공 학 박 사 학 위 논 문

이동객체궤적에 대한 효율적인 범위 및 최근접이웃 질의처리

2015년 2월

부경대학교대학원

컴퓨터공학과

박 영 희

공 학 박 사 학 위 논 문

이동객체궤적에 대한 효율적인 범위 및 최근접이웃 질의처리

지도교수 조 우 현

이 논문을 공학박사 학위논문으로 제출함.

2015년 2월

부경대학교대학원

컴퓨터공학과

박 영 희

박영희의 공학박사 학위논문을 인준함.

2015년 2월 27일

위 원 장 공학박사 정 목 동 (인)

위 원 이학박사 윤 성 대 (인)

위 원 공학박사 류 시 국 (인)

위 원 공학박사 김 종 진 (인)

위 원 공학박사 조 우 현 (인)

목 차

그 림 목 차

이동객체궤적에 대한 효율적인 범위 및 최근접이웃 질의처리

박 영 희

부경대학교 대학원 컴퓨터공학과

요 약

최근 많은 응용프로그램에서 시공간 데이터와 멀티미디어 데이터 등이 사용되고 있다. 스마트폰과 같은 이동 통신 매체의 발달과 LTE, NFC, RFID 등 무선통신의 발달로 실시간으로 이동 객체의 위치데이터를 수집하여 활용하는 위치 기반의 서비스들이 다방면의 개발에 이용되고 있다. 이에 따라 대용량의 이동객체 위치 데이터들을 효율적으로 저장하는 방법과 여러 질의를 좀 더 빠르게 처리할 수 있는 방법들에 대한 연구들이 진행 중이다.

본 논문에서는 이동객체궤적에 대하여 단순화 기법을 사용하여 단순화한 후에 색인구조를 생성하고 이 색인구조를 이용하여 범위질의를 효율적으로 처리할 수 있는 알고리즘을 제안한다. 이동객체궤적의 단순화 기법으로는 Douglas-Peucker 알고리즘을 수정하여 이용한다. 제안된 방법과 기존의 최소 경계 사각형(MBR)을 이용한 색인 방법을 실험을 통하여 비교 및 분석한다.

실험 결과로 제안된 방법에서는 색인 데이터 량이 상대적으로 작아지고 색인 및 질의 처리방법이 간단하며 기존의 방법보다 시공간적으로 효율적임을 확인하였다. 또한 이 색인 구조를 이용하여 최근접이웃 검색질의를 효율적으로 처리할 수 있는 알고리즘을 제안한다. 제안된 방법으로 대용량의 데이터가 더 적은 양의 데이터로 단순화 되고 얼마나 더 효율적으로 질의를 처리하는지 실험을 통하여 확인하였다.

Efficient Range and Nearest Neighbor Query Processing on Moving Object Trajectories

Young-Hee Park

Department of Computer Engineering, Graduate School,

Pukyong National University

Abstract

The management and analysis of spatio-temporal and multimedia data is a hot issue in database research, because such data types are handled in manny applications. Because of the rapid growth of mobile communication and wireless communication, Location-based services are handled in many applications. So, the management and analysis of spatio-temporal data are very important in database research. Also, query processing of such a content is very important for these applications.

This dissertation addresses algorithms that make index structure by using Douglas-Peucker Algorithm and process range query efficiently on moving objects trajectories. We compare and analyze our algorithms and minimum bound rectangle methods by experiments. Our algorithms make smaller size of index structure and process more efficiently. And using

the index structure, we present algorithms that process nearest neighbor search query efficiently on moving objects trajectories.

We compare and analyze our algorithms by experiments. Experiments of this dissertation can be easily checked that our algorithms make small size of index structure and process the query more efficiently.

1. 서론

이동객체란 시간의 변화에 따라 공간적인 위치 및 모양이 연속적으로 변하는 시공간데이터를 말하며 이동객체궤적은 이러한 데이터들이 시간에 따라 연결된 궤적을 말한다. 최근 다양한 응용프로그램에서 위치기반 서비스의 사용이 증가하면서 이러한 데이터들을 관리하고 사용하는 이동객체 데이터베이스에 대한 관심이 증가하고 있다[1,2]. 위치기반 서비스들의 출현으로 공간과 시간 도메인을 다루기 위한 효율적인 질의처리 알고리즘과 기술이 필요하다. 이러한 서비스들의 예로서 교통모니터링, 응급의료서비스, 동물이동특성관찰 등이 있다.

이동객체 질의에는 시간에 따라 과거 질의, 현재 질의, 미래 질의 등이 있다[3]. 이 중에서 현재 질의는 과거 질의 혹은 미래 질의와 같은 방식으로 처리될 수 있다. 따라서 이동객체 질의를 과거질의와 미래질의로 구분할 수 있다. 과거질의는 질의범위 내에 단순히 이동 객체가 있었는지 유무를 검색하는 좌표기반 질의(coordinate-based query)와 질의범위 내에서 특정 이동객체의 궤적정보를 검색하는 궤적기반 질의(trajectory-based query)가 있다.

