UCN-트리: 제한된 망 구조 내의 이동체를 위한 통합 색인

UCN-Tree: A Unified Index for Moving Objects in Constrained Networks

천종현* / Jong-Hyeon Cheon
정명호** / Myeong-Ho Jeong
장용일*** / Yong-Il Jang
오영환**** / Young-Hwan Oh
배해영***** / Hae-Young Bae

요 약

위치 기반 서비스(Location Based Services) 지원을 위해 이동체(Moving Objects)의 위치 정보를 효과적으로 저장 및 검색하기 위한 기술이 요구되었으며, 이러한 이동체를 효율적으로 관리하기 위한 이동체 색인에 대한 연구가 진행되어 왔다. 이러한 이동체 색인들은 실 세계에 응용이 많이 되는 제한된 망 구조(도로, 철도 선로 등)를 따라 움직이는 이동체에 대해 고려가 되어 있지 않기 때문에, 제한된 망 구조 기반 이동체 색인들이 제안되었다.

그러나 이 색인들은 다음과 같은 두 가지 문제점을 가지고 있다. 첫 번째로, 제한된 망 구조에 기반한 이동체 색인은 시간 도메인 별로 나뉘어져 있기 때문에 이동체의 현재부터 과거까지의 위치를 모두 필요로 하는 경우 현재 위치 색인과 과거 위치 색인에서 중복된 공간 탐색을 해야 하는 문제점이 있다. 두 번째로, 이러한 경우 현재 위치 색인과 과거 위치 색인을 모두 구축해 놓아야 하므로 색인의 공간적인 비용 및 갱신 비용에 대한 부담이 따른다.

본 논문에서는 이러한 문제점들을 해결하기 위해 제한된 망 구조 내의 이동체를 위한 통합 색인을 제안한다. 제안 색인은 이동체의 현재 위치뿐만 아니라 과거 위치까지 함께 지원하기 때문에 기존 이동체 색인의 문제점인 이동체의 현재 및 과거 위치 검색 시 별도의 계산 과정이 필요했던 점을 해결한다. 또한 현재 위치 색인 및 과거 위치 색인의 공통된 부분을 통합하였기 때문에 색인을 별도로 유지하는 것 보다 색인의 공간적인 비용 및 갱신 비용을 감소시킨다.

Abstract

To support Location Based Services, the technology to store and search locations information of moving objects effectively was needed. And the study about indexes to manage these moving objects effectively has been done. As these indexes for moving objects was not considered for the objects which are moving along constrained networks such as road and railroad, indexes for the moving objects based on constrained networks was proposed.

■ 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성.지원 사업의 연구 결과로 수행되었음.
■ 논문접수 : 2006. 2. 24 ■ 심사완료 : 2006. 4. 12
* 인하대학교 컴퓨터 · 정보공학과 석사과정(jhcheon@dblab.inha.ac.kr)
** 알티베이스 연구원(mhjeong@altibase.com)
*** 인하대학교 컴퓨터공학부 박사과정(yijang@dblab.inha.ac.kr)
**** 나사렛 대학교 컴퓨터공학부 조교수(yhoh@kornu.ac.kr)
***** 교신저자 인하대학교 컴퓨터공학부 교수(hybae@inha.ac.kr)

But these kinds of indexes have two problems as following. First, as the indexes for the moving objects based on constrained networks is divided according to time domain, when the places of moving objects from the present to the past are needed, the problem to search past indexes as well as present indexes occurs. Second, in this case, we should construct both present indexes and past indexes, so we have no other choice but to spend space cost and reconstruction cost additionally.

This paper proposes A Unified Index for Moving Objects in Constrained Networks to solve these kinds of problems. As this proposed indexes support both present location and past location of moving objects, it can solve the current problems such as when we search present and past location of moving objects, we need a separate processing procedure. And as it consolidated the common parts of current location indexes and past location indexes, we can use less space cost and reconstruction cost than when we maintain indexes separately.

주요어: 이동객체, 위치기반서비스. 통합 색인
Keyword: Moving Object, LBS, Unified Index

1. 서 론

최근 무선 통신과 측위 기술의 발달로 인해 위치 기반 서비스(LBS: Location Based Services)에 대한 요구가 증가되고 있다. 위치 기반 서비스에서는 시간이 흐름에 따라 연속적으로 위치가 변경되는 이동체(Moving Objects)의 위치 정보를 효과적으로 저장 및 검색하기 위한 기술이 요구되었으며, 이러한 이동체를 효율적으로 관리하기 위한 이동체 색인에 대한 연구가 활발히 진행되어 왔다[4, 5, 9, 12, 13, 14, 16, 17, 18]. 그러나 이러한 이동체 색인들은 물류 및 차량 관리, 응급 서비스 등 실 세계에 응용이 많이 되는 제한된 망 구조(도로, 철도 선로 등)를 따라 움직이는 이동체에 관한 고려가 없기 때문에 제한된 망 구조 기반 이동체 색인들이 제안되었다[1, 6, 8, 10, 11].

제한된 망 구조에 기반한 이동체는 이동 영역이 망 구조에 한정되어 있기 때문에 이러한 특징을 반영하여 기존의 색인을 이용하는 것보다 공간 활용도 및 성능을 향상시킨 색인들이 제안되었으며, 크게 현재 위치 색인과 과거 위치 색인으로 분류된다. 현재 위치 색인은 망 구조를 색인으로 미리 구축해 놓고 이 색인의 단말 노드에 대응되는 블록에 이동체를 저장하여 이동체 갱신 시 색인에 갱신되는 부분을 줄여 전체 갱신 비용을 줄인 IMORS(Indexing Moving Objects on Road Sectors)가 있으며, 과거 위치 색인은 제한된 망 구조에 기반한 이동체를 이용하여 기존의 이동체 색인에 비해 공간 활용도 및 검색 성능의 향상을 이룬 FNR-트리(Fixed Network R-Tree) 및 MON-트리(Moving Objects in Networks Tree)가 있다[1, 6, 11].

그러나 이 색인들은 다음과 같은 두 가지 문제점을 가지고 있다. 첫 번째로, 제한된 망 구조에 기반한 이동체 색인은 시간 도메인 별로 나뉘어져 이동체의 현재 위치 또는 과거 위치만을 지원하기 때문에 이동체의 현재부터 과거까지의 위치를 모두 필요로 하는 경우 현재 위치 색인에서 가져온 결과와 과거 위치 색인에서 가져온 결과를 가지고 합병 연산을 해야 한다. 하지만 색인이 별도로 구성되어 있으므로 중복된 공간 탐색을 해야 하는 문제점이 있다. 두 번째로, 이러한 경우 현재 위치 색인과 과거 위치 색인을 모두 구축해 놓아야 하므로

색인의 공간적인 비용 및 갱신 비용에 대한 부담이 따른다. 기존에 시간에 따른 이동체 색인의 통합은 연구 되었으나, 제한된 망 구조 기반의 통합된 이동체 색인에 관한 연구는 없었다. 본 논문에서는 이러한 문제점들을 해결하기 위해 제한된 망 구조 내의 이동체를 위한 통합 색인을 제안한다. 제안 색인의 구조는 다음과 같다. 먼저, 현재 위치 색인 및 과거 위치 색인의 공통된 부분인 제한된 망 구조를 저장·관리하는 부분을 R*-트리로 구축하여 통합하였다[2]. 현재 위치는 이 색인의 단말 노드에 대응되는 블록에 저장하여 이동체의 현재 위치에 대한 지원을 가능하게 한다. 이동체의 현재부터 과거까지의 위치에 대한 지원을 하기 위해서는 가까운 과거 위치를 저장해야 하는데, 이는 위상 정보를 이용한 보간법을 이용하여 폴리라인 단위로 구축하고 그에 대응되는 블록에 저장한다[20]. 마지막으로, 이동체의 과거 위치에 대한 지원을 하기 위해 이동체의 과거 위치를 각각의 가까운 과거 위치 저장 블록에 대응되는 R-트리 색인에 저장한다[7].

제안 색인은 이동체의 현재 위치 뿐만 아니라 과거 위치까지 함께 지원하기 때문에 기존 이동체 색인의 문제점인 이동체의 현재 및 과거 위치 검색 시 별도의 계산 과정이 필요했던 점을 해결한다. 또한 현재 위치 색인 및 과거 위치 색인의 공통된 부분인 제한된 망 구조의 저장·관리 부분을 통합하였기 때문에 색인을 별도로 유지하는 것 보다 색인의 공간적인 비용 및 유지 비용을 감소시킨다.

