추출 연산을 활용한 이동 점 객체 색인 기법
(An Indexing Technique of Moving Point Objects using Projection)

정 영 진 † 장 승 연 † 안 윤 애 †† 류 근 호 †††
(Young Jin Jung) (Seung Youn Jang) (Yoon Ae Ahn) (Keun Ho Ryu)

요 약 현실 세계에서 시간에 따라 연속적으로 위치나 모양이 변하며 이동하는 데이타를 시공간 이동 객체라 한다. 기존의 이동 객체 색인은 R-트리의 구조를 가지기 때문에, dead space, overlap 등 R-트리의 문제점을 그대로 갖고 있을 뿐만 아니라 고려하는 초점에 따라 이 문제가 더 커진다. 따라서, 이 논문에서는 이 문제점을 해결하기 위하여 MPR-트리(Moving Point R-tree)를 제안한다. 제안된 MPR-트리는 추출 연산을 활용하여 특정 시점 질의 및 시공간 범위 질의를 효과적으로 처리하며, 동일한 이동 객체 위치를 시간에 따라 연결리스트로 연결하여 궤적 질의 처리를 용이하게 처리한다. 기존 이동 객체 색인과 비교한 실험으로부터 이동 객체 질의 처리 및 공간 활용에 대해 추출 연산이 유용하게 쓰임을 확인하였다. 제안된 MPR-트리는 LBS, GPS를 활용한 차량 관리 시스템, 항법 시스템 등 이동 객체 관리를 위한 시스템에서 활용될 수 있다.

키워드 : 시공간 데이타베이스, 이동 객체, 이동 객체 색인, 이동 객체 질의

Abstract Spatiotemporal moving objects are changing their positions and/or shape over time in real world. As most of the indices of moving object are based on the R-tree, they have defects of the R-tree which are dead space and overlap. Some of the indices amplify the defects of the R-tree. In the paper, to solve the problems, we propose the MPR-tree(Moving Point R-tree) using Projection operation which has more effective search than existing moving point indices on time slice query and spatiotemporal range query. The MPR-tree connects positions of the same moving objects over time by using linked list, so it processes the combined query about trajectory effectively. The usefulness of the Projection operation is confirmed during processing moving object queries and in practical use of space from experimentation to compare MPR-tree with existing indices of moving objects. The proposed MPR-tree would be useful in the LBS, the car management using GPS, and the navigation system.

Keywords : Spatiotemporal database, Moving objects, Indexing of moving objects, Spatiotemporal query

1. 서 론

현실 세계에서 시간에 따라 연속적으로 위치나 모양이 변하는 데이타를 시공간 이동 객체(spatio-temporal moving objects)라 한다[1, 2].

이러한 이동 객체를 관리하기 위해서는 시간에 따라 연속적으로 변경되는 객체의 정보를 모두 저장해야 하기 때문에, 많은 데이타가 저장되고, 저장된 많은 정보를 효율적으로 검색하는 인덱스 구조가 필요하다.

기존의 이동 객체 색인들은 처리하는 데이타에 따라 크게 두 부분으로 나눈다. 첫째는 이동 객체의 과거 이력정보 및 궤적의 색인이고, 두 번째는 이동 객체의 현재 위치와 가까운 미래 위치의 색인이다[3]. 그리고 이런 응용에 대한 대부분의 이동 객체 색인은 R-트리[4]를 응용한 구조를 가지기 때문에, dead space, overlap 등 R-트리의

이 연구는 대학 IT 연구센터 육성 지원사업의 연구결과로 수행되었음.
† 학생회원 : 충북대학교 전자계산과
yleong@ablab.chungbuk.ac.kr
syjang@ablab.chungbuk.ac.kr
†† 비 회 원 : 충북대학교 전자계산과
yeahn@dblab.chungbuk.ac.kr
††† 종신회원 : 충북대학교 전기전자 및 컴퓨터공학부 교수
khryu@ablab.chungbuk.ac.kr
논문접수 : 2002년 4월 22일
심사완료 : 2002년 10월 25일

문제를 그대로 가지고 있다. 이동 객체의 좌표와 궤적을 둘 다 고려할 경우와 궤적만을 고려하여 노드를 구성할 경우, 오히려 overlap, dead space 등이 더 커지는 경우도 있다.

이 문제점을 해결하기 위하여 이 논문에서는 기존의 R-트리를 수정한 MPR-트리(moving point R-tree)를 제안한다. 제안된 MPR-트리는 3차원 MBB(Minimum Bounded Box)의 각 축에 대한 추출(projection) 연산을 수행하여 MBB의 하위 노드들을 각 축에 대해 순서대로 정렬하여 저장함으로써, dead space를 줄이고 이동 점 객체의 과거와 현재에 대한 특정 시점 질의 및 시공간 범위 질의를 효과적으로 처리한다. 또, 동일한 이동 객체의 위치를 시간에 따라 연결 리스트(linked list)로 연결하여 결합질의 등 궤적 질의를 용이하게 처리할 수 있도록 설계한다. 그리고, 각각의 이동 객체마다 가장 최근의 정보를 저장하고 최신의 속도와 방향을 고려한 삼각함수를 사용하여 가까운 미래의 대략적인 위치를 제공한다.

이 논문의 전체적인 구성은 다음과 같다. 먼저 2장에서는 이동 객체 질의와 이동 객체에 대한 색인들을 소개하고 문제점을 알아본다. 3장에서는 이 논문에서 제안한 MPR-트리의 구조에 대해 설명한다. 4장에서는 추출 연산의 개념 및 그 쓰임에 대해 예를 들어 설명한다. 5장에서는 MPR-트리에서 사용된 삽입, 분할, 검색 알고리즘에 대해서 살펴본다. 6장에서는 이동 객체 관련 질의에 대해 MPR-트리와 기존의 STR-트리, R-트리, 그리고 CR-트리와 비교 실험한다. 마지막으로 7장에서는 결론을 맺는다.

2. 관련 연구

이동 점 객체 색인 구조의 관련 연구로 이동 객체 질의, 그리고 이동 객체 색인에 대한 기존의 연구들을 소개하고 문제점을 파악한다.

이동 객체는 시간이 흐름에 따라 객체의 위치나 영역 같은 공간 정보가 연속적으로 변화하는 객체를 말하며 크게 이동 점(moving point)과 이동 영역(moving region)으로 나눌 수 있다[5]. 이동 점은 시간에 따라 객체의 위치가 변하는 것으로 이런 이동 점의 예는 택시, 배, 비행기, 사람 등이 있고, 이동 영역은 시간에 따라 객체의 위치뿐만 아니라 모양까지 변하는 것으로 이러한 이동 영역의 예는 적조현상, 바다에서의 기름 유출에 따른 오염, 오존층의 변화, 폭풍의 이동 경로 등이 있다.

