The Journal of GIS Association of Korea, Vol.10, No.1, pp. 61-76, March 2002

최소경계사각형 압축 및 해싱 기법을 이용한 PDA용 공간색인

김 진 덕*

A Spatial Index for PDA using Minimum Bounding Rectangle Compression and Hashing Techniques

Jin-Deog Kim*

요 약

최근 무선 인터넷의 급속한 확산과 휴대장치 기술의 발전으로 PDA를 이용한 모바일 지도 서비스가 보편화되고 있다. 지도 서비스를 위한 공간 데이터는 용량이 크고, 공간 연산 또한 비용이 매우 큰 반면 PDA는 저장 장치의 용량이 작고 프로세서의 성능이 낮다. 따라서 PDA용 공간 색인은 작은 규모이면서도 공간 연산의 여과 효율이 좋아야 한다.

이 논문에서는 저용량, 낮은 성능의 PDA에 적합한 공간 색인 구조인 MHF를 제안한다. MHF는 간단한 구조로 공간 효율성을 높이면서, 직접 탐색 방법인 해싱을 도입하여 검색 효율을 높였다. 그리고 2차원 공간 색인의 80%이상을 차지하는 최소 경계사각형(MBR) 정보를 압축하기 위한 HMBR을 제안한다. HMBR은 MBR의 표현량을 약 1/3으로 단축함에도 불구하고, 공간 데이터의 대부분을 차지하는 작은 객체인 경우에는 정량화에 의한 정보 손실이 전혀 없어 여과 효율이 좋다. 실험 결과 색인의 크기, MBR 비교 연산 횟수, 여과 효율, 공간 연산의 수행 시간 측면에서 제안한 HMBR을 이용한 MHF 공간 색인이 PDA에 가장 적합한 구조임을 보여주었다.

주요어 : 지리정보 시스템, 공간 색인, MBR 압축, 해싱, 모바일 지도 서비스

ABSTRACT : Mobile map services using PDA are prevailing because of the rapid developments of techniques of the internet and handheld devices recently. While the volume of

* 동의대학교 컴퓨터공학과 (Dept. of Computer Engineering, Dongeui University)
E-mail : jdk@dongeui.ac.kr

spatial data is tremendous and the spatial operations are time-intensive, the PDA has small size memory and a low performance processor. Therefore, the spatial index for PDA should be small size and efficiently filter out the candidate objects of spatial operation as well. This paper proposes a spatial index for PDA called MHF(Multilevel Hashing File). The MHF has simple structure for storage efficiency and uses a hashing technique, which is direct search method, for search efficiency. This paper also designs a compression technique for MBR, which occupies almost 80% of index data in the two dimensional case. We call it HMBR. Although the HMBR technique reduces the MBR size to almost a third, it shows good filtering efficiency because of no information loss by quantization in case of small objects that occupy a major portion. Our experimental tests show that the proposed MHF index using HMBR technique is appropriate for PDA in terms of the size of index, the number of MBR comparisons, the filtering efficiency and the execution time of spatial operations.

Keywords : GIS, Spatial Index, MBR Compression, Hashing, Mobile Map Service

1. 서 론

최근 네트워크 환경의 급속한 확산과 사용자들의 정보 이용 환경도 다변화됨에 따라 유선망 중심의 기존 PC는 정보 생산 및 창출자(Information Creator) 역할을 수행하며, 정보 소비자(Information Consumer) 역할을 담당할 신 개념의 정보 기기인 휴대용 정보 단말에 대한 요구는 필연적이라 할 수 있다. 특히 PDA는 휴대용 정보 기기의 중심에 서 있으며 그 응용 분야가 무선 인터넷 접속, 업무 처리, 영상 전화, 원격 교육, 오락 등 다양하지만 모바일(Mobile) 지도 서비스 또한 매우 중요한 응용분야 중의 하나이다. 특히 이동 환경에서 실시간 지도 정보 서비스, 위치 기반 시스템(LBS), 지능형 교통 시스템(ITS) 등의 응용을 위해서는 PDA를 이용한 효율적인 공간 데이터 검색이 필수적이며, 이를 위해 공간 색인을 도입해야 한다.

PDA는 기존 PC에 비해 작은 용량의 저장 장치와 낮은 성능의 프로세서를 가진다. 반면 공간 데이터는 대용량이며, 공간 연산 비용 또한 매우 크다. 그러므로 지금까지 연구되어온 PC 기반 공간 색인 기법을 PDA에 맞게 보다 소형화해야 함과 동시에 좋은 여과 효율을 갖도록 수정해야 한다. PC 기반 공간 색인 기법을 PDA에 그대로 적용할 수 없는 이유는 기존 공간 색인 기법이 디스크 기반인 반면 PDA용 공간 색인은 메모리 기반이기 때문이다. 즉, PDA는 서버로부터 자료를 다운받으면 메모리에 존재하며, 메모리상의 데이터에 대한 검색을 수행하게 된다. 전통적인 데이터베이스 시스템에서는 데이터가 보조기억장치의 파일 시스템에 존재한다는 가정 하에 보다 빠른 자료 검색을 위해 색인[4] 및 검색 알고리즘 등이 연구되었고 운용되고 있다. 디스크 기반의

공간색인은 항상 페이지 I/O를 위한 노드 구조를 가지고, 상대적으로 많은 시간이 요구되는 디스크 탐색(Seek) 횟수를 줄이기 위한 클러스터링(Clustering) 기법을 통해 속도향상을 도모한다. 그러나 전자 장치인 메모리 기반 시스템에서는 탐색과 클러스터링이 무의미하므로[1,3] 기존의 색인 구조가 메모리 기반인 PDA에서는 성능 향상 효과가 가져 올 수 없다.

또 다른 이유는 PC 기반 공간 색인 중 가장 성능이 좋은 것으로 평가되고 있는 R-tree 계열은 그 크기가 원시 데이터의 16%~20%에 달할 정도로 매우 크다는 점이다[10]. 또한 동적인 데이터의 변경(Update)을 허용하기 위해 각 노드의 사용율이 70% 미만이다. 그렇지만 PDA용 공간 색인은 주로 검색(Retrieval)이 많고 변경은 서버에서 수행되므로 R-tree와 같은 공간 색인은 적합하지 않다.