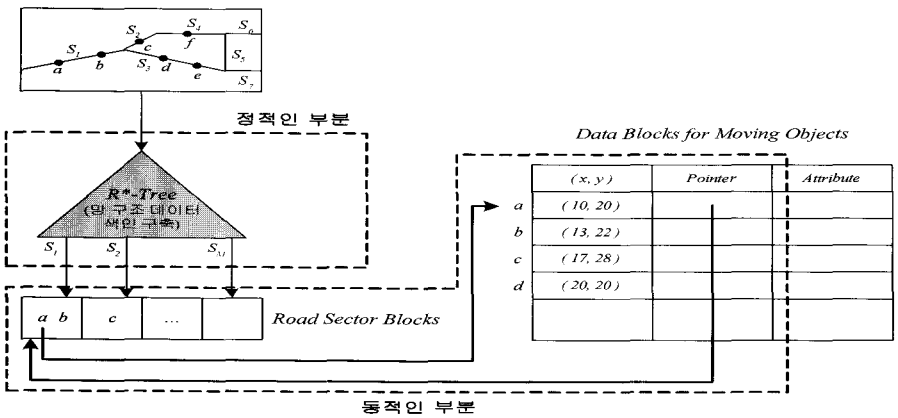
본 논문의 구성은 다음과 같다. 2장은 관련 연구로, 제한된 망 구조 기반 이동체를 위한 현재 위치 색인 및 과거 위치 색인에 대해 설명한다. 3장에서는 본 논문에서 제안하는 제한된 망 구조에 기반한 이동체를 위한 통합 색인에 대한 구조와 이동체 갱신 및 검색에 대한 내용을 다룬다. 4장은 제안 색인의 성능 평가이다. 끝으로 5장에서 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

제한된 망 구조 기반 이동체를 위한 색인은 시간 도메인상 현재 위치 색인 및 과거 위치 색인으로 분류되어 있다. 이 장에서는 본 연구의 기반이 되는 제한된 망 구조 기반 이동체의 현재 위치 색인 및 과거 위치 색인에 대해 기술한다.

2.1 제한된 망 구조 기반 이동체의 현재 위치 색인

제한된 망 구조 기반 이동체의 현재 위치 색인으로 IMORS가 있다.



<그림 1> IMORS의 구조

이는 이동체의 현재 위치 처리에 대한 성능 향상을 위해 동적인 부분(이동체 갱신 시 바뀌는 부분)을 줄이는 목적으로 제한된 망 구조를 이용하였다[11].

색인의 구조는 <그림 1>과 같이 크게 정적인 부분과 동적인 부분으로 나뉜다. 정적인 부분은 망 구조 데이터(<그림 1> S_1 ~ S_M)를 라인 세그먼트로 쪼개어 R*-트리에 저장한 부분이고, 동적인 부분은 R*-트리의 단말 노드 엔트리에 대응되는 로드 섹터 블록(Road Sector Blocks)으로, 이 블록에 이동체 데이터 블록(Data Blocks for Moving Objects)에 해당하는 이동체를 가리키는 포인터(<그림 1> a ~ d)를 저장한다.

갱신을 수행할 때 새로운 로드 섹터 블록의 이동체 위치에 대한 포인터를 이동체 데이터 블록 안에 저장되어 있는 이동체에 반영하고, 새로운 로드 섹터 블록의 이동체 위치에 대한 포인터를 이동체 데이터 블록 안에 저장되어 있는 이동체에 반영한다.

이러한 이동체 갱신 정책은 망 구조가 고정되어 변하지 않는 점과 이동체가 대응되는 라인 세그먼트를 벗어날 때에만 망 구조 색인의 검색 비용이 든다는 점을 통해 이동체 갱신 시에 발생하는 I/O 부하를 감소시킨다.

2.2 제한된 망 구조 기반 이동체의 과거 위치 색인

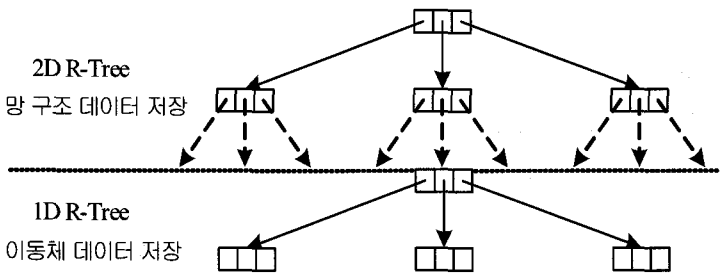
제한된 망 구조 기반 이동체의 과거 위치 색인으로 FNR-트리가 있으며, 색인의 구조는 다음과 같다[6].

<그림 2>와 같이 FNR-트리는 일반적인 R-트리를 기반으로 하고 있으며 망 구조 데이터를 색인으로 구축한 2D R-트리 하나와 이 단말노드 엔트리에 대응되는 다수의 1D R-트리로 구성된다.

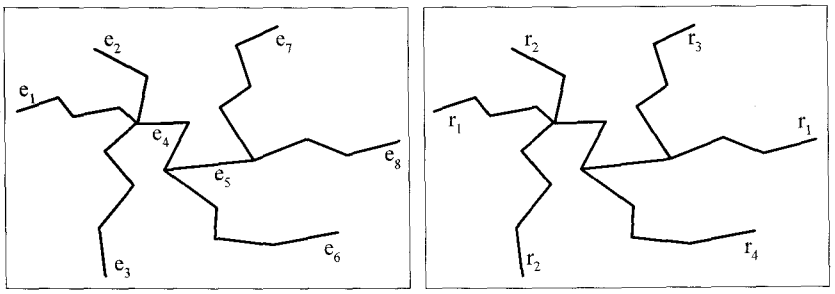
2D R-트리는 IMORS와 같이 라인 세그먼트로 되어 있는 망 구조 데이터를 사용한다. 단말 노드 엔트리는 MBR(Minimum Boundary Rectangle), orientation으로 이루어진다. orientation은 방향성을 나타내는 값으로, 이것은 0 또는 1을 가진다.

FNR-트리의 삽입은 이동체가 주어진 라인 세그먼트를 벗어날 때 실행된다. 따라서, 이동체는 대응되는 라인 세그먼트의 <id, direction, t_{start}, t_{end} >만으로 표현이 가능하다. 여기서 direction은 이는 일반적인 이동체의 과거 위치 정보가 < x_{start}, y_{start}, x_{end}, y_{end}, t_{start}, t_{end} >로 이루어진 것에 비해 이동체 데이터가 차지하는 공간의 크기가 작음을 의미한다.

또한 이동체를 실제 저장하고 있는 1D R-트리는 시간을 중심으로 분할하고, 시간은 계속해서 증가하는 값이기 때문에 이동체의 삽입은 항상 노드의 오른쪽 끝에서 이루어진다. 이러한 1D R-트리의 삽입 결과는 기존의 색인에 비해 공간 활용도를 높인다. 기존의 색인의 공간 활용도가 65%인데 반해,



<그림 2> FNR-트리의 구조



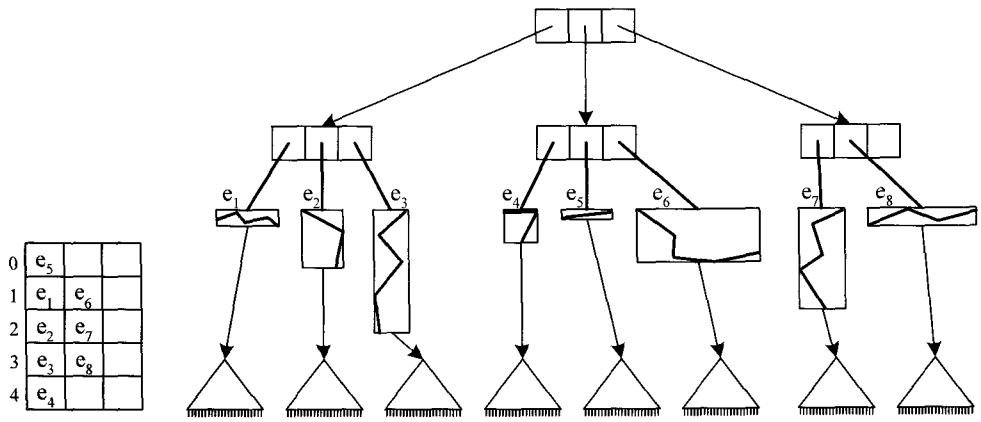
(a) 에지 기반 망 구조 (b) 라우트 기반 망 구조

<그림 3> 에지 기반 및 라우트 기반 망 구조

이 색인의 공간 활용도는 96% 정도로 매우 높다.

또 다른 제한된 망 구조 기반 이동체의 과거 위치 색인으로 MON-트리가 있다. 이 색인 또한 IMORS, FNR-트리와 마찬가지로 제한된 망 구조를 구축하여 사용하고 있지만 이와는 다르게 망 구조 데이터를 에지 기반인 <그림 3>(a) 또는 라우트 기반인 <그림 3>(b)로 저장한다[1].

색인의 구조는 <그림 4>와 같이 top R-트리 하나와 다수의 bottom R-트리로 이루어진다. top R-트리에 망 구조 데이터를 폴리라인의 MBR로 저장하고, 각각의 단말 노드 엔트리에 대응되는 bottom R-트리에 이동체의 위치정보를 저장한다. 이동체의 위치 정보는 폴리라인의 길이를 기반으로 양자화된 값인 포지션 인터벌 $\langle p_1, p_2 \rangle$ $(p_1 \sqsupset 0, p_2 \bullet 1)$ 및, 시간 범위 $\langle t_1, t_2 \rangle$가 들어간다. 또한 이 색인은 <그림 4>의 좌측에 나타나 있는 것과 같이 <polyid, bottreept>로 구성된 해시 구조를 통해 대응되는 bottom R-트리를 빠르게 찾을 수 있는 기능을 제공한다. 여기서 polyid는 폴리라인의 id이며, bottreept는 bottom R-트리를 가리키는 포인터이다. 실제 폴리라인 데이터는 별도의 자료로 저장한다.