이동 객체 질의는 크게 좌표 기반 질의(coordinate-based query)와 궤적 기반 질의(trajectory-based query)로 나눈다[3, 6]. 좌표 기반 질의는 객체 좌표에 대한 질의들로 2차원 평면에 시간 차원을 더한 3차원 공간상의 점(point), 범위(range), 최근접 질의(nearest-neighbor query)가 있다. 예를 들면, "2001년 12월 5일 오후 1시에 어린이 대공원 안에서 놀이객들의 흐름을 파악하시오.", "2001년 10월 3일 오후 1시에 청주 시청에서 가장 가까운 휴양림을 찾으시오" 등이 있다. 궤적 기반 질의는 객체의 움직임에 대한 질의로 "2001년 8월 5일 오전 10시에 청주를 떠나는 택시의 궤적을 찾으시오"와 같이 객체의 이동 정보를 포함하는 위상 질의(topological query)와 "2001년 9월 10일 오전 11시 이후로 북쪽을 향해 100km/h 이상의 속도로 움직이는 객체를 검색하시오."와 같이 객체의 이동에 따라 얻을 수 있는 속도와 방향처럼 객체의 이동에 의해서 유도되어 얻을 수 있는 정보를 포함하는 항법 질의(navigational query)가 있다. 이와 같은 이동 객체 질의는 시간, 공간 및 시공간 변화를 저장하는 자료들을 요구하기 때문에 많은 데이타 접근이 필요하고 사용자에게 빠른 응답을 주기 위하여 효과적인 이동 객체 색인이 필요하다.

기존의 이동 객체 정보를 다루는 색인들은 응용에 따라 크게 두 가지로 나눈다[3]. 그것은 이동 객체의 과거 이력정보 및 궤적의 색인과 이동 객체의 현재 위치와 가까운 미래 위치의 색인이다.

과거 이력 정보 및 궤적의 색인은 Pfoser가 제안한 STR-트리(Spatio-Temporal R-tree)[7], TB-트리(Trajectory Bundle tree)[3] 그리고, 이계영이 제안한 CR-트리(Combined R-tree)[8] 등이 있다. 3차원 MBB(Minimum Bounded Box)구조를 갖는 R-트리에 이동 객체의 궤적 보호 및 공간 속성을 고려한 색인 구조가 STR-트리이다. STR-트리는 저장되는 객체의 수가 많아지면, 성능이 떨어지고, 궤적에 관련된 질의를 제외하면, R-트리보다 좋지 않은 성능을 보인다. TB-트리도 3차원 MBB 구조를 갖지만, 좌표를 생각지 않고 단지 궤적만을 고려한다. 그리고 연결 리스트를 사용하여 이동 객체의 궤적을 시간 순서대로 연결함으로써 궤적질의를 용이하게 처리한다. 트리 구조는 STR-트리와 같지만 궤적 보호를 특히 중요하게 생각하기 때문에, 시공간 범위 질의에서 STR-트리와 R-트리보다 검색 속도가 느리다. CR-트리는 기존의 R-트리에 TB-트리와 같이 연결 리스트를 사용하여 궤적을 연결한 구조를 갖고 있다. 따라서 시공간 범위 질의에서 R-트리 만큼의 성능을 가지며 연결리스트를 사용하여 궤적질의에서도 효과적이다. 그러나 R-트리와 TB-트리를 혼합한 형태이기 때문에 STR-트리나 TB-트리보다 많은 저장공간을 가지는 단점이 있다.

현재 위치 및 가까운 미래의 위치를 다루는 색인은 TPR-트리(Time Parameterized R-tree) 등이 있다[9].

TPR-트리는 이동 객체의 속도와 방향을 고려하여 대략적인 가까운 미래의 위치를 계산해내고, 계산된 위치까지 고려하여 트리를 구성함으로써, 시간에 따라 변화하는 트리의 구조를 좀 더 안정화시켰다.

이동 객체의 좌표와 궤적을 둘 다 고려할 경우와 궤적만을 고려하여 노드를 구성하는 R-트리의 응용들은 오히려 overlap, dead space 등의 단점이 더 커지게 되어 효과적인 검색이 어려워진다. 이 논문에서는 이 문제를 해결하기 위하여 MPR-트리를 제안하였다. 이런 이동 객체 색인들은 위치기반 서비스(Location Based Services) 및 전장 정보 분석[10, 11] 등의 응용 분야에서 시간과 공간 속성을 포함한 시공간 질의 처리[12, 13]를 위해 효과적인 데이타 검색을 해준다.

3. MPR-트리의 구조

MPR-트리는 기존의 R-트리를 기반으로 추출 연산을 사용하여 특정 시점 질의 및 시공간 범위 질의를 효과적으로 처리하는 이동 점 객체 색인이다. 그림 1은 R-트리를 사용하여 이동 객체 궤적을 나타낸 것이다. 이동 객체의 이동을 표현하는 라인 세그먼트를 3차원 MBB로 표현하여 이동 객체의 궤적을 저장한다.

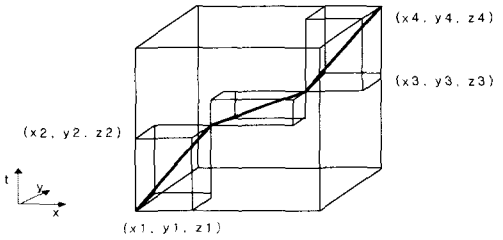


그림 1 R-트리를 사용한 이동 객체 궤적의 표현

하위 노드들로 표현된 라인 세그먼트의 끝점을 서로 연결하면 궤적이 되며, 이동 객체가 지나온 경로를 나타낸다. MPR-트리는 R-트리에서 dead space 및 overlap에 대한 불 필요한 검색을 줄이기 위해 각 축마다 Projection을 갖고, 이동 객체 궤적에 대한 질의를 효과적으로 처리하기 위해 리프 노드 엔트리를 연결 리스트로 연결한 트리이다.

MPR-트리는 그림 2와 같은 넌-리프 노드, 리프 노드, 리프 노드 엔트리로 구성되며, 리프 노드 엔트리를 제외한 노드들은 각 축에 대한 Projection을 가진다. 이 Projection은 하위 노드들의 정보를 각 축에 대해 순서대로 저장하는 곳으로 각 축마다 한 개의 Projection을 가진다. 그리고 각 리프 노드 엔트리는 앞, 뒤 시간대의 동일한 객체를 연결 리스트로 연결하여 궤적에 대한 질의를 처리할 때, 따로 다른 노드를 검색할 필요 없이 바로 처리할 수 있게 한다. 또, 입력된 이동 객체마다 가장 최근의 정보를 저장하고 속도와 방향을 고려한 삼각함수를 사용하여 대략적인 미래의 위치를 제공한다.

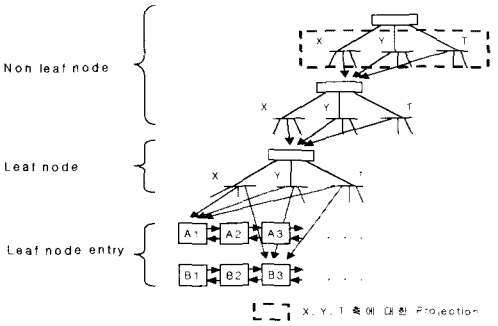


그림 2 MPR-트리의 구조

넌-리프 노드와 리프 노드는 x, y, t 축에 대한 Projection을 통해 하위 노드를 가리킨다. 그리고, 리프 노드 엔트리는 앞, 뒤 시간대에 대해서 연결 리스트로 연결되어 있기 때문에, 이동 객체 궤적에 대한 질의를 처리할 때 유용하다.

이동 객체의 이동을 저장하기 위해 그림 3과 같이 직육면체 형태의 MBB를 사용하여 샘플링 된 이동 객체의 위치를 라인 세그먼트로 저장한다. 이 MBB는 R-트리에서와 마찬가지로 최소 경계 사각형을 말한다. 리프 노드 엔트리는 x, y, t 축에 대한 각각의 경계로 구성되며, 시간 t1의 위치와 시간 t2의 위치에 대한 두 개의 점을 저장한다.