그래서 작은 용량의 PDA 저장장치를 감안해 공간 색인을 사용하지 않고 공간 연산을 수행하는 것을 고려해볼 수도 있지만, 공간 색인을 사용하지 않을 경우 여과 단계가 없으므로 보다 많은 MBR 비교 연산이 낮은 성능의 프로세서에서 수행되어 응답시간이 매우 길어진다는 것은 주지의 사실이다.

따라서 이 논문에서는 공간 효율성(Storage Efficiency)을 극대화되어 소형이며, 여과 효율(Filtering Efficiency)이 좋아 빠른 데이터 검색이 가능한 PDA용 공간 색인 구조를 제시하고자 한다. 구체적으로 작은 용량의 메모리를 고려하고 고속 처리에 적합하도록 단순한 구조이면서, 직접 검색이 가능한 다중 해쉬(Hashing)을 이용하는 공간 색인구조를 제안한다. 그리고 공간 효율성을 극대화하기 위해 공간 색인 중 약 80%를 차지하고 있는 MBR을 압축해서 표현하는 기법을 제안하고, 압축으로 인한 여과 효율의 저하가 발생하지 않음을 실험으로 입증하고자 한다.

실험 평가를 위한 데이터는 Sequoia 2000 벤치마크 데이터[19]를 사용한다. 실험 평가 대상은 기존 R*-tree와 이 논문에서 제안한 다중 해쉬을 이용한 공간 색인이며 각 공간 색인에 대해 MBR 압축 기법을 적용한다. 실험 평가 항목은 공간 색인의 크기, MBR 비교 연산 횟수, 여과 효율, 전체 수행 시간 등이다. 이와 같은 실험 평가를 통해 PDA에 적합한 공간 색인 구조를 파악할 수 있고, 제안한 색인 구조는 최근 활발히 보급되고 있는 모바일 지도 서비스의 핵심 기술이 될 것으로 판단된다.

이 논문의 구성은 다음과 같다. 2장은 디스크 기반 공간 색인 및 메모리 기반 공간 색인에 관한 관련연구를 다루고, 3장에서는 PDA용 메모리 기반 공간 색인 구조를 제시한다. 4장에서는 공간 색인의 크기를 줄이기 위한 MBR 압축법을 제시하고, 5장에서는 제안한 색인 구조 및 기법에 대한 실험 결과를 제시하고 분석하며, 6장에서 결론을 맺는다.

2. 관련연구

점 질의(Point Query)나 영역 질의(Region query)와 같은 공간 연산은 많은 시간이 소요된다. 그래서 공간 데이터베이스에서 공간 연산의 수행 시 성능 향상을 위해

보통 다음과 같은 두 단계로 나누어 처리한다[9,14,17].

1) 여과 단계(Filter Phase) : 공간적인 조건을 만족할 가능성이 있는 후보 객체(Candidate Object)들을 구한다. 일반적으로 여과를 위해 객체를 둘러싸는 최소 사각형인 MBR을 이용한다.
2) 정제 단계(Refinement Phase) : 여과 단계의 후보 객체들에 대해 실제 객체 좌표들을 이용해 공간적인 조건의 만족 여부를 검사하는 최종 과정으로 많은 시간이 소요된다. 따라서 여과 단계에서 가능한 후보 객체의 수를 줄이는 것이 좋다.

공간색인은 여과 단계를 효율적으로 수행하기 위해 사용된다. 지금까지 수많은 공간 색인에 관한 연구가 진행되어 왔다. 그러나 이들 연구의 거의 대부분이 디스크 기반 공간 색인[2,5,6,7,12,13,18,20]이다. 관련 연구 [2]에서는 지금까지 연구된 공간 색인 중 제일 성능이 좋은 것으로 평가되고 있으며, 각종 공간 데이터 유형(점, 선, 면)에 모두 적용할 수 있는 R*-tree에 관한 연구를 진행하였다. R*-tree는 하나의 노드가 디스크의 한 페이지에 해당되며, 각 노드가 인접한 영역을 포함하는 클러스터링의 성질을 지닌다. 그리고 불균일 데이터의 분포(Distribution)에도 좋은 성능을 보이는 것으로 평가된다.

그러나 공간 효율성 측면에서 보면 R*-tree는 균형 트리를 만들기 위해 노드를 분할하고 병합하는 과정에서 디스크의 사용률(Disk Utilization)이 낮아진다. 연구 결과 70%를 넘기 어려운 것으로 언급이 되고 있다[17]. 그리고 R*-tree는 페이지 I/O를 기반으로 하기 때문에 fan-out이 매우 크다. 따라서 자료 밀집도(Data Density)가 상당히 떨어진다. 또한 다단계 트리 구조로 색인 자체의 크기도 매우 크다. 이와 같은 R*-tree의 특징 때문에 공간 효율성이 상당히 떨어져 제한된 용량을 가진 메모리기반 시스템에서는 적절하지 않다. 그리고 R*-tree는 MBR간의 겹침을 허용하기 때문에 데이터 읽기 위주의 검색 및 특정 지역의 대규모 로딩(bulk loading)에 의한 단순 디스플레이에도 부적합하다.

이러한 단점이 존재함에도 R*-tree가 디스크 기반 공간 색인으로 좋은 성능을 보인 이유는 페이지 I/O와 클러스터링 성질이 디스크의 탐색 횟수를 줄이고 하드웨어적인 I/O 메커니즘을 적절히 이용했기 때문이다. 그렇지만 메모리 기반 공간 색인에서는 이러한 디스크 탐색과 I/O 메커니즘은 의미가 없다.

관련 연구 [6]은 그리드 파일을 제시한 것으로 그리드 분할이라는 분할법을 사용하는데 특정 축 위의 범위를 선정하여 그 범위를 이분하면서 전체 공간을 두개의 하위 공간으로 분할시킨다. 공간 분할은 할당된 페이지에 오버플로우가 발생했을 경우이다. 그리드 파일은 그 구조가 간단하며, 정형적인 구조의 배열을 사용하므로 메모리 구조에 쉽게 적용시킬 수 있다. 그리고 R*-tree와 같은 트리 구조 공간 색인에 비해 색인의 크기가 매우 작다는 장점이 있다. 또한 단 두 번 의 디스크 페이지 접근으로 검색이 가능하다는 장점

은 있다.

