論文98-35C-1-4

공간 데이타베이스에서 이동 객체의 탐색기법

(A Search Mechanism for Moving Objects in a Spatial Database)

柳 秉 玖 *, 黃 壽 贊 **, 白 重 煥 ***

(Byung Gu Yu, Soochan Hwang, and Joong Hwan Baek)

요 약

본 논문에서는 다차원 공간 데이타베이스에서 연속적으로 이동하는 객체를 포함하는 공간 객체를 신속히 탐색할 수 있는 탐색 알고리즘을 제시한다. 제안한 탐색 방법은 기존 R-트리의 탐색기법을 객체가 이동하는 경우에 적합하도록 개선한 것으로서 매번 루트 노드로부터 탐색을 시작하는 대신 현재 노드를 기준으로 탐색을 수행하도록 하였다. 즉 현재 노드의 주변 노드들부터 먼저 탐색을 하도록 함으로써 대부분의 탐색에서 동일 노드내의 다른 엔트리 또는 형제 노드에서 탐색을 완료할 수 있도록 하는 것이다. 제안한 탐색기법은 성능평가 결과 탐색 윈도 혹은 탐색을 요청하는 객체가 연속적으로 이동하는 경우 기존 탐색 방법 보다 훨씬 효율적인 성능을 보였다.

Abstract

This paper presents an algorithm for searching an object in a fast way which contains a continuous moving object in multi-dimensional spatial databases. This algorithm improves the search method of R-tree for the case that a target object is continuously moving in a spatial database. It starts the searching from the current node instead of the root of R-tree. Thus, the algorithm will find the target object from the entries of current node or sibling nodes in the most cases. The performance analysis shows that it is more efficient than the existing algorithm for R-tree when search windows or target objects are continuously moving.

I. 서 론

다차원 공간 데이타는 2차원 평면이나 3차원의 육면체 등과 같이 공간적인 정보를 표현하는 데이타를 의미한다. 이러한 데이타를 취급하는 CAD나 지형 정보시스템과 같은 비정형화된 데이타베이스의 화일처리에는 기존 화일기법과는 달리 기본키와 보조키를 구분하지 않고 이들을 대칭적으로 취급하여 같은 수준의 검색을 요구하는 경우가 빈번하다. 즉, 다차원 데이타는 최소한 그 차원 수만큼의 탐색키를 갖게 되는 것이다. 예를 들어 3차원의 객체를 탐색하기 위해서는 X, Y, Z축에 대한 탐색조건을 필요로 한다. 현재 다차원 공간 데이타를 다루는 응용은 우주항공산업, 로보트, 디자인분야 등과 특히 최근 각광을 받고 있는 지리정보처리시스템(GIS)등 매우 광범위한 분야에 적용되고 있으며 따라서 다차원 공간데이타를 취급하기 위한 많은 화일처리 기법들이 활발히 연구되고 있다[1,2,3,5]. 다차원 화일 기법으로 대표적인 것들로는 K-D 트리,

* 學生會員, ** 正會員, 韓國航空大學校 컴퓨터工學科
(Dept. of Telecommunication and Information Eng.)
*** 正會員, 韓國航空大學校 航空通信情報工學科
(Hankuk Aviation University Department of Computer Engineering)

接受日字:1997年11月17日, 수정완료일:1998年1月3日

K-D-B 트리, 격자화일, R-트리 등을 들 수 있다.

K 차원의 트리를 의미하는 K-D 트리는 기본키에 의한 검색과 보조키에 의한 검색을 하나의 구조로 결합한 것으로 기본키와 보조키로 구성된 K개의 키를 이용해 범위탐색이 가능하다[3]. 예를 들어 2-D 트리인 경우 X축의 범위와 Y축의 범위를 결합하여 2차원 평면 사각형 범위의 탐색을 수행할 수 있다.

K-D-B 트리는 다중키 레코드를 효율적으로 검색하기 위한 기법으로 K-D 트리와 B-트리의 특징을 결합한 구조이다. 즉, 노드수가 고정된 다중트리로서 완전균형트리이다[3].

다중키 화일구조인 격자화일은 각 키를 축으로한 공간상에서 격자분할이라는 분할법을 사용하는데 모든 축을 동등하게 돌아가면서 각 축에 해당하는 키값에 의해 공간을 분할하는 화일기법이다. 이 기법은 각 공간 객체를 고차원 공간의 점으로 사상하는 기법으로 2차원상의 두점 (x_1, y_1), (x_2, y_2)를 4차원 공간상의 한점 (x_1, y_1, x_2, y_2)로 표현하게 된다[3,5].

R-트리는 B-트리와 유사하게 인덱스 레코드들로 구성되는 높이균형 트리로서 단말노드들이 데이타 객체에 대한 포인터를 포함하고 있다. R-트리는 완전동적 인덱스이므로 삽입과 삭제가 탐색과 같이 행해질 수 있고 주기적인 재구성도 필요치 않다. 각 내부노드는 (자식의 공간범위, 자식포인터)를 자식에 대한 인덱스로서 자식의 수만큼 유지한다. 또한 모든 단말노드는 (객체의 공간범위, 포인터)를 각 공간 객체에 대한 인덱스로서 유지하고 있다[1,2,3].