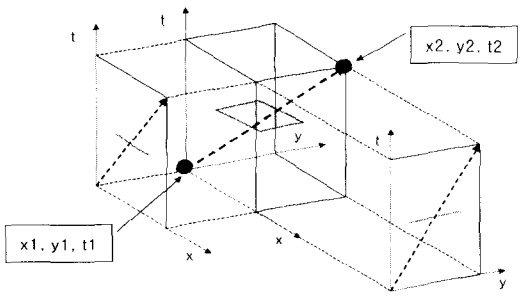


그림 3 리프 노드 엔트리의 구조

MPR-트리의 리프-노드 엔트리의 구조는 그림 3과 같다. 가운데의 작은 사각형과 같은 특정 시점에 대한 질의를 처리할 때, 3차원 공간상의 직육면체를 xt-projection

과 yt-projection을 통해 2 개의 평면으로 나타내고, 각각 x축과 t축, y축과 t축에 대한 보간법을 사용하여 t1과 t2 사이의 시점에 대한 위치를 계산한다. 계산된 위치가 xt 면과 yt 면에서 질의 범위 안에 존재하면, 해당 엔트리의 객체는 특정 시점에 대한 질의를 만족한다.

MPR-트리에서 리프 노드와 넌-리프 노드의 구조는 동일하다. 노드의 데이타를 x축, y축, t축에 대해 추출 연산을 하고 정렬하여, 각 축에 대해 순서대로 저장한다.

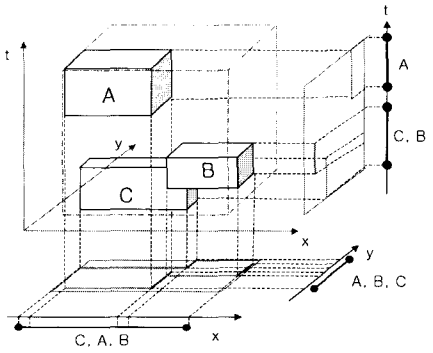


그림 4 리프 노드와 넌 리프 노드의 구조

그림 4를 보면 MPR-트리의 한 노드가 Projection을 사용하여 하위 노드를 포함하는 방식을 보여준다. 3차원 MBB 안에 A, B, C 라는 하위 노드가 있을 때, x, y, t축에 대해 추출 연산을 하며 각 축에 대한 Projection마다 하위 노드를 순서대로 저장한다.

4. MPR-트리의 노드 추출 연산

제안된 MPR-트리에서 하위 노드를 좀 더 효과적으로 검색하기 위해 사용된 추출 연산을 설명한다.

4.1 추출 연산

추출 연산은 삽입 연산 중에 MBB안에 포함된 하위 노드들의 경계를 추출하고 그 경계를 MBB의 각 축의 Projection에 순서대로 정렬하여 저장한다. 그리고, 각 축에 대한 Projection은 정렬된 하위 노드들의 공통된 경계를 저장하는 곳으로 특정 시점 질의 및 시공간 범위 질의에서 효과적인 검색을 하기 위해 사용된다. 이런 Projection을 만드는 추출 연산은 그림 5와 같다.

한 MBB에서 하위 노드를 저장하는 추출 연산은 하위 노드의 경계를 각 축에 대해서 수행한다. 각 축에 대한 Projection은 하위 노드의 정보를 순서대로 정렬하여 저장하기 때문에 특정 시점에 대한 질의나 시공간 범위 질의를 할 때, 질의 범위에 해당하지 않는 노드에 대한 불필요한 검색을 줄여서 검색의 효율을 높일 수 있다.

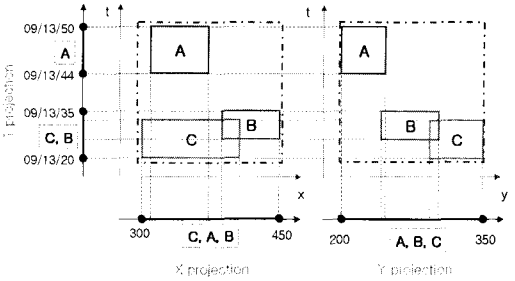


그림 5 MPR-트리의 추출 연산

표 1 MBR과 각 노드들의 경계 정보

	X 축	Y 축	T 축	
			From_time	To_time
MBB	300, 450	200, 350	2001/09/09/13/20	2001/09/09/13/50
A	310, 380	200, 260	2001/09/09/13/44	2001/09/09/13/50
B	390, 450	250, 320	2001/09/09/13/28	2001/09/09/13/35
C	300, 400	300, 350	2001/09/09/13/20	2001/09/09/13/30

그림 5에 있는 각 노드의 경계 정보를 정리하면 표 1와 같다. 기본적인 구조는 R-트리와 같기 때문에 상위 노드 MBB은 하위 노드 A, B, C 의 시공간 경계를 모두 포함한다.

표 2 MBB의 각 축에 대한 Projection 정보

Projection 축	범 위		객체 pointer
X 축	300	450	C, A, B
Y 축	200	350	A, B, C
T 축	2001/09/09/13/20	2001/09/09/13/35	C, B
	2001/09/09/13/44	2001/09/09/13/50	A

상위 노드 MBB의 Projection 정보를 정리하면 표 2와 같다. Projection의 구성을 보면, 하위 노드들의 경계 정보를 모아 특정 간격 안에 해당되는 객체를 순서대로 저장한다. 이 작업을 반복해서 하위 노드들을 모두 각 축에 대하여 순서대로 저장한다. 이 정보는 질의할 때, 각 축에서 질의 범위 안에 해당되는 값을 빠르게 찾아주며 dead space 및 overlap으로 인한 불필요한 검색을 줄일 수 있다. dead space를 최소화하는 것은 검색 할 때 알맞은 데

이타를 찾지 못하는 확률[14]을 감소시킨다.

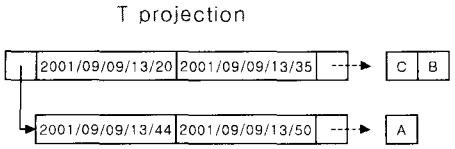


그림 6 T-Projection의 예

t축에 대한 Projection 정보는 그림 6을 보면 알 수 있다. t축에 대한 추출 연산의 장점을 예를 들어 설명한다. 하위 노드 A, B, C를 t축에 대해 추출했을 때 C 노드와 B 노드의 경계가 겹치기 때문에 "2001/09/09/13/20" ~ "2001/09/09/13/35" 와 같은 경계를 만들고, C와 B를 t축에 대한 순서대로 저장하였다. A는 C, B와 경계가 서로 떨어져 있기 때문에, 또 다른 경계를 가진다. 이와 같은 경계는 특정 시점이나 범위 질의를 할 때, 유용하게 쓰인다. 저장된 모든 하위 노드를 검색할 필요 없이 질의 범위와 overlap되는 각 축의 Projection을 찾아 검색하기 때문에 검색의 효율성을 증가시킨다. 이 논문에서는 이동 객체 정보가 시간에 따라 무한정 들어온다고 가정하기 때문에, 공간에 대한 정보보다 시간에 대한 정보를 먼저 처리하는 것이 검색 시간을 줄일 수 있다. 따라서 시공간 범위 질의나 특정 시점에 대한 질의를 처리 할 경우 T-Projection → X-Projection → Y-Projection 순으로 찾는다. 이를 위해 입력과 분할 알고리즘에서도 앞과 같은 순서를 우선시 하여 처리한다. 만약 MPR-트리의 구조를 적용하는 응용분야가 공간에 대해, 또는 특정 축에 대해서 넓은 범위가 존재할 경우 Projection을 검색하는 순서를 다르게 하여 검색 효율을 높일 수 있다.

