DBpia

# 이동체의 궤적 및 현재 위치에 대한 시공간 인덱스

박부식[0], 전봉기, 홍봉희
부산대학교 컴퓨터공학과
{heyman[0],bgjun,bhhong}@pusan.ac.kr

# Spatial-Temporal Indexing of Trajectory and Current Position of Moving Object.

Park Bu-Sik[0], Jun Bong-Gi, Hong Bong-Hee
Dept. of Computer Engineering. Pusan National University

요 약

시간에 따라 연속적으로 위치가 변화하는 객체를 이동체라 한다. 기존의 R-Tree를 사용한 이동체 색인에 관한 연구에서는 현재 위치 질의 시 고비용의 연산이 요구되고, 시간축의 값이 증가하는 방향으로 보고되는 이동체의 위치데이터의 특징을 고려한 노드 분할 정책이 제안되지 않았다. 이 논문에서는 이동체의 현재 위치 및 과거 위치에 대한 색인 방법인 CPTR-Tree(Current Position and Trajectory R-Tree)를 제안한다. 특히, 제안 방법에서 이동체의 현재 위치에 대한 공간차원의 PMBR(Point MBR)을 유지함으로써, 현재 위치 질의 처리시 불필요한 노드 접근 횟수를 줄일 수 있어 성능향상을 할 수 있다. 그리고, 시간축의 값이 증가하는 형태로 보고되는 이동체 위치 데이터의 특징을 고려하여 시간축 분할시 SP(Split Parameter) 분할 방법을 제공함으로써 노드 공간 활용률을 높여 색인의 크기를 줄이고, 공간축 분할시 노드 겹침을 줄이는 동적 클리핑 분할 정책을 제시하여 이동체 과거 위치 검색 효율을 높인다.

## 1. 서론

이동체 위치 데이터는 그 값이 동적으로 변화하는데 특징이 있다. 기존의 정적인 데이터를 대상으로 하는 데이터베이스에서 이러한 데이터를 처리하기 어렵다. 이동체의 위치데이터는 동적이고, 그 수가 많다. 따라서 이동체의 위치데이터를 효율적으로 처리할 수 있는 색인에 대한 연구가 필요하다.

이동체 색인에 관한 연구는 크게 이동체의 현재위치와 미래위치에 관심을 두는 연구[1]와 이동체의 과거위치(궤적)에 관한 연구[2]로 나눌수 있다. 현재 위치에 관한 연구에서는 이동체의 위치를 보고된 시간과 공간 좌표로 이루어진 3차원 점으로 색인하며, 과거 궤적에 관한 연구에서는 이동체의 과거 궤적을 3차원 점들을 연결한 선분(Line Segment)로 표현한다.

[2]의 연구에서 제안한 TB-Tree와 STR-Tree는 노드간의 겹침이 많이 발생하여 과거 영역 질의시 노드 방문 횟수가 많아져 성능이 나쁘다. 이동체 위치 데이터의 특징은 시간이 증가하는 형태로 위치가 보고 되는데 있는데, [3]의 연구에서는 이동체 위치 데이터의 특징을 고려하지 않아 노드 분할 이후 더 이상의 데이터가 삽입되지 않는 노드가 생길 수가 있어 노드의 공간 활용도가 낮아진다. 또한 [2]와 [3]의 연구에서는 현재 위치 검색을 위한 구조가 정의 되어 있지 않다.

이동체의 위치 데이터를 효율적으로 색인 하기 위해서는 검색시 노드간 겹침을 줄여 노드 방문횟수를 줄여야 하고, 현재 위치에 대한 검색을 지원하여야 한다. 또한, 이동체 위치 데이터는 그 개수가 많음으로 색인의 크기를 줄여야 한다. 이 논문에서는 제안 하는 CPTR-Tree는 3D-RTree[3]의 변형으로 현재위치 및 과거위치를 효율적으로 색인하는 것을 목적으로 한다. CPTR-Tree는 현재위치에 대한 PMBR을 유지 함으로서 현재 위치를 효율적으로 검색하는 방법을 제시하고, 과거 영역 분할시 색인의 크기를 줄이기 위해서 노드의 공간 활용도를 높이는 SP분할 정책과, 노드간 겹침을 줄이는 동적 클리핑 분할 정책을 제시하여 이동체의 과거 위치 검색시 노드 방문 횟수를 줄여 검색성능을 향상시킨다.