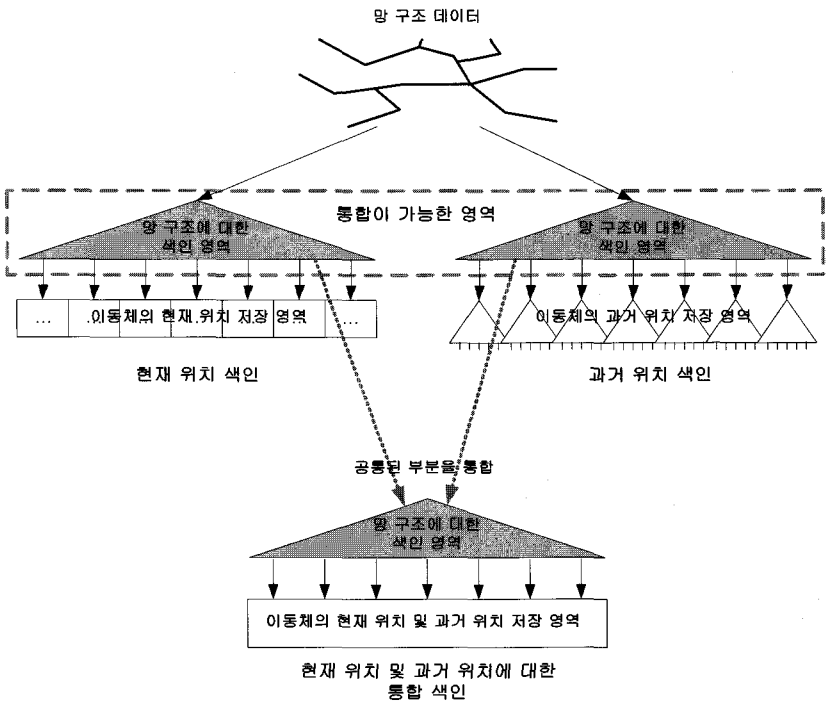
<그림 4> 에지 기반 망 구조로 구축한 MON-트리의 구조

이동체 삽입 연산은 moid(이동체의 id)와 polyid를 필요로 한다. 먼저 알고리즘은 해시 구조에서 대응되는 폴리라인을 찾는다. 폴리라인은 해당되는 polyid를 갖고 있다. 만약 폴리라인이 대응되는 bottom R-트리에 없으면 새로운 색인을 생성하고, bottom R-트리에 MBR이 들어간다. bottom R-트리의 포인터는 해시 구조에 저장된다. 이제 포지션 인터벌 $p = \langle p_1, p_2 \rangle$를 대응되는 폴리라인을 이용하여 생성한다. 그런 후에 $\langle p_1, p_2, t_1, t_2 \rangle$를 bottom R-트리에 삽입한다. 이러한 이동체의 삽입은 제한된 망 구조를 이용함으로 생길 수 있는 다수의 bottom R-트리의 생성을 막고, 양자화된 포지션 인터벌을 통해 기존의 3차원(시작 위치의 x, y 좌표, 끝 위치의 x, y 좌표, 시작 시간, 끝 시간) 이동체를 2차원(시작 위치, 끝 위치, 시작 시간, 끝 시간) 이동체로 바꾸어 저장하므로 같은 공간에 더 많은 이동체를 저장한다.

3. 제한된 망 구조 내의 이동체를 위한 통합 색인

본 장에서는 이동체의 과거 위치 및 현재 위치, 현재부터 과거까지의 위치에 대한 연산을 위한 제한된 망 구조 내의 이동체를 위한 통합 색인인 UCN-트리(Unified Constrained Network Tree)를 제안한다. 이는 현재 위치 색인 및 과거 위치 색인의 공통된 부분인 망 구조를 통합하고, 현재부터 과거까지의 연산을 위한 처리를 포함한 색인이다.

제한된 망 구조 기반 이동체 색인은 크게 과거 위치 색인과 현재 위치 색인으로 나눌 수 있으며, 이 색인들은 망 구조를 구축하여 관리하는 부분을 공통으로 가지고 있다. 따라서, 현재 위치 색인 및 과거 위치 색인의 정적인 부분인 망 구조를 구축하여 관리하는 R*-트리는 통합 색인에서 공유가 가능하다[21].



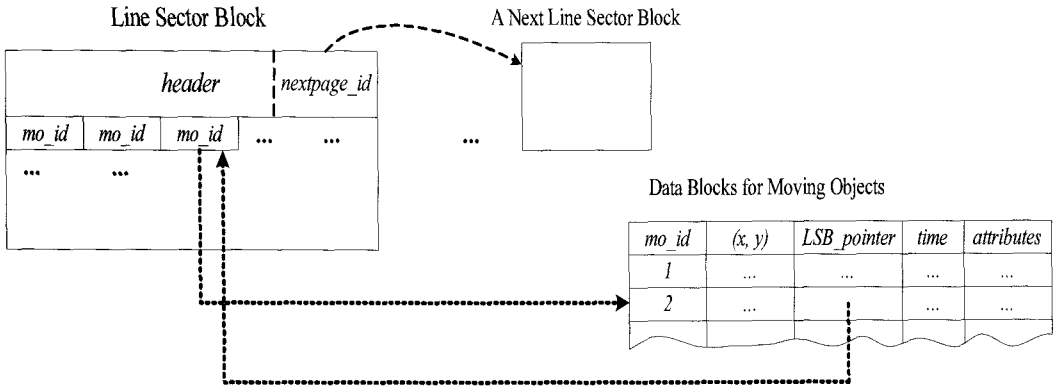
<그림 5> 망 구조 색인을 통합한 색인

header

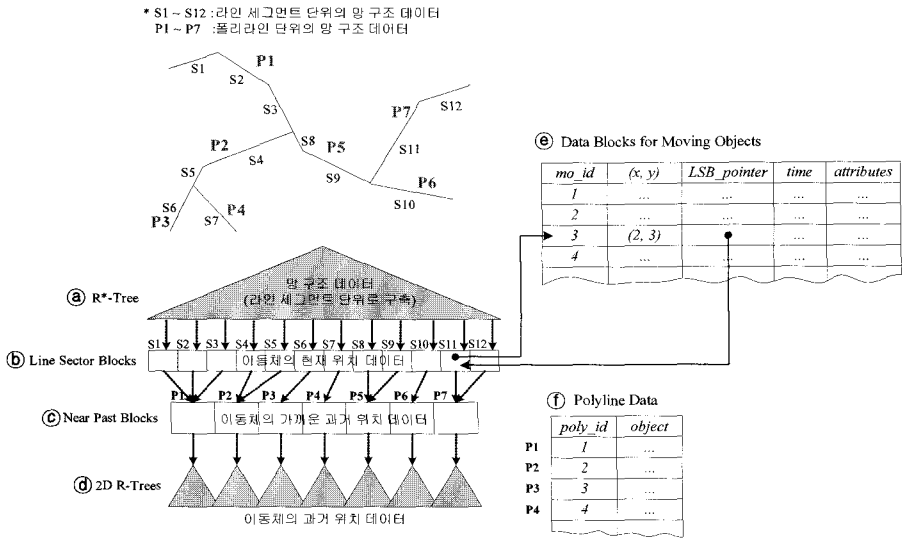
page_id	line_id	poly_id	MBR	direction	mo_count	nextpage_id
4 bytes	4 bytes	4 bytes	8 * 4 bytes(x1, y1, x2, y2)	1 bytes	4 bytes	4 bytes

data

mo_id	x	y
4 bytes	8 bytes	8 bytes

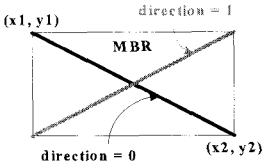


<그림 8> 이동체의 현재 위치 관리 구조



<그림 6> UCN-트리의 구조

line_id	MBR	direction
4 bytes	8 * 4 bytes(x1, y1, x2, y2)	1 bytes



<그림 7> 망 구조 색인 영역(R*-트리)의 단말 노드 엔트리 구조

<그림 5>와 같이 현재 위치 색인과 과거 위치 색인은 망 구조에 대한 색인 영역을 공통적으로 가지고 있기 때문에 이를 통합하여 사용하면 망 구조에 대한 색인 영역의 크기만큼 저장 공간의 감소를 이룬다는 이점 및 망 구조 색인 구축 비용이 감소하는 이점이 있다.

3.1 UCN-트리의 구조

UCN-트리는 기존의 현재 위치 색인과 과거 위치 색인의 공통된 부분을 하나로 통합하여 관리한다. 이는 색인 구성 및 유지 비용을 두 개의 색인을 유지할 때 보다 감소시키기 위함이다. 또한 이동체의 가까운 과거 위치 및 과거 위치에 대한 처리를 통합 관리한다. UCN-트리의 전체적인 구조는 <그림 6>과 같다.

본 절에서는 UCN-트리의 구조를 크게 망 구조 색인 영역, 이동체의 현재 위치 관리 영역, 이동체의 가까운 과거 위치 관리 영역, 이동체의 과거 위치 관리 영역 및 색인에 필요한 데이터 영역으로 나누어 설명한다.

