균등분포 변환을 이용한 공간 색인방법

김동현[O] 문정욱 이기준
부 산 대 학 교 전 자 계 산 학 과
{ dhkim, jwmoon }@quantos.cs.pusan.ac.kr, lik@hyowon.pusan.ac.kr

Spatial Indexing Method by Transformation to Uniform Distribution

Tong-Hyon Kim[O] Jung-Wook Moon Ki-Joune Li
Dept. of Computer Science, Pusan National University

요 약

대부분의 공간 색인 방법들은 공간 객체의 분포가 균일하지 않다는 가정 하에 만들어졌다. 이 가정은 실제의 응용분야에 일반적으로 적용된다. 반면에 균등한 공간 객체의 분포를 위하여 만들어진 공간 색인 방법들도 몇 가지 있다. 예를 들어, 고정 격자 방식의 공간 색인 방법은 균등한 공간 객체에 대한 색인 방법으로 매우 좋은 성능을 보이며, 색인 과정이 매우 단순하다는 장점이 있다. 본 논문에서는 STR을 이용하여 균일하지 않은 분포의 공간 데이터를 균일한 분포로 변형한 다음 고정 격자 방식의 공간 색인을 적용하는 방법을 제안한다. 또한, 본 논문에서 제시한 색인 방법의 질의 처리 성능 예측 모델을 제안하고 실험을 통하여 모델의 정확성 정도를 보여준다. 이 방법에 따르면, 공간 효율성이 매우 증가하고, 빈 공간이 줄어드는 등 공간 색인의 성능이 다른 색인 방법에 비하여 매우 증가하며, 색인 과정이 단순하여 지며 질의 처리 성능이 좋아지는 장점을 가지고 있다.

1. 서론

지리정보시스템과 같은 공간 데이터베이스에서 공간 색인 방법은 데이터의 입, 출력에 기본적으로 사용되며 질의 처리를 위한 중요한 도구이다. 그래서 공간 색인 방법에 대한 많은 연구가 있어 왔다. 격자 파일이나(grid file), 고정 격자 파일(fixed grid file)이 그것인데[5, 6, 7, 8], 이러한 방법들은 실제의 비균일 분포의 데이터를 처리하는 데는 역부족이었다. 그래서 비균일 분포의 데이터를 효율적으로 처리하기 위한 공간 색인 방법으로 R-tree 계열의 공간 색인 방법이 제안되었다[1, 2, 3, 4]. 하지만 고정 격자 색인 방법이 단순하여 구현과 관리가 쉬운 데 반해, R-tree 계열의 공간 색인 방법은 비교적 복잡한 구조와 과정을 통하여 색인을 수행하므로 Buddy-tree와 같은 비균일 분포의 공간 데이터에 대해서도 좋은 성능을 보이는 고정 격자 색인 방법이 소개 되기도 했다[15].

격자 파일을 이용한 공간 색인 방법들은 단순하여 구현하고 관리하기가 쉽다는 장점이 있지만 이와 같은 장점에도 불구하고 대부분의 응용분야에서 주어지는 실제 공간 객체의 분포가 비균일 분포이기 때문에 사용되는 데는 문제가 있다[9].

본 논문에서는 비균일 분포의 데이터를 균일 분포의 데이터로 변형하여 격자 파일로 처리할 수 있는 방법을 제안하고 질의 처리 성능에 대한 모델을 제시한다. 본 고정 격자 방법을 이용한 공간 색인은 질의의 여과 단계(filtering)에서는 디스크 접근이 필요 없고, 정제 단계(refinement)에서 데이터 블록에 대한 디스크 접근만이 요구되어 매우 성능이 뛰어나다.

본 논문은 다음과 같이 구성되어 있다. 먼저 2장에서 고정 격자 공간 색인 방법의 장점과 단점에 대해서 알아본다. 3장에서는 균등 분포 변환을 이용한 고정 격자 방법에 대해서 균등 분포로 변환하는 방법에 대해서 알아보고 균등 분포로 변형된 데이터의 균등 분포 정도를 살펴본 뒤, 질의 처리 방법, 질의 처리 성능 예측의 순으로 알아본다. 4장의 성능평가에서는 현재 가장 많이 사용되는 공간 색인 방법인 R*-tree와 본 논문에서 제시하는 방법의 성능을 비교해 보고, 3장에서 제시한 질의 처리 성능 예측 모델을 실험 결과와 비교해 본다.

2. 고정 격자 공간 색인 방법의 장점과 단점

공간 색인의 성능은 공간 객체의 분포에 많은 영향을 받는다. 또한 대부분의 응용분야에서 주어지는 공간 객체의 분포는 비균일적이기 때문에, 공간 색인은 비균일적인 공간 객체의 분포에 대해서도 좋은 성능을 보여야만 한다.

공간 객체의 비균일 분포의 문제를 해결하기 위해

서 R-tree 계열의 방법은 비교적 복잡한 구조와 과정을 통하여 색인을 수행한다. 먼저, 색인을 위해서는 트리 구조를 가지는 자료구조를 필요로 하고, 공간 질의 처리를 위해서는 이 자료구조에 대한 복잡한 처리 과정이 요구된다. 또한 공간 색인을 위해서는 이 트리 자료구조의 방문을 위하여 많은 디스크 접근이 필요하다.