그러나 이러한 화일기법들은 주어진 탐색 윈도에 포함되는 객체들을 신속히 탐색하기 위한 비연속적이고 정적인 탐색기법으로서, 탐색 윈도나 탐색 대상객체가 계속적으로 이동하는 응용에 적용한다면 적절한 성능을 보장하기가 매우 어렵게 된다. 이러한 응용으로는 항공기 시뮬레이터의 항법 지리정보 탐색이나 이동통신분야, 교통정보시스템 등 이동하는 물체와 고정시설 사이에 정보교환을 필요로 하는 모든 분야가 해당될 것이다.

예로써 항공기 시뮬레이터의 경우를 살펴보기로 하자. 항공기는 비행중 공항이나 관제탑, 지상관제소 등과 계속적으로 통신을 하며 비상시를 대비하여 가까운 공항이나 비상활주로 등의 위치를 계속적으로 파악을 하고 있어야 한다. 그러나 지상관제소는 항공기와 통신이 가능한 범위를 가지고 있으므로 항공기가 이동함에 따라 교신이 가능한 관제소가 계속적으로 변하게 된다. 즉, 이 경우 항공기가 탐색 대상객체가 되며 지상관제소의 관제 범위가 공간 데이타 객체로 표현될 수 있는 것이다.

따라서 본 논문에서는 다차원 공간데이타베이스에서 연속적으로 이동하는 객체를 포함하는 공간 객체를 신속히 탐색할 수 있는 공간탐색 알고리즘을 제시하고자 한다. 본 논문에서 제시한 탐색 알고리즘은 R-트리를 기본 구조로 이용하면서 본 대학에서 개발한 항공기 시뮬레이터의 구성요소인 항법지리 데이타베이스 시스템에 실제로 적용한 것이다.

본 논문의 구성은 2장에서 항공기 시뮬레이터의 개요를 설명하고 3장에서는 새로운 탐색 알고리즘을 제시한다. 그리고 4장에서는 새로운 알고리즘의 성능을 분석하고 끝으로 5장에서 결론을 맺는다.

Ⅱ. 항공기 시뮬레이터

이 장에서는 다차원 공간 데이타베이스에서 이동 객체에 대한 실시간 탐색의 응용으로서 항공기 시뮬레이터의 구성요소인 항법지리(navigational geography) 데이타베이스 시스템을 소개하기로 한다.

항공기 시뮬레이터란 실제 항공기와 똑같은 상황 및 동작으로 지상에서 비행훈련을 실시할 수 있는 장치이다. 실제 항공기에 의한 비행훈련의 경우 훈련자의 비행기술 미숙으로 인한 사고의 위험이 따르고, 비행훈련에 따른 인적, 물적 경비가 많이 드는 단점이 있다. 그러나, 실제 항공기에 의한 본격적인 비행훈련전에 항공기 시뮬레이터를 사용하여 훈련할 경우, 훈련자가 그 비행기에 대해 사전숙지가 가능하고, 또한 비행훈련 경비 및 시간을 감축시킬 수 있는 장점을 갖는다[4].

항공기 시뮬레이터에서 항법에 관련된 기본적인 기능은 조종사로부터 항법시설의 주파수를 입력받아서 그 주파수를 갖는 지상관제소를 연결하여 주고 필요한 경우 각도와 거리를 계산하여 각종 계기를 제어하는 것이다. 이때 지상 관제소는 항공기로부터 통신이 가능한 범위 내에 위치해야만 연결된다. 이 기능을 수행하기 위한 수행과정은 그림 1에 제시되어 있다. 새로운 관제소를 연결하기 위해 조종사가 주파수를 입력하는 경우도 있지만 항공기의 위치는 계속적으로 변하므로 항공기의 현위치에서 통신이 가능한지를 계속 검사하기 위해서는 항공기의 위치가 바뀔 때마다 그림 1의

순서도 과정을 수행해야 한다. 이때 지상관제소의 정보는 항법지리 데이타베이스로 구성되어 있으며 항공기와 통신 가능한 관제소를 찾기 위해 데이타베이스를 탐색해야 한다.

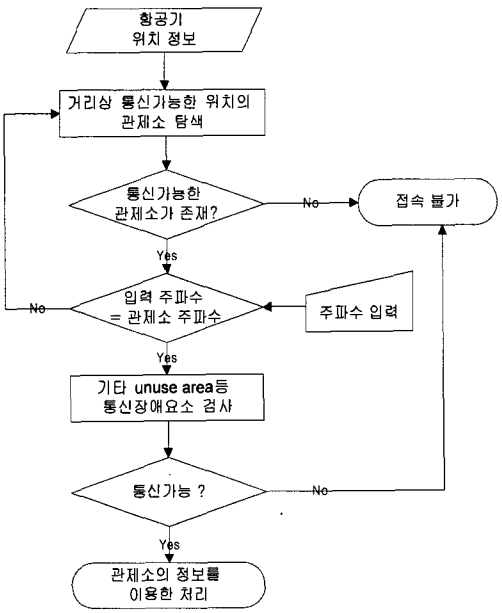


그림 1. 항공기 시뮬레이터의 기초기능 수행과정.
Fig. 1. Procedure for the basic functions of flight simulator.

항공기 시뮬레이터를 위한 항법지리 정보로는 비행에 따른 위치정보와 항공기를 통제하기 위한 지상관제소의 위치정보 및 관제범위, 그리고 항공기의 전파 수신범위 등의 정보가 필요하다. 이러한 정보들을 데이타베이스화 하기 위해서는 공간데이타를 처리할 수 있는 화일기법이 필요하며 항공기 시뮬레이터의 특성상 실시간 탐색이 요구된다[4]. 항공기와 지상 관제소와의 관련성은 아래 그림 2와 같이 나타낼 수 있다.