4.2 추출 연산을 이용한 검색

이 절에서는, 특정 시점에 대한 시공간 질의를 처리할 때 추출 연산이 어떻게 유용하게 쓰이는 지를 표 2의 데이타를 활용하여 설명한다.

(질의 1) "2001년 9월 9일 오후 1시 30분에 X 축 : 200 ~ 320, Y 축 : 250 ~ 300의 범위 안에 있는 객체를 모두 검색하시오"

표 3 질의 1 의 각 축에 대한 정보 체크

검색하는 축	조건에 만족하는 객체
T	C, B
X	C, A, B
Y	A, B, C

질의 1에 답하기 위해 체크된 각 축과 하위 노드 정보는 표 3과 같다. 검색할 때, 먼저 t축에 대한 Projection에서 질의 범위를 체크한다. 각 Projection의 링크는 하위 노드들의 집합을 가리키고 있으므로, 해당되는 Projection의 링크를 찾을 때, 단지 각 링크의 경계만을 체크하기 때문에 그 링크에 해당하는 하위 노드들을 검색하지 않는다. 질의한 시점을 포함한 경계 ("2001/09/09/13/20"~"2001/09/09/13/35")를 찾게 되면 해당 노드의 x, y 축의 Projection에서 같은 방식으로 찾아 질의에 답한다. 그 후에 T-Projection의 다음 링크의 경계 ("2001/09/09/13/44"~"2001/09/09/13/50")와 비교했을 때, 질의한 시점을 포함하지 않기 때문에 검색을 마친다. 각 Projection의 링크는 시간에 대해서 정렬되어 있으므로 해당 범위를 지나면 더 이상 검색할 필요가 없어, 검색을 마치게 된다. 결국 C, B 노드만 검색하여, 질의에 답한다.

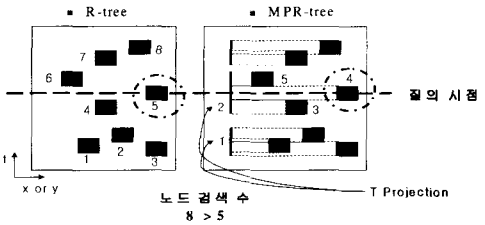


그림 7 질의 1에 대한 노드 검색 수 비교

질의 1에 대한 검색을 R-트리와 비교하면 그림 7과 같다. R-트리는 MBB에 포함된 하위 노드를 모두 검색해야 하지만, MPR-tree는 각 축에 따라 정렬된 하위 노드의 경계정보를 갖고 있기 때문에 모든 노드를 검색할 필요가 없다.

(질의 2) "2001년 9월 9일 오후 1시 40분에 객체의 위치를 모두 검색하시오"

표 4 질의 2의 각 축에 대한 정보 체크

검색하는 축	조건에 만족하는 객체
T	

질의 2에 답하기 위해 각 축에 대해 체크된 객체는 표 4와 같다. 우선 T-Projection 에 대해서 체크했을 때, 단지 T-Projection의 링크의 경계를 검색하는 것에 의해서 조건에 만족하는 객체가 없다는 사실을 알 수 있다. 만약 하위 노드 집합의 경계를 저장하는 Projection이 없다면, 검색되는 노드의 하위 노드를 모두 검색한 후에 질의를

마치게 되어 불필요한 검색을 하게 된다.

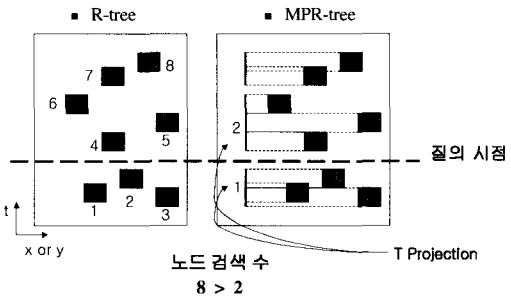


그림 8 질의 2에 대한 노드 검색 수 비교

질의 2에 대한 검색을 R-트리와 비교하면 그림 8과 같다. 이것은 dead space에 대한 질의이다. 하위 노드의 공통된 경계를 저장하는 Projection이 유용하게 쓰임을 알 수 있다. 또한 정렬된 정보는 dead space 뿐만 아니라 overlap된 부분에 대한 추가적인 검색에서도 위와 같은 이유로 불필요한 검색을 줄인다.

5. MPR-트리의 연산 알고리즘

이장에서는 MPR-트리의 삽입, 분할, 검색 등의 연산 알고리즘을 설명한다. MPR-트리는 C 언어로 구현되었으며, 시간에 따라 위치가 변하는 이동 점 객체의 과거와 현재의 위치 질의를 다루며, 공간은 유한하고, 시간은 무한하다고 가정한다. 따라서 검색할 때, 알고리즘에서 t축 -> x축 -> y축의 순서로 데이타를 입력, 분할, 검색한다. 이 연산은 시간에 대한 제약사항을 먼저 처리하여 상대적으로 많은 시간에 대하여 불필요한 후보들을 먼저 제거한다.

5.1 삽입 알고리즘

삽입 알고리즘의 목적은 저장된 데이타 중, 입력받은 이동 객체의 시간과 가장 가까운 시간대의 노드를 찾아 저장하는 것이다. 이렇게 하면 각 노드를 시간에 따라 분류하기 좋기 때문에, 시간에 대한 범위를 검색하기 좋다. 일단 시간에 대한 순서에 맞게 노드를 정하고, 해당 노드의 x축, y축의 순서를 고려하여 각 축의 Projection에 저장한다.

MPR-트리의 삽입 알고리즘은 알고리즘 1에서 알 수 있다. 입력받은 리프 노드 엔트리의 시간대와 가장 가까운 노드를 찾아 입력한다. 일단 저장할 노드가 정해지면 각 축에 대한 Projection에 정렬하여 저장한다. 그리고 궤적에 대한 질의 처리를 위해 동일한 객체 넘버(the identifying number of object)를 갖는 객체를 찾아 연결 리스트로 연결한다. 또, 가까운 미래의 위치 질의를 위해서 가장 최근의 이동 객체 정보를 갱신한다.