2장에서는 관련연구와 문제점에 대해 언급하고, 3장에서는 이동체의 모델링과, CPCR-Tree의 자료구조에 대해 기술한다. 4장에서는 제안 방법인 CPTR-Tree의 알고리즘을 다룬다. 5장에서 결론 및 향후 연구를 기술한다.

## 2. 관련연구

이동체의 위치를 색인 하기 위한 구조로서 R-Tree를 사용한 색인구조와, 그 변형을 사용한 TB-Tree와 STR-Tree를 사용한 색인 구조가 있다[2]. 이러한 구조에서는 현재 위치를 검색하기 위한 구조가 정의되어 있지 않다. 이동체의 현재 위치 색인과, 과거 궤적 색인의 기존 연구로서 2D-RTree + 3D-RTree를 사용한 연구가 있다[4]. [4]에서는 이동체의 현재 위치는 2D-Rtee에 점 객체를 삽입하여 색인 하고, 과거 위치는 3D-RTree에 선분을 삽입 하여 색인 한다. 이러한 경우에 이동체가 새로운 위치를 보고 하게 되면, 2D-RTree에 삽입, 삭제 연산이 발생하고, 3D-RTree에 삽입 연산이 발생함으로, 새로운 위치 삽입 시 고비용이 소모 된다. 또한, 현재 위치 바로 이전의 위치를 검색할 경우 역시 이전위치에 대한 버퍼질의를 해야 함으로서 검색 성능이 나쁘다.

이동체의 위치 데이터를 위한 시공간 색인으로서 HR-Tree[5]와 MV3R-Tree[6]가 있다. HR-Tree는 이동체들의 이동이 빈번하지 않은 경우에는 효율적이지만, 이동이 많이 발생하는 경우 단말 노드 및 비단말 노드를 새로 생성해야 하고, 특히 영역 질의의 성능이 저하되는 문제를 가진다. MV3R-Tree는 MBR-Tree와 단말 노드들로 구성되는 보조적인 3DR-Tree를 결합하여 타임스탬프 질의와 영역 질의를 효율적으로 처리할 수 있지만, 버전 분할에서 발생하는 중복되는 데이터 때문에, 3DR-Tree보다 대략 1.5배 색인의 크기가 큰 문제가 있다.

## 3. 이동체의 모델링과 CPTR-Tree의 자료구조

### 3.1 이동체의 모델링

이 논문에서 연속적으로 이동하는 이동체의 궤적을 이산적 모델로 모델링 한다. 이산적 모델은 이동체의 위치 정보를 보고된 위치와 보고 시간인 유한개의 3차원 점으로 표현한다. 그리고 이산적 모델에서는 보고주기 사이의 갭(Gap)으로 인하여 정확한 현재 위치를 알 수 없기 때문에 이동체의 현재 위치는 가장 최근에 보고한 공간상의 위치로 정의한다. 이 논문에서 현재 위치를 표현 하기 위한 TN시간(Time Now)의 정의와 새로운 데이터 형식을 정의한다.



Copyright (C) 2005 NuriMedia Co., Ltd.
www.dbpia.co.kr

[정의1]
*TN Time* : 연속적으로 변하는 현재 시간을 나타내는 추상적인 시간 값. TN Time으로 표현된 시간은 항상 현재 시간을 나타낸다.
Time T: {t | 0 < t < ∞ or TN}
가변선분 : 선분을 나타내는 좌표에 TN Time이 있을 경우
고정선분 : 선분을 나타내는 좌표에 TN Time이 아닌 다른 시간 값을 갖는 경우

아래 [그림1]은 색인에 삽입 되어 있는 가변선분 및 고정선분을 보여준다. 이동체가 이동 중에 있을 경우 이동체의 현재 위치를 알 수 없음으로 현재 위치를 표현 하기 위해 가변선분을 정의하고, 가변선분은 공간좌표상 시작점과 끝점이 같은 선분으로서 논리적으로 시간이 지남에 따라 선분의 길이가 늘어난다.