그러나 공간 데이터의 분포가 극단적인 치우침(Skewed Data)이 발생했을 경우 그리드 배열의 갯수도 엄청나게 증가하고 그 결과 색인의 크기가 매우 커짐을 알 수 있다. 또한 각 그리드 셀이 전체 데이터 공간에 대해 분할 되므로 사영역(Dead Space)이 많이 발생하고 데이터 검색 시 효율이 많이 떨어진다.

최근에는 메인 메모리 데이터베이스에서 사용할 수 있는 색인[8,11,15,16,18]에 대한 연구도 진행되고 있다. 관련 연구 [11]에서 연구된 T-tree는 메모리에 존재하는 문자 데이터를 검색하기 위한 메모리 기반 1차원 색인이다. T-tree는 이진 트리의 구조를 유지하면서 각 노드는 다중 값(multi value)을 갖는다. 이러한 구조를 가지는 것은 링크 필드를 최소화하고 메모리의 공간 효율성을 높이고자 함이다. 그러나 이 연구는 1차원 문자 데이터에 적용되는 색인으로서, 다차원의 공간 데이터에는 적용될 수 없다. 즉, 공간 데이터는 문자 데이터와 달리 순서화가 불가능하기 때문에 T-tree를 공간 색인에 그대로 적용할 수 없다.

관련 연구 [8]에서는 메모리 기반 공간 색인인 CR-tree를 제시하고 캐쉬를 보다 효율적으로 활용하기 위한 방안을 제시하고 있다. CR-tree는 메모리의 공간 효율성을 높이기 위해 R-tree의 노드 구조를 변경한 것이다. 각 노드의 압축을 위해 R-tree 노드에서 MBR 데이터를 압축하여 표현하는 기법을 사용하고 있다. 그러나 이 공간 색인은 메모리의 공간 효율성은 좋지만, 그 하부구조 자체는 여전히 R-tree를 사용하고 있기 때문에 앞에서 살펴본 R*-tree의 문제점을 그대로 내포하고 있다. 또한 정량화 방법을 사용한 MBR 압축 기법은 공간 효율성은 좋지만 여과 효율의 저하로 비용이 큰 정제 단계에 참여하는 개수의 증가로 나타난다. 정제 단계에 참여하는 개수의 증가는 프로세서 능력이 낮은 PDA에서는 치명적인 단점이다. 5장의 실험 평가에서 보다 자세히 다룬다.

지금까지 대표적인 디스크 기반 공간 색인과 최근 시도되고 있는 메모리 기반 색인들에 대해 살펴보았다. 그러나 아직까지 공간 데이터를 위한 PDA용 공간 색인 구조가 제시된 바가 없고, 그 성능평가 또한 수행되지 않았다. 그래서 공간 데이터의 변경보다는 단순 디스플레이 및 점 질의(Point Query) 위주의 공간 데이터를 검색하는 PDA 맵 서비스를 위한 메모리 기반 공간 색인에 관한 연구를 수행하고자 한다. 이 논문에서 다루는 공간 데이터는 다양한 응용 분야에서 가장 많이 사용되는 다각형(polygon) 레이어이다.

3. PDA용 메모리 기반 공간 색인

기존 PC보다 저용량, 저성능의 PDA 기반 지도 서비스가 원활하게 되려면 공간 색인이 다음과 같은 조건을 만족시켜야 한다.

첫째, 작은 용량의 메모리에 맞게 공간 색인의 크기가 작아야 한다. 그러므로 원시 데이터에 대한 공간 색인의 크기가 매우 큰 R-tree 계열은 적합하지 않으며, 비

교적 단순한 구조를 갖는 공간 색인이어야 한다.

둘째, 잦은 변경보다는 점 질의와 같은 단순 검색 성능이 좋아야 한다. PDA기반 맵 서비스는 공간 데이터에 대한 변경 작업이 거의 없고 검색 위주이므로 여과 단계의 MBR 비교 연산 횟수가 적은 검색 방법을 채택해야 한다.

셋째, PDA 윈도우에 맞는 사각영역의 디스플레이를 위해 대규모 로딩이 쉬워야 한다. 일반적으로 공간 색인 구조가 정규 분할(Regular Decomposition)일 때 특정 지역의 대규모 로딩이 쉬우며, R*-tree와 같은 비정규분할 공간 색인은 대규모 로딩을 위해 여러 개의 탐색 경로를 추적해야 하는 불편함이 있다.

넷째, 불균일 분포(Non uniform distribution) 데이터를 잘 처리할 수 있어야 한다. R*-tree와 함께 널리 사용되는 그리드 파일은 불균일 분포 데이터에 대한 성능이 떨어지는 단점이 있다.

그러므로 기존의 디스크 기반 공간 색인인 R*-tree와 그리드 파일은 PDA용 공간색인으로서 부적절함을 알 수 있다. 이 논문에서는 위와 같은 네 가지 조건을 만족하는 다단계 해쉉 파일 형식의 공간 색인 구조를 제안한다. 이 공간 색인 구조는 비교적 간단한 구조로서 공간 효율성이 높고, 직접 탐색 방법인 해쉉을 도입하여 MBR 비교 연산 횟수를 줄이며, 정규 분할 방식이며, 불균일 분포 데이터를 처리할 수 있다. 이 논문에서 제안하는 공간 색인을 간단히 MHF(Multilevel Hashing File)이라 명명한다.

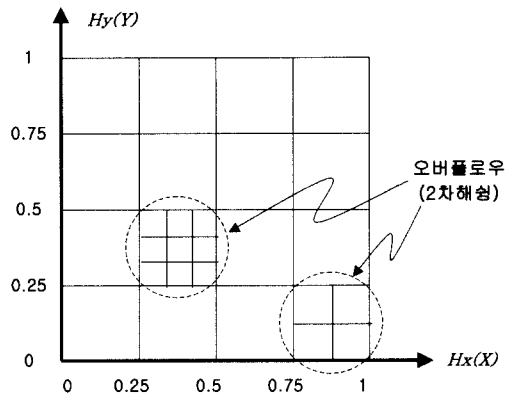
3.1 색인 구조

MHF는 각 객체의 X 값과 Y 값을 기준으로 각각 해쉉하여 주어진 해쉉 버켓에 할당한다. 해쉉 함수는 아래와 같다. 그림 1과 같이 해쉉 테이블은 2차원 구조이며, 오버플로우가 발생했을 경우 2차해쉉을 수행하여 하나의 버켓에 들어가는 객체의 수를 일정하게 유지하도록 하였다.