하지만 공간 객체의 분포가 균일 하게 되어 있으면 위와 같은 복잡하고 많은 시간이 요구되는 과정은 필요 없게 된다. 균일한 분포의 데이터에 대해서 고정격자 공간 색인 방법을 이용하면 아래 그림 1에서와 같이 하나의 격자는 하나의 디스크 블록에 해당되므로 질의 영역에 겹쳐지는 디스크 블록의 데이터만 읽으면 된다. 그러므로 고정 격자 공간 색인 방법은 R-tree의 중간 노드에 해당하는 디스크 블록의 방문이 필요 없다는 장점이 있다.

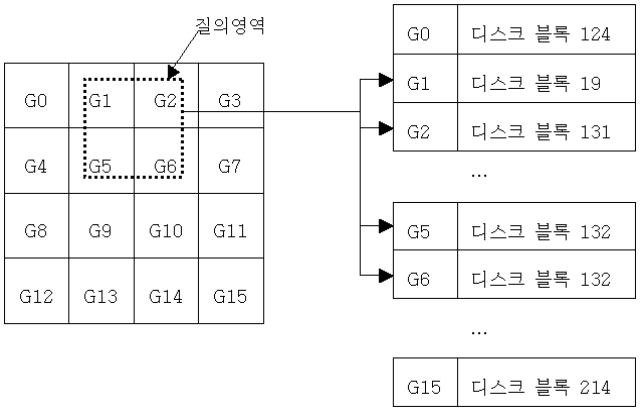


그림 1. 고정 격자를 이용한 공간색인

그러나 고정 격자 방법의 가장 큰 단점은 사장 공간(dead space)으로 인한 저장 공간 효율성의 저하이다. 앞에서 언급한 바와 같이, 공간 객체의 분포는 비균일적이다. 이것은 각 격자에 속하는 공간 객체의 수가 서로 다르다는 것을 의미한다. 즉, 격자의 밀도 $density(G_i)$ 가 서로 다르다는 것을 의미한다. 그런데, 하나의 격자는 하나의 디스크 블록에 해당되므로 다음과 같은 조건을 만족하도록 격자가 구성되어야 한다. 여기서 a 는 하나의 격자의 넓이를 의미한다.

$$\frac{Bf_{\max}}{d_{\max}} \geq a \qquad \text{-----} (1)$$

여기서 $Bf_{\max}$ 는 디스크 블록의 최대 저장 객체 수(Blocking Factor)이고, $d_{\max}$ 는 아래 식과 같다.

$$d_{\max} = Max(density(G_i), i = 0, n-1) \quad \text{-----} (2)$$

식(1)은 $Bf_{\max}$ 는 정해져 있으므로, 최대 밀도가 높아지면 하나의 격자의 넓이는 작아지고 결과적으로 격자의 개수와 디스크 블록의 수 N_{block} 는 증가된다는 것을 의미한다. 즉, 식(3)와 같이, 저장 공간의 효율성(Storage Utilization)이 떨어지므로, 주어진 질의 영역에 걸쳐지는 격자의 수가 많아지고, 격자에 대응되는 디스크 블록의 수가 많아져 색인 과정에 필요한 디스크 접근은 없지만 데이터 블록을 위한 디스크 접근 횟수가 많아진다는 것을 의미한다.

$$N_{block} = \frac{A}{a}, \ (A: \text{전체 영역의 넓이})$$
$$N_{block} \geq A \cdot \frac{d_{\max}}{Bf_{\max}} \qquad \text{-----} (3)$$

만일 비균일 정도, 즉 공간 객체의 집중도가 커지면, $d_{\max}$ 가 커지고 결과적으로 디스크 블록의 수도 함께 많아지게 된다.

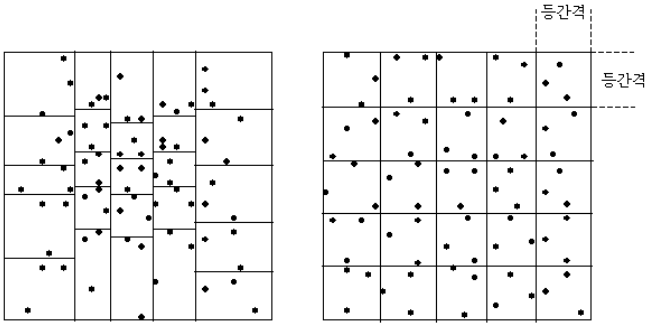
본 논문에서는 비균일적으로 분포되어 있는 공간 객체의 위치를 적절하게 변환하여 균일 분포로 만드는 방법을 제안한다. 즉, 균일 분포로 만들어, $d_{\max}$ 의 값을 가능한 평균밀도 d_{mean} 에 근접하게 만들어 저장 공간의 효율성을 증가시키고, 결과적으로 공간 질의 처리에서 요구되는 데이터 블록의 접근 횟수를 줄이는 방법을 제안한다.

3. 균일 분포 변환을 이용한 고정 격자 방법

비균일 분포의 데이터를 균일하게 변형하는 방법은 여러 가지가 있을 것이나, 본 연구에서는 들로니 삼각 분할(Delaunay Triangulation)을 이용한 탄성변형과 STR을 이용한 변환에 대해 연구하였다. 그 결과 STR을 이용한 변환이 데이터를 더 균일 분포에 가깝게 변환하여 본 논문에서는 STR을 이용한 방법을 소개한다.