그림 2에서 실제 원으로 나타나는 지상관제소의 전파도달범위는 R-트리로 구성하기 위해 내접사각형으로 구성하며, 항공기의 위치는 이러한 R-트리 공간데이타에 대한 탐색 윈도로 이용된다. 항공기의 위치는 하나의 점으로서 표현 가능하므로 이 응용에서 탐색 윈도의 이동은 점의 이동이 된다. 하지만 항공기가 어떤 탐색범위를 가지며 이동한다면 탐색 윈도는 범위가 될 수도 있다. 여기서 연속적으로 이동하는 항공기의 위치에 의한 통신가능 관제소의 탐색은 기존 R-트리의 탐색 기법으로는 비효율적이므로 연속적으로 이동하는 항공기를 효율적으로 탐색하기 위한 알고리즘이 필요하다.

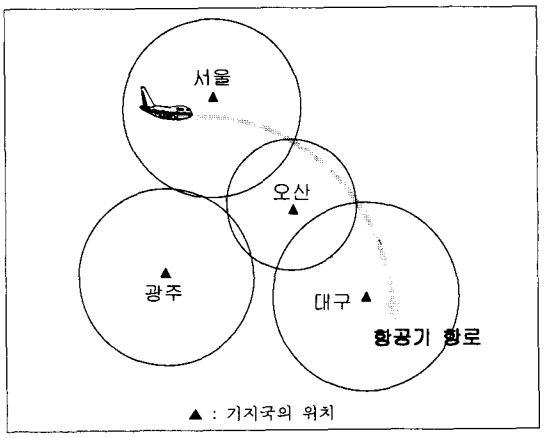


그림 2. 항공기와 지상 관제소와의 관련도
Fig. 2. A relation between an airplane and ground stations.

Ⅲ. 연속 이동 객체를 위한 R-트리 탐색 기법

1. R-트리

R-트리는 B-트리와 유사하게 인덱스 레코드들로 구성되는 높이균형트리로서 단말노드들이 데이타 객체에 대한 포인터를 포함하고 있다. R-트리의 구조는 공간 탐색시 적은 수의 노드들만을 방문하여 원하는 객체를 탐색할 수 있도록 설계되었다. 인덱스는 완전히 동적인 성질을 갖기 때문에 삽입, 삭제가 탐색과 같이 행해질 수 있고 주기적인 재구성도 필요치 않다. 2차원 공간 객체의 분포 예와 이를 R-트리로 표현한 예가 그림 3에 있다.

트리에서 한 노드에 저장 가능한 최대 엔트리 수를 M이라 하고 $m(\leq M/2)$은 엔트리의 최소 수를 명세하는 매개변수라 할 때 R-트리는 다음과 같은 성질을 만족한다[2,3].

① 모든 단말노드는 루트가 아닌 한 m과 M 사이의 인덱스 레코드들을 포함한다.

② 단말노드의 각 인덱스 레코드 (I,튜플식별자)에서 I는 튜플식별자가 가리키는 n-차원의 데이타 객체를 포함하는 최소 경계 사각형이다.

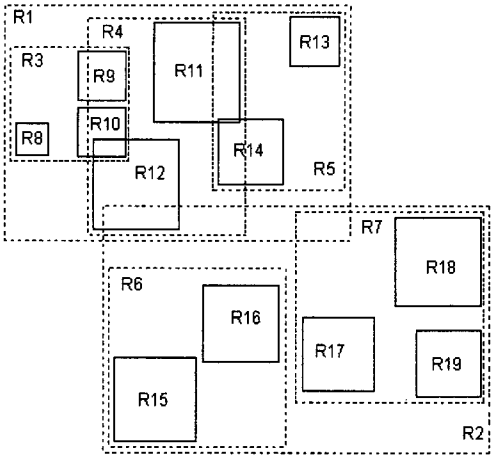
③ 단말노드가 아닌 모든 노드는 루트가 아닌 경우 m과 M사이의 자식노드들을 갖는다.

④ 단말노드가 아닌 노드의 각 엔트리(I, 자식포인터)에서 I는 자식노드들의 사각형을 공간적으로

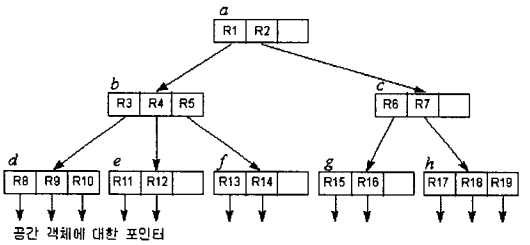
포함하는 최소의 사각형이다.

⑤ 루트노드는 단말노드가 아니라면 적어도 두 개의 자식노드를 갖는다.

⑥ 모든 단말노드는 동일한 레벨에 위치한다.



(a) 범위 사각형의 구조



(b) 트리 구조 및 탐색의 예

그림 3. R-트리 예
Fig. 3. An example of R-tree.

R-트리에서 검색은 B-트리와 유사하게 루트부터 트리를 탐색해 내려간다[2,3]. 인덱스 엔트리 E의 범위 사각형 부분을 E.I로 나타내며 엔트리 E의 튜플식별자 혹은 자식 포인터부를 E.p로 표현한다면 루트가 T인 R-트리에서 탐색점(탐색 윈도) S를 포함하는 공간 데이터 객체를 찾는 과정은 다음과 같다. 다음 알고리즘은 면적이 있는 객체의 검색 알고리즘[2]을 점 데이터에 대한 검색으로 수정한 것이다.