- Hx(x) = int[(x - Xmin)/(Xmax-Xmin)*Nx]
  단, Nx는 X축 버켓의 수
- Hy(y) = int[(y - Ymin)/(Ymax-Ymin)*Ny]
  단, Ny는 Y축 버켓의 수

그러므로 불균일 분포의 데이터가 일 경우에도 점 질의와 영역 질의에 큰 손실 없이 적절한 응답시간을 보인다. 2차 해쉉 함수는 1차 해쉉 함수와 동일하지만 min, max값이 전체 데이터 공간이 아니라 오버플로우가 발생한 버켓의 최저값과 최고값으로 대치된다. 이와 같은 해쉉은 오버플로우가 발생하지 않을 때가지 하위로 계속 진행된다. 버켓의 용량(M)은 객체의 개수, 원하는 공간 색인의 크기 및 검색 수행 시간에 따라 결정할 수 있다. M 값이 크면 공간 색인의 크기가 줄어들지만 MBR 비교 연산 횟수는 증가하며, M 값이 작으면 그 반대 현상이 나타난다.

그림 2에 나타난 바와 같이 색인 구조의 헤더는 전체 맵의 크기(Extent)와 각 축의 버켓의 수(Nx,Ny) 정보를 포함하며, 버켓 엔트리는 버켓내의 객체의 수와 포인터로 구성된다. 포인터값은 객체의 수에



[그림 1] 해쉬 테이블

따라 아래와 같이 구분된다.

- [# of Obj. = 0] Empty Bucket Condition : Pointer = NULL;
- [1 <= # of Obj. <= M] Normal Condition : Pointer = 해당 버켓을 가리키는 포인터

포인터가 가리키는 곳에는 아래와 같이 해당 버켓에 포함되는 객체들 각각에 대해 MBR과 실제 객체를 가리키는 포인터가 존재한다.

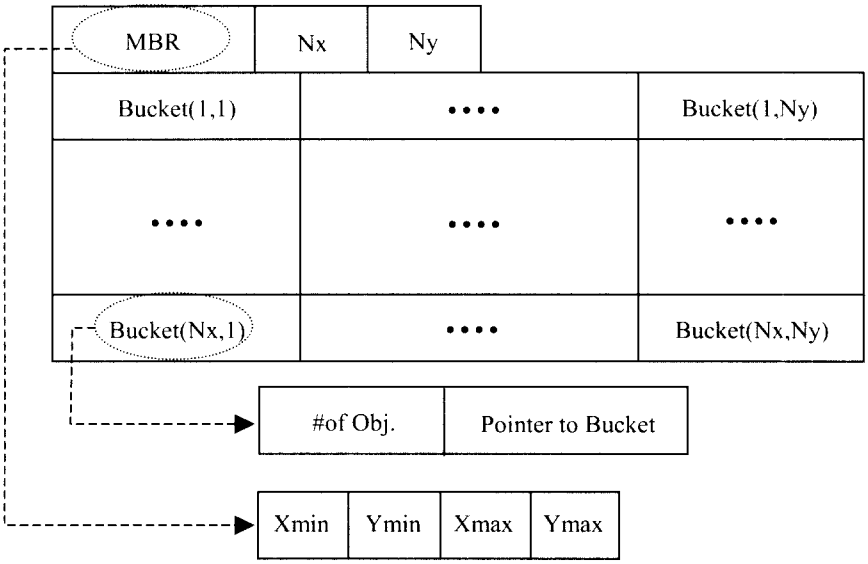
(MBR of Object, Pointer to Object) * #ofObj.

- [# of Obj. > M] Overflow Condition : Pointer = 하위 해쉬 테이블의 Header 포인터

3.2 공간 연산 알고리즘

여기서는 PDA 맵 서비스 응용에서 가장 많이 사용되는 점 질의(Point Query)와 영역 질의(Region Query)에 대한 알고리즘을 제시하고자 한다.

점 질의는 주로 마우스로 특정 객체를 클릭하여 해당 속성 정보를 검색하거나



[그림 2] Header 구조

```
Algorithm Point_Query( Point, Header )
  Read Header
  X = Hx(Point.x); Y = Hy(Point.y);
  Read Bucket(X,Y)
  CASE( # of Obj. )
    [ 0 ] : "Not Found"
    [ > M ] : Header = Pointer_to_Bucket;
              Point_Query(Point, Header) // Recursive Call
    [ 1~M ] : S = Pointer_to_Bucket;
              FOR( all Obj ∈ S )      // 1부터 # of Obj까지
                  IF( Obj.MBR ∩ Point ≠ ∅ ) Result += Obj.
end of Algorithm
```

추가 질의를 수행하고자 할 때 주로 사용된다. 점 질의의 입력 인자는 마우스 지점 좌표(Point)와 공간 색인의 헤더 포인터(Header)이며, 출력값은 선택된 객체들(Result)이다.

영역 질의는 사각형의 질의 영역내에 있는 모든 객체를 검색하는 질의이다. 영역 질의의 입력 인자는 사각형 질의 영역(Region)과 Header 포인터이며, 출력값은 질의 영역 내의 모든 객체(Result)들이다.

MHF는 점 질의에 대해 아주 간단한 해쉉으로 원하는 데이터를 취할 수 있는 방법이며, 특히 색인 구조가 정규분할이기 때문에 특정 지역을 디스플레이 하기 위한 대규모 로딩 작업이 영역 질의에 의해 직관적으로 처리될 수 있다. 또한 불균일 분포의 오버플로우 지역은 재귀호출로 간단히 수행된다.

```
Algorithm Region_Query( Region, Header )
  Read Header
    Xlow = Hx(Region.Xmin); Xmax = Hx(Region.Xmax);
    Ylow = Hy(Region.Ymin); Ymax = Hy(Region.Ymax);
    FOR( X = Xlow ~ Xmax )
    FOR( Y = Ylow ~ Ymax )
      {
       Read Bucket(X,Y)
       CASE( # of Obj. )
        [ 0 ] :  Continue
        [> M] : Header = Pointer_to_Bucket;
                Region_Query(Region, Header) // Recursive Call
        [1~M] : S = Pointer_to_Bucket;
                FOR( all Obj  S )          // 1부터 # of Obj까지
                    IF( Obj.MBR ∩ Region ≠ ∅ ) Result += Obj.
      }
end of Algorithm
```

