시스템에서 사용되고 있다. 3차원 포인트 클라우드를 위한 3차원 인덱싱 기법으로는 R-tree의 3차원 확장인 3D R-tree를 들 수 있다. 3차원 포인트 클라우드에 대하여 쿼드트리와 3D R-tree를 적용한 결과, 쿼드트리가 3D R-tree에 비하여 생성속도는 빠르지만 갱신속도는 3D R-tree가 빠르며 메모리 사용량은 3D R-tree가 적다고 보고되었다(Schön, 2009). 한편, 옥트리(octree)는 쿼드트리의 3차원 확장으로서 3차원 자료 인덱싱 기법으로 보다 유망한 방식으로 소개되었다(Schön, 2009). 아울러 옥트리는 3차원 또는 항공 레이저 스캐너로부터 취득된 포인트 클라우드의 세그멘테이션(segmentation)에 응용된 바 있다(Woo, 2002; Wang, 2004; Cho, 2008).

본 연구에서는 3차원 지상 레이저 스캐너로부터 취득된 포인트 클라우드로부터 효율적인 포인트 탐색을 수행하기 위한 인덱싱 방법으로서 3D R-tree와 옥트리를 비교하였다. 포인트 클라우드의 각 포인트로부터 일정 거리 이내의 포인트를 조회하는 방식으로 탐색을 수행하였으며, 탐색 시간 및 메모리 사용량을 측정하였다.

2. 본 론

2.1 3D R-tree

R-tree는 그 변형 방식들과 함께 GIS분야에서 2차원 좌표를 갖는 지형공간자료의 인덱싱을 위해 널리 활용되고 있다(이득우 외, 2009; 이기영 외, 2004; 안성우 외, 2006). R-tree는, 자료를 감싸는 최소 경계 직사각형(minimum bounding rectangle, MBR)을 수개의 MBR로 분할하여 각 MBR들을 대변하는 노드(node)들을 계층적으로 연결한 트리(tree) 구조이다. 최하위 노드는 리프 노드(leaf node)라 불리며 리프 노드가 아닌 일반 노드들은 각각 두 가지 정보를 가진다. 이 두 가지 정보는 하위 노드 정보(예, 포인터)와 하위 노드들을 둘러싸는 MBR이다. 리프 노드도 각각 두 가지 정보를 가지며, 실제 자료(예, 포인트)에 접근하기 위한 정보(예, 포인터)와 그 자료들을 둘러싸는 MBR이다. 각 MBR의 크기는 포함되는 자료의 분포에 다르며 각각의 노드들은 인덱스 용량(index capacity) 이내의 하위 노드를 포함한다. 인덱스 용량(index capacity)은 리프 용량(leaf capacity)와 더불어 R-tree의 탐색 성능에 영향을 미치는 중요한 요소이다. 인덱스 용량은 하나의 일반 노드가 포함할 수 있는 최대의 하위 노드 수를 의미하며 리프 용량은 각 리프가 관할할 수 있는 최대 자료 수를 의미한다. 인덱스 용량을 작게 설정할 경우 같은 계층에 속하는 하위 노드들을 탐색하기 위한 시간이 감소한다. 그러나 인덱스 용량을 너무 작게 설정할 경우 계층 수가 크게 증가하여 리프 노드에 접근하기 위한 과정이 길어지므로 탐색 시간이 증가하게 된다. 리프 용량을 작게 설정할 경우 탐색 영역을 감소시켜 검색 속도를 증가시킬 수 있으나, 너무 작게 설정할 경우 계층 수 증가로 인한 탐색 시간 증가를 유발할 수 있다.

R-tree는 다차원 공간자료의 인덱싱에 활용될 수 있으며 특별히 3차원 좌표를 갖는 공간자료의 인덱싱에 사용되는 형태가 3D R-tree이다(Zhu, 2007). 본 연구에서는 3차원 좌표를 갖는 포인트를 저장하고 탐색할 수 있으며 C++로 구현된 3D R-tree를 사용하였다. 아울러, 여러 인덱스 용량을 적용하여 주어진 3차원 포인트 클라우드의 탐색에 적합한 인덱스 용량을 도출하였다.