3.1.1 망 구조 색인 영역

망 구조 색인 영역은 이동체의 위치 갱신 및 검색에 사용된다. UCN-트리는 망 구조 데이터를 R*-트리(<그림 6> ⓐ)에 구축한다. R*-트리는 공간 객체의 겹침을 최소화 하여 공간 검색 성능을 높이는 색인으로, 검색에 주로 사용되고 한 번 구축이 되면 거의 바뀌지 않는 망 구조 데이터에 적합한 색인이다[2].

이 색인의 단말 노드에 들어가는 데이터 엔트리의 구조는 <그림 7>과 같다. line_id는 라인 세그먼트의 식별자로, 이를 이용하여 대응되는 Line Sector Block을 해시 연산을 통해 찾을 수 있다[15]. MBR은 라인 세그먼트를 포함하는 최소 사각형을 이루는 두 점의 좌표로 이루어진다. 마지막으로, direction은 라인 세그먼트의 방향을 나타낸다. 라인 세그먼트는 왼쪽 위에서 오른쪽 아래 방향(direction = 0)으로, 또는 왼쪽 아래에서 오른쪽 위 방향(direction = 1)으로만 존재하게 된다

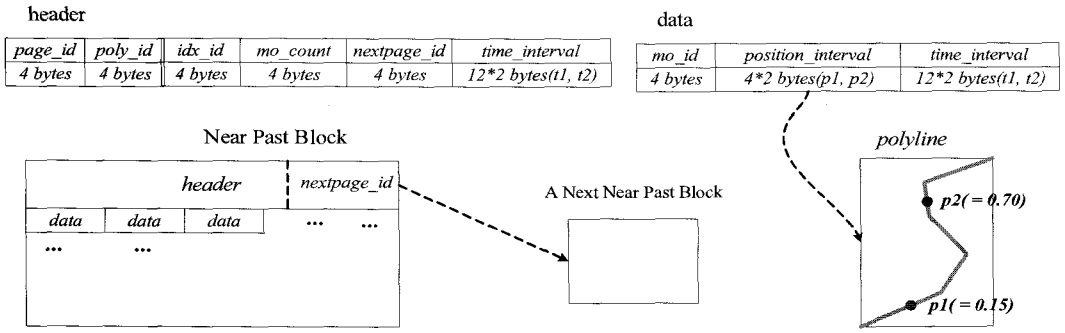
3.1.2 이동체의 현재 위치 관리 영역

이동체의 현재 위치에 대한 갱신 및 검색을 위한 구성 요소로, 여기에는 이동체의 현재 위치 정보가 저장된다.

이동체의 현재 위치는 Line Sector Block (<그림 6>ⓑ)에서 관리한다. Line Sector Block의 구조는 <그림 8>과 같다. 이는 디스크 페이지 단위의 블록으로 구성되어 있으며, 여기에 저장되는 포인터는 Data Blocks for Moving Objects에 저장되어 있는 이동체를 가리키는데, 여기서도 서로 양방향으로 참조가 가능하도록 포인터를 갖고 있다. 이는 갱신 부하를 줄이기 위한 방법으로, 이동체의 갱신을 다루는 3.2절에서 자세히 설명한다. 하나의 Line Sector Block은 page_id(페이지 번호), line_id(대응되는 라인 세그먼트의 id), poly_id(대응되는 폴리라인의 id), MBR, direction, mo_count(저장된 이동체의 개수), nextpage_id(다음 페이지 번호)로 구성된 헤더와 다수의 이동체 위치 데이터(mo_id, x, y : mo_id는 이동체의 식별자)로 구성되어 있다.

3.1.3 이동체의 가까운 과거 위치 관리 영역

이 구성요소는 이동체의 현재부터 과거까지의 갱신 및 검색 시에 사용되며, 과거 위치 색인 영역에 저장하기 위한 버퍼 영역으로 활용된다.

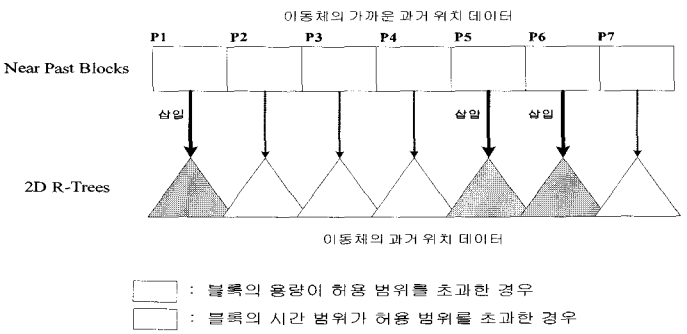


<그림 9> 이동체의 가까운 과거 위치 관리 구조

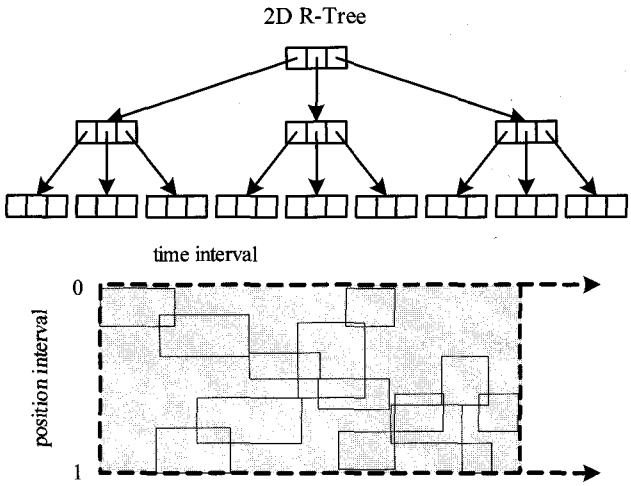
이동체의 가까운 과거 위치는 Near Past Block(<그림 6> ©)에서 관리한다. Near Past Block의 구조는 <그림 9>와 같다. Near Past Block은 현재부터 가까운 과거까지의 이동체 위치정보를 Line Sector Block으로부터 얻어와 mo_id, position_interval, time_interval로 구성하여 저장하는데, 여기서 position interval은 대응되는 폴리라인의 길이를 기반으로 0부터 1까지의 실수형 데이터로 변경한 값이다. 헤더 정보에는 page_id(페이지 번호) 및 poly_id(대응되는 폴리라인의 id), idx_id (대응되는 2D R-트리의 id)와 mo_count(저장된 이동체의 개수), nextpage_id(다음 페이지를 가리키는 링크), time_interval(이 블록에 저장된 전체 시간 범위)를 가지고 있다.

이 구성 요소의 특징은 이동체의 현재 위치 정보를 이용하여 현재부터 가까운 과거 위치까지의 정보를 생성하고, 이동체 갱신의 부하를 줄이는 데 있다. 변환 과정에서 이동체의 위치 정보는 폴리라인 단위의 위치 정보로 변환하는데, 이는 라인 세그먼트 단위로 분리되어 있는 이동체 위치 정보를 더 큰 범위의 데이터인 폴리라인 단위의 데이터로 변환하여 대응되는 2D R-트리를 줄이기 위함이다. 생성 방법은 위상 정보를 이용한 보간법을 이용한다[20].

이동체의 과거 위치 정보는 계속해서 생성되는 대용량 데이터이며, 삭제가 없이 계속 축적되는 데이터이다. 따라서 과거 위치 정보를 효율적으로 저장하는 기술이 필요한데, Near Past Block는 이동체의 현재부터 가까운 과거 위치까지의 정보를 마치 버퍼 영역처럼 저장하고 있다가 블록의 용량 또는 시간 범위에 따라 주기적으로 삽입하는 방식으로 갱신의 부하를 줄인다.



<그림 10> Near Past Block에서 2D R-트리로 과거 위치 정보 삽입



<그림 11> 2D R-트리

<그림 10>과 같이 P1, P6 블록의 용량이 설정된 허용 범위를 초과하거나 P5 블록의 시간 범위가 허용 범위를 초과한 경우 이 블록에 저장되어 있는 이동체의 과거 위치 데이터들을 대응되는 과거 위치 색인에 삽입하고 블록의 내용은 삭제한다.

3.1.4 이동체의 과거 위치 관리 영역

UCN-트리의 과거 위치 관리는 2D R-트리(<그림 6>ⓓ)에서 관리한다[7]. FNR-트리의 2D R-트리는 MON-트리보다 성능이 다소 떨어지지만[1], 본 논문에서 제안하는 통합 색인의 구축에 용이하다. 이 색인 구조는 이동체의 과거 위치를 위한 색인으로 기존의 공간 데이터를 저장하는 색인인 R-트리구조와 동일하며, 색인 자신과 대응되는 폴리라인의 id를 가지고 있다. 이는 폴리라인의 개수만큼 생성되며 Near Past Block의 과거 위치 데이터를 주기적으로 저장한다. 따라서, 단말 노드 데이터의 엔트리 구조는 Near Past Block의 데이터 구조와 동일한 mo_id, position_interval, time_interval로 구성된다.

이 색인에 저장되는 데이터는 <그림 11>과 같은 형태이다. 이는 position interval 값이 0부터 1까지로 제한되어 있고 시간은 계속해서 증가하기 때문이다.