4. MBR의 압축 표현법

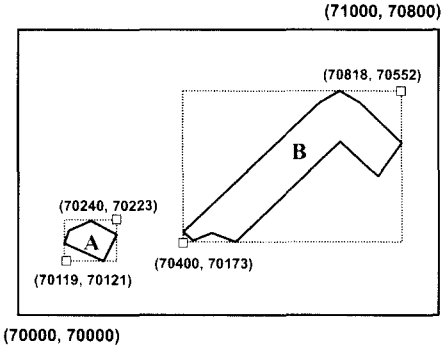
GIS 자료가 지닌 특성은 도형자료의 양이 일반적으로 대단히 크다는 것이다. 그리고 공간 데이터의 색인시 여과 단계의 효율성을 위해 객체의 MBR을 이용한다. 객체의 MBR 표현을 위해 좌하점과 우상점의 두점을 이용한다. 공간객체의 각 축의 좌표값은 최소 4바이트의 정수 또한 실수로 표현하므로 MBR은 16바이트를 필요로 한다. 따라서 공간 색인 자체도 일반 문자용 색인보다 매우 크다. 이러한 문제점은 보조기억장치 보다 상대적으로 크기가 작은 메모리 기반 시스템에서는 반드시 해결되어야 한다. 즉, 공간 색인 중 상당 부분을 차지 하는 MBR의 압축이 필요하며, 아울러 공간 효율성 향상에 따른 여과 효율의 저하가 적어야 한다.

이 논문에서는 다음과 같은 몇 가지 MBR 압축 방법을 이용하고자 한다. 우선 상대(Relative) 좌표값을 이용한 압축법, 관련 연구 [8]에서 제시한 정량화(Quantizaiton)에 의한 MBR 압축 방법, 그리고 이 논문에서 새롭게 제안하는 MBR의 크기를 고려한 혼합(Hybrid) 표현법이다. 편의상 각각 RMBR, QMBR, HMBR이라 명명한다.

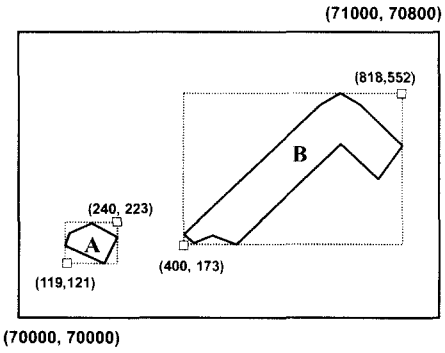
이 논문에서는 단지 공간 색인의 크기를 줄이기 위한 MBR의 압축방법을 다룬다. 공간 데이터 자체의 압축은 고려하지 않는다. 그렇지만 제시한 압축방법이 공간 데이터의 좌표 표현에서도 사용될 수 있음을 밝힌다.

4.1 RMBR : 상대좌표값을 이용한 MBR

MBR을 컴퓨터로 표현하기 위해 전술한 바와 같이 적어도 16바이트의 저장공간이 필요하다. 그렇지만 기준값을 이용한 상대 좌표계를 이용할 경우 각 좌표점은 보다 적은 바이트를 사용하면서 표현될 수 있다. 예를 들어 그림 3의 (a)는 절대 좌표값을 이용한 MBR 표현으로 16바이트가 사용되는 반면, 그림 3의 (b)와 같이 상대 좌표계를 이용할 경우, 각 X,Y 좌표를 각각 2바이트로 표현할 수 있다. 그 결과 저장공간을 절반으로 줄일 수 있다.



(a) 절대 좌표값(MBR)



(b) 상대 좌표값(RMBR)

[그림 3] 절대 좌표값과 상대 좌표값 표현

4.2 QMBR : 정량화에 의한 MBR

앞에서 설명한 상대 좌표계를 이용한다면 MBR 표현량이 반으로 줄어든다. 그러나 관련연구 [8]에서 제시한 방법을 도입하여 각 MBR의 좌표값을 정량화하면 표현량을 더 줄일 수 있다. 그림 4의 (a)는 정량화(13,9등분) 기법을 이용한 QMBR 표현법을 나타낸 것이다. 정량화 단계가 256이하이면 각 좌표값을 1바이트로 표현 가능하므로 QMBR은 4바이트의 표현량이 필요로 하여 기존 MBR에 대해 4배의 압축효과가 있다.

그러나 QMBR은 MBR의 커짐 현상이 발생하고 이는 정제 단계에 참여 하는 객체의 수가 증가하며 결과적으로 전체 수행 시간의 증가를 초래한다. 예를 들어 그림 4의 (a)에서 객체 A의 QMBR은 기존 MBR의 두배 정도의 면적을 가진다. MBR의 커짐 비율은 객체의 크기가 작을수록 더욱 심하다. 이와 같이 정제 단계에 참여하는 객체의 수가 증가함은 프로세서 성능이 매우 낮은 PDA에서는 매우 큰 단점이다.

4.3 HMBR : MBR의 크기를 고려한 혼합 표현

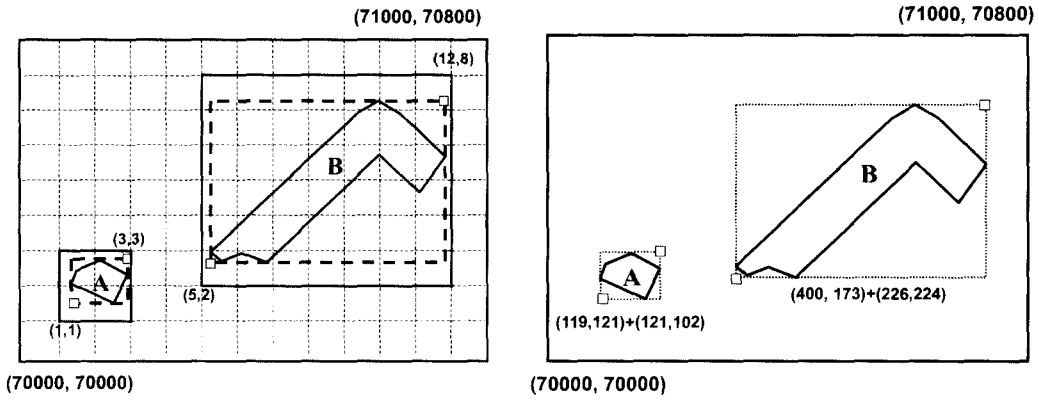
이 논문에서 제시하는 HMBR 표현법은 아래와 같이 MBR의 좌하점은 상대 좌표계처럼 표현하되 우상점을 MBR의 크기로 표현하는 혼합 표현법이다. 이 방법은 MBR의 정확도는 적절히 유지하면서 데이터의 양을 줄이는 방법이다.