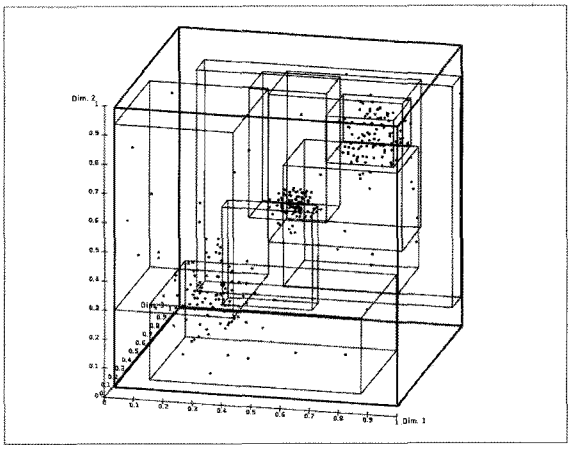

그림 1. 3D R-tree

2.2 옥트리

옥트리는 쿼드트리를 3차원으로 확장한 것으로서, 공간을 8개의 직육면체로 분할하여 각 하위공간을 대변하는 노드(node)들을 계층적으로 연결한 트리(tree) 구조이다. 즉, 각 노드는 동일한 크기와 형상의 직육면체를 의미하며 8개의 하위 노드를 갖는다. 일반 노드는 하위 노드 정보(예, 포인터)와 하위 노드들을 둘러싸는 MBR 정보를 포함한다. 리프 노드는 실제 자료(예, 포인트)에 접근하기 위한 정보(예, 포인터)와 그 자료들을 둘러싸는 MBR 정보를 포함한다. 옥트리는 사용자가 설정한 깊이

(tree-depth)만큼의 계층을 형성하며 이 계층의 수가 옥트리의 성능에 가장 큰 영향을 미친다. 계층의 수가 많아지면 각 리프노드가 대변하는 영역의 크기가 줄어들어 탐색 시간을 감소시킬 수 있으나, 너무 크게 설정할 경우 리프 노드에 접근하기 위한 과정이 길어지므로 탐색 시간이 증가하게 된다.

본 연구에서는 옥트리의 메모리 사용량을 줄이기 위하여 각 노드에 MBR 정보를 저장하지 않고 전역 변수로 선언된 MBR을 포인터 형태로 전달하는 방식을 사용하였다. C++로 구현된 옥트리 노드의 수도 코드(pseudo code)는 다음과 같다.

```
class CNode{
        CNode * pChild; // 하위 노드에 대한 포인터
        PointList * pList; // 포인트를 저장하는 리스트에 대한 포인터
        void AddPoint(Point *pt, MBR * pMBR); // 포인트 삽입 함수
        PointList * GetPointList(double X, double Y, double Z, MBR * pMBR); // (X, Y, Z)를 포함하는 노드(정사면체) 내의 포인트 리스트 반환 함수
}
```

포인트 삽입 함수에는, 8개의 하위 노드 중 삽입할 포인트가 포함될 노드를 선택하는 기능과 MBR 정보를 선택된 노드의 MBR로 갱신하는 기능이 포함된다. 포인트의 삽입은 최상위 노드로부터 리프 노드까지 재귀적 형태로 이루어지며 이때 갱신된 MBR 정보가 포인터 형태로 전달된다. 포인트 리스트 반환 함수에도 포인트 삽입 함수와 유사한 하위 노드 탐색 방식이 사용되었다.

