제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

# 이동체의 궤적과 현재 및 미래위치 검색을 위한 색인

전희철°, 안성우, 김진덕*, 홍봉희
부산대학교 컴퓨터공학과°, 동의대학교 컴퓨터공학과*,
e-mail : {hcjeon°, starnme}@pusan.ac.kr ,jdk@dongeui.ac.kr*, bhhong@pusan.ac.kr

# An Index for Querying Trajectory, Current and Future Position

Hee-Chul Jeon°, Sung-Woo Ahn, Jin-Deog Kim*, Bong-Hee Hong
Dept. of Computer Engineering. Pusan National University
Dept. of Computer engineering. Dongeui University*

요 약

최근 이동통신과 GPS 기술의 발달로 위치기반 서비스에 대한 요구 및 관련된 연구가 활발히 진행되고 있다. 이동체 색인에 관한 기존 연구는 시간도메인에 따라 과거궤적에 대한 색인과 현재 및 미래위치색인 등으로 구분된다. 그러나 실 세계 응용에서는 과거 궤적뿐만 아니라 현재 및 미래 위치를 모두 필요로 하는 경우가 많기 때문에 각 시간도메인 별로 두 개의 색인을 유지해야 하며 이 방법은 불필요한 비용을 필요로 한다. 이 논문에서는 이동체의 과거궤적과 현재 및 미래위치를 하나의 색인에 유지하는 기법과 과거와 현재 및 미래에 대한 질의처리방법, 그리고 이동체의 이동 속성을 고려한 삽입과 분할정책을 적용한 새로운 색인을 제안하였다.

## 1. 서론

오늘날 무선이동통신 기술의 발달로 PDA(Personal Digital Assistants), 휴대폰 등의 무선이동기기들이 보편화 되었다. 이러한 기술은 GPS(Global Positioning System)기술의 발달에 힘입어 최근 위치기반서비스(LBS : Location Based Service)를 이용한 서비스가 날로 늘어나고 있다.

위치기반서비스는 사용자들로부터 전송되어 오는 수많은 위치정보를 효율적으로 저장, 검색, 관리할 수 있는 색인연구가 그 핵심이라 할 수 있다. 지금껏 관련된 연구가 활발히 진행되고 있으며 다양한 연구결과가 발표되었다. 이러한 기존 연구는 위치정보를 다루는 방식에 따라 첫째, 위치정보를 점으로 표현하는 이동체의 현재위치에 대한 R-Tree 와 해쉬 기반 색인들이 있고, 둘째로 선분(Line Segment)을 사용하여 과거궤적을 인덱싱 하는 3DR-Tree[3]와 TB-Tree, STR-Tree[2], 셋째, 이동체의 이동정보를 선형함수로 표현하여 현재 및 미래위치에 대한 검색을 지원하는 TPR-Tree[1]와 PMRQuad-Tree 등의 3 가지로 분류할 수 있다.

실 세계의 서비스에서는 이동체의 과거이동정보와 함께 현재위치정보도 중요시되고 있으며 과거와 현재위치정보를 동시에 필요로 하는 응용이 많다. 그러나 기존 연구에서는 두 시간도메인을 모두 포함하는 색인에 대한 연구가 없기 때문에 과거궤적에 관한 색인과 현재위치에 대한 색인을 따로 유지하는 방식을 사용하지만 이 방법은 삽입과 검색에 있어 불필요한 비용을 필요로 한다. 본 논문에서는 이동체의 과거위치를 표현하는 선분과 현재위치정보를 나타내는 시간에 대한 선형함수를 하나의 노드에 함께 저장하는 방법과 과거와 현재 및 미래시간을 동시에 포함하는 질의처리 시 두 개의 색인을 유지할 때 발생하는 불필요한 노드탐색비용과 부가비용을 줄인 질의처리 방법을 적용한 새로운 색인을 제안한다.

2 장에서 관련연구에 대해 조사하고 3 장에서는 과거궤적을 위한 색인과 현재 및 미래위치를 위한 색인을 별도로 유지할 때의 문제점에 대해 살펴본다. 4 장에서는 이 논문에서 제안하는 색인에 대해 설명하고 5 장에서 결론 및 향후 연구로 끝을 맺는다.

