이동 객체 위치 예측 시스템을 위한 효율적인 미래 인덱싱 기법†

An Efficient Future Indexing Technique for the Moving Object Location Prediction System

이강준*, 김정준, 한기준
Kang-Joon Lee*, Joung-Joon Kim, Ki-Joon Han
건국대학교 컴퓨터·정보통신공학과
{kjlee*, jjkim9, kjhan}@db.konkuk.ac.kr

요 약

최근 도로 네트워크 환경에서 이동 객체 위치 정보를 관리하고 이동 객체의 미래 위치를 예측하는 이동 객체 위치 예측 시스템의 필요성이 나날이 증가되고 있다. 이동 객체 위치 예측 시스템은 교통 관제 및 다양한 응급 상황 시 이동 객체의 미래 위치를 신속히 예측하기 위해 사용되며, 보다 편리한 위치 기반 서비스의 제공을 가능하게 해준다. 이러한 시스템을 위한 대부분의 미래 인덱싱 기법은 일반적으로 이동 객체의 미래 위치 예측을 위해 과거 이동 궤적을 이용하고 있다. 그러나, 수많은 이동 객체의 과거 이동 궤적 관리가 어렵고, 실시간으로 변화하는 이동 객체의 미래 궤적을 반영하기 위한 방대한 미래 인덱스의 갱신 요청으로 인해 인덱스 유지 비용이 증가하여 미래 위치 질의 요청에 대한 신속한 처리 성능이 떨어지게 된다.

따라서 본 논문에서는 이동 객체 위치 예측 시스템에서 방대한 이동 객체의 과거 이동 궤적으로부터 효율적으로 미래 위치를 예측하기 위해 셀 기반의 미래 인덱싱 방법인 PFCT-Tree(Probability Future Cell Trajectory-Tree)를 제시한다. PFCT-Tree는 방대한 과거 이동 궤적을 셀 단위로 재구성하여 인덱스 크기를 줄이고, 셀 내부 경험치를 기반으로 장기간 질의 시 빠른 미래 위치를 예측할 수 있다. 또한 신속한 미래 이동 궤적의 갱신 속도를 향상시키기 위해 미래 시간을 미래 궤적과 분리하여 인덱싱함으로써 위치 예측 오류로 인한 미래 인덱스 갱신 비용을 최소화 할 수 있다. 마지막으로 실험을 통해 도로 네트워크 환경에서 PFCT-Tree가 기존 인덱싱 기법들보다 갱신 및 검색 성능이 우수함도 입증하였다.

1. 서론

최근 위치 정보 기술의 발달은 다양한 위치 기반 서비스(LBS: Location Based Services)의 발전을 가져왔다[1]. 위치 기반 서비스에서는 기본적으로 이동 객체 위치 정보 관리 인덱스를 사용하는데, 보다 효율적인 이동 객체 관리를 위해 이동 객체의 미래 위치를 예측할 수 있는 미래 인덱스의 필요성이 높아지고 있다[2,3].

도로 네트워크 환경에서 이동 객체의 위치를 효율적으로 예측하기 위해서는 과거의 이동 객체의 궤적을 이용한다. 이동 객체의 과거 궤적은 매우 근접한 미래 궤적을 예측하게 해주고, 예측된 미래 궤적

† 본 연구는 서울시 산학연 협력사업의 신기술 연구개발 지원사업의 지원으로 수행되었음.

을 기반으로 이동 객체의 미래 위치를 예측하게 한다. 이와 같은 미래 위치 예측 방법은 미래 궤적을 예측하여 생성하는 과정과 생성된 미래 궤적에서 미래 위치를 예측하는 과정으로 이루어진다. 이때 기존 미래 인덱스의 경우 방대한 과거 궤적에 대한 경험치를 유지하는 비용이 크고, 인덱싱된 미래 궤적이 도로 상황에 따라 빈번하게 변화되므로 이동 객체의 수가 증가할수록 미래 궤적의 갱신 비용이 커지는 문제점이 있다[4].

본 논문에서는 이러한 미래 인덱스의 문제점을 해결하기 위해 과거 궤적 경험치의 유지 비용을 줄이고, 미래 궤적의 빈번한 갱신 부하를 최소화하는 셀 기반의 미래 궤적 인덱싱 기법인 PFCT-Tree(Probability Future Cell Trajectory-Tree)를 제시한다.