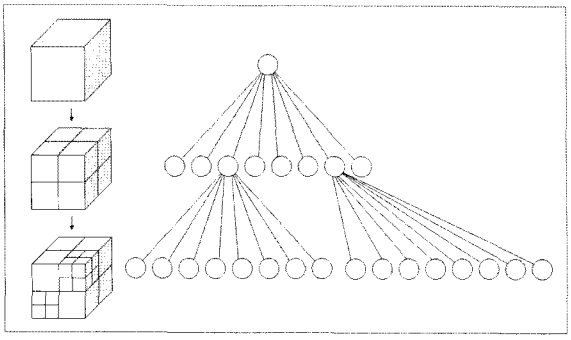

그림 2. 옥트리

2.3 탐색 구성

3D R-tree와 옥트리의 성능 비교를 위하여, 각 포인트로부터 일정 거리 이내에 존재하는 포인트를 조회하는 기능을 구현하였다. 이와 같은 기능은 3차원 포인트 클라우드의 노이즈 포인트 인식, 3차원 개체의 메쉬 생성 또는 표면 법선 벡터 계산 등에 직, 간접적으로 사용될 수 있다. 그림 3a와 같이 인덱싱 방법을 사용하지 않을 경우, 특정 포인트로부터 인접 포인트를 검색하기 위해서는 포인트 클라우드 전체를 검색해야 한다. 3D R-tree(그림 3b)나 옥트리(그림 3c)를 사용할 경우, 검색 영역이 축소되어 조회해야 할 포인트의 총량이 감소한다.

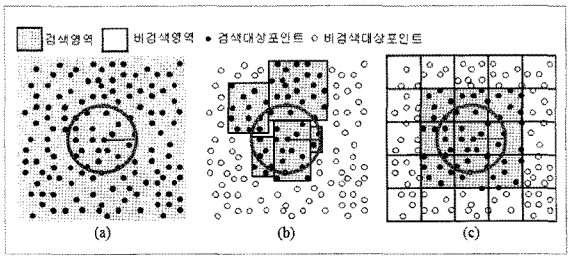

그림 3. 탐색 영역:
(a) 인덱싱 방법 미사용, (b) 3D R-tree, (c) 옥트리

3. 적용 및 분석

실제 건물과 석탑을 대상으로 스캐닝한 3차원 포인트 클라우드를 대상으로 3D R-tree와 옥트리의 성능을 평가하였다. 이를 위하여 포인트 클라우드를 3D R-tree와 옥트리를 입력하고 그 중 10,000개의 포인트에 대하여 2.3절에서 제시한 탐색 시간을 측정하였다. 탐색에는 1cm부터 5cm까지의 다섯 개의 검색 반경을 사용하였다. 3D R-tree의 경우, 먼저 인덱스 용량을 변화시켜 최소의 탐색 시간이 소요되는 인덱스 용량을 도출한 후, 최소의 탐색 시간이 소요되는 리프 용량을 도출하였다. 옥트리의 경우 계층 수를 변화시켜 최소의 탐색 시간이 소요되는 계층 수를 도출하였다. 실험에 사용된 자료와 처리 시스템의 제원은 각각 표 1, 표 2와 같으며 포인트 클라우드의 형태는 그림 4, 그림 5와 같다.

표 1. 포인트 클라우드 제원

지상 레이저 스캐너	Leica scan station 2	
스캐닝 대상물	건물 한 동	석탑 한 동 및 주변 일부
대상물 규모	△x = 15.11m, △y = 12.57m, △z = 13.16m	△x = 6.69m, △y = 6.73m, △z = 5.45m
포인트 수	6,484,350개	5,199,821개