(1) 내부노드 탐색

- 노드의 각 엔트리 E에 대해 E.I가 S를 포함하는지 검사
- S를 포함하는 엔트리 E의 E.p가 가리키는 노드를 루트로 하는 서브트리에 대해 (1), (2)의 과정을 반복

(2) 외부노드 탐색 (단말노드 탐색)

- 노드의 각 엔트리 E에 대해 E.I가 S를 포함하는지 검사
- 포함한다면 E는 조건에 맞는 레코드를 가리키는 인덱스 엔트리이다.

2. 기존 R-트리 탐색기법의 문제점 및 개선 방안

만약 분기율이 m이고, 데이타 객체의 수가 N이라면 R-트리에서 하나의 객체를 탐색하는 시간은 R-트리의 높이($=\lceil \log_m N \rceil$)에 비례한다. 그러나 실제로 항공기 시뮬레이터의 경우 매단위 시간마다 항공기의 위치가 변경되고 매번 루트에서 리프까지의 탐색을 필요로 하므로 기존 탐색방법을 그대로 사용한다는 것은 매우 비효율적이다.

기존 R-트리 탐색기법의 경우 시간지연의 주된 요소는 탐색이 언제나 루트로부터 이루어진다는데 있다. R-트리는 균형트리[1,2,3]이므로 탐색은 항상 단말노드에서 종료하고 트리의 깊이가 깊어지면 탐색속도는 느려질 수밖에 없다. 항공기 이동의 특성상 매 단위 이동마다 탐색이 이루어지고 단위 시간 동안에는 일정한 짧은 거리를 이동하게 되므로 탐색의 목표는 공간적으로 이전 탐색의 결과에 인접해 있게 된다. 따라서 탐색결과는 동일 노드나 그 노드 주변의 형제 노드들 중에서 탐색될 확률이 아주 높다는 것은 R-트리의 특성상 자명한 것이다. 이러한 특성을 고려하여 탐색시에 인접한 노드부터 점차 멀리 있는 노드로 탐색해 나가는 방법을 이용하면 대부분의 경우는 인접 노드에서 탐색을 종료할 수 있을 것이다.

또한 아무리 주변 노드에서 발견될 확률이 높다고 하더라도 단위이동을 매우 작게 잡는다면 상당히 많은 탐색요구가 연속해서 들어오므로 약간의 지연시간도 매우 큰 오버헤드로 작용할 소지가 있다는 것이다. 이러한 경우 이동 객체가 어느 사각형에 들어오면 사각형과 이동단위의 크기에 따라 현재의 사각형의 범위 안에 머무르는 시간을 결정할 수 있다. 즉 "(사각형횡단길이/단위이동길이) = 이동 객체가 같은 사각형에 머무를 수 있는 단위이동 횟수" 라는 관계가 성립된다. 이 관계를 이용해서 만일 다른 특정 객체가 아닌 동일한 객체에 대해서는 매 단위 이동마다 데이타베이스를 탐색할 필요 없이 현재 속한 사각형이 이동 객체를 포함하는지만 검사하면 된다. 이것은 2차원의 경우 2회의 비교연산, 3차원의 경우 3회의 비교연산과

같이 차원수만큼의 비교연산만으로 충분하다.

이와 같은 개선방안들을 적용하여 R-트리에서 연속 이동 객체를 탐색하기 위한 과정을 표현하면 다음과 같다. 여기서 인덱스 엔트리 E의 범위 사각형 부분을 E.I로 나타내며 엔트리 E의 자식 포인터부는 E.p로 표현한다.

(1) 탐색점 S와 현재 엔트리 E의 E.I와 범위 비교 연산을 통해 포함관계를 조사
(2) S가 E.I에 여전히 포함되어 있다면 종료
(3) 아니면 S를 포함하는 인덱스 엔트리를 탐색
  (3.1) 현재 노드의 다른 엔트리에 속해 있으면 해당 엔트리 E를 반환후 종료
  (3.2) 아니면 부모 노드로 이동하여 S가 속해 있는 엔트리 E를 탐색
    (3.2.1) S를 포함하는 엔트리 E의 포인터부 E.p가 가리키는 자식 노드를 루트로하는 서브트리에 대해 탐색을 수행
    (3.2.2) S를 포함하는 단말노드의 엔트리 E를 찾으면 해당 엔트리 E를 반환후 종료
    (3.2.3) S를 포함하는 엔트리가 없다면 다시 부모 노드로 이동하여 (3.2)의 과정을 반복

앞에서 설명한 연속 이동 객체의 탐색 방법을 그림 4의 탐색점 이동 경로와 그림 3의 2차원 R-트리를 예로 설명하기로 한다.