본 논문의 구성은 다음과 같다. 제 2 장에서는 기존의 확률 기반 미래 궤적 예측 인덱스 및 미래 궤적 기반 미래 인덱스 기법에 대하여 분석한다. 제 3 장에서는 이동 객체의 미래 위치 예측을 위한 셀 기반의 미래 인덱스인 PFCT-Tree에 대해 설명하고, 제 4 장에서는 실험을 통한 성능 평가 결과를 알아본다. 마지막으로 제 5 장에서는 결론에 대해 언급한다.

2. 관련연구

본 장에서는 이동 객체의 미래 위치를 예측할 수 있는 기존의 확률 기반 미래 궤적 인덱스 및 미래 궤적 기반 미래 인덱스 기법에 대해 분석한다.

2.1 PLM(Prediction Location Model)

도로 네트워크 환경에서 이동 객체의 위치를 예측하는데 있어 과거 이동 객체의 경험은 매우 중요한 자료가 된다. 실세계에서 이동 객체는 주기성을 가지고 있는데, 예를 들어 출근길, 퇴근길, 업무상 다니는 길, 버스 노선, 지하철 노선 등 이동 객체는 자주 다니는 길을 이용하는 특성을 가지고 있다. 이런 주기성은 이동 객체의 미래 위치 예측 오차를 줄이는데 이용될 수 있다. 이런 경험적 데이타를 활용하여 서버는 객체가 다음에 이동할 도로 세그멘트를 예측할 수 있으며, 또한 예측 시간도 늘릴 수 있다.

PLM은 이동 객체가 과거 교차로에서 도로 세그먼트를 선택한 확률 통계를 이용하여 확률 매트릭스(Probability Matrix)를 관리한다[5]. 또한 PLM은 확률 매트릭스를 이용하여 이동 객체의 현재 위치로부터 가능한 궤적 탐색 트리를 생성하게 되며, 이러한 트리를 이용하여 이동 객체가 앞으로 지나갈 도로 세그먼트의 연결인 미래 궤적을 예측한다.

PLM의 확률 매트릭스는 주기성을 가지는 이동 객체의 미래 궤적을 예측하는데 매우 효율적인 확률적 접근 방법을 제시하고 있다. 이와 같은 방식은 교차점에서 이동 객체의 경험에 따른 미래 도로 세그먼트 예측을 가능하게 한다. 그러나 교차점마다 이동 객체별 확률 매트릭스를 관리하므로 확률 변수의 수가 많아지고, 예측 기간이 길어질수록 궤적 탐색 트리가 방대해지므로 미래 궤적 예측을 위한 탐색 시간이 길어지는 단점이 있다.

2.2 FT-Quadtree

FT-Quadtree는 미래 궤적을 X축-시간축, Y축-시간축 2개의 인덱스로 구성한다. 이렇게 구성된 인덱스에서 미래 궤적의 세그먼트들은 같은 영역을 공유하는 빈도수가 증가하는데, FT-Quadtree는 이를 이용하여 Quadtree 내의 공유된 세그먼트들을 하나의 대표 세그먼트로 관리하는데, 이를 공유 궤적 세그먼트(Shared Trajectory Segment)라 부른다[6]. 공유 궤적 세그먼트는 같은 지역, 같은 시간에 집중된 미래 궤적으로 인한 갱신 시 불필요한 노드 병합 및 분할을 줄여준다.

FT-Quadtree는 이동 객체의 시간 변화에 따른 인덱스 갱신에 대해 매우 효율적인 방법을 제시하고 있다. 또한, 같은 시간에 같은 공간을 차지하는 이동 객체의 궤적에 대해 생기는 수많은 Quadtree 분

할로 인한 갱신 부하를 효율적으로 처리하게 해준다. 그러나 서로 다른 이동 객체의 궤적이 각기 다른 시간 및 공간을 차지할 경우 인덱스 크기가 매우 커지고, 시간의 변화에 따른 인덱스 갱신 부하를 공유 궤적 세그먼트를 이용하여 부분적으로 줄일 수는 있으나, 시간 변화로 생긴 궤적이 다른 궤적들과 100% 공유되지 않기 때문에 결국 Quadtree 인덱스 갱신 부하가 발생하는 단점이 있다.