3.1 균일 분포 변환

비균일 분포를 가지는 데이터를 균일 분포로 변환하는 방법에는 STR을 이용한다[11]. STR을 이용한 균일 분포 변환은 STR의 각 격자에 같은 개수의 데이터를 넣고, STR로 나누어지는 공간을 동일한 크기의 공간으로 변환하는 것이다. 즉, 그림 2과 같이 하나의 격자 내에 있는 공간 객체의 위치를 변환하는 것이다.



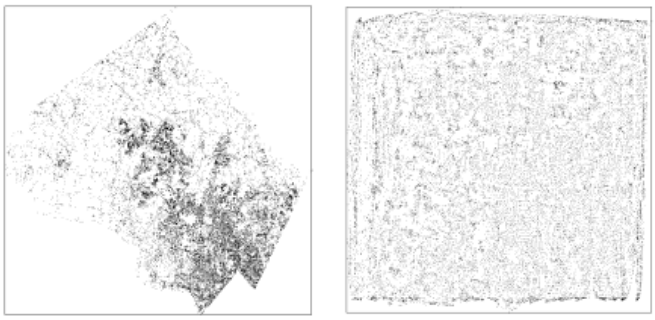
(a) 변환 전 객체 분포 (b) 변환 후 객체 분포

그림 2. STR을 이용한 균일 분포 변환

그림 2-a의 변환 전 객체 분포는 중앙에 데이터가 집중되어 있다. 그러나 그림 2-b의 변환 후, 공간 객체의 분포가 거의 균등함을 알 수 있다.

3.2 균일 분포 변환 결과

우선, 공간 질의 처리를 위해서는 변형을 위한 보조적인 정보가 필요하다. 탄성변형의 경우는 제어점의 좌표와 제어점 연결 정보가 필요하고, STR을 이용한 방법은 STR의 스케일 정보가 필요하다.



(a) 변환 전 (b) 변환 후

그림 3. 균일 분포 변형 예

그림 3는 균일 분포로 변형한 예이다. 그림 3-a는 미국 몽고메리 지역의 데이터이고, 그림 3-b는 그림 3-a의 데이터를 STR을 이용하여 변형을 수행한 것이다. 이를 히스토그램으로 나타낸 것이 아래 그림 4와 같다.

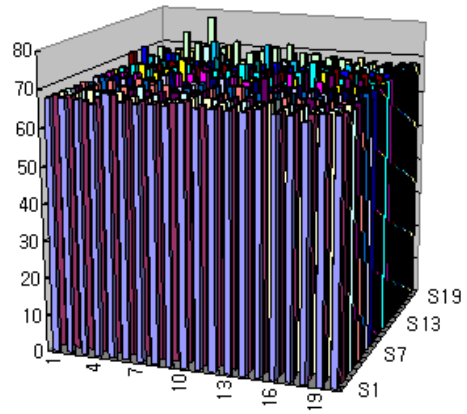


그림 4. 변형 후 분포

그림 3, 4에서 확인한 것 같이 STR을 이용한 변환에서 거의 균일한 분포에 가깝게 변환되었다.

3.3 질의 처리

공간 질의가 요청되면, 공간 객체와 마찬가지로 공간 질의의 위치도 변형되어야 한다. 질의를 변형하는 방법은 그림 5와 같다.

그림 5에서 보듯이 STR에 의한 변형에서 하나의 영역 질의가 주어질 경우 각 격자에 해당하는 여러 개의 영역 질의들로 분할되고, 각각의 분할된 영역 질의들은 STR 격자 정보에 의해서 변형되게 된다.

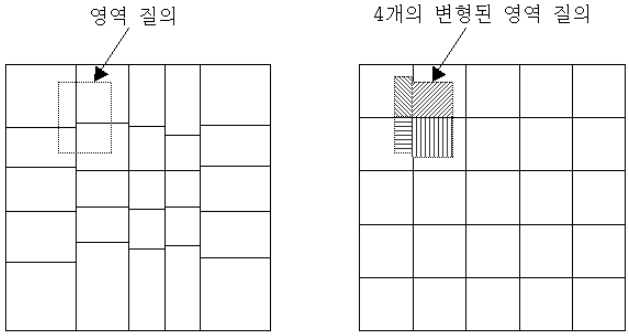
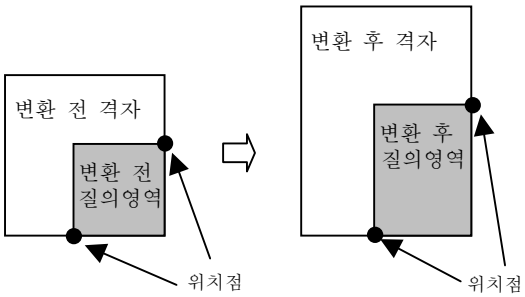


그림 5. STR을 이용한 균등 분포 변환에서의 영역 질의 변환

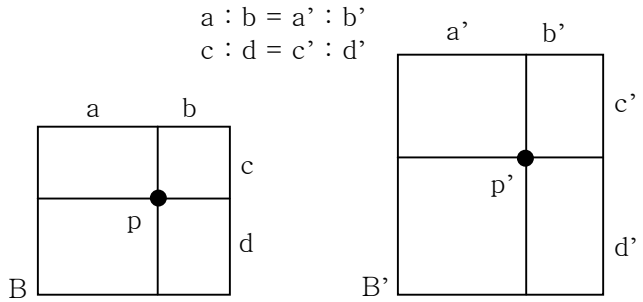
그림 5에서 보듯이 STR에 의한 변형에서 하나의 영역 질의가 주어질 경우 각 격자에 해당하는 여러 개의 영역 질의들로 분할되고, 각각의 분할된 영역 질의들은 STR 격자 정보에 의해서 변형되게 된다.