## 2. 관련연구

STR-Tree, TB-Tree[2] 는 이동체의 궤적을 선분으로 표현한 색인으로 R-Tree 의 변형이다. 이 색인은 현재위치 검색이 고려되지 않았고 궤적을 보호하기 위해 공간근접성을 고려하지 않기 때문에 영역질의 성능이 좋지 않다. 3DR-Tree[3]는 R-Tree 를 3 차원으로 확장한 구조로써 역시 현재위치 검

색이 고려되지 않았고 분할 시 시간도메인의 특성에 대해 고려되지 않았기 때문에 공간 효율성이 떨어진다.

이동체의 과거궤적 및 현재 위치에 대한 기존 연구로 2+3DR-Tree[6]가 있다. 2DR-Tree 를 사용하여 현재 위치를 점 객체로 표현하고 3DR-Tree 를 이용하여 과거궤적을 선분으로 표현하는 방법이지만 이 구조에서 새로운 위치보고가 있을 때 2DR-Tree 에서 삭제와 삽입 연산이 발생하고 3DR-Tree 에서 다시 삽입연산이 발생하므로 삽입에 많은 부하가 걸리고 2DR-Tree 의 현재위치 색인에서는 이동체가 최근 보고한 위치를 현재 위치라고 정의하고 있으나 이는 실제 현재위치와 오차가 크다고 할 수 있다. 또한 과거 궤적에 대한 3DR-Tree 의 경우 위에서 설명했던 시간도메인에 대한 고려가 없다는 단점을 그대로 가진다.

TPR-Tree[1]는 이동체를 시간에 대한 선형함수로 표현하는 색인구조이다. 이동체의 위치를 단순한 좌표로 저장하지 않고 방향과 속도를 나타내는 벡터를 저장하여 특정 임계값 이하의 변화일 경우 업데이트 하지 않음으로써 업데이트 횟수를 획기적으로 줄였고 시간에 대한 선형 함수를 사용하여 현재 및 가까운 미래위치에 대한 검색이 가능한 구조이지만 과거궤적에 대한 고려가 없다는 단점이 있다.

## 3. 두 개의 색인 유지 시의 문제점

3 장에서는 기존연구의 문제점 즉, 과거궤적을 위한 색인과 현재 및 미래위치를 위한 색인을 별도로 유지할 때 질의처리와 삽입에서의 문제점을 설명한다.

### 3.1 질의처리 비용문제

이동체의 과거위치와 현재 및 미래위치를 위해 별도의 색인을 유지하는 경우 "5 분 전부터 현재까지 서울역 광장 앞을 통과한 차량을 찾아라" 와 같이 동시에 두 색인에 대한 접근이 필요한 질의가 요청되면 [그림 1]처럼 각 색인에 대한 노드탐색 비용과 그 결과를 병합하는 부가적인 비용을 필요로 하게 된다.

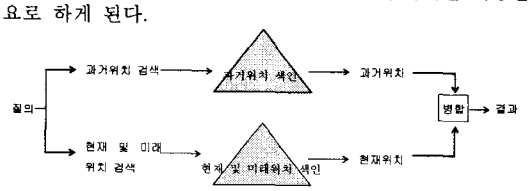


그림 1. 별도 색인 유지 시 질의처리비용

만약 과거위치와 현재 및 미래위치를 하나의 색인에 유지한다면 [그림 1]과 같은 복잡한 과정이 필요 없어지게 되고 각각의 색인에서 노드를 탐색하지 않아도 된다. [그림 2]는 각각의 경우에 대한 노드 탐색 경로의 예를 보여준다.

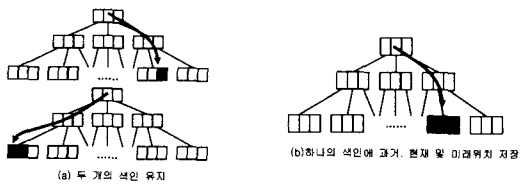


그림 2. 노드탐색 경로

[그림 2]의 (a)는 과거위치와 현재 및 미래위치에 대한 별도의 색인을 유지할 때 두 번의 질의가 발생해야 하는 것과 각각의 검색을 위한 노드탐색이 필요함을 보여준다. 일반적으로 동일한 이동체의 과거위치와 현재 및 미래위치간에는 공간 근접성이 있고 일반적인 공간색인에서 공간근접성이 있는 데이터는 동일노드 혹은 그 형제노드(같은 부모노드를 가지는 노드)에 저장될 확률이 매우 높다. 다시 말해 공간 근접성이 있는 정보가 색인상에 비슷한 위치를 가질 때 검색에 필요한 노드탐색 경로는 짧아진다. 따라서 [그림 2]의 (a)처럼 각각의 색인에 대해 검색을 하는 것은 [그림 2]의 (b)와 같이 과거위치와 현재 및 미래위치를 공간근접성에 기반하여 하나의 색인에 저장하는 경우에 비해 노드탐색에 필요한 I/O 횟수가 많아진다는 문제가 있다.