3. PFCT-Tree

본 장에서는 셀 기반 미래 인덱스인 PFCT-Tree의 구조, 셀 궤적 예측, 그리고 PFCT-Tree의 알고리즘에 대해 설명한다.

3.1 PFCT-Tree의 구조

기존 미래 인덱스는 미래 궤적 예측 속도가 느리다는 문제점과 특정 지역에 집중된 미래 궤적들을 인덱스에 삽입 시 연속적인 셀 분할이 일어나며, 빈번한 미래 궤적의 변화에 따른 인덱스 갱신 부하가 높은 문제를 가진다. PFCT-Tree는 이러한 문제점을 해결하기 위해 셀 단위 미래 궤적 관리 방법을 사용한다.

PFCT-Tree는 이동 객체의 미래 궤적 예측을 위해 과거 궤적을 셀 단위 궤적으로 저장하여 빠른 장기간 미래 궤적 예측을 가능하게 하고, 예측된 미래 궤적 관리를 위해 셀 단위 미래 궤적을 공간적인 속성과 비공간적인 속성으로 분리하여 인덱싱함으로써 이동 객체의 빈번한 시간 변화에 따른 인덱스 갱신 부하를 감소시킨다.

이를 통해 PFCT-Tree는 이동 객체의 경로가 서버에 미리 보고되지 않는 환경에서 이동 객체의 미래 위치 예측을 가능하게 해주고, 이동 객체가 이동 시 예측된 미래 셀 궤적과 현재 위치 및 시간을 점검하여 차이가 생길시 중앙 서버에 정보를 전송하여 미래 인덱스를 갱신하게 된다.

PFCT-Tree의 구조는 그림 1과 같다.

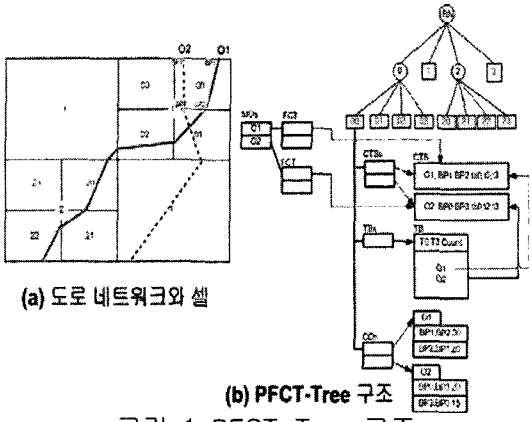


그림 1 PFCT-Tree 구조

그림 1에서 보는바와 같이 두 객체 O1, O2의 미래 궤적의 위치 정보를 셀 단위로 분할하여 PFCT-Tree에 저장하고, 미래 궤적의 셀 단위 진입 시간 및 진출 시간 정보를 PFCT-Tree의 시간 버킷에 저장하고, 진입, 진출 빈도를 셀 객체 CO에 저장하고 있다. 이동 객체의 미래 궤적은 여러 개의 셀을 지나게 된다. 이때, 도로와 셀 영역이 만나 진입한 경계점과 진출한 경계점을 만들게 된다. 이렇게 셀과 만나면서 생기는 경계점과 진입, 진출 시간을 셀 궤적 세그먼트 CTS라 부른다. 또한 이와 같은 CTS들이 모여 셀 궤적을 구성하며, 셀 궤적은 PFCT-Tree에 삽입된다.

3.2 PFCT-Tree의 셀 궤적 예측

PFCT-Tree는 미래 궤적 예측을 위해 셀 확률 매트릭스를 이용한다. 셀 확률 매트릭스는 이동 객체의 미래 궤적을 생성하기 위한 셀 확률 변수를 가지고 있다. 셀 확률 변수는 이동 객체가 지나간 경계점만을 가지고 생성되는데, 이러한 셀 확률 변수를 이용하여 미래 궤적 예측을 위한 탐색이 수행된다. 그림 2는 n개의 진입, 진출 도로 세그먼트를 가진 셀 c에서의 셀 확률 매트릭스를 보여준다.

$$M(C,t_k)=\begin{bmatrix} P(R_1|R_1) & P(R_2|R_1) & \cdots & P(R_n|R_1) \\ P(R_1|R_2) & P(R_2|R_2) & \cdots & P(R_n|R_2) \\ \cdots & \cdots & \cdots & \cdots \\ P(R_1|R_n) & P(R_2|R_n) & \cdots & P(R_n|R_n) \end{bmatrix}$$