(a) 변환 전 격자와 해당 질의 영역 (b) 변환 후 격자와 해당 질의 영역

그림 6. 격자 변환에 따른 영역 질의 변환

그림 6에서 분할된 질의는 2개의 위치점으로 구성되어 있다. 이 때, 질의 영역의 변환은 위치점이 격자 내부에서 가지는 위치 비율을 유지하면서 이루어진다.



(a) 변환전 격자의 위치점 (b) 변환후 격자의 위치점

그림 7. 격자 변환에 따른 위치점 변환

분할된 질의의 변환을 위하여 질의를 구성하고 있는 위치점을 그림 7과 같이 위치 비율을 유지하며 좌표 이동을 할 수 있는데, 이러한 격자 변환 식은 위치 비율을 사용하여 간단하게 구할 수 있다. 그림 7-a의 격자를 B, 그림 7-b의 격자를 B'라고 하자. B의 격자의 왼쪽, 아래 좌표를 $(B_{ll.x}, B_{ll.y})$, 오른쪽,

위의 좌표를 $(B_{ur.x}, B_{ur.y})$라 하면 가로, 세로 길이 $B_{\Delta x}$, $B_{\Delta y}$는 아래의 식(4, 5)과 같다. 마찬가지로 B'의 왼쪽, 아래 좌표는 $(B'_{ll.x}, B'_{ll.y})$, 오른쪽, 위 좌표를 $(B'_{ur.x}, B'_{ur.y})$라 할 수 있고, 또한 가로, 세로 길이 $B'_{\Delta x}$, $B'_{\Delta y}$는 식(6, 7)과 같다.

$$B_{\Delta x} = B_{ur.x} - B_{ll.x} \quad \text{-----} (4)$$
$$B_{\Delta y} = B_{ur.y} - B_{ll.y} \quad \text{-----} (5)$$
$$B'_{\Delta x} = B'_{ur.x} - B'_{ll.x} \quad \text{-----} (6)$$
$$B'_{\Delta y} = B'_{ur.y} - B'_{ll.y} \quad \text{-----} (7)$$

그림 8은 그림 7과 같은 격자 변환에 따른 위치점 변환을 알고리즘으로 나타낸 것이다.

알고리즘 격자 변환에 따른 위치점 변화

입력 B_{ll}, B_{ur} 격자 B의 양 끝 점
B'_{ll}, B'_{ur} 격자 B'의 양 끝 점
$p(p.x, p.y)$ 격자 B 내의 위치점
출력 $p'(p'.x, p'.y)$ 격자 B' 내의 위치점

알고리즘 시작

// 격자 B의 가로, 세로 길이
$B_{\Delta x} = B_{ur.x} - B_{ll.x}$
$B_{\Delta y} = B_{ur.y} - B_{ll.y}$
// 격자 B'의 가로, 세로 길이
$B'_{\Delta x} = B'_{ur.x} - B'_{ll.x}$
$B'_{\Delta y} = B'_{ur.y} - B'_{ll.y}$

// 격자 B' 내의 위치점 $(p'.x, p'.y)$

$$p'.x = \frac{(p.x - B_{ll.x}) \cdot B'_{\Delta x}}{B_{\Delta x}} + B'_{ll.x}$$

$$p'.y = \frac{(p.y - B_{ll.y}) \cdot B'_{\Delta y}}{B_{\Delta y}} + B'_{ll.y}$$

알고리즘 끝

그림 8. 격자 변환에 따른 위치점 변환 알고리즘

그림 8에서 보는 바와 같이 하나의 분할된 영역 질의는 STR 격자 정보와 격자 변환식에 의해 쉽게 변형될 수 있다.

3.4 질의 처리 성능 예측

공간 데이터베이스에서 공간 질의 처리는 고비용의 작업이다. 이 때, 질의 처리 순서를 최적화하면 질의 처리 비용을 줄일 수 있다. 질의 처리 순서 최적화를 위해서 필요한 것이 질의 처리 성능 예측이며 이를 위해서 많은 연구가 있다[10, 12, 13, 14].

본 논문에서는 고정 격자 방법에서의 질의 처리 성능 예측 모델을 제시한다. 고정 격자 방법에서 질의 처리 비용은 디스크 참조 횟수라고 할 수 있으며, 고정 격자 방법에서 디스크 참조 수는 그림 9에서 보듯이 주어진 질의가 겹쳐지는 격자의 수와 같다. $S = S_1 * S_2$이며 $S_i : [0,1)$이다. B는 균등 분포에 가깝게 변환된 데이터를 고정 격자 방법으로 저장하였을 때 하나의 격자, 즉, 디스크 한 블록이다. 그리고 q_s는 질의의 한 변의 길이를 말하며 질의는 정사각형이다.