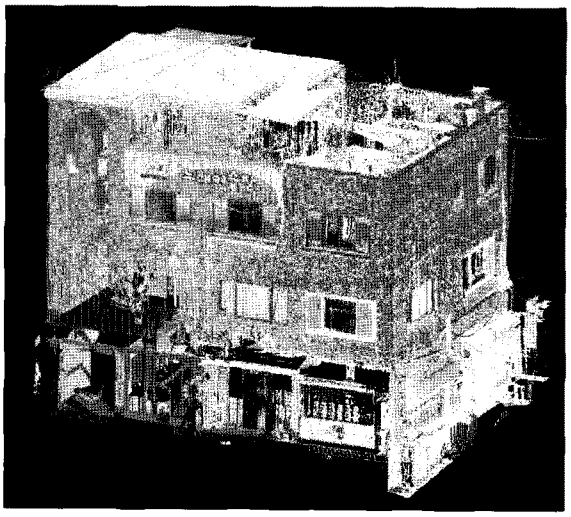

그림 4. 스캐닝 대상 건물

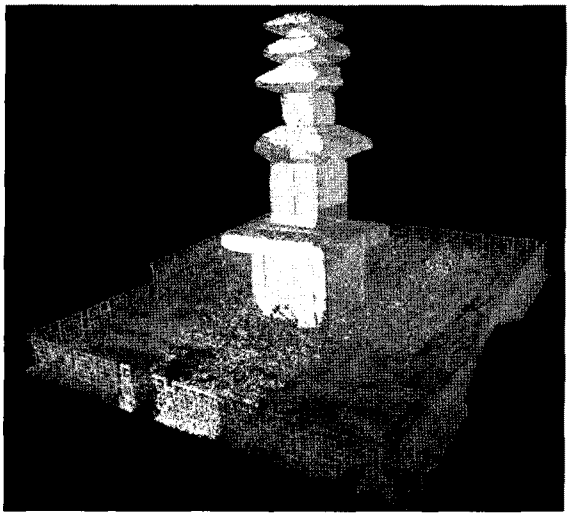

그림 5. 스캐닝 대상 석탑

표 2. 처리 시스템 및 실험 사양

시스템 제원	CPU : Intel core 2 duo E8200 (2.67GHz) RAM : 8.0 GB OS : Windows 7 (64 bit edition)
코딩 언어	C++
검색 반경	1cm, 2cm, 3cm, 4cm, 5cm
R-tree 인덱스 용량	4, 6, 8, 10, 20, 40, 60
R-tree 리프 용량	4, 6, 8, 10, 20, 40, 60
옥트리 계층 수	7, 8, 9, 10, 11

3.1 3D R-tree

건물 및 석탑의 포인트 클라우드에 대하여 리프 용량을 우선 10으로 고정하고 4부터 60까지의 7가지 인덱스 용량(표 2)에 대하여 탐색 시간을 측정하였다. 건물의 경우, 표 3에서와 같이 인덱스 용량이 10인 경우 가장 적은 시간이 소요되었다. 표 3의 결과에 의해 인덱스 용량을 10으로 고정하고 4부터 60까지의 7가지 리프 용량에 대하여 탐색 시간을 측정하였다. 표 4에서 알 수 있듯이, 탐색반경 1cm에서는 리프 용량이 10인 경우, 탐색반경 2cm에서는 리프 용량이 20인 경우, 탐색반경 3~5cm에서

표 3. 건물에 대한 3D R-tree 탐색 시간(초)(리프 용량 = 10)

인덱스용량	1cm	2cm	3cm	4cm	5cm
4	8.56	12.02	15.97	20.43	25.31
6	5.51	7.98	10.94	14.31	18.03
8	5.03	7.42	10.26	13.50	17.07
10	4.81	7.06	9.75	12.81	16.28
20	5.21	7.41	10.04	13.04	16.41
40	5.75	8.07	10.77	13.84	17.29
60	6.21	8.59	11.40	14.48	17.91

는 리프 용량이 40인 경우 가장 적은 시간이 소요되었다. 3D R-tree의 생성 시간 및 메모리 사용량은 표 5와 같다. 석탑의 경우, 표 6에서와 같이 인덱스 용량이 20인 경우 가장 적은 시간이 소요되었으며, 이에 따라 인

표 4. 건물에 대한 3D R-tree 탐색 시간(초)(인덱스 용량 = 10)

리프용량	1cm (sec)	2cm (sec)	3cm (sec)	4cm (sec)	5cm (sec)
4	5.57	9.07	13.28	18.24	23.81
6	5.36	8.18	11.54	15.39	19.67
8	5.23	7.91	11.02	14.58	18.53
10	4.79	7.05	9.75	12.80	16.26
20	4.81	6.84	9.16	11.78	14.77
40	4.94	6.87	9.03	11.48	14.17
60	4.96	7.00	9.21	11.69	14.44

표 5. 건물에 대한 3D R-tree 구축 시간/메모리 사용량(초)(인덱스 용량 = 10)