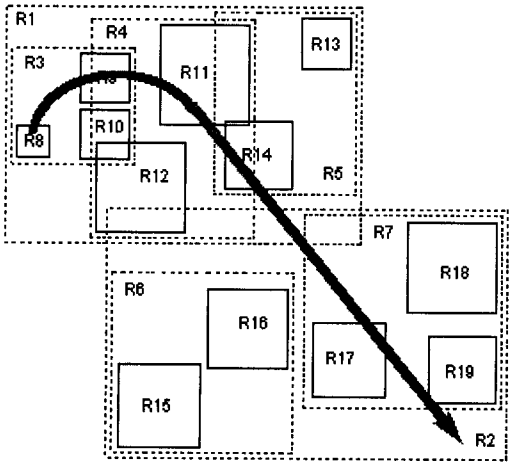


그림 4. R-트리 상에서의 탐색점 이동
Fig. 4. Movement of a searching point in R-tree.

```
Check_Range (position, E)
/* R-트리에서 탐색 위치 position이 현재 노드(전역 변수
   current_node에 의해 포인트)의 인덱스 엔트리 E의 사각형 범
   위에 포함되어 있는 지를 검사한다. 만약 인덱스 엔트리 E의 사
   각형 범위(E.I)를 벗어난 경우 함수 Search_Object를 호출하여
   position이 포함된 노드와 해당 인덱스 엔트리를 탐색한다. */
{
 if (position ∈ current_node->E.I)
 /* 현재 노드의 현재 엔트리 사각형 범위 내에 있는 경우 */
       return (E);
 else E = Search_Object(position);
       return (E);
 }

Search-Object(position)
/* R-트리에서 탐색 위치 position을 포함하는 노드와 해당 인덱
   스 엔트리를 탐색한다. 탐색은 현재 노드의 인접 엔트리에서부터
   시작하여 형제 노드들로 점차 탐색 범위를 넓혀 간다. */
{
 for each index entry E in current_node {
       if (position ∈ E.I) {
            entry_found = E; /* 탐색 위치 position을 포함하
            는 인덱스 엔트리 발견 */
            if (current_node ≠ leaf) {
            /* 발견한 엔트리가 단말노드에 속하지 않은 경우 해
               당 엔트리가 가리키는 서브트리로 탐색을 진행 */
                  current_node = E.p;
                  entry_found = Search_Object(position);
            }
            return (entry_found);
       } /* end of if (position .. */
 } /* end of for */
 /* 현재 노드의 엔트리에서 position을 포함하는 엔트리를 발견하
    지 못한 경우 현재 노드의 부모 노드로 이동하여 부모 노드의
    엔트리들을 탐색 */
 current_node = current_node->parent;
 entry_found = Search_Object(position);
 return (entry_found);
}
```

그림 5. R-트리에 대한 탐색 알고리즘
Fig. 5. A search algorithm of R-tree.

일반적인 R-트리 탐색의 경우 탐색점(연속적으로 이동하고 있는 질의자의 위치)이 노드 d의 공간 객체 R8의 범위내에 존재하는 상황에서 질의를 가했을 때 탐색은 루트 노드 a로부터 시작하여 노드 b, d의 순으로 검색해 나갈 것이다. 반면에 연속 이동 객체를 위한 탐색기법을 적용하면 R8의 범위 사각형 내에 탐색점이 존재하는지 x, y 좌표 2회의 비교연산으로 끝난다. 탐색점이 계속 이동하여 R8의 범위를 벗어나 R9의 범위로 들어간 경우도 기존 R-트리의 탐색 알고리즘은 마찬가지로 루트에서부터 탐색을 시작할 것이다

($a \rightarrow b \rightarrow d$). 하지만 연속 이동 객체를 위한 탐색기법의 경우 현재 노드 d의 인접 엔트리들만을 먼저 탐색함으로써 추가의 트리순회 없이 탐색을 끝낼 수 있다.

다시 탐색점이 R11로 이동한 경우는 현재 노드 d에서 해당 객체를 찾지 못하고 탐색은 노드 d의 부모 노드인 b로 진행하고 자식 노드인 e에서 해당 엔트리를 발견하고 탐색을 종료한다. 계속하여 탐색점이 경로를 따라 R11, R14, R17, R19로 이동함에 따라 R-트리 상에서의 노드 순회는 $e \rightarrow b \rightarrow f \rightarrow b \rightarrow a \rightarrow c \rightarrow h$와 같이 이루어진다.

이와 같은 실시간 탐색 기법을 구체적 알고리즘으로 표현하면 그림 5와 같다.

Ⅳ. 탐색 알고리즘의 성능평가 및 비교

1. 성능평가

성능평가를 위해 N개의 단말노드와 분기율 m을 가진 일반적인 R-트리에서 임의의 탐색점이 이동하는 경우를 고려하기로 한다. R-트리 상에서 데이타 객체들의 평균 지름을 r이라 하고 탐색점이 n회 이동한다고 하면 총 n회의 탐색 접근 요구가 발생한다.

(1) 기존 R-트리 탐색기법의 성능

R-트리에 대한 기존 탐색 방법은 루트 노드로부터 단말노드까지의 트리 경로를 따라 가며 탐색점을 포함하는 공간 데이타 객체를 탐색한다. 그러므로 탐색점이 이동할 때마다 트리 깊이만큼의 노드를 방문하게 되며 탐색 성능은 방문 노드의 수 즉, 트리의 깊이에 비례하게 된다. N개의 노드로 구성된 m 차원 R-트리에서 트리의 깊이 h는 $\lceil \log_m N \rceil$이다. 따라서 n 단위의 거리를 연속적으로 이동하는 데이타를 탐색하는 성능은 식 (1)과 같이 표현된다. 여기서 한 노드에서의 연산 시간을 의미하는 kc는 k 차원의 공간 범위와 점 데이타의 비교연산 시간들로 구성된다.

$$T_s = n \cdot kc \cdot \lceil \log_m N \rceil = n \cdot kc \cdot h \qquad (1)$$

(2) 개선된 탐색기법의 성능

개선된 탐색기법을 이용하면 탐색점이 현재 속한 사각형의 범위를 벗어나는지 검사하고 만약 벗어난 경우는 주변 노드에 대한 탐색을 시행한다. 탐색점은 임의의 위치로 이동할 수 있으므로 기존 사각형의 범위를 벗어나는 경우 현재 노드의 다른 엔트리로 갈 것인지 아니면 그 외의 다른 노드들로 갈 것인 지의 확률이 필요하다.