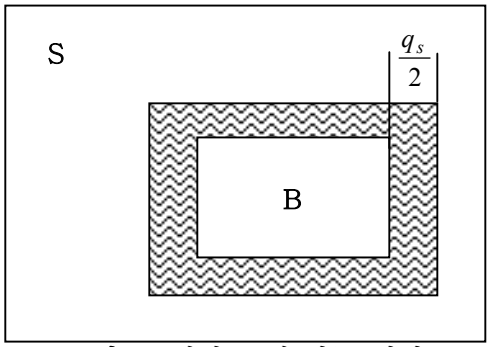


그림 9. 질의 존재 가능 영역

그림 9에서 B는 한 개의 격자이고, q_s에 겹쳐지게 되는 격자의 개수는 S_i의 격자의 개수(n_i)와 질의 크기의 곱이 된다. S의 한 변 S_i의 격자의 개수 n_i는 식(8)과 같이 구할 수 있다.

$$n_i = \left(\frac{N}{Bf_{\max} \cdot \alpha} \right)^{\frac{1}{2}} \quad \text{-----} (8)$$

S의 전체 격자의 개수는 식(8)에서와 같이 전체 데이터의 개수(N)에 한 디스크 블록의 평균 객체 수로 나누어 구할 수 있다. $Bf_{\max}$는 한 디스크 블록에 들어갈 수 있는 최대 객체 수이며, α는 실제 디스크 공간 활용도이다.

길이 q_s와 같은 고정 격자의 개수(n_{q_s})는 S_i의 격자의 개수와 질의 크기의 곱이므로 식(9)와 같이 구할 수 있다.

$$n_{q_s} = \left(\frac{N}{Bf_{\max} \cdot \alpha} \right)^{\frac{1}{2}} \cdot q_s \quad \text{----} (9)$$

고정 격자 방법에서 질의 존재 가능 영역에 있는 디스크 블록의 수가 곧 디스크 참조 횟수이므로, 고정 격자 방법의 성능은 다음과 같이 예측할 수 있다.

$$DA = \left(1+\left(\frac{N}{Bf_{\max}\cdot\alpha}\right)^{\frac{1}{2}}\cdot q_s\right)^2 \qquad \text{-----} (10)$$

위 식에서 N과 $Bf_{\max}$의 값은 정해져 있는 값이므로, 질의 크기가 클 때는, 디스크 공간 활용도가 디스크 접근 횟수에 영향을 많이 미치고, 질의 크기가 작아지면 디스크 참조 횟수가 거의 1에 가까워 진다는 것을 예상할 수 있다.

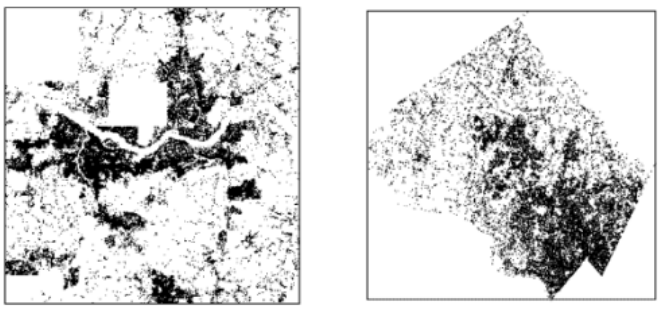
4. 성능 평가

비균등 분포의 데이터를 균등 분포의 데이터로 변환한 뒤, 고정 격자 색인 방법을 이용하여 실험하였다. 성능 평가를 하기 위한 환경, R*-tree와의 비교 그리고 질의 처리 예측과 그 결과에 대해서 알아보도록 한다.

4.1 성능 평가 데이터 및 환경

본 논문에서는 고정 격자 방법의 성능 평가를 하기 위해 두 개의 실제 데이터와 한 개의 합성 데이터를 사용하는데, 두 가지 실제 데이터는 그림 15와 같다. 합성 데이터는 10^6개 점으로 이루어져 있다.

본 논문의 실험에서 고정 격자 방법을 실험 할 때, 스케일 정보를 주기억 장치에 저장하여 실험하고, 이 때, 스케일 정보를 저장하기 위한 주기억 장치의 크기는 디스크의 한 블록의 크기와 같은 4Kbytes로 하였다. 스케일 정보의 양이 4Kbytes를 넘으려면 데이터의 개수가 최소 349,184개를 넘어야 한다. 그래서 합성 데이터로 10^6개의 점 객체를 가지는 데이터를 만들었다.



(a) 서울 데이터 (b) 몽고메리카운티 데이터

그림 10. 성능 평가 데이터

그림 10-a는 서울 지역의 점 데이터(68,736)이며, 그림 10-b에 비해 데이터가 균등하게 분포하고 있으나, 중앙에 다소 집중되어 있으며 몇 군데는 군사 지역으로 데이터가 없어 완전한 균등 분포 데이터로는 보기 힘든 데이터이다. 이에 반해, 그림 10-b는 미국 몽고메리카운티 지역의 점 데이터(27,282개)이며, 중앙 부분에 데이터가 집중되어 있으며 외곽 지역에는 데이터가 전혀 없는 대표적인 비균등 분포 데이터이다. 10^6개의 합성 데이터는 비균등 분포의 데이터로 객체가 너무 많아 그림으로 그렸을 때, 전체 데이터 영역에 데이터가 있는 것처럼 보여 그림을 첨부하지 않았다.