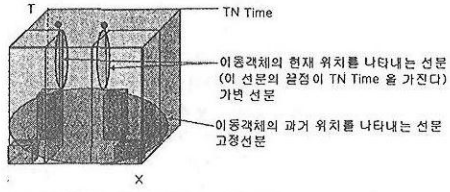


그림 1. 색인 상에 유지되는 현재 위치 및 과거 위치

### 3.2 CPTR-Tree의 자료구조

현재 위치의 검색은 시간이 고정된 공간차원에 대한 범위 질의로 볼 수 있고, 현재 위치를 표현 하는 선분이 속한 MBR(Minimum Bounding Rectangle)를 3차원 MBR로 표현할 경우, 현재 위치 질의 시 탐색영역이 증가할 수 있어서 성능이 낮아진다. [그림2]에서 보는 바와 같이 이동체의 움직임을 나타내는 선분을 3차원 MBR로 표현 하게 되면, 실제로 현재 위치를 나타내는 선분이 MBR내에서 차지하는 영역이 작을 수 있음에도 불구하고, MBR은 크게 표현 된다. 이것은 현재 위치를 나타내는 2차원 영역에 대해서 사장공간(Dead Space)을 증가시키고, 현재 위치 질의 시 성능을 저하시키는 원인이 된다. 따라서 이 논문에서 제안하는 CPTR-Tree는 현재 위치를 나타내는 가변라인의 끝점의 공간차원 좌표(TN Time의 공간좌표)를 PMBR(2차원 MBR)로 유지 함으로서, 현재 위치 검색 성능을 높인다. [그림2]는 PMBR을 나타내고 있다.

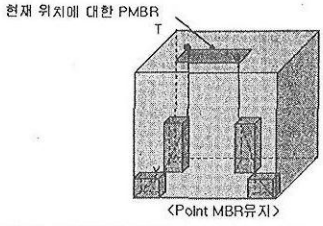


그림 2. 이동체의 현재 위치에 대한 점 MBR

CPTR-Tree의 단말 노드는 아래와 같은 형태의 엔트리 Ci개를 가진다. Ci는 단말 노드의 용적률이다.

(LMBR, O_id, Trajectory ID)

LMBR(Line MBR)은 이동체의 움직임을 나타내는 선분에 대한 (Xlow, Ylow, Tlow, Xhigh, Yhigh, Thigh)형태의 MBR이고, O_id는 실제 객체의 객체 식별자 이다. Trajectory ID는 궤적 식별자 이다. CPTR-Tree의 중간 노드는 아래와 같이 표현 된다.

(LMBR, PMBR, Child-Pointer)

LMBR은 Child-Pointer가 가르키는 자식 노드의 모든 엔트리의 LMBR의 합이다. PMBR(PMBR)은 자식 노드가 TN Time을 포함하는 엔트리를 가지고 있을 경우, TN Time을 제외한 공간 좌표만으로 이루어진 2차원 MBR이다. PMBR 또한 Child-Pointer가 가르키는 자식 노드의 모든 엔트리의 PMBR의 합이다.

## 4. CPTR-Tree의 알고리즘

이 장에서는 CPTR-Tree의 검색, 삽입, 분할 알고리즘을 제안한다. 이 논문에서 제안 하고자 하는 색인 구조는 아래와 같은 특징이 있다.

- 이동체의 현재 위치와, 과거 위치 색인이 가능하다.
- 이동체의 현재 위치 검색을 PMBR(Point MBR)을 사용하여 함으로서 현재 위치 검색 성능이 높다.
- 색인의 공간활용도를 높이는 분할 정책을 사용하여, 색인의 크기가 작고, 공간 분할시 동적 클리핑 분할 정책을 사용함으로서 노드 겹침을 줄여 과거 위치 검색 성능이 좋다.