여기서 R-트리의 분기율이 m인 경우 탐색점이 속한 인덱스 엔트리를 포함한 현재 노드 이외의 다른 노드로 분기할 확률을 구하도록 한다.

단말노드에서 동일 노드내의 인덱스 엔트리 외의 다른 노드로 분기할 확률이 가장 높은 지점은 부모 노드가 표현하는 사각형의 모서리 4개 부분에 존재하고 나머지 경우는 부모 노드의 각 면에 존재한다. 그리고 부모 노드의 경계선에 인접하지 않고 있는 공간 범위의 경우는 다른 노드로 이동하지 않는다. 즉 현재 노드에 있는 엔트리 중의 하나로 이동하게 된다. 그림 6은 이 관계를 도식화한 것이다. 전파의 도달 범위는 원형이므로 공간 객체의 범위 사각형은 정사각형으로 간주한다.

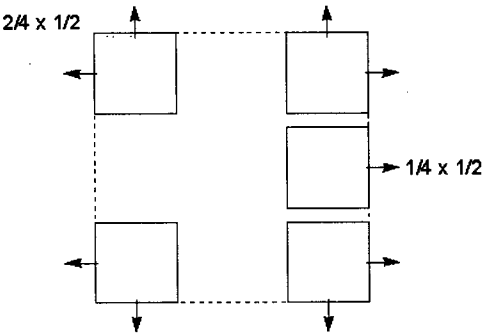


그림 6. 분기확률
Fig. 6. Branching probability.

그림 6에서 형제가 아닌 다른 노드로 분기할 확률인 P_{out}은 다음과 같이 구할 수 있다.

$$P_{out} = \frac{4}{m} \cdot \frac{1}{2} \cdot \frac{1}{2} + \frac{m-4}{m} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{m+4}{8m}$$

그리고 현재 노드의 다른 엔트리로 이동할 확률 P_{in}은 다음과 같다.

$$P_{in} = 1 - P_{out} = 1 - \frac{m+4}{8m} = \frac{7m-4}{8m}$$

이러한 확률 하에서 탐색점이 현재 포함된 사각형에서 벗어날 때 현재 노드의 다른 엔트리로 가는 경우와 다른 노드로 갈 경우는 앞의 확률을 이용하여 구할 수 있다. 먼저 현재 노드의 다른 엔트리로 이동하는 경우는 어느 엔트리로 이동했는 지를 알기 위한 범위 비교 연산 시간 kc가 필요하고, 현재 노드의 내부로 이동할 확률을 고려하면 이에 필요한 시간은 $P_{in} \cdot kc$가 된다.

다음으로 만약 같은 부모 노드의 형제 노드로 이동하는 경우를 고려하도록 한다. 먼저 탐색은 현재 노드에서 외부로 나가(P_{out}) 부모 노드로 이동하고 그 노드의 내부에 존재함을 확인한다(P_{in}). 즉, 부모 노드의 다른 엔트리에 포함되어 있음을 확인하는 것이다. 그리고 그 엔트리의 포인터를 따라 자식 노드로 이동하여 해당 엔트리를 발견하게 된다. 따라서 범위비교에 필요한 시간은 $3 \cdot kc$이고 확률을 고려하면 $P_{out} \cdot P_{in} \cdot 3 \cdot kc$가 된다. 이와 같은 방법으로 그 밖의 경우도 순환적으로 계산할 수 있다. 현재 노드 외에 다른 노드로 분기할 때 최악의 경우는 루트를 거치는 경로이다. 그러면 탐색점의 이동에 따라 현재 노드 이외의 다른 노드로 이동하는 경우 평균 처리시간은 식 (2)와 같이 구할 수 있다.

한번 사각형의 범위를 벗어날 때 평균 처리시간 :

$$\begin{aligned} & P_{in} \cdot kc + P_{out} \cdot (P_{in} \cdot kc \cdot 3 + P_{out} \cdot (P_{in} \cdot kc \cdot 5 + P_{out} \\ & \cdot (P_{in} \cdot kc \cdot 7 + \cdots + P_{out} \cdot (P_{in} \cdot kc \cdot (2h-1))) \cdots)) \\ & = \sum_{i=1}^{h} (P_{out})^{i-1} \cdot P_{in} \cdot kc \cdot (2i-1) \\ & = P_{in} \cdot kc \, [2 \sum_{i=1}^{h} i \cdot (P_{out})^{i-1} - \sum_{i=1}^{h} (P_{out})^{i-1}] \\ & = P_{in} \cdot kc \cdot [\frac{2(1-(P_{out})^h)}{(1-P_{out})^2} + \frac{(2h+1) \cdot (P_{out})^h - 1}{1-P_{out}}] \\ & = P_{in} \cdot kc \cdot [\frac{2(1-(P_{out})^h)}{(P_{in})^2} + \frac{(2h+1) \cdot (P_{out})^h - 1}{P_{in}}] \\ & = kc \cdot [\frac{2(1-(P_{out})^h)}{P_{in}} + (2h+1) \cdot (P_{out})^h - 1] \end{aligned} \quad (2)$$