고정 격자 방법과 성능 비교를 하기 위해서 본 논문에서는 대표적인 공간 색인 방법인 R*-tree를 사용한다[4]. R*-tree는 현재 가장 많은 응용 프로그램에서 사용되고 있으며, 많은 연구가 진행되어 여러 분야에서 뛰어난 성능을 보이고 있다. 본 논문에서는 비균일 분포의 데이터를 STR을 이용하여 균일 분포로 변환한 뒤, 고정 격자 공간 색인 방법을 적용하여 R*-tree와 성능을 비교할 것이다.

본 성능 평가에서는 공간데이터의 색인 방법으로 R*-tree를 사용할 경우 색인 정보를 저장할 파일이 필요하다. 이 색인 정보를 중간 노드와 단말 노드로 구분하여 디스크 저장 용량과 디스크 참조 횟수를 비교할 것이고, 고정 격자 정보를 구분하여 R*-tree와 비교할 것이다.

4.2 R*-tree와 고정 격자 방법의 비교

표 1은 디스크 블록 크기를 4Kbytes로 정하고, 서울 데이터와 몽고메리카운티 데이터, 합성 데이터를 사용하여, R*-tree와 고정 격자에서 필요로 하는 디스크 공간을 알아보기 위한 실험 결과이다.

(단위 : byte)

데이터	R*-tree			고정격자		
	중간 노드	단말 노드	전체 크기	스케일	격자	전체 크기
서울 데이터 (점 68,736개)	16,384	2,007,040	2,023,424	2,951	1,484,820	1,487,771
몽고메리카운티데이터 (점 27,282개)	12,288	1,015,808	1,028,096	1,065	5,563,499	5,564,564
합성 데이터 (점 1,000,000개)	184,320	27,447,296	27,631,616	12,626	66,750,419	66,763,045

표 1. 데이터에 따라 필요한 저장 공간

표 1는 서울 데이터와 몽고메리카운티 데이터에서 R*-tree의 색인정보의 크기가 1Mbytes에서 2Mbytes인데 비해서, 고정 격자의 스케일 정보의 크기는 R*-tree의 중간노드의 크기의 최대 18% 정도로 주기억장치에 저장할 수 있을 정도의 작은 크기임을 보여주고 있다. 그러므로, 이 이후의 성능 비교에서 고정 격자 방법의 스케일 정보를 주기억장치에 저장하여 R*-tree의 색인정보 참조 수와 고정 격자 방법의 격자 정보의 참조 수 만을 디스크 참조 수로 비교한다.

표 2, 3, 4는 디스크 블록 크기를 4Kbytes로 정하고, 각 데이터에 대하여 1000개의 균등 분포 질의와 데이터 분포 질의를 수행하였을 경우의 서울 데이터, 몽고메리카운티 데이터, 합성 데이터의 디스크 블록 참조 수를 나타내고 있다. 고정 격자 방법의 실험에서는 스케일 정보 만을 주기억 장치에 저장하여 실험하였으며 스케일 정보를 저장하기 위한 주기억 장치의 크기를 디스크의 블록 크기와 같은 4Kbytes로 하였다. 표 2, 3, 4의 마지막 열인 B/A는 R*-tree를 이용하였을 때의 디스크 방문 횟수와 고정 격자를 이용하였을 때의 디스크 방문 횟수를 비교하기 위해서, 고정 격자 방식을 이용하였을 때의 디스크 방문 횟수

를 R*-tree를 이용하였을 때의 디스크 방문 횟수로 나누어 준 값이다.

서울 데이터(점 68,736개)					
질의크기 (길이)	R*-tree			고정격자	B/A
	중간노드	단말노드	합계(A)	격자(B)	
10^{-1}	1.173	8.672	9.845	5.494	0.558
10^{-2}	1.026	1.232	2.258	1.286	0.570
10^{-3}	1.007	0.836	1.843	1.024	0.557
10^{-4}	1.013	0.790	1.803	1.000	0.555
10^{-5}	1.009	0.813	1.822	1.000	0.549

표 2. R*-tree와 고정 격자 방법의 디스크 참조 수
서울 데이터, 균등분포질의

몽고메리 데이터(점 27,282개)					
질의크기 (길이)	R*-tree			고정격자	B/A
	중간노드	단말노드	합계(A)	격자(B)	
10^{-1}	1.222	5.157	6.379	3.055	0.479
10^{-2}	1.131	0.725	1.856	1.150	0.620
10^{-3}	1.107	0.526	1.633	1.009	0.618
10^{-4}	1.111	0.495	1.606	1.003	0.625
10^{-5}	1.099	0.514	1.613	1.000	0.620

표 3. R*-tree와 고정 격자 방법의 디스크 참조 수
몽고메리카운티 데이터, 균등분포질의

합성 데이터(점 1,000,000개)					
질의크기 (길이)	R*-tree			고정격자	B/A
	중간노드	단말노드	합계(A)	격자(B)	
10^{-1}	2.641	80.802	83.443	52.080	0.589
10^{-2}	1.255	3.468	4.723	2.787	0.590
10^{-3}	1.146	1.322	2.468	1.128	0.457
10^{-4}	1.119	1.136	2.255	1.021	0.453
10^{-5}	1.132	1.114	2.246	1.002	0.446

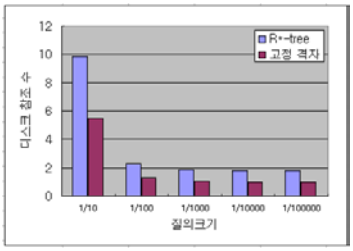
표 4. R*-tree와 고정 격자 방법의 디스크 참조 수
합성 데이터, 균등분포질의

표 2, 3, 4를 보면 스케일 정보만 주 기억장치에 있을 때, 고정 격자 방법의 디스크 참조 수가 R*-tree의 디스크 참조 수의 약 절반이 되는 것을 알 수 있다.