3.1.5 색인에 필요한 데이터 영역

UCN-트리는 실제 이동체 데이터를 저장하는 테이블과 서로 연결되어 있다. 이 구성요소는 일반적인 데이터베이스의 테이블과 같으며, 실제 이동체 데이터들을 저장한다(<그림 6>ⓔ).

mo_id	*(x, y)*	*LSB_pointer*	*time*	*attributes*
4 bytes	*8 * 2 bytes*	*4* 2 bytes*	*12 bytes*	-

<그림 12> 이동체 데이터의 구조

하나의 이동체 데이터가 갖는 속성은 <그림 12>와 같다. mo_id는 이동체 데이터의 식별자이며 (x, y)는 이동체 데이터의 현재 위치 좌표 값이다. LSB_pointer는 Line Sector Block 내의 이동체 위치 데이터를 가리키는 포인터로, 페이지 번호와 페이지 내의 위치 값으로 이루어진다. time은 시간 값이다. 마지막으로, attributes는 기타 속성 필드이다.

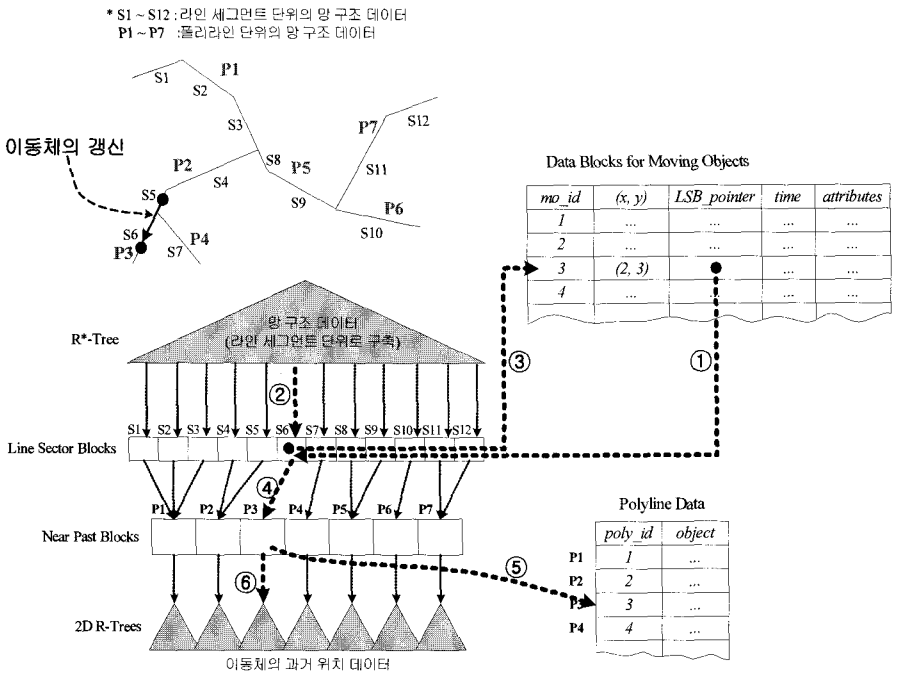
UCN-트리는 망 구조 데이터를 실제 저장하고 있는 테이블을 이용한다(<그림 6>ⓕ). 이는 라인 세그먼트 단위로 나뉘어져 구축되어 있는 R*-트리, Line Sector Block에서, 폴리라인 단위로 구축되어 있는 Near Past Blocks 또는 2D R-트리s를 참조하기 위해 사용되며, 공간 좌표를 갖고 있는 검색 질의를 폴리라인 단위의 검색 질의로 변환하는 데에 사용된다.

3.2 이동체의 갱신 연산

본 절에서는 UCN-트리의 이동체 갱신 연산에 대한 내용을 다룬다. UCN-트리에서 이동체의 갱신은 크게 이동체의 현재 위치 갱신, 이동체의 현재 위치 갱신을 가까운 과거 위치로 생성, 가까운 과거 위치를 과거 위치 색인 영역에 저장하는 과정으로 진행된다.

먼저, UCN-트리는 <mo_id, x, y, t>을 입력값으로 받아서 갱신한다. 전체적인 갱신 연산 과정은 <그림 13>의 번호 순서대로 다음과 같이 진행된다.

① Data Blocks for Moving Objects에서 이동체의 위치 및 시간을 갱신하고 LSB_pointer를 통해 대응되는 Line Sector Block을 찾은 다음 만약 갱신될 이동체가 갱신되기 이전에 있던 라인 세그먼트를 벗어나면 ②, ③ 과정 수행



<그림 13> UCN-트리의 갱신 연산 과정

② R*-트리에서 새로운 Line Sector Block을 찾아서 이동체를 삽입하고 이전의 Line Sector Block에 있던 이동체를 가까운 과거 위치 정보로 생성을 위해 임시 저장하고 Line Sector Block에서는 제거

③ 이동체가 삽입된 새로운 Line Sector Block의 포인터를 갱신

④ 갱신되기 이전의 이동체와 새롭게 갱신된 이동체를 대응되는 Polyline Block에 삽입

⑤ Near Past Block은 Polyline Data를 참조하여 두 개의 이동체를 이용하여 가까운 과거 위치 정보를 생성하여 삽입

⑥ Near Past Block은 과거 위치 정보를 관리하는 2D R-트리에 이동체의 가까운 과거 위치 정보를 주기적으로 삽입(3.1절의 <그림 10> 참조)

먼저, 이동체의 현재 위치에 대한 갱신과정을 나타내는 <그림 13>의 ①, ②, ③ 과정에 대해 살펴본다. 이는 (알고리즘 1)과 같은 과정으로 수행된다.

이동체의 현재 위치 갱신은 먼저 Data Blocks for Moving Objects(이하 DBMO)에서 갱신하는 것으로 시작한다. 이동체의 현재 위치 갱신은 이전에 저장되어 있던 이동체의 위치를 삭제하고 새로운 값으로 바꾸기 때문에 이를 과거 위치 정보로 활용하기 위해서는 삭제하기 전에 과거 위치를 관리하는 부분으로 저장해야 한다(01번째 줄).

그 다음 이동체의 식별자를 통해 대응되는 Line Sector Block(이하 LSB)을 가져온다(02번째 줄). 과거 위치 정보로 활용하기 위해 대응되는 라인 세그먼트의 id를 저장한다음(03번째 줄), 이 LSB에 대응되는 라인 세그먼트에 이동체의 위치가 속해 있는지 여부를 평가한다(04번째 줄).

```
Input
m:        새로운 위치를 가진 이동체
Output
om:               갱신되기 이전의 이동체
om_line_id:       갱신되기 이전의 이동체에 대응되는 라인 세그먼트의 id
m_line_id:        새로운 위치를 가진 이동체에 대응되는 라인 세그먼트의 id
Begin
01 :   om ←update moving object data using m
02 :   LSB←get line sector block using m's identifier
03 :   om_line_id ← line-id of LSB;
04 :   if m is not on LSB line segment
05 :      remove m from LSB
06 :      LSB ← find LSB that m is to be inserted from R*-tree
07 :      LSB_pointer ← insert m into LSB
08 :      update old-LSB_pointer as LSB_pointer
09 :   else
10 :      LSB_pointer ← get LSB_pointer from m
11 :      update current position of m using LSB_pointer
12 :   end if
13 :   m_line_id←Line-id of LSB
End
```

(알고리즘 1) Update current Position

만약 그렇지 않다면, 이는 새로운 라인 세그먼트 영역 안에 저장되어야 하므로 LSB안에 있는 이동체 위치 데이터를 지운 후(05번째 줄) 다시 R*-트리에서 새롭게 들어갈 LSB를 검색한 다음(06번째 줄), 해당하는 LSB에 이동체의 위치를 삽입한다(07번째 줄). 이 때 DBMO에 저장되어 있는 이동체의 LSB_pointer도 LSB에 새롭게 삽입된 이동체 데이터의 위치로 바꾸어 준다(08번째 줄).

만약 이동체가 라인 세그먼트에 그대로 속해 있다면 LSB안에 저장되어 있는 이동체의 위치 값 만을 바꾸어 준다(10~11번째 줄).

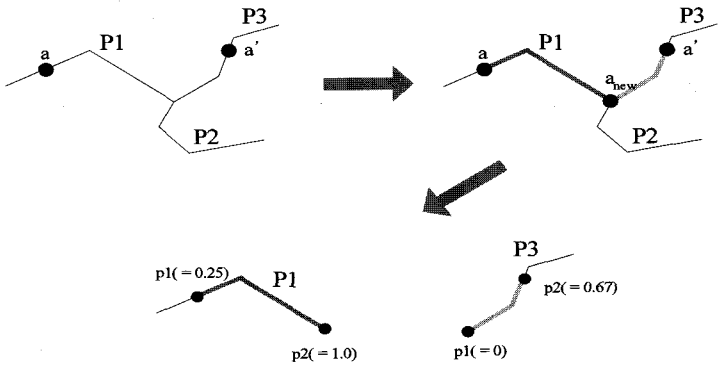
마지막으로, 새롭게 삽입된 이동체에 대응되는 라인 세그먼트를 저장한다(13번째 줄). 다음으로 이동체의 가까운 과거 위치에 대한 갱신(<그림 13>의 ④, ⑤ 과정)에 대해 살펴본다. 이는 이동체의 현재 위치 갱신 이후에 바로 실행되며 (알고리즘 2)와 같은 과정으로 수행된다.