5.2. 분할 알고리즘

분할 알고리즘은 한 노드의 객체들이 서로 다른 시간대에 있으면, 시간에 따라 과거의 객체는 과거의 노드로 최근의 객체는 최근의 노드로 나누어 저장한다. 그리고, 만약 한 노드의 객체들이 모두 같은 시간대에 있다면, 검색

```
Algorithm insert_node(class node *root, class node *entry)
input : root// Node of MPR-tree
        entry    // Information of moving objects
method:
  if root is null then
     make root's bound with entry's bound
     make x, y t projection
  else
     if entry's bound beyond root's bound then
        modify root bound with entry bound
     endif
     if root->node_count < M then
        if root is the parent node of the entry then
           add a entry to x, y, t projection of root in sort of time
        elseif
           find child node of root to meet a close entry
           insert_node(found child node of root, entry)
        endif
     endif
  endif
  connect to trajectory of same objects which have same ono for trajectory query
  update lastest information of moving objects for near future query
end
```

알고리즘 1 MPR-트리의 삽입 알고리즘

순서에 따라서 t축 다음 순서인 x축에 따라 분할한다.

```
Algorithm split_node(class node *root)
input: root  // node to split
method:
    if root has brother node then
        put brother nodes of root into node_array
    endif
    if all child nodes of root have same time period then
        put child nodes of root into imsi_node1, imsi_node2 in
        order of x coordinate
        if root is top then
            set root with null
            insert_node(root, imsi_node1)
            insert_node(root, imsi_node2)
        else
            insert node_array into top of tree
            insert_node(top of tree, imsi_node1)
            insert_node(top of tree, imsi_node2)
        endif
    else
        put child nodes of root into imsi_node1, imsi_node2 in
        order of time
        if root is top then
            set root with null
            insert_node(root, imsi_node1)
            insert_node(root, imsi_node2)
        else
            insert node_array into top of tree
            insert_node(top of tree, imsi_node1)
            insert_node(top of tree, imsi_node2)
        endif
    endif
end
```

알고리즘 2 MPR-트리의 분할 알고리즘

MPR-트리의 분할 알고리즘은 알고리즘 2와 같다. 궤적에 대한 결합 질의 등을 처리할 때는 특정 시점 질의와 같은 객체마다 연결된 연결 리스트를 통해 쉽게 검색할 수 있기 때문에, 분할 알고리즘에서는 단지 특정 시점 질의나 시공간 범위 질의의 효율적인 검색을 위하여 시간과 공간만을 고려하여 분할한다. 이때, MBB의 각 Projection이 정렬된 하위 노드의 정보를 갖고 있기 때문에, R-트리보다 더 적은 비용으로 분할 할 수 있다.

5.3. 검색 알고리즘

검색 알고리즘은 삽입, 분할 알고리즘에서 고려한 바와 같이 시간(t축)에 대해서 먼저 검색하고, x축, y축 순으로 각 축의 Projection을 따라 검색한다.

노드의 Projection을 활용한 시공간 범위 검색 알고리즘은 알고리즘 3과 같다. t축, x축, y축에 대한 Projection을 모두 활용하는 것은 질의 처리할 때, 오히려 좋지 않은 성능을 나타내기 때문에, 여기서는 시간이 무한대인 것을 고려하여 t축에 대한 Projection 만을 고려하였다. x축, y축에 대한 추출 연산은 공간 범위만을 고려한 질의에서 효과적으로 사용될 수 있다.

6. 실험 및 평가

이 장에서는 시공간 범위 질의 등과 같은 이동 객체 관

```
Algorithm search_node(class node *root)
input:  root      // non leaf node or leaf node
output: node  // leaf node entry ( the result of search )
method:
  input spatio-temporal range bound to query
  check root bound with inputted query range bound
  search_time_projection(root, query)
end

Algorithm search_time_projection(class node *root, class range *query)
input:  root    // non leaf node or leaf node
        query   // query range
method:
  for each projection of time projection of root
      if time period of query include time projection node of root then
          if root is leaf node entry then
              if root node is included in query bound then
                  print root node      // result
                  return;
              else if spatial bound of query intersect spatial bound then
                  search_time_projection(intersected child node, query)
              endif
          else
              search_time_projection(intersected child node, query)
          endif
      else if time period of query intersect time projection node of root then
          search_time_projection(intersected child node, query)
      endif
  endfor
end
```

알고리즘 3 MPR-트리의 검색 알고리즘

련 질의에 대해서, 이 논문에서 구현한 MPR-트리의 노드 접근 수를 기존의 이동 객체 색인과 비교 실험하였다.

6.1 실험 환경

실험에 사용한 기존의 이동 객체 색인은 STR-트리, R-트리, CR-트리이고, 질의 종류로는 특정 시점 질의, 시공간 범위 질의, 궤적 관련 질의를 사용한다. 실험에 사용되는 이동 객체 데이타는 10분 간격으로 이동 객체의 위치와 속도를 랜덤하게 생산하는 이동 점 객체 데이타 생성기를 사용하였다. 질의할 때 변경하는 데이타는 다음과 같다.

표 5 질의 매개 변수

매개변수	내 용
이동 객체의 수	이동 점 객체 데이타 생성기에서 이동 객체를 생성하는 시간을 늘려서 많은 수의 이동 객체 정보가 입력됐을 때, 각각에 질의에 대한 노드 접근 수를 체크한다.
특정 시점에 대한 질의의 경우 -질의 시간대	질의하는 시간대를 변경시켜, 전체 시간의 범위 중 특정 시점 질의에 대한 노드 접근 수를 체크한다.
시공간 범위 질의의 경우 -질의 범위	시간과 공간에 대해서 질의하는 범위를 변화시키면서 노드 접근 수를 체크한다.
궤적에 대한 결합 질의의 경우 -질의 범위	질의하는 시간대를 변경시켜, 시간 범위에 따른 노드 접근 수를 체크한다.

실험에서 이동 객체 질의와 관련하여 고려될 매개변수는 표 5와 같다. 궤적과 관련한 결합 질의는 특정 시점 질의나 시공간 범위 질의를 먼저 처리한 후에 각 엔트리에 연결된 연결 리스트에 따라 검색하면 되므로, 이 장에서는 특정 시점 질의와 시공간 범위 질의를 주로 다룬다.

6.2 이동 점 객체 데이타 생성기

나라별 경계, 호수, 도로, 공원, 등과 같은 공간 객체들은 웹에서도 공개된 실제 데이타를 구할 수 있다. 하지만 자동차나 비행기와 같은 이동 객체에 대해서는 공개된 실제 데이타가 없으므로 이동 객체와 관련한 응용 분야의 실험에서는 보통, GSTD(Generator of Spatio-Temporal Datasets)[15, 16]를 사용하기도 한다. GSTD는 이동 객체 수, 객체의 속도, 객체의 이동 범위 등을 매개변수로 받아, 그에 맞게 이동 객체 데이타를 생성한다. 또한, GSTD는 빌딩과 같이 거의 위치가 변하지 않는 기반구조도 생성하여, 되도록 실제 데이타의 움직임을 나타내도록 하였다. 이런 기반구조는 이동 객체 위치에 대한 질의 처리에서 고려되도록 하였다.

실험에 사용된 이동 점 객체 데이타 생성기는 GSTD에서 다루지 않은 일반적인 시간에 대한 정보까지 활용하여 데이타를 생산한다. 이는 질의할 때, "2001년 10월 9일 오전 7시 32분의 이동 객체 위치를 검색하시오"와 같이 좀 더 실제와 같이 질의 할 수 있다. 이동 점 객체 데이타 생성기는 이동 객체가 자신의 위치를 10분 간격으로 샘플링하여 보낸다고 가정하고, 이 10분 안에 최대 30km 까지 움직이는 이동 객체 위치 정보를 생성한다. 방향은 16방위를 사용하고, 공간 좌표의 경계는 (X : 0 ~ 40000, Y : 0~40000)로 한정한다.

6.3 실험 및 성능 평가

이 장에서는 제안한 MPR-트리와 기존의 STR-트리, R-트리, CR-트리를 가지고, 아래 표와 같이 이동 객체 관련 질의에 대한 노드 접근 수를 비교 및 평가한다.

실험할 질의 종류와 실험 데이타는 표 6에 나타나 있다. 이와 같은 질의에 대해 10만, 25만, 50만, 75만, 100만 개의 이동 점 객체 데이타를 입력시키고 노드 접근 수를 체크하여 각 트리의 성능을 비교하고 평가한다.