개선된 탐색기법은 현재 속한 범위를 벗어나는 경우만 탐색이 시행되고 범위를 벗어나는 총 횟수는 n회 탐색시 n거리를 진행한다고 할 때 평균 $n/(r/2)$가 된다. 그 이유는 단말노드의 평균지름이 r일 때 평균 $r/2$정도의 거리를 진행하였을 때 경계선을 통과한다고 볼 수 있기 때문이다. 최초의 탐색이 $h \cdot kc$가 걸리고 매 이동마다 kc가 걸리므로 식 (3)과 같은 탐색 성능을 얻을 수 있다.

$$\begin{aligned} Ts &= \text{첫 노드 탐색 시간} + \text{사각형 내부 이동시간} + \frac{2n}{r} \cdot \text{(범위를 벗어난 경우 처리시간)} \\ &= h \cdot kc + n \cdot kc + \frac{2n}{r} \cdot kc \cdot [\frac{2(1-(P_{out})^h)}{P_{in}} + (2h+1) \cdot (P_{out})^h - 1] \end{aligned} \quad (3)$$

2. 성능 비교

기존 방법과 본 논문에서 제시한 탐색 성능을 비교하기 위해 비교 연산시간 kc를 1로 놓고 차원 k를 2차원으로 가정한다. 표 1은 분기율 m을 4로 하고 단말노드의 평균반지름 r을 20과 40으로 각각 가정했을 때 탐색회수(단위이동회수) n과 공간 객체의 수(단말노드수) N에 따른 탐색시간의 비교 결과를 제시한 것이다. 또한 탐색 성능을 구체적으로 보이기 위해 탐색회수(n), 데이타 객체의 수(N), 분기율(m), 객체 범위 반경(r) 값들이 변경되는 경우 성능의 변화를 그림 7에 그래프로 나타내었다. 분기율이 고정된 경우 데이타 객체의 수 N의 증감은 바로 트리의 높이와 연관되므로 그림 7(b)의 경우는 변수로 트리의 높이를 지정하였다.

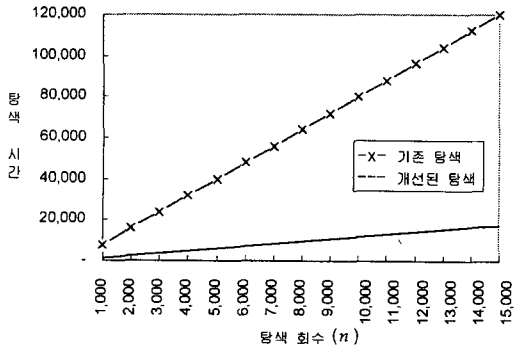
표 1. 기존 탐색기법과 개선된 탐색 기법의 비교

Table 1. Comparison between the existing search method and the improved method.

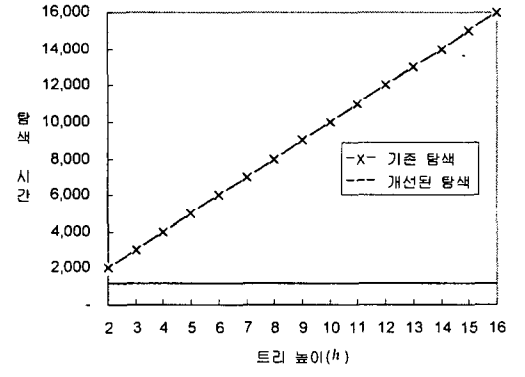
탐색 회수 (n)	평균 반경 (r)	객체 수 (N)	기존 탐색	개선된 탐색	비율
1,000	20	4^5	2,000	1,183	0.59
		4^6	4,000	1,173	0.29
		4^7	8,000	1,175	0.15
10,000	20	4^5	20,000	11,815	0.59
		4^6	40,000	11,695	0.29
		4^7	80,000	11,675	0.15
1,000	40	4^5	2,000	1,093	0.55
		4^6	4,000	1,089	0.27
		4^7	8,000	1,091	0.14
10,000	40	4^5	20,000	10,908	0.55
		4^6	40,000	10,850	0.27
		4^7	80,000	10,841	0.14

그림 7(a)와 (b)의 성능 분석 결과에서 보인 것과 같이 기존 탐색은 모든 단위 이동마다 R-트리를 탐색하므로 탐색 회수와 데이타의 수 N에 비례하여 탐색시간이 증가함을 알 수 있다. 그러나 개선된 탐색은 탐색 회수에는 비례하여 증가하지만(그림 7(a)) 객체의 수에는 거의 관련이 없음을 보이고 있으며(그림 7(b)) 객체의 수가 많아질수록 기존 탐색에 비해 더 좋은 성능을 보임을 알 수 있다. 이는 객체의 수가 많아지면 트리의 높이가 커지고 따라서 매번 트리의 루

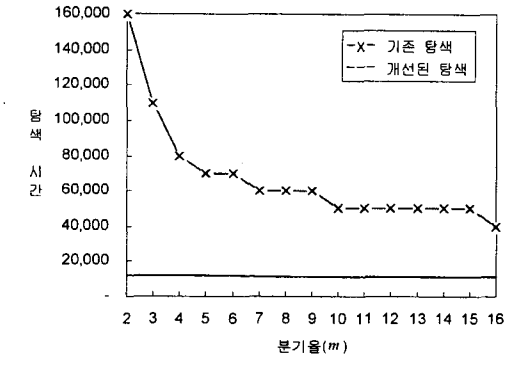
트로부터 탐색을 할 필요가 없는 개선된 탐색기법이 더 효율적인 것은 자명한 것이다. 또한 트리의 분기율의 경우는 분기율이 커질수록 트리의 높이가 작아지므로 기존 탐색기법은 탐색 시간이 줄어들지만 개선된 탐색의 경우는 그림 7(b)의 경우와 유사한 결과를 보인다. 마지막으로 객체의 범위는 기존 탐색기법의 성능에는 영향을 주지 않는다. 그러나 개선된 탐색기법의 경우 범위가 커질수록 다른 범위로 이동할 확률이 줄어들므로 조금씩 탐색 성능이 향상되는 것을 보인다.