그림 2. 셀 확률 매트릭스

그림 2에서 시간 t_k는 현재 시간을 뜻하며, R_1, R_2, ... R_n은 셀에 진입하고 진출

하는 도로 세그먼트를 의미한다. 또한 P($R_1|R_1$)은 현재 도로 세그먼트 R_1으로 진입한 후 R_1으로 진출할 확률을 의미하며, P($R_2|R_1$)은 R_1으로 진입한 이동 객체가 R_2로 진출할 확률을 뜻한다. 이와 같은 진입, 진출 데이타가 셀 확률 매트릭스의 확률 변수로 사용된다. Bayesian 이론에 의하여 식 1을 얻을 수 있다.

$$P(b, a) = P(b|a) \cdot P(a) \qquad (식\ 1)$$

그러므로 R_1->R_2로 갈 확률 P(R_2, R_1)는 P($R_2|R_1$) • P(R_1)이며, P(R_1)은 1이므로 P($R_2|R_1$)의 값을 셀 확률 매트릭스에서 구할 수 있다.

3.3 PFCT-Tree의 알고리즘

본 절에서는 PFCT-Tree의 자료 구조와 미래 궤적 예측, 삽입, 갱신, 검색 알고리즘에 대해 설명한다.

3.3.1 PFCT-Tree 자료구조

PFCT-Tree는 이동 객체를 관리하는 MO 구조체, 이동 객체의 미래 셀 궤적을 관리하는 FCT 구조체, 셀 관리를 위한 Cell 구조체, 셀 확률 정보 관리를 위한 CO 구조체, 미래 셀 궤적의 세그먼트 정보를 가지는 CTS 구조체, 그리고 시간 버킷을 관리하는 TB 구조체로 구성된다. 그림 3은 PFCT-Tree의 자료구조를 보여 준다.

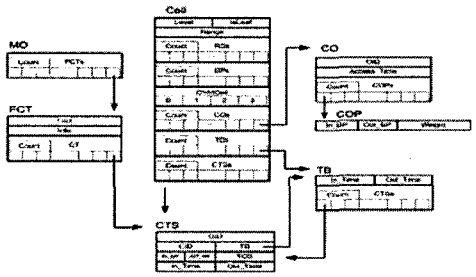


그림 3. PFCT-Tree 자료구조

MO 구조체는 이동 객체의 FCT(미래 셀 궤적) 리스트 FCTs를 가지고 있다. FCT는 CTS(셀 궤적 세그먼트)의 리스트 CTSs를 가지며, OID(객체 식별자), 객체 정보 Info를 가진다.

Cell은 셀 분할 레벨인 Level, 단말 셀 여부인 IsLeaf, 셀 영역인 Range, 셀 내부 도로 세그먼트들의 리스트인 RSs, 경계점 리스트인 BPs, 자식 셀 포인터인 ChildCell, 셀 객체 정보를 가지는 COs, 미래 시간 관리를 위한 TB(시간 버킷)의 리스트인 TBs, 그리고 CTSs를 가진다. CTS는 객체 OID, CID(셀 식별자), 셀로 진입한 시간과 진출한 시간을 관리하는 TB, IN_BP(진입 경계점), OUT_BP(진출 경계점인), 그리고 셀 내부의 궤적을 코드화하는 TCB(궤적 제어 블록)를 가진다.

TB는 시간 버킷 전체의 최소 진입 시간인 In_Time, 최대 진출 시간인 Out_Time, 그리고 CTSs를 가진다. CO(셀 객체)는 OID와 접근시간, 그리고 COP(셀 객체 확률 변수)들의 리스트를 가진다. COP는 IN_BP와 OUT_BP에 대한 접근빈도를 가진다.