가. 특정 시점 질의

특정 시점 질의는 "2001년 8월 13일 오후 4시 32분의 543번 택시의 위치를 구하시오"와 같이 한 시점에 대한 위치 질의이다. 먼저, 검색 비용이 과거의 시점 질의와 현재의 시점 질의를 처리할 때 달라지는지 알아보기 위해,

표 6 질의 데이타

질의 종류	매개 변수	실험 데이타 범위				
특정 시점 질의	질의 위치	10 %	25 %	50 %	75 %	90 %
	공간 범위	10 %	25 %	50 %	75 %	100 %
시공간 범위 질의	공간 범위	10 %	25 %	50 %	75 %	100 %
	시간 범위	1 %	5 %	10 %	15 %	20 %
	시간, 공간 범위	시간 1 % 공간 10 %	시간 5 % 공간 25 %	시간 10 % 공간 50 %	시간 15 % 공간 75 %	시간 20 % 공간 100 %
궤적 질의	질의 위치	시간 10 %			시간 20 %	

각 트리에 입력된 데이타의 전체 시간 중 10%, 25%, 50%, 75%, 90% 지점에 대해 질의한다.

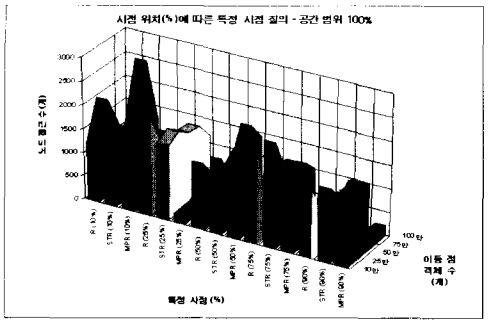


그림 9 시점에 따른 특정 시점 질의

그림 9는 각 시점에 따른 노드 접근 수를 나타낸다. STR-트리와 R-트리가 시간축의 50%, 90% 지점에 대한 검색에서, 다른 시점보다 더 적은 노드 접근 수를 나타내는 반면, MPR-트리는 이보다 적은 노드 접근 수로 어떤 시점에서도 동일한 노드 접근 수를 보였다. 시점에 따른 특정 시점 질의에 대한 노드 접근 수는 MPR < R < STR 이다.

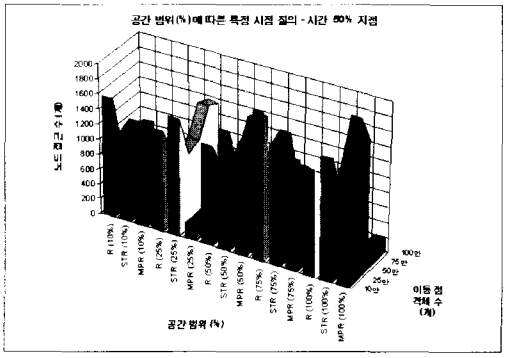


그림 10 공간 범위에 따른 특정 시점 질의

동일한 시점(시간 50% 지점)에서, 공간 범위를 다르게 했을 때의 결과는 그림 10과 같다. 현재로 갈수록, R-트리가 STR-트리보다 더 좋은 결과를 보였고, MPR-트리는 동일한 노드 접근 수를 보이면서 가장 적은 노드 접근 수를 기록했다. 공간 범위에 따른 특정 시점 질의에 대한 노드 접근 수는 MPR < R < STR 이다. 위의 실험 결과에서 MPR-트리는 시간에 대해 고르게 분할되어 있으며, t축에 대한 Projection이 유용하게 쓰임을 알 수 있다.

나. 시공간 범위 질의

시공간 범위 질의는 "2001년 9월 9일 오후 1시 40분부터 오후 5시까지 청주 지역에 있던, 버스들을 검색하시오"와 같이 시간과 공간에 대한 범위를 정하여 검색하는 것이다. 먼저, 공간 범위에 따른 노드 접근 수를 알아보기 위해, 전체 공간 범위 중 10%, 25%, 50%, 75%, 100% 범위에 대해 질의한다.

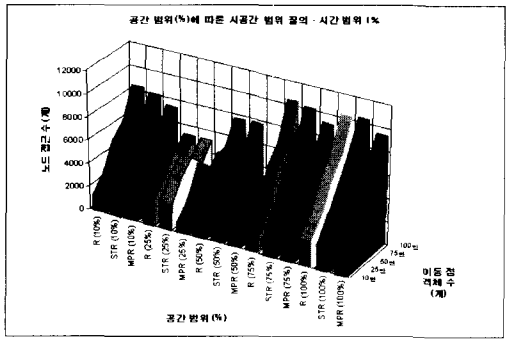


그림 11 공간 범위에 따른 시공간 범위 검색

시간범위 1%에서 공간 범위에 따른 시공간 범위 검색의 결과는 그림 11과 같다. STR-트리, R-트리는 거의 비슷한 결과를 보인다. 그리고, MPR-트리는 공간 범위가 100%가 아닐 경우에는 비교적 좋은 결과를 보인다. 공간 범위에 따른 시공간 범위 검색 질의에 대한 노드 접근 수는 MPR < R ≤ STR 이다.

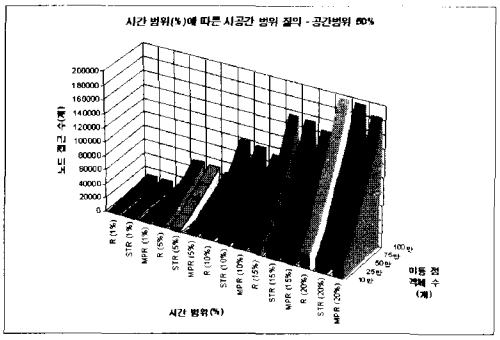


그림 12. 시간 범위에 따른 시공간 범위 검색

공간 범위 50%에서 시간 범위에 따른 시공간 범위 검색의 결과는 그림 12와 같다. 시간 범위가 1%일 때를 제외하면 트리 모두가 비슷한 성능을 보인다. 시간 범위에 따른 시공간 범위 검색 질의에 대한 노드 접근 수는 MPR < R ≒ STR 이다. R-트리와 STR-트리의 노드 접근 수의 차이 거의 없다.

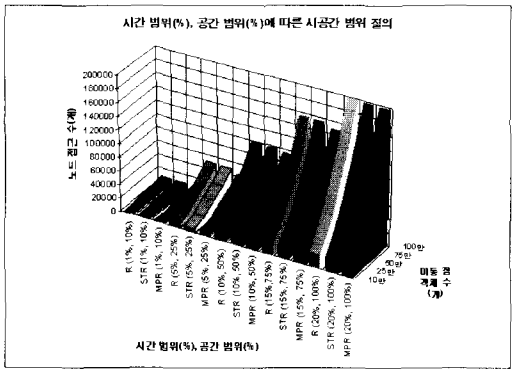


그림 13 시간과 공간 범위에 따른 시공간 범위 검색

시간과 공간 범위에 따른 시공간 범위 검색의 결과는 그림 13과 같다. STR-트리와 R-트리는 거의 비슷하고 MPR-트리는 그보다 좀 더 나은 성능을 보인다. 시간과 공간 범위에 따른 시공간 범위 검색 질의에 대한 노드 접근 수는 MPR ≤ R ≒ STR 이다.