### 3.2 삽입비용문제

이동체 색인에 있어서 관리해야 할 이동체의 수가 많을 경우 시스템의 처리성능을 벗어나는 위치보고가 발생하면 데이터베이스에 저장되지 않고 소실되는 문제가 있다. 특히 색인의 삽입비용이 클 경우 더욱 문제가 된다.

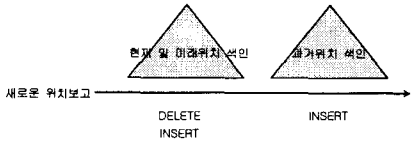


그림 3. 별도 색인 유지시의 삽입비용

[그림 3]에서 볼 수 있듯이 이동체가 새로운 위치를 보고할 경우 현재 및 미래위치 색인에서는 동일 이동체의 직전위치를 검색하여 삭제한 후 루트노드로 부터 삽입연산을 수행해야 하고 과거위치 색인에서 다시 삽입연산을 수행해야 한다. 그러나 동일 이동체의 직전위치와 새로운 위치는 공간 근접성이 매우 높기 때문에 현재 및 미래위치색인에서 이동체의 직전위치를 삭제하는 연산과 과거위치색인에서의 삽입연산은 현재 및 미래위치와 과거위치를 하나의 색인에 유지 할 경우 동일노드 내에서 이루어질 수 있으므로 노드탐색에 필요한 I/O 횟수를 줄일 수 있다.

## 4. HCR-Tree(History and Current R-tree)

4 장에서는 이 논문이 제안하는 새로운 색인에 대해 용어정의, 과거데이터와 현재 및 미래를 나타내는 선형함수를 동시에 저장할 수 있는 새로운 노드구조와 이동체의 이동속성을 고려한 삽입과 분할정책, 그리고 질의처리방법 등으로 나누어 소개한다.

### 4.1 용어정의

이 논문에서는 현재 및 미래위치를 표현하기 위해 시간에 대한 선형함수[1]를 사용하고 과거위치를 저장하기 위해 선분을 사용한다. 이 두 가지 형태의 위치정보를 하나의 색인에 유지하기 위해서 다음과 같이 시간과 위치정보 형식을 정의한다.

[정의 1]

$T_{uc}$ : 이동체의 최근 보고시간

*Closed-line(CL)* : 과거궤적에 해당하는 선분

$(x_1, y_1, t_1, x_2, y_2, t_2)$

*Opened-line(OL)* : 이동체의 이동속성을 나타내는 벡터

$(x, y, v_x, v_y, t)$

*Bounding rectangle(BR)* : 노드 내 혹은 하위 노드들의 *CL* 과 *OL* 을 포함하는 시간에 대해 동적인 영역

$(BR_T, BR_B, BR_L, BR_R, v_T, v_B, v_L, v_R, t_1, t_2)$

CL 은 3 차원상의 두 점을 연결한 선분으로써 이동체의

과거궤적을 표현하고 OL 의 x, y 는 $T_{uc}$ 시간의 이동체의 위치를 나타내고 $v_x$ 와 $v_y$ 는 이동체의 이동속성을 표현한다. 이 논문에서 제안하는 색인의 한 노드는 OL 과 CL 을 동시에 저장해야 하므로 [1]에서 제시한 TPBR 을 [정의 1]의 BR 과 같이 재정의 한다. [그림 4]는 BR 이 OL 들의 이동방향과 속도를 고려하여 시간이 지남에 따라 확장되는 것을 보여준다. 예를 들어 $BR_T$ 는 노드내의 OL 들의 $v_y$ 중 가장 큰 값을 가지는 $v_T$ 의 속도로 시간이 흐름에 따라 확장하고 나머지 $BR_L$, $BR_R$, $BR_B$ 등도 같은 원리에 따라 확장한다. BR 의 $t_1$ 은 노드 내에 있는 이동체의 가장 오래된 보고시간으로 설정되고 $t_2$ 는 삽입 혹은 질의발생시간으로 결정된다.