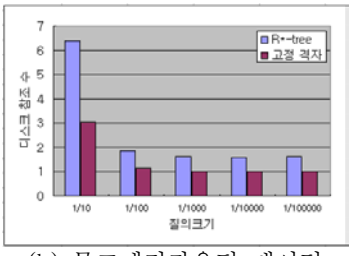
또한, 고정 격자 방법의 성능이, 몽고메리카운티 데이터, 서울 데이터, 합성 데이터 순으로 몽고메리카운티 데이터에서 가장 나쁜 성능을 보이는 것을 알 수 있다. 이것은 몽고메리카운티 데이터에는 사장 공간(dead space)이 많아서, 이 사장 공간이 최소 영역 사각형(MBR, Minimum Bounding Rectangle)을 사용하는 R*-tree에 유리한 요소로 작용하고 있기 때문이다.

그리고 표 2, 3, 4 모두에서 질의 크기가 10^{-4}이하가 되면 고정 격자 방법에서 디스크 접근 횟수가 거의 1이 되는 것을 볼 수 있다. 이 이유는 질의 크기가 점점 작아져 질의가 거의 하나의 격자에 걸쳐지기 때문이고, 고정 격자 방법이 중간 노드 구조를 가지지 않기 때문에 결국 하나의 디스크 블록을 방문하게 되기 때문이다.

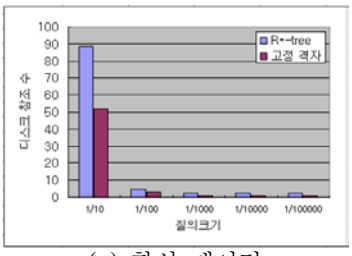
그림 11은 고정 격자의 스케일 정보만을 주기억장치에 저장하여 실험한 결과를 도표로 나타낸 것이다.



(a) 서울 데이터



(b) 몽고메리카운티 데이터



(c) 합성 데이터

그림11. 균등 분포 질의에 대한
R-tree와 고정 격자 방법의 디스크 접근 횟수 비교

이 때, 합성 데이터인 경우에 R*-tree를 이용한 방법보다 고정격자 방법을 이용한 경우가 더 좋은 성능을 보이며 질의 크기가 더 작아질 수록 고정 격자 방법의 성능이 점점 더 좋아지는 것을 볼 수 있다. 그림 11에서 보듯이, 고정 격자 방법의 디스크 참조 수는 R*-tree의 디스크 참조 수의 평균 55%이다.

4.3 고정 격자 방법의 질의 처리 예측과 결과

식(10)에서 보듯이 고정 격자 방법의 질의 처리 예측 모델에서 디스크 공간 활용도는 질의 처리 성능에 중요한 요소로 작용한다. 각 데이터의 디스크 공간 활용도를 조사해 보았더니 표 6과 같았다.

디스크 공간 활용도	서울 데이터	몽고메리 데이터	합성 데이터
최적	0.895875	0.987727	0.969438
실제	0.895875	0. 987727	0.615953

표 5. 디스크 공간 활용도

표 5에서의 최적 디스크 공간 활용도란 $N/Bf_{\max}$

개의 디스크 블록에 각 데이터를 골고루 나누어 넣었을 때의 디스크 공간 활용도이다. 그리고 실제 디스크 공간 활용도는 STR을 이용하여 데이터의 균일한 정도를 높이고 고정 격자 방법으로 디스크에 저장 하였을 때의 실제 디스크 공간 활용도이다.

표 5를 보면 몽고메리카운티 데이터, 서울 데이터, 합성 데이터 순으로 디스크 공간 활용도가 좋아지고 있다. 서울 데이터가 가장 좋은 이유는 서울 데이터는 점 데이터 68,736개로 이루어져 있기 때문에 스케일 정보의 개수를 최적의 값으로 균일 분포로 만들 수 있었기 때문이다. 그리고 몽고메리카운티 데이터역시 최적의 스케일 정보의 개수로 균일 분포화 하였지만 원래 데이터가 비균일 정도가 심해 균일 분포화 성능이 좋지 않기 때문이다. 마지막으로 합성데이터는 점 객체 10^6개로 이루어져 있는데 데이터가 너무 많아 스케일 정보를 디스크 한 블록의 크기인 4Kbytes까지 밖에 만들 수 없었기 때문이다.

표 6은 식(10)로 구한 고정 격자 방법으로 실험하였을 때의 균등 분포 질의 처리 성능(디스크 접근 횟수)을 예측한 값과 실험한 값으로 나타낸 표이다.