리프용량	구축 시간 (sec)	메모리 사용량 (MB)
4	460.81	629.13
6	355.31	426.63
8	323.51	323.88
10	306.70	274.74
20	260.70	164.21
40	245.47	95.61
60	245.25	73.28

표 6. 석탑에 대한 3D R-tree 탐색 시간(초)(리프 용량 = 10)

인덱스용량	1cm	2cm	3cm	4cm	5cm
4	4.84	7.51	11.16	15.77	21.49
6	3.19	5.22	7.99	11.63	16.16
8	3.01	4.93	7.54	11.04	15.40
10	2.97	4.78	7.31	10.64	14.83
20	2.94	4.73	7.21	10.48	14.57
40	3.27	5.07	7.61	10.91	14.97
60	3.63	5.46	8.01	11.33	15.49

표 7. 석탑에 대한 3D R-tree 탐색 시간(초)(인덱스 용량 = 20)

리프용량	1cm (sec)	2cm (sec)	3cm (sec)	4cm (sec)	5cm (sec)
4	2.99	5.30	8.68	13.29	19.24
6	3.12	5.11	7.93	11.69	16.42
8	3.06	5.00	7.63	11.15	15.59
10	2.95	4.74	7.20	10.48	14.56
20	2.98	4.66	6.92	9.85	13.48
40	3.26	5.13	7.55	10.60	14.33
60	3.91	5.91	8.53	11.74	15.58

표 8. 석탑에 대한 3D R-tree 구축 시간/메모리 사용량(초)(인덱스 용량 = 20)

리프용량	구축 시간 (sec)	메모리 사용량 (MB)
4	331.49	439.25
6	285.61	316.29
8	253.62	242.47
10	244.50	218.75
20	208.49	133.02
40	195.80	80.22
60	199.40	61.80

덱스 용량을 20으로 고정하고 4부터 60까지의 7가지 리프 용량에 대하여 탐색 시간을 측정하였다. 표 7에서 알 수 있듯이, 탐색반경 1cm에서는 리프 용량이 20인 경우, 탐색반경 2~5cm에서는 리프 용량이 40인 경우 가장 적은 시간이 소요되었다. 3D R-tree의 생성 시간 및 메모리 사용량은 표 8과 같다.

3.2 옥트리

7부터 11까지의 계층 수에 대하여 탐색 시간을 측정하였다. 건물의 경우, 표 9에서와 같이 탐색반경 1~3cm에서는 계층 수가 9인 경우 가장 적은 시간이 소요되었고, 탐색반경 4~5cm에서는 계층 수가 8인 경우 가장 적은 시간이 소요되었다. 석탑의 경우, 표 11에서와 같이 탐색반경 1cm에서는 계층 수가 8, 9일 때 동일하게 가장 적

표 9. 건물에 대한 옥트리 탐색 시간(초)

계층 수	1cm	2cm	3cm	4cm	5cm
7	0.25	0.34	0.44	0.56	0.68
8	0.10	0.16	0.25	0.35	0.46
9	0.06	0.13	0.24	0.40	0.61
10	0.08	0.27	0.65	1.26	2.16
11	0.25	1.24	3.41	7.19	12.99

표 10. 건물에 대한 옥트리 구축 시간/메모리 사용량(초)

계층 수	구축 시간 (sec)	메모리 사용량 (MB)
7	1.68	178.152
8	2.03	237.320
9	2.61	460.688
10	3.74	1093.922
11	5.35	2145.664

표 11. 석탑에 대한 옥트리 탐색 시간(초)

계층 수	1cm	2cm	3cm	4cm	5cm
7	0.20	0.36	0.55	0.79	1.08
8	0.10	0.23	0.42	0.68	1.02
9	0.10	0.33	0.76	1.45	2.46
10	0.25	1.23	3.43	7.25	13.10
11	1.22	7.38	22.14	48.87	90.75

표 12. 석탑에 대한 옥트리 구축 시간/메모리 사용량(초)

계층 수	구축 시간 (sec)	메모리 사용량 (MB)
7	1.37	162.379
8	1.68	224.570
9	2.25	498.629
10	3.24	1106.746
11	4.59	2018.863

은 시간이 소요되었고, 탐색반경 2~5cm에서는 계층 수가 8인 경우 가장 적은 시간이 소요되었다. 옥트리의 생성 시간 및 메모리 사용량은 건물에 대해서는 표 10, 석탑에 대해서는 표 12와 같다.