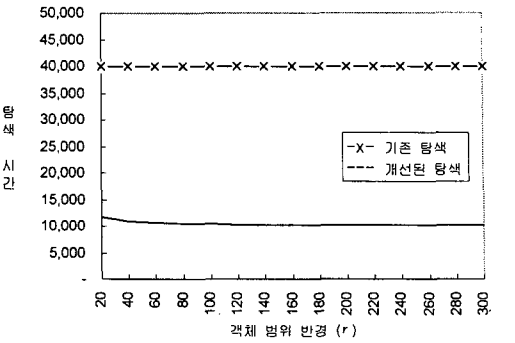
(a) m = 4, h = 8 (N = 4^8), r = 20



(b) m = 4, n = 1,000, r = 20



(c) n = 10,000, N = 4^8, r = 20



(d) n = 10,000, m = 4, h = 4 (N = 4^4)

그림 7. 개선된 탐색기법의 성능
Fig. 7. Performance of the improved search method.

V. 결 론

본 논문에서는 공간 데이타를 탐색하는 응용에 있어서 탐색을 요청하는 객체가 이동하는 경우에 실용적인 탐색 성능을 낼 수 있는 개선된 공간 데이타 탐색기법을 제시하였다. 제안한 탐색 방법은 R-트리를 기본 구조로 기존 R-트리의 탐색기법을 객체가 이동하는 경우에 적합하도록 개선한 것으로서 매번 루트 노드로부터 탐색을 시작하는 대신 현재 노드를 기준으로 탐색을 수행하도록 하였다. 즉 현재 노드의 주변 노드들부터 먼저 탐색을 하도록 함으로써 대부분의 탐색에서 동일 노드내의 다른 엔트리 또는 형제 노드에서 탐색을 완료할 수 있도록 하는 것이다. 개선된 탐색기법은 성능평가 결과 탐색 윈도 혹은 탐색을 요청하는 객체가 연속적으로 이동하는 경우 기존 탐색방법 보다 훨씬 효율적인 성능을 보였다.

논문에서 제시한 탐색 알고리즘은 이동하는 객체가 공간 데이타에 대한 정보를 요구하는 모든 경우에 적용 가능하며 실제로 본 대학에서 개발한 항공기 시뮬레이터의 항법시스템에 개선된 탐색 알고리즘을 적용하여 사용하고 있다. 그 외에도 이동 통신분야에서 이동하는 통신기기와 통신국사이의 연결이나, 도로상에 이동하는 자동차에 대한 교통정보 제공 등에도 적용될 수 있을 것으로 예상된다.

참 고 문 헌

[1] T. Sellis, N. Roussopoulos and C. Faloutsos, "The R+-tree : A Dynamic Index

For Multi-Dimensional Objects," Proceedings of the 13th VLDB Conference, Brighton, 1987

[2] A. Guttman, "R-tree: A Dynamic Index Structure for Spatial Searching," Proceedings of SIGMOD '84 pp 47-57, 1984

[3] 이석호, 화일처리론, 정익사, 1992

[4] 백중환, 황수찬, 김칠영, 황명신, 항공기 시뮬레이터용 항공 전기/전자 모의실험 소프트웨어 개발 (최종보고서), 한국항공대학교, 1994

[5] H. Samet, The Design and Analysis of Spatial Data Structure. Addison Wesley. 1990

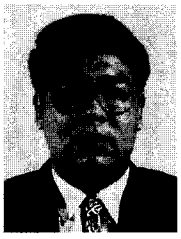
[6] B. P .Zeigler, Multifacetted Modeling and Discrete Event Simulation, Academic Press, 1984

[7] E. Horowitz, S. Sahni and S. Anderson-Freed, "Fundamentals of Data Structure in C", Computer Science Press, 1993

저 자 소 개

柳 秉 玖(學生會員)

1995년 2월 한국항공대학교 컴퓨터공학과 졸업(공학사). 1997년 2월 한국항공대학교 대학원 컴퓨터공학과 졸업(공학석사). 1997년 ~ 현재 미국 일리노이 공대 박사과정 재학중. 주관심분야는 데이타베이스, 지리정보시스템 등임



黃 壽 贊(正會員)

1984년 2월 서울대학교 전자계산기공학과 졸업(공학사). 1986년 2월 서울대학교 대학원 컴퓨터공학과 졸업(공학석사). 1991년 2월 서울대학교 대학원 컴퓨터공학과 졸업(공학박사). 1995년 ~ 1996년 미국 캘리포니아주립대 방문교수. 1991년 ~ 현재 한국항공대학교 컴퓨터공학과 부교수. 주관심분야는 멀티미디어 데이타베이스, 객체지향 데이타베이스, 지리정보시스템 등임

白 重 煥(正會員) 第 34卷 S編 第 11號 參照