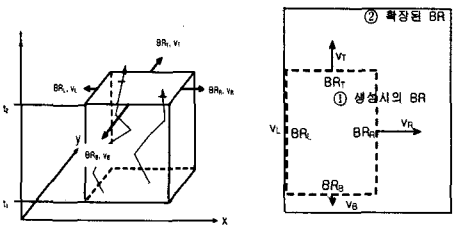


그림 **4 Bounding rectangle**

### 4.2 노드구조

[정의 1]의 OL 과 CL 을 동일 노드에 저장하기 위해서는 새로운 노드 구조가 필요하다. [정의 1]의 OL 과 CL 은 각각 5 개와 6 개의 변수에 해당하는 저장공간을 필요로 하므로 큰 차이가 없다. 또한 이동체가 새로운 위치를 보고함에 따라 CL 의 비율이 OL 보다 커지므로 저장공간의 차이로 인한 공간효율의 감소는 매우 작다. 이 논문에서는 이러한 특징을 이용하여 다음과 같은 노드구조를 제안한다.

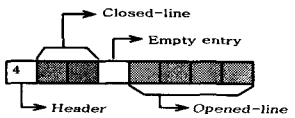


그림 **5. HCR-Tree** 의 노드 구조

[그림 5]는 노드 내의 CL 들을 노드의 왼쪽에 저장하고 OL 들을 오른쪽에 저장한 후 OL 의 개수를 노드의 헤더에 표시하는 방법을 설명한다. 과거위치에 대한 질의 수행 시에는 노드 왼쪽의 CL 만을 참조하게 되고 현재 및 미래위치에 대한 질의 수행 시는 헤더에 표시된 OL 의 개수를 확인하여 OL 들만을 참조하며 앞의 두 가지 모두에 해당하는 경우에는 전체 엔트리를 참조하여 검색을 진행한다.

### 4.3 질의처리

이 논문에서 제안하는 새로운 색인은 영역질의 성능이 좋은 3DR-Tree 에 기반한다. 따라서 [정의 2]와 같이 새로운 색인은 영역질의를 문제의 대상으로 한다.

[정의 2]
*Range Query* : $Q = (R, T_{q1}, T_{q2})$
$R$: 공간상의 질의영역 $(x_1, y_1, x_2, y_2)$
$T_{q1}$, $T_{q2}$: 질의상의 요구시간 간격
**a.** $(T_{q1} < T_{q2} < T_{uc})$
**b.** $(T_{q1} < T_{uc} < T_{q2})$
**c.** $(T_{uc} < T_{q1} < T_{q2})$

[정의 2]의 a 는 질의가 요구하는 정보가 모두 과거의 위치정보라는 것을 의미한다. 이 경우 색인 내의 모든 OL 은 무시되고 BR 은 [그림 4]의 ① 과 같이 확장을 고려하지 않고 기존 R-Tree 기반 색인의 검색알고리즘을 사용한다. [정의 2]의 b 는 질의가 요청하는 정보가 과거와 현재 및 미래에 걸친 위치정보라는 것을 의미한다. 이 경우는 CL 과 OL 이 모두 참조되어야 하고 BR 은 [그림 4]의 ② 와 같이 $T_{q2}$ 에 대해 확장되고 노드 내의 OL 또한 $T_{q2}$ 를 사용하여 CL 로 취급되어 R-Tree 기반 색인의 검색 알고리즘을 사용하여 검색한다. [그림 6] 은 [정의 2]의 c 와 같이 미래에 대한 질의처리 방법을 설명하고 있으며 이 경우 노드 내의 OL 들만을 참조하게 되므로 CL 만이 저장된 노드와 그 하부노드는 탐색하지 않아도 된다.

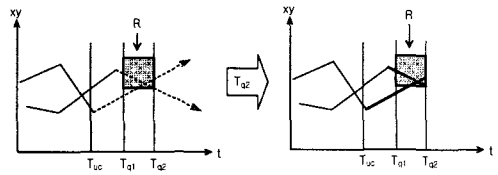


그림 **6.** $T_{uc} < T_{q1} < T_{q2}$ 일 경우의 질의처리

이 논문에서 제안하는 색인은 CL 과 OL 을 하나의 색인에 유지하고 각각에 대한 질의를 한번의 질의로 처리하게 됨으로써 3.1 절에서 언급한 과거와 현재 및 미래를 동시에 포함하는 질의처리 시 병합에 의한 추가비용을 제거할 수 있고 과거위치를 위한 색인과 현재 및 미래위치를 위한 색인을 별도로 유지할 때 보다 노드탐색에 필요한 I/O 횟수가 감소하게 된다.