3.3 결과 분석

그림 6은 건물에 대한 3D R-tree와 옥트리의 탐색 시간 결과(표 4와 표 9)을 바탕으로 5개의 탐색 반경에 대한 각각의 최소 시간을 비교한 그래프이며, 그림 7은 석탑에 적용한 경우(표 7과 표 11)의 비교 그래프이다. 건물의 경우 3D R-tree가 옥트리에 비하여 최소 약 30배(탐색반경=5cm)에서 최대 약 80배(탐색반경=1cm) 많은 시간이 소요되었다. 석탑의 경우 3D R-tree가 옥트리에 비하여 최소 약 13배(탐색반경=5cm)에서 최대 약 30배(탐색반경=1cm) 많은 시간이 소요되었다. 아울러 최소 탐색 시간을 나타낸 리프용량 및 계층 수를 기준으로, 포인트 클라우드 입력 및 트리 작성 시간을 비교하였다. 건물의 경우 3D R-tree가 약 245초~307초(표 5) 소요된 반면 옥트리는 약 2초(표 10) 소요되어 100배 이상의 차이를 나타내었다. 석탑의 경우에도 3D R-tree는 약 208초~245초(표 8) 소요된 반면 옥트리는 약 2초(표 12)가 소요되어 역시 100배 이상의 차이를 나타내었다. 반면에 메모리 사용량은, 건물의 경우 3D R-tree가 약 96MB~275MB(표 5) 소요되었으나 옥트리는 약 237MB~461MB(표 10) 소요되었다. 또한, 석탑의 경우 3D R-tree는 약 133MB~219MB(표 8) 소요되었으나 옥트리는 약 225MB~499MB(표 12) 소요되었다. 따라서 3D R-tree가 옥트리에 비하여 메모리 사용량이 다소 적음을 확인할 수 있었다.

3D R-tree의 최적 인덱스 및 리프 용량과 옥트리의 최적 계층 수는 모든 자료에 적용할 수 있는 일반적인 수치로는 볼 수 없으며, 3D R-tree의 경우 코드 구현의 최적화를 통해 속도 향상을 기대할 수 있을 것으로 판단된다. 그러나 옥트리는 3D R-tree에 비하여 생성 및 탐색 속도가 우수하며 3D R-tree는 옥트리보다 메모리 효율적이라고 충분히 판단할 수 있다. 아울러, 동일한 포인트 클라우드의 지속적인 사용이 필요할 경우, 3D R-tree(또는 옥트리)의 최적 인덱스 및 리프 용량(또는 최적 계층 수)을 미리 분석함으로써 효율적인 처리를 수행할 수 있을 것으로 판단된다.