### 4.1 검색 알고리즘

CPTR-Tree의 검색 방법은, 기존의 R-Tree의 검색 방법과 유사하다. 검색 알고리즘은 이동체의 현재 위치 검색과 이동체의 과거 위치를 검색으로 나눈다. 이동체의 현재 위치 검색일 경우에는 PMBR을 이용하고, 이동체의 과거 위치 검색은 LMBR을 이용하여 검색한다.

[그림3] 은 현재 위치 검색시 성능향상의 예를 보여주고 있다. [그림3]에서 보는 바와 같은 이동체의 현재 위치 질의를 처리 하고자 할 경우, 기존의 방법을 사용하면 [그림3 (a)]처럼 질의 영역에 이동체가 없음에도 불구하고 노드 접근을 해야 하지만, [그림3 (b)]처럼 PMBR을 유지함으로써 불필요한 노드 접근을 하지 않을 수 있다.

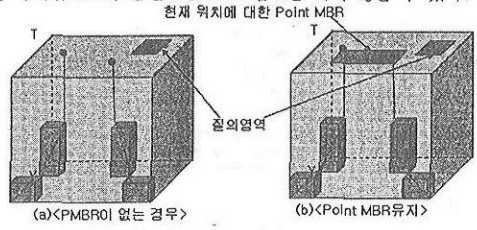


그림 3. 현재 위치 검색방법

```
Algorithm Search(node Root, rect w)
S1. Search nonleaf nodes:
    If W has TN Time
        Invoke Search for every entry whose PMBR intersects
        the query window w.
    If W do not have TN Time
        Invoke Search for every entry whose LMBR intersects
        the query window w.
S2. Search leaf Nodes:
        Report all the entries that intersect the query window
        w as candidate.
```

### 4.2 삽입 알고리즘

이동체의 현재 위치가 새롭게 보고 된 경우 색인에 삽입하는 과정은 다음과 같이 2단계로 나누어 진다. 기존에 보고된 현재 위치를 과거 위치로 갱신 하는 과정과, 새로 보고된 현재 위치를 삽입하는 과정으로 이루어 진다.

새로운 위치의 삽입은 새로운 선분을 나타내는 MBR을 삽입하는 것으로 볼 수 있다. 이동체가 보고 하는 선분은 고정선분이 되는데, 현재 위치를 나타내기 위해서, 고정 선분에서 유도된 가변선분을 생성하여 삽입한다. 가변 선분의 MBR은 이동체가 보고한 최후의 공간 좌표와, 보고한 시간과, TN Time을 포함한 MBR이 된다. 이동체의 다음 위치 보고 시, 기존의 가변선분이 고정선분으로 갱신되고 현재 위치를 나타내는 가변 선분이 삽입된다.

[그림4]는 현재 위치 삽입 예제를 보여 주고 있다. [그림4]에서 왼쪽 그림은 삽입 이전의 상태이고, 오른쪽 그림은 삽입 이후의 상태이다.