이동체의 가까운 과거 위치는 새롭게 갱신된 이동체와 갱신 이전의 이동체를 이용하여 생성한다. 두 이동체는 <그림 14>와 같이 서로 다른 폴리라인 상에 존재할 수 있으므로, 다수의 새로운 과거 위치 정보가 생성될 수 있다. 이는 위상정보를 이용한 이동체의 보간법 알고리즘을 이용하여 생성한다[20].(01번째 줄) 그 다음 생성된 이동체의 개수만큼 루프를 돌면서 이에 맞는 Near Past Block(이하 NPB)을 찾는다(03번째 줄). 이 때 고려되어야 할 사항이 있는데, 삽입될 NPB에 시공간 상으로 이어진 기존의 이동체가 존재하는 경우에는 합병 연산을 하고(05번째 줄), 존재하지 않으면 삽입한다(07번째 줄).

마지막으로, 이동체의 과거 위치 정보는 NPB에 대응되는 2D R-트리에 저장하는데, 이는 별도의 프로세스가 주기적으로 NPB를 체크하여 NPB가 허용된 범위를 초과하거나 허용된 시간 범위를 초과하게 되면 저장하는 방식을 취한다. 이는 이동체의 과거 위치

```
Input
m:              새로운 위치를 가진 이동체
om:             갱신되기 이전의 이동체
m_line_id:      새로운 위치를 가진 이동체에 대응되는 라인 세그먼트의 id
om_line_id:     갱신되기 이전의 이동체에 대응되는 라인 세그먼트의 id
NewObj_n:       폴리라인 단위로 구성된 가까운 과거 위치 이동체
Begin
01 :    NewObj ← convert line segment of m_line_id and om_line_id
           into polyline segment
02 :    while NewObj_n is not empty
03 :       NPB←find near past block using NewObj_n
04 :       if find linked object in NPB using NewObj_n
05 :          merge NewObj_n and object of NPB
06 :       else
07 :          Insert NewObj_n into NPB
08 :       end if
09 :    end for
End
```

(알고리즘 2) update near past position



a	:	갱신되기 이전의 이동체
a'	:	새롭게 갱신된 이동체
a_{new}	:	폴리라인 단위로 구분하기 위한 임시 이동체
P1~P3	:	폴리라인
p1~p2	:	이동체 과거 위치 정보의 position interval

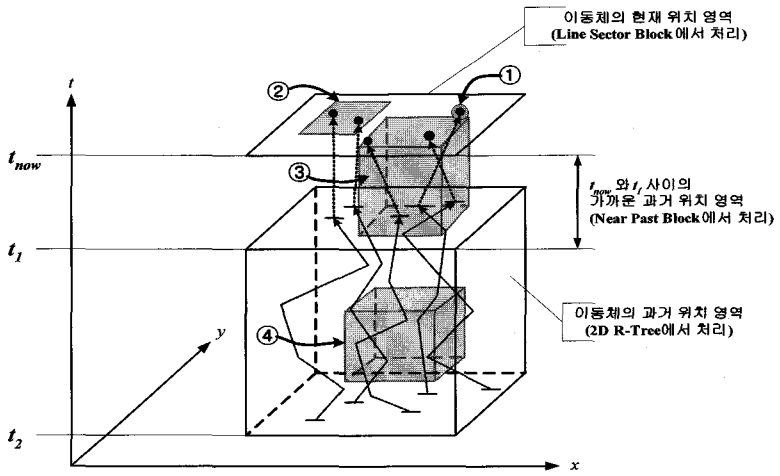
<그림 14> 두 개의 이동체를 이용한 과거 위치 정보의 생성

갱신이 자주 일어나게 되어 발생하는 I/O 부하를 감소시키기 위함이다(3.1절의 <그림 10> 참조). 이 때 저장되는 데이터는 NPB에서 생성하는 데이터형식과 동일하다.

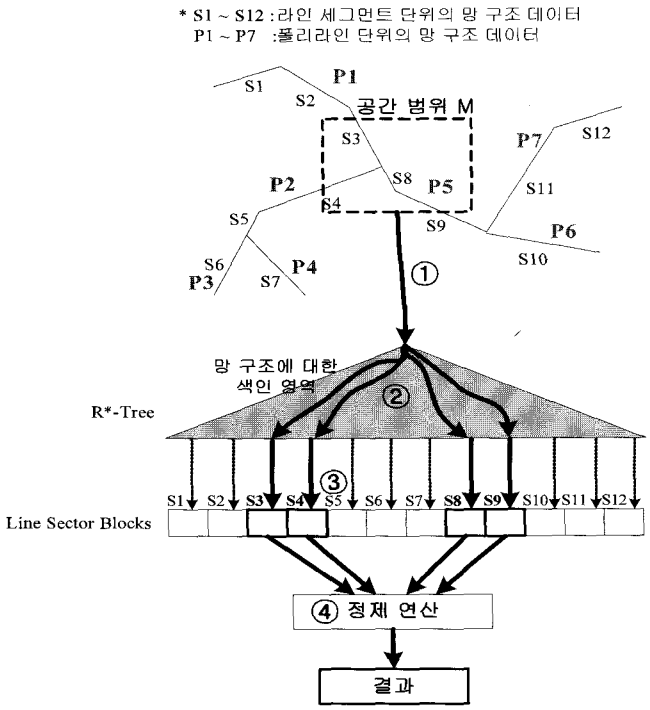
3.3 이동체의 검색 연산

본 절에서는 UCN-트리에서 지원하는 이동체의 검색 연산에 대해 설명한다. UCN-트리는 <그림 15>와 같은 이동체의 현재 및 과거 위치 검색을 지원한다[8].

<그림 15> ① 형태의 질의는 id검색 질의로, "이동체의 id가 x인 이동체의 현재 위치를 구하라" 와 같은 형태이다. 이 때 결



<그림 15> 이동체의 검색 연산



<그림 16> 현재 위치 이동체의 검색 과정(<그림 15> ②형태의 질의)

과는 이동체 하나의 위치 정보가 된다. 이와 같은 검색은 이동체의 위치 값이 아닌 id만으로 검색이 가능하기 때문에 Data Blocks for Moving Object에서 이동체를 검색하여 그 이동체의 x, y좌표 값을 결과로 보내주면 된다.

<그림 15> ② 형태의 질의는 현재 위치 범위 검색 질의로, "공간 범위 M(= MBR 영역)에 존재하는 모든 이동체의 현재 위치를 구하라" 와 같은 형태의 질의이다.

이동체의 현재 위치에 대한 검색 과정은 <그림 16>의 번호 순서대로 다음과 같이 진행되며, 이 때의 결과는 다수의 이동체의 위치 정보가 된다.

① 공간 범위 M(MBR 영역)을 입력 값으로 받는다.
② R*-트리에서 공간 범위 M에 대응되는 라인 세그먼트를 검색한다.
③ 라인 세그먼트에 대응되는 Line Sector Block을 검색한다.
④ 검색된 Line Sector Block에 저장되어 있는 이동체에 대해 정제 연산을 한 후 결과로 내보낸다.

<그림 15> ③ 형태의 질의는 이동체의 현재부터 과거까지의 위치 검색 질의로, "공간 범위 M(= MBR 영역)에 속하고 현재부터 t_a 에 속하는 모든 이동체 궤적을 구하라" 와 같은 형태의 질의이다. 이 때 결과는 범위 내에 속하는 모든 이동체의 궤적이 된다.

<그림 15> ④ 형태의 질의는 과거 위치 범위 검색 질의로, "공간 범위 M(= MBR 영역)에 속하고 시간 t_a ~ t_b에 속하는 모든 이동체 궤적을 구하라" 와 같은 형태의 질

의이다. 이 때 결과는 범위 내에 속하는 모든 이동체의 궤적이 된다.

<그림 15> ③ 형태의 질의와 <그림 15> ④ 형태의 질의에 대한 처리는 UCN-트리에서 다음과 같이 <그림 17>의 번호 순서대로 동일하게 진행된다.

① 공간 범위 M(MBR 영역)을 입력 값으로 받는다.

② R*-트리에서 공간 범위 M에 대응되는 라인 세그먼트를 검색하고, 대응되는 Line Sector Block을 검색한다. 이 때, 시간 도메인이 현재를 포함하고 있지 않다면 이 과정은 생략 가능하다.