### 4.4 삽입정책

이동체가 새로운 위치를 보고하여 삽입이 발생하면 해당 이동체의 직전 OL 이 저장된 노드를 검색해야 한다. 이 논문에서는 이동체의 직전위치가 저장된 노드를 검색하는 FindNode[2] 알고리즘을 사용한다. 검색된 OL 은 새로운 OL'을 사용하여 CL 로 갱신되고 새로운 OL'은 루트 노드로부터 삽입연산을 수행해야 한다. 새로운 CL 은 기존 OL 의 x, y, $t$ 와 새로운 OL' 의 $x'$, $y'$ $t'$으로 구성된다. 새로운 CL (x, y, t, x', y', t')은 노드의 빈 엔트리중 가장 왼쪽에 저장되고 기존 OL 은 노드에서 삭제된다. 이러한 일련의 연산들은 모두 동일 노드에서 이루어지므로 3.2 절에서 언급한 삽입비용 중 현재위치색인에서의 직전위치삭제연산과 과거위치색인에서의 삽입연산이 동일노드 내에서의 업데이트연산으로 해결되므로 삽입에 필요한 비용이 감소된다.

#### 4.4.1 이동체의 이동속성을 고려한 삽입정책

이동체의 새로운 위치보고 OL'은 위치정보(x, y) 와 이동속성($v_x$, $v_y$)로 이루어진다. OL'을 이동 속성을 고려하지 않고 위치정보 만을 고려하여 삽입하게 되면 동일 이동체의 다음 위치보고발생에 따라 OL 을 CL 로 변환하게 될 때 많은 중첩(Overlap)과 사장영역(Deadspace)를 유발할 수 있다. 이 논문에서는 위치정보와 이동속성을 함께 고려하는 방법으로 이동체의 예측된 미래위치와 새로운 위치보고 OL'의 위치정보를 연결한 선분을 삽입하는 것을 고려한다.

[그림 7]의 (a)는 이동체의 위치만을 고려하여 삽입할 경우를 설명하고 있다. 이 경우 새로운 위치정보는 A 에 포함되지만 B 를 향해 이동하고 있으므로 동일 이동체의 다음 위치 보고 시 A 와 B 는 많은 중첩영역이 발생하게 된다. [그림 7]의 (b)는 $T_{estimate}$ 이후의 예측된 위치를 사용하여 선분의 형태로 삽입을 하게 될 경우 B 로 삽입되는 것을 보여

준다. $T_{estimate}$ 는 삽입 시 이동체의 위치를 예측하기 위해 별도로 정의된 시간 값이다.

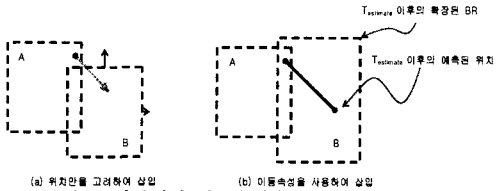


그림 7. 이동체의 이동속성을 고려한 삽입

### 4.5 분할정책

이 논문에서는 색인의 공간 활용도를 개선하기 위해 [7]의 연구에서 제안한 동적 분할정책을 고려한다. 동적 분할정책은 이동체의 위치데이터의 시간도메인의 값이 증가하는 방향으로만 저장된다는 점에 착안하여 노드 내의 데이터 분포에 따라 시간축비 균등분할을 고려한 분할기법이다. 이 논문에서는 분할도메인 선정방법과 시간축분할 시 새로운 노드에 저장되는 데이터 선정방법에 있어서 이동체의 이동속성을 고려한 변경된 동적 분할정책을 제안한다.

#### 4.5.1 이동체의 이동속성을 고려한 분할도메인 선정방법

[정의 1]의 BR 은 노드 내의 OL 과 CL 을 포함하고 OL 의 이동속성에 따라 확장한다. BR 이 포함하는 CL 들의 이동속성이 유사할 경우 BR 은 한쪽 방향으로만 확장하게 되어 시간이 지남에 따른 BR 의 영역확장이 감소된다. 이 논문에서는 BR 의 네 변중 확장하는 변의 개수를 $S_{axis}$ 라 정의하고 이 값에 따라 공간분할 혹은 시간축비균등분할을 수행한다.