- HMBR = Xmin(2byte) + Ymin(2byte) + X 크기값(1byte) + Y크기값(1byte)

MBR의 크기값을 1바이트로 표현하기 위해 다음과 같은 방법을 도입한다.우선 MBR의 크기값이 임계값() 이하이면 크기값을 그대로 표현하고, 크기값이 임계값보다 크면 정량화된 값으로 표현하는 것이다. 이때, 임계값 는 255 - n이며, n은 정량화단계로서 버켓 공간을 n 등분함을 의미한다. 그리고 정량화 크기 는 버켓공간/n이며, 정량화된 값은 + (MBR의 크기) / 이다.

예를 들어 그림 4의 (b)는 절대좌표값으로 표현된 그림 3의 (a)를 HMBR로 표현한 것이다. 객체 A는 크기값을 그대로 표현한 경우이며, 객체 B는 정량화된 값으로 크기를 표현한 경우이다. 객체 A의 MBR 크기가 X축으로 121이고 Y축으로 102이다. 객체 B는 X축으로 418, Y축으로 379이다. 그러므로 객체 A의 HMBR은 (119,121,121,102)이다. 반면, 객체 B의 크기값은 1바이트의 범위를 벗어나 정량화해서 표현한다. 따라서 n이 50이라고 가정하면 는 205이고 는 20이 되어 객체 B의 HMBR은 (400, 173, 226, 224)로 표현된다.

결론적으로 HMBR은 6바이트로 표현가능하고 전체적인 MBR의 표현량이 줄어들게 된다. 대부분의 공간 객체의 크기는 일정 기준치 이하이며 그와 같은 경우에는 값의 오차가 전혀 없다. 다만 객체의 크기가 매우 클 경우에는 MBR의 커짐 현상이 발생하지만 그 빈도가 절대적으로 낮아 전체적인 여과 효율의 저하는 거의 없다.



(a) QMBR (b) HMBR

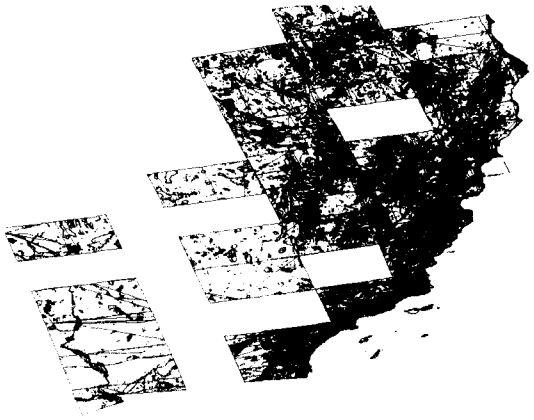
[그림 4] QMBR과 HMBR

이와 같이 MBR 압축 기법을 사용했을 때 점 질의와 같은 공간 연산 수행을 위해 압축된 MBR을 복구하는 과정에서 다소의 연산이 필요하다. 그러나 여러 개의 압축된 MBR을 복구하지 않고, 오히려 질의 좌표를 변환하면 압축에 의한 부하를 간단히 줄일 수 있다.

5. 분 석

5.1 실험 평가 환경

실험 평가 데이터는 현재 공간 데이터베이스 연산의 벤치마크 데이터로 널리 이용되고 있는 Sequoia 데이터[19]이다. 이 데이터는 GIRAS Land use/ Land cover를 표현한 다각형의 집합이며, 총 38개 레이어의 조합이다. 그러므로 점 질의에 의한 결과 값이 2개 이상 존재할 수도 있다. 그림 5는 이들 레이어를 복합하여 출력한 것이며, 불균일 분포 상태이다. 표 1에서 Sequoia 데이터의 특성(객체의 숫자, 각 객체의 점의 개수)을 정리하였다.



[그림 5] 실험 데이터 Sequoia

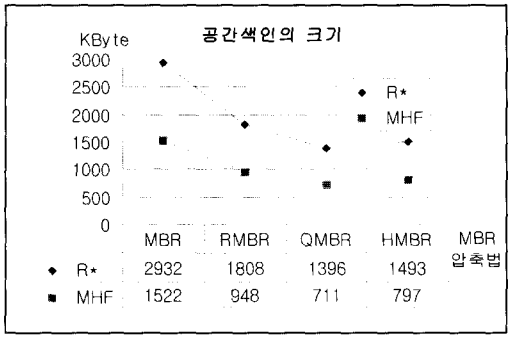
<표 1> Sequoia 데이터의 특성

# of objects	Min (#of points)	Max (#of points)
79607	3	8400

실험을 위한 하드웨어로는 Compaq iPAQ PDA를 이용하였으며, Microsoft embedded Visual C++ 개발 환경에서 구현하였다.

5.2 공간 색인 크기 차이

전술한 바와 같이 PDA를 위한 공간 색인의 크기는 작을수록 좋다. 그림 6은 각 공간 색인 구조에 다양한 MBR 압축표현을 적용한 경우의 색인의 크기를 나타낸 것이다. R*-tree는 노드의 크기를 512바이트로 하였고, MHF는 1차 버켓 용량(M)을 50으로, 2차 부터는 5씩 감소시켰다. QMBR의 정량화 단계는 256이다.



[그림 6] 공간 색인의 크기

실험 결과 공간 효율성 측면에서 다음과 같은 결론을 내릴 수 있다. 첫째, 공간 색인 구조를 평가해 볼 때 같은 MBR 표현법을 사용했을 경우 이 논문에서 제시한 MHF가 R*-tree보다 공간 효율성 측면에서 좋은 것으로 나타났다. 예를 들어 그림 6에서 같은 QMBR 표현법을 사용하는 경우의 R*-tree와 MHF의 크기를 비교해 보면 MHF이 거의 50% 수준의 작은 용량을 차지함을 알 수 있다. 이는 R*-tree가 다단계 구조이며, 균형 트리를 유지하기 위해 공간 효율성이 떨어지기 때문이다.