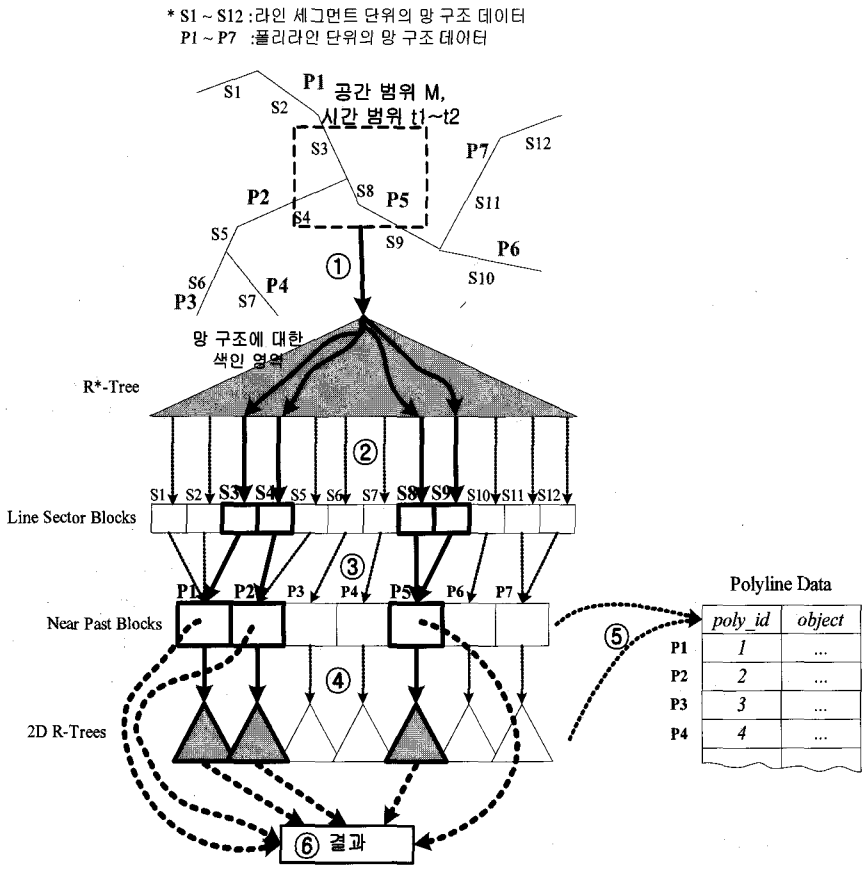
③ Line Sector Block에 대응되는 Near Past Block을 검색한다. 이 때 시간 도메인의 범위가 대응되는 블록에 해당하지 않으면 이 과정은 생략한다.

④ Near Past Block에 대응되는 2D R-트리를 검색한다.

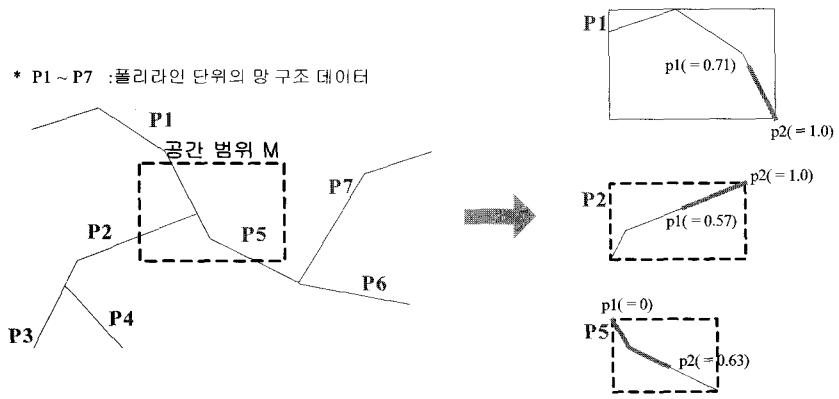
⑤ 폴리라인 데이터가 있는 테이블을 참조하여 질의의 형식을 폴리라인 단위의 질의로 바꾸어 처리한다.

⑥ Near Past Block 및 2D R-트리에서 검색된 결과를 내보낸다.

검색 과정 중, <그림 17> ⑤ 과정의 경우에



<그림 17> 이동체의 과거 위치 검색



<그림 18> 공간 범위를 폴리라인 단위의 범위로 변형

대해 자세히 살펴본다. 과거 위치 데이터는 현재 위치 데이터와 달리 폴리라인 단위로 변형하여 저장되기 때문에, 주어진 공간 영역을 이에 맞게 변환하는 과정을 수행해야 한다.

<그림 18>에서 좌측의 공간 범위 M의 경우 일반적인 MBR영역(x1, x2, y1, y2로 이루어진 사각형)이기 때문에, 과거 위치 검색을 위해서는 이를 <그림 18>의 우측과 같이 폴리라인 단위로 변형하여야 한다. 공간 범위 M은 P1, P2, P5에 속해 있기 때문에 이와 같이 총 3개의 영역으로 분리되어 변형된다.

3.4 이동체의 삽입 및 삭제 연산

본 논문에서는 이동체의 삽입 및 삭제 연산은 다루지 않는다. 이동체는 자신의 위치만을 갱신할 뿐 삭제가 일어나지 않으며, 이동체의 삽입은 색인의 변화와 관계 없이 이동체 데이터를 관리하는 테이블에서 일어나고, 이후의 이동체 위치 관련 연산은 이동체의 갱신 과정에 포함되기 때문이다[4, 5, 8, 9].

4. 성능평가

본 장에서는 제안 색인인 UCN-트리와 기존의 현재 위치 색인 및 과거 위치 색인과의 성능을 실험을 통하여 평가한다.

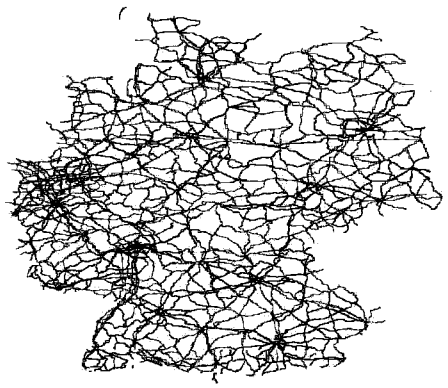
4.1 실험 환경

실험에 사용된 시스템의 하드웨어 및 소프트웨어 환경은 <표 1>과 같다.

<표 1> 성능 평가에 사용된 시스템 환경

기종	IBM PC 호환
CPU	Pentium IV 2.4GHz
메인 메모리	1GB
하드 디스크	160GB
운영 체제	Windows XP Professional
개발 언어	Visual C++ 6.0

실험에 사용된 이동체는 Brinkhoff가 제안한 '제한된 망 구조 기반 이동체 생성기'를 이용하여 생성하였다[3]. 이 이동체 생성기는 제한된 망 구조 기반 이동체 색인의 성능 평가에 일반적으로 사용된다[1, 6, 11]. 실험에 사용된 망 구조 데이터는 기존의 제한된 망 구조 기반 이동체 색인인 MON-트리의 성능 평가에서 사용되었던 망 구조 데이터 셋이다[1, 19]. 이 망 구조 데이터 셋은 총 30,674개의 라인 세그먼트로 이루어져 있으며, 5,342개의 폴리라인으로 되어 있다. 데이터의 크기는 812,171bytes 이다.



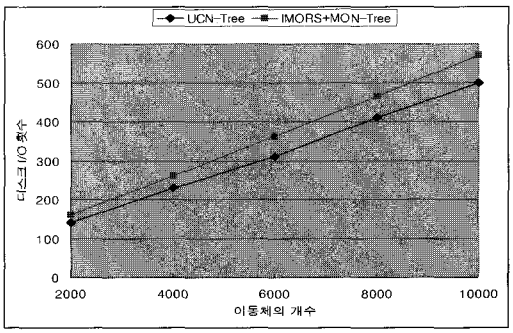
<그림 19> 실험에 사용된 망 구조 데이터

비교 대상 색인은 제안 색인인 UCN-트리와 IMORS+MON-트리이다. IMORS+MON-트리는 2.1 절과2.2 절에서 다루었던 현재 위치 색인인 IMORS의 결과와 과거 위치 색인인 MON-트리의 결과를 가지고 합병 연산을 수행한 것이다.

4.2 실험평가

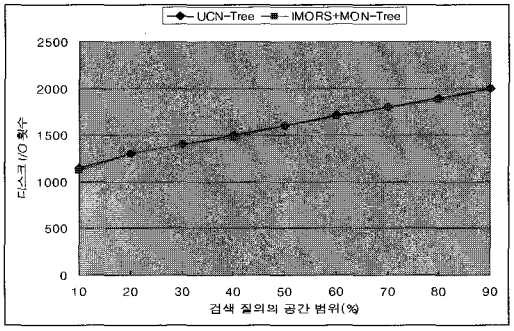
다음은 이동체의 갱신에 대한 성능 평가이다. 이동체 데이터는 2,000개에서 최대 10,000개까지 생성하며, 성능에 대한 비교는 처리 시간이 아닌 디스크 I/O의 횟수로 정하였다. 이는 색인의 성능이 디스크의 I/O에 가장 많은 영향을 받기 때문이다. 성능 평가 비교 기준은 디스크 I/O 횟수 이므로 이동체의 속도나 방향의 조건을 달리하여도 같은 결과를 보였다.

<그림 20>의 결과와 같이, UCN-트리는 기존의 색인의 공통된 부분인 제한된 망 구조 색인 영역을 통합하였기 때문에 기존의 색인보다 디스크 I/O가 30회에서 80회 정도 적게 발생한다. 이는 색인을 통합하여 관리하는 것이 색인을 별도로 구축하여 관리하는 것 보다 갱신 성능이 뛰어나다는 것을 증명한다.



<그림 20> 이동체의 갱신에 대한 성능 비교

다음은 이동체의 현재 위치 검색에 대한 성능 평가이다. 이동체 데이터는 총 10,000개이며, 검색 질의의 공간 범위는 전체 공간 범위의 10%에서 90% 사이로 생성한다.