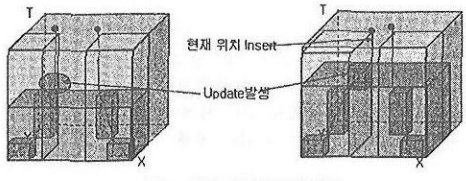


그림 4. 현재 위치 삽입 예제



Copyright (C) 2005 NuriMedia Co., Ltd.
www.dbpia.co.kr

```
Algorithm Insert(node root, rect r)
/* 보고된 위치를 나타내는 r과 r에서 유도된 가변 Line Segment를
삽입한다.
TN_rect : 현재 위치를 가진 가변선분
Previous_TN_rect:현재 위치 보고 이전에 가졌던 가변선분 */
I1. Create Unfixed Line Segment TN_rect(x,y,t,x,y,TN Time)
derived from r.
I2. Select Node L to insert TN_Rect.
   Choose_Leaf(node root, rect TN_Rect)
I3. Insert TN_Rect into Node L and Update Previous_TN_Rect.
   If L has an previous_TN_Rect
      If L has an empty slot
         Update Previous_TN_Rect by r&Insert TN_rect
      If L is Full
         Update Previous_TN_Rect by r&Split(L, TN_rect) and
         Create LL
   If L doesn't have previous_TN_Rect
      PN := FindNode(node N, rect r)
      Update PN by r and invoke AdjustTreeMBR()
      If L has an empty slot
            Insert TN_rect
      If L is Full
         Invoke Split(L, TN_rect) and create LL
I4. Propagate Changes upward
   Invoke AdjustTreeMBR(L, LL)
         PMBR,LMBR Propagation
I5. Grow Tree Taller
   If node split propagation caused the root to split, create a
   new root whose children are the two resulting nodes.
```

### 4.3 분할 축 선정 정책

이동체의 위치 보고는 시간축의 값이 증가하는 방향으로 순차적으로 이루어진다. CPTR-Tree에서는 새로운 위치 데이터가 보고되면 이전의 가변 선분이 고정선분으로 갱신되고, 가변 선분이 삽입된다. 이와 같은 특징으로 인하여 가변 선분을 포함하는 노드는 갱신 및 삽입이 진행되는 동적노드이고, 고정선분만을 포함하는 노드는 더 이상의 삽입이 없는 고정노드이다.

기존의 R-Tree와 R*-Tree의 변형을 이용한 색인은 이동체 위치 데이터의 시간 도메인에 대한 특징을 고려하지 않았기 때문에 노드의 공간활용도가 50%~60%정도로 공간활용도가 낮았다. CPTR-Tree는 이와 같은 문제점을 해결하기 위하여 시간축 분할시 가변선분과 고정선분으로 비균등 분할을 수행하며, SP(Split Parameter)값을 사용하여 고정선분을 포함하는 고정노드의 공간활용도를 높이는 분할 정책을 제안한다.

> [정의2]
> SP(Split Parameter)값: 분할축 선정시 기준이 되는 노드 용적율 보다 작은 값.
> 1 <= SP <= Node capacity/2

SP 값에 의한 분할축 선정 방법은 다음과 같다.

① 시간축 분할 (가변선분의 개수 <= SP)
가변선분을 가지는 노드와 고정선분을 가지는 두 개의 노드로 분리하고, 이것을 시간축 분할이라 한다.

② 공간축 분할 (가변선분의 개수 > SP)
공간분할은 분할될 노드간의 중복이 최소화 하는 동적 클리핑 분할 정책을 사용한다.

### 4.4 동적 클리핑 분할 정책

분할 정책을 설명하기 위해 3차원 공간을 2차원으로 그리기로 한다. [그림5]에서 가로 축은 x,y축을 의미하고, 세로 축은 t축을 의미한다.

R*-Tree[7]의 분할 정책은 분할시 노드의 겹침을 최소화 하는 방법을 제공한다. 노드의 겹침을 최소화 함으로서 검색 성능을 높일 수 있다. [7]의 연구에서, 한 노드에 포함되어야 하는 최소한의 엔트리 수가 노드 용적율의 40%정도인 것이 최적이라고 실험적으로 증명되어 있다. 그러나 이동체의 궤적 데이터 자체의 문제로 노드간의 겹침이 발생한다.

이 논문에서는 겹침영역을 유발시키는 선분에 대해서 [그림5]와 같은 동적으로 클리핑을 시도하여 겹침영역을 줄이는 분할정책을 제안한다. 분할 축을 선정하기 위하여 다음과 같이 엔트리를 정렬하여 3개의 그룹으로 나눈다. (M:노드용적율, m: 노드당 최소 엔트리 개수)

① 각각의 축에 대하여, 엔트리들의 최소값과, 최대값에 대해여 엔트리를 정렬한다

② 작은 값과 큰 값을 차례대로 m개씩 가지는 두 그룹(G1, G2)과, 나머지 M-2m+1개를 가지는 그룹(G3)으로 나눈다.