6.3.3 궤적 질의

궤적 질의는 "2001년 9월 15일 오후 10시 35분에 청주에 있던 택시들의 궤적을 현재까지 검색하시오"와 같이 어떤 특정 시점부터 원하는 시간대까지의 궤적을 찾는 결합질의[3]를 말한다.

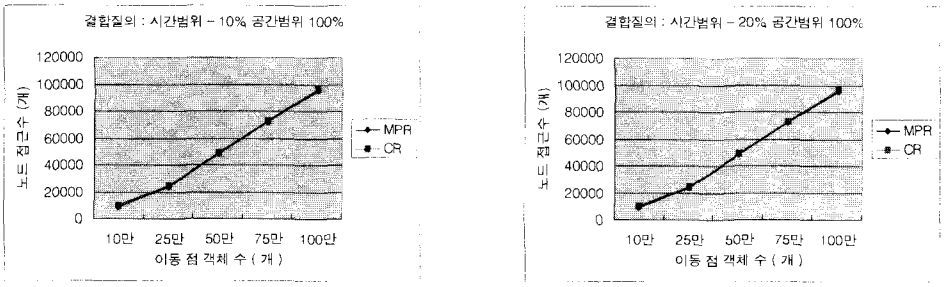
TB-트리를 제안한 Dieter Pfoser는 궤적 질의에 대해 STR-트리, R-트리, TB-트리를 비교하였고, 그 중 TB 트리가 결합질의에 대해 가장 뛰어난 성능을 보였다[3]. 이계영은 R-트리에 연결 리스트로 이동 객체 궤적을 연결하는 CR-트리를 제안했다 그리고, 결합질의에 대한 실험에서 TB-트리보다 overlap이 적은 R-트리가 시공간 범위 질의에서 더 빠른 검색 속도를 보이며 결합질의에서 좀더 좋은 성능을 보였다[8]. 이 논문에서 제안한 MPR-트리는 기존의 R-트리, STR-트리보다 특정 시점 질의 및 시공간 범위 질의에서 좀 더 효과적이고, 이로 인해 궤적에 관련된 결합질의에서도 좀 더 좋은 성능을 보인다.

시간 범위에 따른 결합질의 결과는 그림 14와 같다. 먼저 특정 시점에 대한 이동 객체를 검색하고, 검색된 이동 객체에 대해서 시간 범위 10%, 20% 에 대한 이동 객체 궤적을 검색하였다. CR-트리와 MPR 트리의 노드 검색 수는 거의 비슷하지만, MPR-트리가 좀 더 나은 성능을 보인다. 시간 범위에 따른 궤적 질의에 대한 노드 접근 수는 MPR≤CR 이다.

6.3.4 실험 결과 분석 및 고찰

앞의 실험을 정리해 보면, 이 논문에서 제안한 MPR-트리가 기존의 STR-트리, R-트리보다 특정 시점 질의, 시공간 범위 질의에서 좀더 좋은 성능을 보였고, 이로 인해 궤적에 대한 결합질의에서도 기존의 CR-트리보다 빠른 검색 속도를 보였다. 이는 결합질의가 특정 시점 질의와 시공간 범위 질의를 먼저 처리하여 원하는 객체 정보를 얻고, 해당 객체의 궤적을 찾는 질의이므로, 세 종류의 트리가 똑같이 연결 리스트를 사용했을 경우 특정 시점 질의와 시공간 범위 질의에서 속도 차이가 나기 때문이다. 특히 좋은 성능을 보인 특정 시점 질의는 시간과 공간 범위와 관계없이 동일한 검색 속도를 나타내었다. 이는 MPR-트리가 시간 축에 대해 분할이 잘 되었기 때문이라고 생각된다. 시공간 범위 질의에서도 시간 범위가 작을수록 좋은 성능을 보이지만, 시간과 공간 범위가 넓어지고, 많은 이동 객체 정보가 입력되면, 기존의 STR 트리와 R-트리와 비슷한 성능을 보였다. 그리고, 미래 위치 질의는 빠른 검색 속도보다도 교통 체증이나 사고 또는 재해에 대해 얼마나 효과적이고, 정확한지를 고려해야 하기 때문에 기존의 색인과 노드 접근 수를 비교하지 않았다.

실험에서 사용한 Projection은 t축에 대한 것뿐이므로 각 노드와 연결 리스트 그리고, T Projection만을 고려하여 계산된 MPR-트리의 저장 공간과 동일하게 궤적을 다



(a) (b)

그림 14 시간 범위에 따른 결합질의

루는 CR-트리의 저장 공간과 비교했을 때, 약 0.9배 적은 공간을 차지했다. 또, R-트리, STR-트리와 저장공간을 비교하기 위해서 연결 리스트를 제외하고 각 노드와 T-Projection만을 고려하여 MPR-트리의 저장공간을 계산하였다.

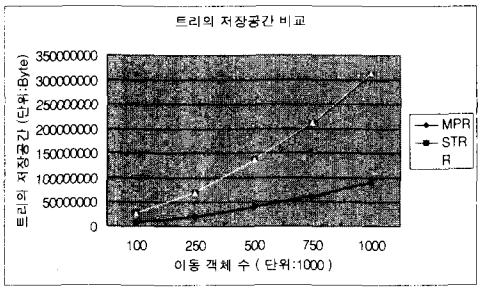


그림 15 이동 객체 수에 따른 각 트리의 저장 공간 비교

그림 15는 MPR-트리, R-트리, STR-트리의 저장공간을 나타낸다. MPR-트리의 저장공간이 상대적으로 적은 것을 알 수 있는데, 이는 각 축마다 정렬된 하위 노드들의 정보를 가진 MPR-트리의 Projection이 분할 알고리즘에서 효과적으로 사용되면서 노드의 공간분할에 도움을 준 것으로 생각된다. 각 Projection은 이미 정렬된 정보를 갖고 있기 때문에, 분할할 때 따로 비용을 들일 필요가 없고 응용 분야에 따라 그 정렬된 정보를 그대로 이용하면 된다.

제 6.3절의 실험결과에서 보였듯이 제안된 MPR-트리는 추출 연산을 사용하여 노드의 하위 노드를 각 축에 대해서 순서대로 정렬하여 저장함으로써, dead space, overlap으로 인한 불필요한 검색을 감소시켰다. 따라서, 2차원 평면에 시간차원을 더한 3차원 공간에서의 이동 점 객체에 대한 특정 시점 질의 및 시공간 범위 질의를 효과적으로 처리함을 알았다.

7. 결 론

기존의 이동 객체 정보를 다루는 색인들은 대부분 R-트리를 응용한 구조를 가지기 때문에, dead space, overlap 등 R-트리의 문제를 그대로 가지고 있다. 2차원 공간 객체를 다루는 R-트리의 MBB에 시간 차원을 더한 3차원 MBB를 사용할 경우와 단지 궤적만을 고려하여 노드를 구성할 경우에는 오히려 overlap, dead space 등의 단점이 더 커지고 효과적인 검색을 방해한다.