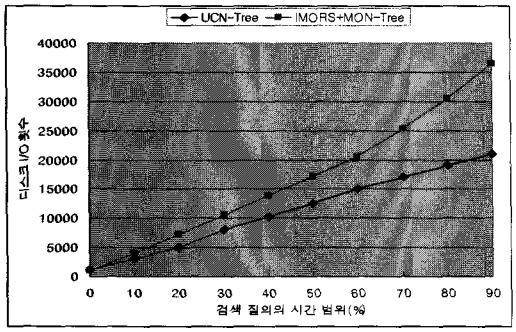


<그림 21> 이동체의 현재 위치 검색에 대한 성능 비교

<그림 21>의 결과와 같이, 이동체의 현재 위치 검색에 대한 성능은 UCN-트리와 기존의 색인의 차이가 거의 없다. 이는 UCN-트리가 이동체의 현재 및 과거 위치를 모두 지원해도 현재 위치 색인에 비해 성능이 떨어지지 않음을 증명한다. 그러나, 제안기법의 과거위치 색인은 FNR-트리와 MON-트리의 성능적 차이 때문에 다소 떨어진다[1].

다음은 이동체의 현재부터 과거까지의 위치 검색에 대한 성능 평가이다. 이동체 데이터는 총 10,000개이며, 각각의 이동체는 5초 단위의 시간 범위로 되어 있는 5,000개의 과거 위치 궤적을 가진다. 검색 질의의

공간 범위는 전체 공간 범위의 10%로 제한하였다. 시간의 범위는 현재부터 전체 시간 범위의 90% 사이로 생성한다.



<그림 22> 이동체의 현재부터 과거까지의 위치 검색에 대한 비교

<그림 22>의 결과와 같이, 이동체의 현재부터 과거까지의 위치 검색에 대한 성능은 UCN-트리가 기존의 색인에 비해 약 10%에서 70% 정도 좋은 성능을 보인다. 이는 기존의 색인에 있었던 문제점인 이동체의 현재부터 과거까지의 위치에 대한 검색 시 발생하는 망 구조 색인 영역의 중복 탐색 비용을 제안 색인인 UCN-트리에서 감소시켰기 때문이다.

5. 결론 및 향후 연구

본 논문에서 제안한 제한된 망 구조 내의 이동체를 위한 통합 색인은 망 구조 기반 이동체의 현재 위치 색인 및 과거 위치 색인의 공통된 부분인 망 구조를 저장·관리하는 부분을 공유하였다. 이는 기존 색인의 문제점인 색인을 별도의 시간 도메인 영역별로 구축할 경우에 발생하는 색인의 공간적인 비용 및 유지 비용에 대한 부담을 줄이고 갱신 과정에서 따르는 망 구조 색인 영역의 중복 탐색을 줄여 갱신 성능을 향상시켰다. 또한 현재 위치에 대한 색인, 과거 위치에 대한 색인 및 현재 위치의 갱신으로 발생하는 가까운 과거 위치에 대한 생성 및 처리 기능을 통합하였다. 이는 기존 색인에서 이동체의 현재부터 가까운 과거까지의 위치를 검색할 경우 중복된 공간 탐색이 발생하는 문제점을 해결하여 기존 색인에 비해 검색 성능을 향상시켰다.

본 제안 색인은 기존의 제한된 망 구조 기반 이동체 색인과 성능 평가를 하였다. 실험 결과와 같이 본 제안 색인은 기존의 현재 위치 색인 및 과거 위치 색인과 비교했을 때 이동체의 갱신 성능이 기존의 색인보다 우수함을 나타내었으며, 이동체의 현재 위치 검색에 대한 성능 평가를 통해 통합 색인을 구축하여도 각 색인의 성능과 비슷하다는 것을 보였다. 또한 기존 색인에 비해 이동체의 현재부터 과거까지의 위치 검색 성능을 크게 향상시킴을 증명하였다.

본 논문의 향후 연구는 도로 교통 정보 안내 시스템과 같은 실제 응용 시스템에 적용하기 위해 제안한 색인을 도로 망 정보에 대한 통계 정보 수집 및 분석을 지원하도록 확장하는 것이다.

참고문헌

1. V.T. Almeida, R.H. G□ting,"Indexing the Trajectories of Moving Objects in Networks," Conf. on Scientific and Statistical Database Management, 2004.
2. N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger,"The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles,"In Proc. of the ACM Special Interest Group on Management of Data Conf, 1990, pp. 322-331.
3. T. Brinkhoff, "A framework for generating network-based moving objects," GeoInformatica, Vol.2, No.6,

2002, pp. 153-180.
4. M. Erwig, R.H. G□ting, M. Schneider, and M. Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases," GeoInfomatica Vol.3, No.3, 1999, pp.269-296.
5. L. Forlizzi, R.H. G□ting, E. Nardelli, M. Schneider, "A Data Model and Data Structures for Moving Objects Databases," In Proc. of the ACM Special Interest Group on Management of Data Conf, 2000, pp. 319-330.
6. E. Frentzos, "Indexing Objects Moving on Fixed Networks," Symp. on Spatial and Temporal Databases, 2003, pp. 289-305.
7. A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," In Proc. Of the ACM Special Interest Group on Database Int'l. Conf. on Management of Data, 1984.
8. R.H. G□ting, V.T. Almeida, Z. Ding, "Modeling and querying moving objects in networks,"Fernuniversit□t Hagen, Fachbereich Informatik, 2004.
9. R.H. G□ting, M.H. B□hlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," ACM Transactions on Database Systems, Vol.25, No.1, 2000, pp. 1-42.
10. C.S. Jensen, D. Pfoser, "Indexing of network constrained moving objects," In Proc. of the Int' l. ACM Symp. on Advances in Geographic Information Systems, 2003.
11. K. Kim, S. Kim, T. Kim, K. Li, "Fast indexingand updating method for moving objects on road networks,"In Proc. of the Web Information Systems Engineering Workshops, 2003.
12. D. Kwon, S. Lee, S. Lee, "Indexing the Current Positions of Moving Objects Using the Lazy Update R-Tree," In Mobile Data Management, 2002, pp. 113-120.
13. M.A. Nascimento, J.R.O. Silva, "Towards historical R-Trees," In Proc. of the ACM Symp. on Applied Computing, 1998, pp. 235-240.
14. D. Pfoser, C.S. Jensen, Y. Theodoridis, "Novel Approaches in Query Processing for Moving Object Trajectories,"In Proc. of the Int'l. Conf. on Very Large Data Bases, 2000, pp. 395-406.
15. A. Silbershatz, H.F. Korth, S. Sudarshan, "DATABASE SYSTEM CONCEPTH 4TH EDITION," McGrawHill, 2002, pp. 465-478.
16. Z. Song, N. Roussopoulos, "Hashing Moving Objects," In Mobile Data Management, 2001, pp. 161-172.
17. Y. Tao, D. Papadias, "MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries,"In Proc. of the Int'l. Conf. on Very Large Data Bases, 2001, pp. 431-440.
18. Y. Theodoridis, M. Vazirgiannis, and T. Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications," In Proc. of the IEEE Conference on Multimedia Computing and Systems, 1996.

19. The R-TreePortal, http://www.rtreeportal.org/roads.zip
20. 심태정, 김재홍, 정원일, 장용일, 배해영, "위상 정보를 이용한 갱신 정책과 불확실한 위치 정보에 대한 추정 기법," 정보처리학회 춘계학술대회 논문집, Vol.10, No.1, 2003, pp. 1651-1654.
21. 정명호, 장용일, 박순영, 오영환, 배해영, "도로 네트워크 기반 이동체의 통합 색인," 한국정보과학회 추계학술대회 논문집, Vol.31, No.2, 2004, pp. 4-6,.

천종현
2005년 인하대학교 산업공학과(공학사)
2005년~현재 인하대학교 대학원 컴퓨터·정보공학과(석사과정)
관심분야: 메인 메모리 데이터베이스, 공간 데이터베이스 클러스터, 그리드 데이터베이스

정명호
2003년 인하대학교 컴퓨터공학부(공학사)
2005년 인하대학교 컴퓨터·정보공학과(공학석사)
2005년~현재 알티베이스 연구원
관심분야: 실시간 데이터베이스시스템, 클러스터 데이터페이스시스템

장용일
1997년인하대학교 전자계산공학과 (공학사)
2001년인하대학교 컴퓨터공학부 (공학석사)
2003년~현재 인하대학교 컴퓨터공학부 박사과정
관심분야: 웹 & 모바일 GIS, 공간 데이터베이스 클러스터, 위치기반서비스, 그리드 데이터베이스

오영환
1993년 인하대학교 전자계산공학과(공학사)
1997년 인하대학교 전자계산공학과(공학석사)
2001년 인하대학교 전자계산공학과(공학박사)
2000년~2001년 ㈜케이지아이 시스템 개발부 부장
2002년~현재 나사렛대학교 컴퓨터공학부 조교수
관심분야: 공간 데이터베이스, 데이터베이스 시스템의 보안, GIS

배해영
1974년 인하대학교 응용물리학과(공학사)
1978년 연세대학교 대학원 전자계산학과(공학석사)
1989년 숭실대학교 대학원 전자계산학과(공학박사)
1985년 Univ. of Houston 객원교수
1992년~1994년 인하대학교 전자계산소 소장
1982년~현재 인하대학교 컴퓨터공학부 교수
1999년~현재 지능형GIS연구센터 센터장
2000년~현재 중국 중경우전대학교 대학원 명예교수
2004년~현재 인하대학교 정보통신대학원 원장
관심분야: 분산 데이터베이스, 공간 데이터베이스, 지리정보 시스템, 멀티미디어 데이터베이스 등