분할축 선정 기준은 다음과 같다.

① G1.MBR과 G2.MBR이 X,Y축 모두에 겹침이 있으면 겹침이 작은 축을 선정한다.

② G1.MBR과 G2.MBR이 X,Y축 중 하나의 축에 겹침이 발생하면, 겹침이 없는 축을 선정한다.

③ G1.MBR과 G2.MBR이 X,Y축 중 겹침이 발생하는 축이 모두 없으면 G3의 엔트리들을 X,Y축으로 투영(Projection)하여 엔트리간의 겹침이 최소인 축을 선정한다.

동적 클리핑 분할 정책은 다음과 같다.

① G3의 남은 엔트리들을 각각의 그룹으로 분배해서 가장 적은 클리핑을 발생시키는 분배방법으로 분할 한다.

② 중복을 유발하는 엔트리는 클리핑 한다.

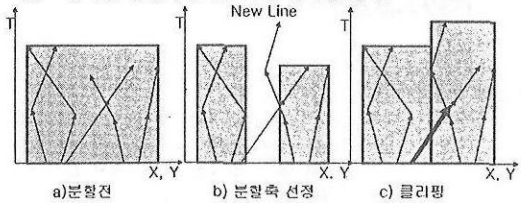


그림 5. 동적 클리핑 분할 정책

### 5. 결론 및 향후 연구

이 논문에서 CPTR-Tree를 사용하여, 이동체의 현재 위치와 과거 위치를 색인 하는 방법에 대하여 제안하였다. 이동체의 현재 위치와, 과거 위치를 한 색인에 관리 함으로서, 이동체 위치 삽입 비용을 줄였다. 이동체의 현재 위치에 대해서 PMBR을 유지함으로써 현재 위치 검색시 필요한 노드 접근 횟수를 줄일 수 있어 성능향상이 있다. 그리고 이동체 위치 데이터의 특성을 반영한 노드 분할 정책으로, 시간축 분할시 노드 공간 활용도를 높임으로서 색인의 크기를 줄였고, 공간축 분할시 노드겹침영역을 줄이는 동적 클리핑 분할 정책을 제안하여 이동체 과거 영역에 대한 검색 성능을 높였다.

향후 연구로서 SP값의 변화에 따른 노드 공간 활용도의 변화를 분석하고, 공간축 분할시 노드겹침영역을 줄이는 최적의 방안에 대해 연구가 필요하다. 그리고 구현을 통한 성능 평가가 필요하다.

### 6. 참고문헌

[1] S. Saltenis, C. S. Jensen, S.T. Leutenegger, and M. A. Lopez, " Indexing the Positions of Continuously Moving Objects." , In Proc. ACM SIGMOD on Management of data, p331 - 342, 2000.

[2] Pfoser, D., Jensen, C., Theodoridis Y., " Novel Approaches to the Indexing of Moving Object Trajectories" , In Proc. Of the 26th Int' l Conference on VLDB, pp. 395-406, 2000.

[3] Yannis Theodoridis, " Spatio-Temporal Indexing for Large Multimedia Applications" , In Proc. Of the 3rd IEEE Conf. on Multimedia Computing and Systems, pages 441-448, June 1996

[4] Mario A. Nascimento, Jefferson R. O. Silva, Yannis Theodoridis" Evaluation of Access Structures for Discretely Moving Points." Spatio-Temporal Database Management,171-188 1999

[5] M. A. nascimento, J. R. O. Silva, " Towards Historical R-Trees" , Proceedings of ACM-SAC, 1998

[6]Yufei Tao,Dimitri Papadias:MV3R-Tree:A Spatio-Temporal Access Method for Timestamp and Interval Queries.VLDB 2001: 431-440

[7] Norbert Beckmann,Hans-Peter Kriegel,Ralf Schneider, Bernhard Seeger," The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles." SIGMOD Conference 1990: 322-331



Copyright (C) 2005 NuriMedia Co., Ltd.
www.dbpia.co.kr