둘째, MBR 압축 방법에 따른 공간 색인의 크기를 비교해 볼 때 예상대로 QMBR이 좋다. 공간 색인의 크기는 비록 QMBR을 적용한 MHF 방식이 가장 작지만, 여과 효율은 그렇지 않음을 5.4절에서 자세히 설명한다. 또한 HMBR 방식도 좋은 압축효과가 있었다.

5.3 MBR 비교 연산 횟수

점 질의의 검색 성능을 의미하는 여과 단계의 MBR 비교 연산회수는 표 2와 같다. 우선 전체적으로 이 논문에서 제안한 MHF가 R*-tree에 비해 전반적으로 좋은 성능을 보여주었다. 구체적으로 높은 공간 효율성을 보이는 HMBR과 QMBR을 적용했을 경우에서 상대적으로 MHF가 R*-tree보다 절반 수준의 비교 연산만을 요구하는 좋은 성능을 보였다.

둘째, QMBR을 적용했을 경우 다른 방법보다 많은 비교 연산 횟수를 기록했다. 이는 앞에서 설명한 MBR의 커짐 현상으로 인한 성능 저하임을 실험결과로 알 수 있다. 특히 QMBR을 적용한 R*-tree는 더욱 더 좋지 않은 성능을 보였는 데 이는 다단계 구조이면서 영역 겹침을 허용하므로 상위 단계부터 MBR의 커짐 현상이 나타나기 때문이다.

결론적으로 여과 과정의 MBR 비교 횟수 실험 결과로서 색인 구조에서는 MHF가 보다 좋은 성능을 보이고, MBR 압축

법에서는 HMBR 방법이 QMBR 방법에 비해 훨씬 좋음을 확인할 수 있다. 영역 질의의 검색 성능 또한 크게 차이가 나지 않아 생략한다.

<표 2> MBR 비교 연산 횟수

색인구조 / MBR표현법	R*-tree	MHF
MBR	83.58	61.56
RMBR	104.16	61.56
QMBR	157.22	71.68
HMBR	112.42	62.86

5.4 MBR 표현법에 따른 여과 효율

공간 색인의 목적은 여과 단계를 보다 빠른 시간 내에 수행함과 동시에 정제 단계에 참여하는 후보 객체의 개수를 최소화함에 있다. 특히, PDA의 낮은 프로세서 성능을 감안할 때 후보 객체 수가 전체 수행 시간에 지대한 영향을 미친다. 표 3은 점 질의 및 영역 질의(전체 데이터 공간의 0.1%, 0,4%, 1%)를 수행할 때 여과 단계를 마친 후의 후보 객체 수의 평균치를 나타낸 것이다. 실험 결과는 공간 색인 구조와는 독립적이며, 단지 MBR 표현법에 의존적이다. 실험 결과 정량화에 의한 영역 커짐이 발생하는 QMBR 방법은 다른 MBR 표현법에 비해 후보 객체의 수가 매우 많음을 알 수 있다. 특히, 점 질의와 질의 영역이 매우 작을 때는 QMBR 방법이 더 좋지 않은 성능을 보임을 알 수 있다. 반면, HMBR은 비록 정량화 방법을 사용하기 하지만 매우 큰 객체에만 적용되어 후보 객체의 수에서는 MBR이나 RMBR과 거의 대응한 수치를 보여 매우 좋은 성능을 보였다.

<표 3> 후보 객체의 수

질의 영역 / MBR 표현법	PointQuery	0.1%	0.4%	1%
MBR	3.2	187.4	521.3	1038.9
RMBR				
QMBR	7.3	318.5	672.5	1266.7
HMBR	3.5	201.3	552.3	1073.0

5.5 질의 수행 시간

여기서는 QMBR과 HMBR 방법의 성능 평가를 위해 두 방법을 R*-tree, MHF에 각각 적용하여 질의 수행의 평균 시간을 측정해 보았다. 표 4는 질의 수행 시간을 정리한 것이다. 실험 결과 동일 조건일 경우 MHF가 R*-tree보다 좋은 성능을 보였다. 그리고 HMBR 표현법이 QMBR에 비해 평균 40%의 성능 개선이 있었다. 왜냐하면 QMBR은 표 3에 나타난 바와 같이 후보 객체의 수 증가로 인해 연산 비용이 매우 큰 정제 단계의 수행 횟수가 증가 했기 때문이다. 정제 단계에서도 포함 관계(Containment Relationship)은 비교적 간단히 처리되지만 겹침 관계(Overlap Relationship)는 많은 연산 시간이 소요된다. QMBR은 MBR의 커짐 현상으로 겹침 관계에 있는 후보 객체 수를 많이 양산하여 전체 연산 수행 시간이 매우 증가한 것으로 보여진다.

<표 4> 질의 수행 시간(단위 : ms)

질의 영역 / MBR 표현법	PointQuery	0.1%	0.4%	1%
R*-tree(QMBR)	106.2	901.7	1278.3	1934.9
R*-tree(HMBR)	69.1	543.8	881.2	1215.2
MHF(QMBR)	78.8	792.3	1159.1	1773.2
MHF(HMBR)	46.8	451.8	782.9	1103.0

이상과 같은 모든 실험 결과를 종합해 볼 때 이 논문에서 제안한 공간 색인 구조인 MHF가 R*-tree에 비해 평균적으로 절반 이하의 저장 용량으로 표현 가능하여 좋은 공간 효율성을 보여 주었으며, 제안한 MBR 표현법인 HMBR 방법이 여과 효율측면과 전체 수행 시간에서 좋은 성능을 보여 PDA에 적합한 방법이라 판단된다. 다만, 공간 효율성 측면에서는 QMBR 방법이 조금 더 좋은 성능을 보이지만 여과 효율의 저하로 인한 전체 수행 성능이 상당히 떨어짐을 알 수 있었다.

물론 실험 데이터의 특성에 따라 실험 결과가 차이가 많을 것이지만, 이 논문에서 사용한 실험 데이터는 벤치마크 데이터로 가장 많이 사용되며, 실제 데이터(Real Data)와 유사한 특성을 가지므로 실험 결과는 보다 신뢰성이 있을 것으로 사료된다.