질의 크기	서울 데이터		몽고메리 데이터		합성 데이터	
	예측값	실험값	예측값	실험값	예측값	실험값
10^{-1}	6.250	5.494	7.840	3.055	57.775	52.080
10^{-2}	1.323	1.286	1.392	1.150	2.756	2.787
10^{-3}	1.030	1.024	1.036	1.009	1.136	1.128
10^{-4}	1.002	1.000	1.004	1.003	1.014	1.021
10^{-5}	1.000	1.000	1.000	1.000	1.000	1.002

표 6. 질의 처리 성능의 예상값과 실험값

표6에서 예측값이 평균 13%의 오차를 보인다. 서울 데이터에서 예측값과 실험값이 평균 3.5%의 오차를 보이고, 합성 데이터에서 평균 1.9%의 오차를 보이는데 비해서, 몽고메리카운티 데이터는 평균 36%, 최대 156%의 오차를 보여 최악의 결과를 보여준다. 이 이유는 질의 크기가 변형되기 전의 질의 크기값을 비용 모델에 사용하였고, 데이터의 편중도가 심하여 디스크 활용도가 낮았기 때문이다.

5. 결론

본 논문에서는 고정 격자 방법에 의한 공간 색인 방법의 단순성과 효율성을 이용하기 위하여, 비균일적인 공간 객체를 균일적인 공간 객체의 분포로 전환하여 기존의 고정 격자 방법의 공간 색인을 효과적으로 이용하는 방법을 제안하였다. 또한 이 방법을 이용하여 질의 처리를 할 때의 비용 모델을 제시하고 실험과 비교하여 보았는데 비용 모델과 실험 결과가 평균 13%의 오차를 보여 정확성을 입증하였다.

이 방법은 다른 종류의 공간 색인 방법과 달리, 색인 구조를 위한 디스크 접근이 거의 필요 없고, 대부분의 디스크 참조가 데이터를 읽기 위한 디스크 접근에 필요하므로 성능이 100%에서 200% 정도 향상된다. 또한 색인 방법이 단순하여, 구현 및 기존의 데이터베이스 관리 시스템에 통합하는 작업이 간편하여 진다.

그러나, 이 방법은 점 객체에 대하여서만 적용이 가능하다. 또한 아직까지는 2차원 공간의 객체에 대해서만 적용을 하였다. 따라서, 이를 비점객체와 3차원 이상의 다차원 공간의 객체에 적용하는 연구가 계속하여 이루어져야 한다. 그리고 고정 격자 방법의 스케일 정보가 커질 겨우, 효율적인 디스크 저장 방법에 대한 연구도 계속되어야 한다.

6. 참고문헌

[1] Antonin Guttman, “R-trees: a dynamic index structure for spatial searching.”, Proc. ACM SIGMOD Conf., pp.47-57, June 1984

[2] Timos Sellis, Nick Roussopoulos and Christos Faloutsos, “The R+ -tree: a dynamic index for multi-dimensional objects.”, Proc. VLDB Conf., pp507-518, England, September 1987

[3] Ibrahim Kamel and Christos Faloutsos, “Hilbert R-tree: An Improved R-tree Using Fractals”, Proc. VLDB Conf., pp.500-509, 1994

[4] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, “The R*-tree: An Efficient and Robust Access Method for Points and Rectangles”, Proc. ACM SIGMOD Conf., pp.322-331, 1990

[5] Jurg Nievergelt and Hans Hinterberger, “The Grid Files: An Adaptive, Symmetric Multikey File Structure”, ACM TODS, 9(1), pp.38-71, 1984

[6] Henk M. Blanken, Alle IJbema, Paul Meek and Bert van den Akker, “The Generalized Grid File: Description and Performance Aspects.”, Proc. ICDE, pp.380-388, 1990

[7] Andreas Hutflesz, Hans-Werner Six and Peter Widmayer, “Twin Grid Files: Space Optimizing Access Schemes”, Proc. ACM SIGMOD Conf., pp.183-190, 1983

[8] Michael Freeston, “The BANG file: a new kind of grid file”, Proc. ACM SIGMOD Conf., pp260-269, 1987

[9] Christos Faloutsos and Ibrahim Kamel, “Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension.”, Proc. PODS, pp.4-13, 1994

[10] Ibrahim Kamel and Christos Faloutsos, “On Packing R-trees”, CIKM, pp.490-499, 1993

[11] Scott T. Leutenegger, Mario A. Lopez and Jeffrey Edgington, “STR: A Simple and Efficient Algorithm for R-Tree Packing”, Proc. ICDE, pp.497-506, October 1997

[12] Bernd-Uwe Pagel, Hans-Werner Six, Heinrich Toben, and Peter Widmayer, “Towards an Analysis of Range Query Performance in Spatial Data Structures.”, Proc. PODS, pp.214-221, 1993

[13] Yannis Theodoridis and Timos Sellis, “A Model for the Prediction of R-tree Performance”, Proc. PODS, pp.161-171, 1996
[14] Christos Faloutsos, Timos Sellis and Nick Roussopoulos, “Analysis of Object Oriented Spatial Access Methods”, Proc. ACM SIGMOD Conf., pp.426-239, 1987
[15] Bernhard Seeger and Hans-Peter Kriegel, “The Buddy-Tree: An Efficient and Robust Access Method for Spatial Data Base Systems”, Proc. VLDB Conf., pp.590-601, 1990