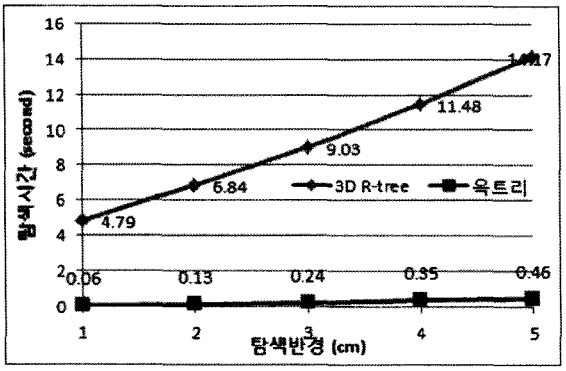

그림 6. 건물에 대한 3D R-tree와 옥트리의 탐색시간 비교

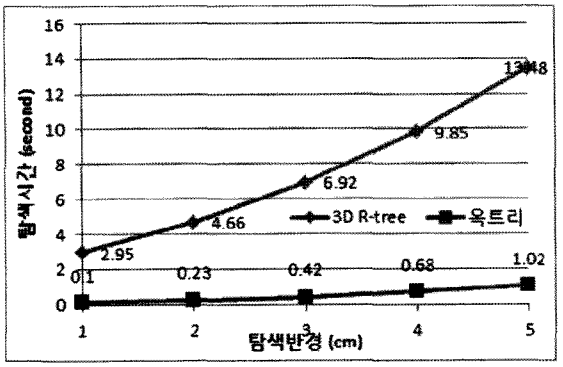

그림 7. 탑에 대한 3D R-tree와 옥트리의 탐색시간 비교

4. 결 론

본 연구에서는 3차원 지상 레이저 스캐너로부터 취득된 포인트 클라우드로부터 효율적인 포인트 탐색을 수행하기 위한 인덱싱 방법으로서 3D R-tree와 옥트리를 비교하였다. 포인트 클라우드의 각 포인트로부터 일정 거리 이내의 포인트를 조회하는 방식으로 탐색을 수행하였으며, 탐색 시간 및 메모리 사용량을 측정하였다. 실제 건물과 석탑을 대상으로 취득된 포인트 클라우드에 적용한 결과, 옥트리는 3D R-tree에 비하여 생성 및 탐색 속도가 우수하며 3D R-tree는 보다 메모리 효율적임을 확인할 수 있었다. 3D R-tree는 인덱스 용량과 리프 용량이, 옥트리는 계층 수가 탐색 성능을 좌우함을 확인하였으며, 주어진 자료에 대한 최적의 수치를 도출할 수 있었다.

향후 연구 과제로서, 3D R-tree의 경우 구현된 코드의 최적화와 파생 구조를 활용한 탐색 속도 향상을 목표로 하고 있다. 옥트리의 경우, PC의 메모리 용량을 초과하는 대용량 포인트 클라우드의 처리를 위하여 병렬처리 컴퓨터의 도입 및 적용을 고려하고 있다.

감사의 글

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2010-0006802)

참고문헌

안성우, 홍봉희, 안경환, 이창우 (2006), 이동체 데이터베이스를 위한 R-tree 기반 메인 메모리 색인의 설계 및 구현, 한국공간정보시스템학회 논문지, 제 8권, 제 2호, pp. 53~73.

이기영, 윤재관, 한기준 (2004), HBR-Tree를 이용한 실시간 모바일 GIS의 개발, 한국공간정보시스템학회 논문지, 제 6권, 제 1호, pp. 73~85.

이득우, 강홍구, 이기영, 한기준 (2009), DGR-Tree : u-LBS에서 POI의 검색을 위한 효율적인 인덱스 구조, 한국공간정보시스템학회 논문지, 제 11권, 제 3호, pp. 55~62.

Cho, H., W. Cho, et al., (2008), 3D Building Modeling Using Aerial LiDAR Data, *Korean Journal of Remote Sensing*, Vol. 24, No. 2, pp. 141-152.

Cho, W., Jwa, Y.S., Chang, H.J. and Lee, S.H. (2004), Pseudo-grid Based Building Extraction Using Airborne Lidar Data, *International Archieve Photogrammetry and Remote Sensing*, Vol. 35, No. B3, pp. 378-381.

Han, S.H., Lee, J.H. and Yu, K.Y. (2007), An Approach for Segmentation of Airborne Laser Point Clouds Utilizing Scan-Line Characteristics, *ETRI Journal*, Vol. 29, No. 5, pp. 641-648.

Wang, M. and Tseng, Y-H, (2004), Lidar data segmentation and classification based on octree structure, *Geo-Imagery Bridging Continents*, ISPRS, Istanbul, Turkey.

Woo, H., E. Kang, et al., (2002), A new segmentation method for point cloud data, *International Journal of Machine Tools and Manufacture*, Vol. 42, No. 2, pp. 167-178.

Zhu, Q., J. Gong, et al., (2007), An efficient 3D R-tree spatial index method for virtual geographic environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 62, No. 3, pp. 217-224.

(접수일 2010. 12. 27, 심사일 2011. 02. 08, 심사완료일 2011. 02. 08)