6. 결 론

이 논문의 목적은 현재 많이 응용되고 있는 PDA용 맵 서비스에 적합한 공간 색인을 도출하고자 함에 있다. 이와 같은 목적에 부합하기 위한 공간 색인의 조건은 보다 작은 용량이며, 점 질의와 같은 공간 연산의 응답시간이 빨라야 한다는 것이다.그래서 이 논문에서는 PDA용 메모리 구조에 맞은 간단한 구조로 공간 효율성이 높으면서 직접 탐색 방법인 해쉬을 도입한 공간 색인 구조로서 MHF를 제안하였다. 그리고 공간 효율성을 높이기 위한 방안으로서 공간 색인의 80% 이상을 차지하는 MBR 정보를 압축하는 방법을 제안하였다. 제안한 MBR의 크기를 고려한 혼합 표현법인 HMBR은 그 표현량을 줄이면서 공간 데이터의 대부분을 차지하는 크지 않은 객체인 경우에는 오차가 전혀없다는 장점이 있다.

실험 결과 지금까지 디스크 기반 공간 색인 구조 중 최고의 성능을 보이는 R*-tree가 메모리 기반 PDA에서는 그렇지 않음을 알 수 있었다. 이를 통해 R*-tree가 디스크 기반 공간 색인으로서 페이지 I/O와 클러스터링을 적절히 표현하지만 메모리 기반 시스템에서는 오히려 단점으로 작용함을 알 수 있었다. 이 논문에서 제안한 공간 색인인 MHF가 공간 효율성과 검색 성능에서 R*-tree에 비해 좋은 성능을 보여주었다. 또, 이 논문에서 제안한 MBR 압축법인 HMBR이 여타의 방법에 비해 압축 효과도 뛰어나면서 검색 효율도 좋은 것으로 평가되었다.

구체적으로 MHF는 공간 효율성 측면에서는 R*-tree에 비해 약 50% 이하의 작은 용량을 차지하며, MBR 비교 연산 횟수 측면에서는 절반 정도로 줄어들었다. 그리고 HMBR을 이용할 경우 공간 색인의 크기가 기존 MBR을 이용할 경우에 비해

50%이상 감축 효과가 있었다. 또한 HMBR 방법은 공간 연산 수행 시간에서도 평균 40% 정도 개선되었다.

실험 평가를 통해 제안한 색인 구조가 저용량, 저성능의 PDA에 적합한 공간 색인 구조임을 알 수 있었고, 최근 활발히 보급되고 있는 모바일 지도 서비스, 지능형 교통 시스템(ITS), 위치 기반 시스템(LBS) 등의 핵심 기술이 될 것으로 판단된다.

참고문헌

[1] A. Ailamaki, D.J. Dewitt, M.D. Hill, D.A. Wood, 1999, "*DBMSs on a modern processor : where does time go?*", Proc. of Int. Conf. on VLDB, pp. 510-521

[2] N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger, 1990, "*R*-tree : An Efficient and Robust Access Method for Points and Rectangles*", Proc. of Int. Conf. on ACM SIGMOD, pp. 322-331

[3] P. Boncz, S. Manegold, M. Kersten, 1999, "*Database architecture optimized for the new bottleneck : memory access*" Proc. of Int. Conf. VLDB, pp. 54-65

[4] D. Greene, 1989, "*An Implementation and Performance Analysis of Spatial Data Access*", Proc. of Int. Conf. on Data Engineering, pp. 606-615

[5] A. Guttman, 1984, "*R-trees: A dynamic index structure for spatial searching*", Proc. of Int. Conf. on ACM SIGMOD, pp. 47-57

[6] K. Hinrichs, J. Nievergelt, 1983, "*The Grid File : A Data Structure designed to Support Proximity Queries on Spatial Objects*", Proc. of Int. Conf. on Graph Theoretic Concepts in Computer Science, pp. 100-113

[7] E.G. Hoel, H. Samet, 1992, "*A Qualitative Study of Data Structures for Large Line Segment Databases*", Proc. of Int. Conf. on ACM SIGMOD, pp. 205-214

[8] K.H. Kim, S.K. Cha, K.J. Kwon, 2001, "*Optimizing multidimensional index trees for main memory access*", Proc. of Int. Conf. on ACM SIGMOD

[9] R. Laurini, D. Thompson, 1992, "*Fundamentals of Spatial Information Systems*", Academic Press, pp. 680

[10] T.J. Lehman, M.J. Carey, 1986, "*Query processing in Main memory Database management system*", Proc. of Int. Conf. on Management of Data, ACM SIGMOD, pp. 239-250

[11] T.J. Lehman, M.J. Carey, 1986, "*A Study of index structures for main memory database management system*", Proc. of Int. Conf. on VLDB, pp. 294-303

[12] H. Lu, B.C. Ooi, 1993, "*Spatial Indexing : Past and Future*", IEEE Data Engineering Bulletin, Vol. 16, No. 3, pp 16-21

[13] J. Neivergelt, H. Hinterberger, 1984, "*The Grid File : An Adaptable, Symmetric Multikey File Structure*", ACM Transaction on Database Systems, Vol. 9, No. 1, pp. 38-71

[14] B. Pagel, H. Six, H. Toben, P. Widmayer, 1993, "*Towards an Analysis of Range Query Performance in Spatial Data Structures*", Proc. of Int. Conf. on ACM SIGMOD, pp. 214-221

[15] J. Rao, K.A. Ross, 1999, "*Cache conscious indexing for decision-support in main memory*", Proc. of Int. Conf. on VLDB, pp. 78-89

[16] J. Rao, K.A. Ross, 2000, "*Making B+-trees cache conscious in main memory*", Proc. of Int. Conf. on ACM SIGMOD, pp. 475-486

[17] H. Samet, 1990, "*The Design and Analysis of Spatial Data Structures*", Addison Wesley, pp. 507

[18] A. Shatdal, C. Kant, J.F. Naughton, 1994, "*Cache conscious algorithms for relational query processing*", Proc. of Int. Conf. on VLDB, pp. 510-521

[19] M. Stonebraker, J. Frew, K. Gardels, J. Meredith, 1993, "*The SEQUOIA 2000 Storage Benchmark*", Proc. of Int. Conf. on ACM SIGMOD, pp. 2-11

[20] K.Y. Whang, R. Krishnamurthy, 1991, "*The Multilevel Grid Files - a Dynamic Hierarchical Multidimensional File Structure*", Proc. of Int. Conf. on Database Systems for Advanced Applications, pp. 449-459