3.3.2 미래 궤적 예측 알고리즘

그림 4는 PFCT-Tree의 미래 궤적 예측 알고리즘을 보여준다. 그림 4에서 라인 1은 결과 값을 초기화하며, 라인 2는 Depth가 주어진 Depth_Limit에 이르면 재귀적 호출을 종료한다. 라인 3~6은 주어진 oid를 가지는 COP의 총 빈도수 totw, 가장 빈도수가 큰 fcop, 두 번째로 빈도수가 큰 scop를 각각 구한다. 라인 7~9는 셀의 진입 경계점과 fcop, scop의 진출 경계점으로 이뤄진 셀 궤적을 후보 경로에 추가한다. 라인 10에서 미래 궤적 예측 알고리즘의 재귀 호출을 수행하며, 라인 11에서 미래 궤적일 확률이 가장 높은 궤적을 찾는다.

```
Function PFCT_Prediction( Depth, C, OID, In_BP, Prob, CT )
Input  Depth:Prediction Level, C:Cell, OID:Object ID,
       In_BP:Enter BP, Prob:Probability, CT:Cell Trajectory
Output nct:New Cell Trajectory
Define Static Variable Cand_Prob_CTs //Cell Trajectory List wit
       probability

Begin
 1. If Depth is zero then Cand_Prob_CTs initialize
 2. If Depth > Depth_Limit then //Depth_Limit is the prediction
       boundary depth limit
       Add (CT,Prob) to Cand_Prob_CTs and Return
 3. COP[] cops = COPs with in_bp of CO with oid of C
 4. Float totw = SUM(COP.Weight) in cops
 5. COP fcop = FIND COP with the largest Weight in cops
 6. COP scop = FIND COP with the secondly largest Weight in
       cops
 7. CT fct = Add New Cell Trajectory Segment CTS with (C,
       In_BP, fcop.Out_BP) to CT
 8. PFCT_Prediction( Depth+1, fcop.Out_BP,C, OID,
       fcop.Out_BP, Prob * fcop.Weight / totw, fct)
 9. CT sct = Add New CTS with (C, In_BP,scop.Out_BP) to CT
10. PFCT_Prediction( Depth+1, scop.Out_BP,C, OID,
       scop.Out_BP, Prob * scop.Weight / totw, sct)
11. CT nct = Find Cell Trajectory CT that have the highest
       probability value in Cand_Prob_CTs
12. Return nct
End
```

그림 4. 미래 궤적 예측 알고리즘

그림 5는 PFCT-Tree의 미래 궤적 삽입 알고리즘을 보여준다. 그림 5에서 라인 1,

2에서 입력된 궤적을 미래 셀 궤적 FCT로 변환한 후 MO 리스트에 추가한다. 라인 3에서는 FCT를 순회하여 셀 궤적 세그먼트 cts를 얻는다. 라인 4~9는 oid를 가지는 cell의 cop를 찾은 후 cop의 가중치를 증가한다. 라인 10에서 cts의 진입 시간 및 진출 시간을 저장할 시간 버킷 TB을 찾는다. 라인 11,12에서 새로운 tb의 생성 및 tb에 더 이상 cts를 저장할 수 없으면, split_time_bucket()함수를 호출하여 TB를 둘로 분할한다. split_time_bucket()는 out_time 기준으로 시간 버킷을 분할하는 동작을 수행한다. 라인 7,8에서는 cts를 tb와 cts의 셀에 저장한다.

```
Algorithm PFCT_Insert( OID, Traj )
Input OID:Moving Object ID, Traj : Trajectory
Begin
 1. PFCT fct = Change trajectory Traj to cell trajectory
 2. Add fct to MO
 3. For each Cell Trajectory Segment cts in fct do //Retrive all
      cell trajectory segment in future cell trajectory FCT
      gradually.
 4.   Find cell object co with OID at cts.Cell
 5.   If there is no co exists then
        CO co = New (CO) and Set OID to co
 6.   Set access time to co
 7.   COP cop = Find cell object probability variable that have
        cts.In_BP and cts.Out_BP in COPs of co
 8.   If there is no cop exists then
        COP cop = New (COP) and Set cts.In_BP and cts.Out_BP
        to cop
 9.   Increment cop.Weight by one
10.   TB tb = Find time bucket with times of cts at cts.Cell.TBs
11.   If there is no tb exists then
        TB tb = New (TB) and Set (cts.in_time, cts.out_time) to tb
        Add tb to cts.Cell.TBs
12.   If tb is full then split_time_bucket( cts, tb )
13.   Add cts to tb
14.   Add cts to cts.Cell
End
```