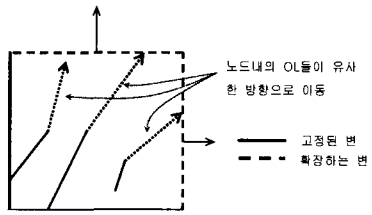


그림 8. BR 의 네 변 중 이동하는 변과 고정된 변

[그림 8]은 점선과 화살표로 표현된 OL 들이 유사한 이동속성을 가지게 됨에 따라 $S_{axis}$ 가 2, 즉 BR 의 두 변만이 확장되는 것을 보여준다.

이 논문에서는 $S_{axis}$ 의 값이 2 이하일 경우 시간축분할을 수행하고 그렇지 않을 때 공간분할을 수행한다.

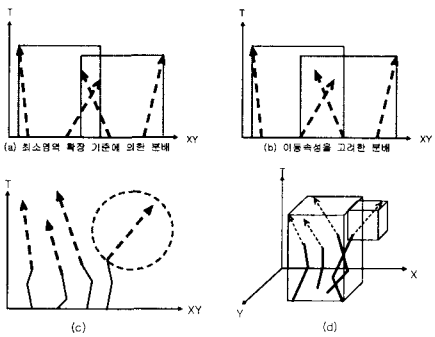


그림 9. 이동체의 이동속성을 고려한 분할정책

분할축 선정의 결과로 공간분할이 선택되면 4.4.1 절에서 소개한 $T_{estimate}$ 를 사용하여 OL 을 CL 로 취급하여 PickSeed[3] 알고리즘을 수행한 후 노드 내의 CL 들을 최소영역확장 기준에 의해 분배한 다음 남아있는 OL 들을 이동체의 이동속성을 고려하여 분배한다. [그림 9]의 (a)는 이동속성을 고려하지 않은 분배이며 이 경우 시간이 지남에 따라 중첩영역이 커지게 된다. (b)는 이동속성을 고려한 분배로써 분할시점의 중첩영역은 (a)에 비해 크지만 시간이 지남에 따라 BR 이 확장하게 되면 (a)의 경우보다 중첩영역이 작아지게 된다.

[그림 9]의 (c)는 시간축분할시 새로운 노드에 저장될 객체로써 OL 들의 이동속성 평균값과 가장 차이가 많은 객체가 선정된 것을 보여준다. 노드 내의 OL 들은 이동체의 다음 위치보고 시 CL 로 업데이트 되고 이때 이동속성이 유사한 OL 만을 포함하게 되면 BR 의 영역이 작아지게 된다. (d)는 이 논문이 제안하는 변경된 시간축분할의 결과를 보여준다.

## 5. 결론 및 향후 연구

이 논문에서는 영역질의 성능이 좋은 3DR-Tree 를 기반으로 하여 과거궤적과 현재 및 미래위치에 대한 검색을 지원하는 새로운 색인(HCR-Tree)을 제안하였고 이동체의 이동속성을 고려한 삽입과 분할정책, 그리고 이동체의 최근보고시간을 기준으로 과거와 현재 및 미래위치에 대한 질의를 처리하는 방법에 대해 설명하였다.

향후 연구로서 첫째, 분할도메인 선정 시 CL 들과 OL, 이동체의 이동속성을 모두 고려한 분할도메인 선정기준에 대한 연구가 필요하고 둘째, 이 논문이 제시하는 과거위치와 현재 및 미래위치를 동시에 저장하고 검색하는 기법을 3DR-Tree 뿐만 아니라 선분객체를 사용하는 다른 이동체 색인들에 적용하여 이 논문이 제안하는 방법이 질의성능에 미치는 영향을 연구해볼 필요가 있다. 마지막으로 이 논문에서 제안한 색인에 대한 구현과 이를 통한 성능평가가 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] 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

[5] Betty salzberg, vassilis j. tsotras, " Comparison of Access Methods for Time-Evolving Data" ACM Computing Surveys, Vol.31, No.2, pp.158-221, 1999

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

[7] 이창헌, 임덕성, 홍봉희, "KDB-Tree 를 사용한 이동체 색인의 동적 변경", 한국정보과학회 데이터베이스 연구회 2002학술발표논문집, 제18권 2호, p117-124