이와 같은 문제를 해결하기 위해서 이 논문에서는 MPR-트리를 제안하였다. 제안된 MPR-트리는 2차원 평면에 시간 차원을 추가한 3차원 공간에서의 이동 점 객체를 다루었다. 그리고, 3차원 MBB의 각 축에 대한 추출 연산을 수행하여 각 MBB의 하위 노드들을 각 축에 대해 순서대로 정렬하여 저장함으로써, 이동 점 객체의 과거와 현재에 대해서 특정 시점에 대한 질의 및 시공간 범위 질의를 효과적으로 처리하였다. 각각의 축에 대한 Projection 노드는 해당 범위의 하위 노드를 가리키고 있기 때문에, 검색할 때 일일이 하위 노드를 검색할 필요가 없다. 그리고, 동일한 이동 객체의 위치를 시간에 따라 연결 리스트로 연결하여 결합질의 등 궤적에 대한 질의 처리를 용이하게 할 수 있도록 설계하였다. 이를 통해 dead space를 줄이고, 검색 효율을 높일 수 있음을 확인하였다. 기존의 STR-트리와 R-트리와 비교했을 때, 특히 특정 시점 질의에서 뛰어난 성능을 보였고, 전체적으로 고른 노드 접근 수를 보였다. 결합질의에 대한 실험에서는 MPR-트리가 TB-트리보다 좋은 성능을 보인 CR-트리보다 조금 더 나은 성능을 보였다. 이는 특정 시점 질의와 시공간 범위 질의에서 기존의 이동 객체 색인 보다 나은 성능을 보였기 때문이다.

기존의 이동 객체 색인과 저장 공간을 비교했을 때, 비교적 적은 공간을 차지함을 알 수 있는데, 이는 각 축에 대한 Projection이 분할 알고리즘에서의 비용을 줄일 뿐만 아니라 노드들의 overlap을 효과적으로 줄여주기 때문이다. 실험 결과를 종합해 봤을 때, 추출연산은 모든 축에 적용되는 것보다, 각 응용마다 중요한 한 개의 축을 선택하여 실행하는 것이 가장 좋은 형태로 생각된다. 이 형태에서는 입력 및 갱신할 때, R-트리보다 한 축의 Projection을 체크하는 시간이 더 필요한 단점이 있지만, 분할 및 질의 처리 그리고, 공간 활용 측면에서 장점을 보인다.

이 연구에서는 모의 데이타를 가지고 실험하였다. 추가적으로 우리는 실제 데이타를 가지고 실험하여 이동 객체 정보의 불확실성 처리와 궤적 질의 및 범위 질의 등, 좀 더 실생활과 가까운 질의 처리를 위한 실험을 수행할 예정이다.

참 고 문 헌

[1] Martin Erwig, Ralf Hartmut Guting, Markus Schneider and Michalis Vazirgiannis, "Spatio - Temporal Data Types : An Approach to Modeling and Querying Moving Object in Databases," CHOROCHRONOS Technical Report CH-97-08, December, 1997.

[2] Ralf Hartmut Guting, Michael H. Bohlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis, "A

Foundation for Representing and Querying Moving Objects," ACM Transactions on Database Systems, Vol. 25, No.1, pp. 1-42, March, 2000.
[3] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis, "Novel Approaches in Query Processing for Moving Objects," CHOROCHRON OS Technical Report CH-00-3, February, 2000.
[4] Guttman, "A.:R-trees: a Dynamic Index Structure for Spatial Searching," In Preceedings of the ACM-SIGMOD Conference on the Management of Data, pp. 47-57, 1984.
[5] Luca Forlizzi, Ralf Hartmut Guting, Enrico Nardelli, and Markus Schneider, "A Data Model and Data Structures for Moving Objects Databases," Proc. ACM SIGMOD Conf., Dallas, Texas, pp. 319-330, 2000.
[6] Martin Erwig, and M. Schneider, "Developents in Spatio-Temporal Query Languages," In Procee dings of DEXA Workshop on Spatio-Temporal Data Models and Languages, 1999.
[7] Dieter Pfoser, Yannis Theodoridis, and Christian S. Jensen, "Indexing Trajectories of Moving Point Objects," CHOROCHRONOS Technical Report CH-99-03, October, 1999.
[8] 이계영, 권동섭, 송병호, 이석호, "이동 객체의 궤적 검색을 위한 인덱스 구조", 제28회 한국정보과학회 추계 학술발표회, pp. 217-219, 2001년 10월.
[9] S. Saltenis, C. Jensen, S. Leutenegger and M. Lopez. "Indexing the Positions of Continuously Moving Objects," In Proc. of the 19th ACM-SIGMOD Int. Conf. on Management of Data, Dallas, Texas, 2000.
[10] K. H. Ryu and Y. A. Ahn, "Application of Moving Objects and Spatiotemporal Reasoning", TimeCenter TR-58, 2001.
[11] Seong Seung Park, Yun Ae Ahn, Keun Ho Ryu, "Moving Objects Spatiotemporal Reasoning Model for Battlefield Analysis" Proceeding on MGA2001, Vol. 33, No. 4, pp. 108~113, April 1, 2001.
[12] Dong Ho Kim, Keun Ho Ryu, Hong Soo Kim, "A spatiotemporal database model and query language", The Journal of Systems and Sortware 55, pp. 129-149, 2000
[13] Dong Ho Kim, Keun Ho Ryu, Chee Hang Park "Design and implementation of spatiotemporal database query processing system", The Journal of Systems and Sortware 60, pp. 37-49, 2002
[14] Rasa Bliujute, Christian S. Jensen, Simonas Saltenis, Giedrius Slivinskas, "R-Tree Based Indexing of Now-Relative Bitemporal Data," VLDB 1998: pp. 345-356, 1998.
[15] Yannis Theodoridis, Mario A. Nascimento. "Generating Spatiotemporal Datasets," on the WWW. SIGMOD Record, 29(3): pp. 39-43, September 2000.
[16] Dieter Pfoser and Christian S. Jensen, "Querying the Trajectories of On-Line Mobile Objects," CHOROCHRONOS TECHNICAL REPORT 57, June 6, 2000.

정 영 진
2000년 충북대학교 전자계산학과(이학사). 2002년 충북대학교 대학원 전자계산학과(이학석사). 2002년 ~ 현재 충북대학교 대학원 전자계산학과 박사과정. 관심분야는 이동 객체 데이타베이스, 이동 객체 색인, 지리정보 시스템

장 승 연
2001년 충북대학교 경영정보학과, 멀티미디어공학(경영학사, 공학사). 2003년 충북대학교 대학원 전자계산학과(이학석사). 관심분야는 시공간 데이타베이스, 이동 객체 데이타 베이스, 지리정보 시스템

안 윤 애
1993년 한남대학교 전자계산공학과(공학사) 1996년 충북대학교 대학원 전자계산학과(이학석사) 2003년 충북대학교 대학원 전자계산학과(이학박사). 관심분야는: 시공간 데이타베이스, 모바일 데이타베이스, 지리정보 시스템, 지식기반 시스템

류 근 호
1976년 숭실대학교 전산학과 졸업. 1980년 연세대학교 공학대학원 전산전공(공학석사) 1988년 연세대학교 대학원 전산전공(공학박사) 1976년 ~ 1986년 육군군수 지원사 전산실(ROTC 장교), 한국전자통신 연구원(연구원), 한국방송통신대전산학과(조교수) 근무. 1989년~1991년 Univ. of Arizona Research Staff. (TempIS 연구원, Temporal DB) 1986년 ~ 현재 충북대학교 전기전자 및 컴퓨터공학부 교수. 관심분야는 시간 데이타베이스, 시공간 데이타베이스, Temporal GIS, 객체 및 지식기반 시스템, 지식기반 정보검색 시스템, 데이타마이닝, 데이타베이스 보안 및 Bio-Informatics