그림 5. PFCT-Tree 삽입 알고리즘

그림 6은 PFCT-Tree의 미래 궤적 갱신 알고리즘을 보여준다. 그림 6에서 라인 2에서는 기존의 미래 셀 궤적 CT의 미래 시간을 입력된 미래 셀 궤적 NCT의 시간으로 갱신한다. 라인 2~4에서는 셀 궤적 세그먼트 cts, ncts, 시간 버킷 t, 그리고 tb의 시간 범위인 it, ot를 얻는다. 라인 5~6은 tb의 시간 범위를 갱신한다. 라인 7은 빠른 검색을 위해 시간 버킷을 out_time순으로 정렬한다. 마지막으로 라인 8에서 cts의 시간을 변경한다.

```
Algorithm PFCT_Update( CT, NT )
Input CT,NT:Furture Cell Trajectory Segment
Begin
1. For i =0 to number of CTS in CT do
2.     CTS cts = CT[i] and CTS ncts = NT[i]
3.     TB t=cts.TB
4.     it = NT[i].in_time and ot = NT[i].out_time
5.     If it < t.in_time then t.in_time = it
6.     If ot > t.out_time then t.out_time = ot
7.     Sort t by out_time for fast binary search
8.     cts.in_time = it and cts.out_time = ot
End
```

그림 6. PFCT-Tree 갱신 알고리즘

그림 7은 PFCT-Tree의 미래 궤적 검색 알고리즘을 보여준다. 그림 7에서 라인 1~2에서 입력된 셀 영역 r이 검색 영역 sr 내에 있지 않고, Quadtree의 중간 노드면 함수를 빠져나간다. 라인 3에서 셀 c에서 입력된 시간 범위에 있는 모든 시간 버킷을 tbs에 구한다. 라인 5~6에서는 tbs의 모든 시간 버킷을 순회하여 시간 범위의 객체를 구한 후 result에 추가한다. 라인 7에서는 PFCT_Search 함수를 재귀적으로 호출한다.

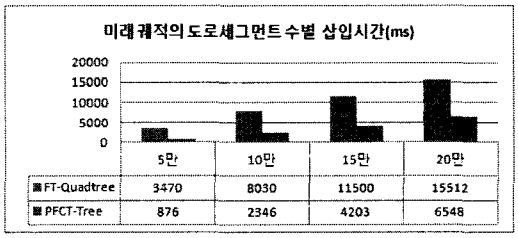
```
Function PFCT_Search( result, r, sr, tr, c )
Input result : OID List, r:Range, sr:Search Range, tr:Time
      Range, c : Cell
Begin
1. If r does not intersect with sr then Return
2. If c.Leaf then Return
3. TB[] tbs = Find time bucket tb with tr.in_time and
     tr.out_time at c.TBs
4. For each tb in tbs do
5.     oid = Search object id between tr.in_time and
       tr.out_time in tb
6.     Add oids to result
7. For i=0 to 3 do
     PFCT_Search( result, range of c.Cell[i], sr, tr, c.Cell[i] )
End
```

그림 7. PFCT-Tree 검색 알고리즘

4. 성능 평가

본 장에서는 본 논문에서 제시한 PFCT-Tree의 성능 평가를 살펴본다. 성능 평가는 20만개 도로 세그멘트 데이타의 삽입 시간, 갱신 시간, 검색 시간을 FT-Quadtree와 비교하였고 미래 궤적 예측 시간을 PLM과 비교하였다.

그림 8은 미래 궤적의 도로 세그먼트 수별 삽입 시간을 보여준다.



	5만	10만	15만	20만
FT-Quadtree	3470	8030	11500	15512
PFCT-Tree	876	2346	4203	6548

그림 8. 삽입 성능 평가

그림 8에서와 같이 PFCT-Tree는 셀 궤적에 대한 미래 시간 정보 및 가중치 삽입만이 이루어지므로 FT-Quadtree에 비해 상대적으로 삽입 시간이 매우 짧다.

그림 9는 인덱싱된 모든 미래 궤적의 시간을 100초씩 지연시킬 경우의 미래 궤적의 도로 세그먼트 수별 갱신 시간을 보여준다.

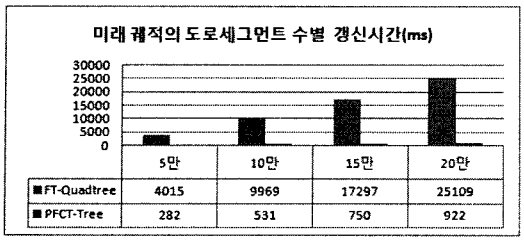


그림 9. 갱신 성능 평가

그림 9와 같이 PFCT-Tree는 갱신 시 별도의 인덱스 재구성이 없고 시간 버킷 내의 미래 시간 속성 값의 변경만이 발생하므로 FT-Quadtree에 비해 상대적으로 갱신 시간이 매우 짧다.

그림 10은 검색 성능 시간 평가를 보여준다.

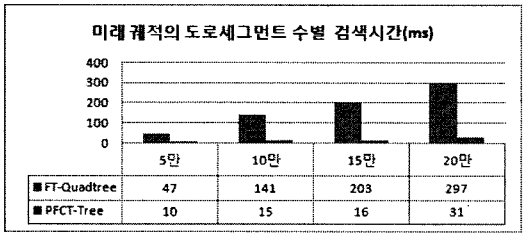


그림 10. 검색 성능 평가

그림 10과 같이 PFCT-Tree는 미래 궤적을 셀 단위로 관리하므로 노드 수가 적어 FT-Quadtree에 비해 공간 인덱스 검색이 매우 빠르며, 미래 시간 검색 시 미래 시간 버킷에 대한 이진 검색을 수행함으로 매우 빠른 검색이 가능하다.

그림 11은 예측 생성 시간 평가를 보여준다.

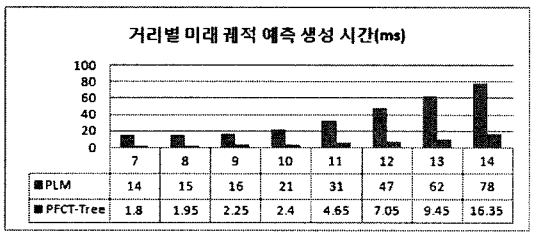


그림 11. 예측 성능 평가

그림 11과 같이 PFCT-Tree는 기존의 미래 궤적 예측 인덱스인 PLM에 비해 5배의 미래 궤적 예측 시간이 단축되는 것으로 나타났다.

5. 결론

이동 객체 위치 예측 시스템에서 미래 궤적 기반의 미래 인덱스는 이동 객체의 미래 위치 예측을 위해 방대한 과거 궤적 정보를 유지해야 되며, 수많은 이동 객체가 보내는 위치 정보 갱신 요구로 인한 서버 인덱스 갱신 부하는 전체 미래 인덱스 성능을 감소시킨다.

이와 같은 문제점을 해결하기 위해 본 논문에서는 효율적인 미래 궤적 관리를 위한 셀 기반의 미래 인덱스인 PFCT-Tree를 제시하였고, 또한 성능 평가를 통해 PFCT-Tree가 기존 미래 인덱싱 기법보다 우수함을 입증하였다. PFCT-Tree는 이동 객체 위치 예측 시스템에서 대용량의 이동 객체의 미래 위치 예측 요구를 신속하게 서비스하기 위한 미래 인덱스로써 사용될 수 있다.

참고문헌

[1] 최혜옥, "위치 기반 서비스 (LBS: Location-Based Services)," 제3회 공간정보 워크샵, 2002, pp.5-22.

[2] Agarwal, P. K., Arge, L., and Erickson, J., "Indexing Moving Points," Proc. of the Symposium on Principles of Database Systems, 2000.

[3] 이강준, *이동 객체 위치 예측을 위한 효율적인 궤적 인덱싱 기법*, 건국대학교 대학원 박사학위 논문, 2006.

[4] Civilis, A., Jensen, C. S., and Pakalnis, S., *Techniques for Efficient Road-Network-Based Tracking of Moving Objects*, DB Technical Report 10, 2005.

[5] Karimi, H. A., and Liu, X., "A Predictive Location Model for Location-Based Services," Proc. of the 11th ACM Int. Symp. on Advances in Geographic Information Systems, 2003, pp.126-133.

[6] Ding, R., Meng, X., and Bai, Y., "Efficient Index Maintenance for Moving Objects with Future Trajectories," Proc. of the 8th Int. Conf. on Database Systems for Advanced Applications, 2003, pp.183-192.