좌표기반 질의는 시공간 삼차원으로 이루어진 공간상에서 좌표조건을 만족하는 이동객체들을 검색하는 질의이다. 점 질의는 시공간상에서 주어진 하나의 점과 관련된 이동 객체의 유무를 검색한다. 범위질의는 특정 범위 내에 이동객체의 유무를 검색한다. 근접이웃 질의는 주어진 점이나 이동객체궤적으로부터 가장 가까운 k개의 이동객체들을 검색한다. 궤적기반

질의는 위상질의(topological query)와 항해질의(navigational query)로 나눌 수 있다. 위상 질의는 이동 객체 궤적의 움직임을 추적하는 질의이다. 이 질의는 이동 객체가 질의범위에 들어왔는지, 나갔는지, 통과했는지, 우회했는지를 검색한다. 항해 질의는 이동객체의 궤적정보를 통하여 속도나 방향 등을 계산한다. 미래 질의는 이동 객체의 현재의 위치, 속도, 방향 등을 기반으로 이동 객체의 미래 위치를 예측하는 질의이다. 미래 예측에 대한 정확성 여부가 매우 중요하나, 현재까지는 만족스러운 정확성을 보장하지는 못하고 있다.

시공간데이터를 생성하는 단말기들은 많은 양의 이동객체궤적 데이터를 생성하며, 이를 통한 위치기반 서비스는 대량의 데이터를 효율적으로 처리할 수 있는 적절한 색인구조가 필요하다. 이동객체궤적 데이터에 대한 색인구조를 생성하는 방법은 다양하며, 현재까지 B^+-트리, 최소경계사각형(minimum bounding rectangle: MBR) 근사법, R-트리, 다항식 근사법, 사분트리 등의 여러 가지 색인 방법이 제시되었다[4-10]. 그리고 각 색인구조를 이용하는 범위 질의처리 알고리즘들과[11-16] 최근접이웃 질의처리 알고리즘들이 제시되었다[17-23].

본 논문에서는 Douglas-Peucker 알고리즘[24]을 이용하여 이동객체궤적을 단순화하고 이를 이용하여 색인구조를 생성하고 이 색인구조에 대하여 범위질의(range query)를 처리하는 알고리즘을 고안한다. 지금까지 널리 연구된 MBR방식과 그 성능을 비교 및 분석하여 본 논문의 방법이 더 단순하고 적은 양의 데이터를 생성하며 범위질의 처리속도가 빠른 이점이 있음을 보인다. 또한 이 색인구조를 응용하여 이동객체궤적 데이터에 대한

최근접이웃 질의(nearest neighbor query)를 처리하는 알고리즘을 제안한다. 제안된 알고리즘이 질의를 얼마나 더 효율적으로 처리하는지 실험을 통하여 비교분석 하였고, 더 적은양의 데이터 처리를 통하여 질의처리 속도가 더 빠르다는 장점을 확인한다. 원래 시간(t)과 이차원좌표(x, y) 데이터를 가지는 삼차원의 시공간 데이터를 다루어야 하나, 알고리즘과 실험의 단순성을 위하여 시간(t)과 일차원좌표(x) 데이터만을 가지는 이차원 데이터로 축소하여 알고리즘을 설계하고 실험하였다. 이는 단지 실험의 편의를 위한 것으로 실제 삼차원데이터를 처리하는 알고리즘으로 확장하는 것은 면적계산(시간, x)을 체적계산(시간, x, y)으로 변환하면 가능하므로 간단하다.

본 논문의 구성은 다음과 같다. 제2장에서는 이동객체궤적 데이터를 색인방법과 이 색인구조를 이용하여 범위 및 최근접이웃 질의처리에 대한 관련 연구들을 살펴보고 제3장에서는 색인구조를 생성하는 알고리즘과 이 색인구조를 이용하는 범위 질의처리 알고리즘과 최근접이웃 질의처리 알고리즘을 제안한다. 제4장에서는 제안하는 범위 및 최근접이웃 질의처리 알고리즘들의 효율성을 입증하기 위해 실험을 통하여 기존 방법과의 성능 평가를 수행한다. 마지막으로 제5장에서는 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

이 장에서는 이동객체궤적 데이터를 색인하는 관련 연구를 먼저 살펴보고 그 다음에 범위 질의를 처리하는 여러 방법들과 최근접이웃 질의를 처리하는 여러 방법들을 조사한다.

2.1 색인구조

이동객체궤적에 대한 질의는 범위질의, 최근접이웃검색, k-최근접이웃검색[21,22,25-40]등이 있는데 이러한 질의들을 효율적으로 처리하기 위해서 색인구조를 생성하여 이용한다. 이러한 색인구조를 생성하기 위해서 어떠한 모양의 데이터 구조와 근사 방법을 사용하는지는 아주 중요한 요소가 되며 색인구조 검색 효율에 영향을 미친다.

주로 쓰이는 공간색인 방법으로는 이동객체궤적 데이터가 포함된 공간을 분할하여 색인하는 사분트리와 해싱을 이용한 방법[8]과 이동객체궤적 데이터를 다항식으로 근사하여 검색 공간을 줄이는 방법[6,7], 이동객체궤적 데이터를 최소경계사각형(MBR)으로 분할하여 R-트리 기반의 색인구조를 생성하는 방법[5], B^{+}-트리를 기반으로 하는 방법[4]과 그리드 구조[38]와 같은 데이터 분할 색인이 있다.

2.1.1 사분트리(quadtree)

사분트리(quadtree)는 각 사분면에 겹치는 rectangle의 개수가 페이지의 용량보다 작을 때까지 검색공간은 재귀적으로 사분면으로 분할된다[39]. 사분면은 각각 North West(NW), North East(NE), South West(SW), South East(SE)로 부르고 색인은 4진 트리로 표현한다. 각 단말노드는 하나의 디스크 페이지와 연관된다. 아래 그림 1는 사분트리의 예이다. 단말노드들은 부합하는 rectangle과 함께 표시를 하였고 각 페이지의 용량은 4개의 엔트리라고 가정을 한다. 사분트리는 자식노드의 수가 4로 고정됨으로 디스크 페이지로 사상하는 것이 쉽지 않다.

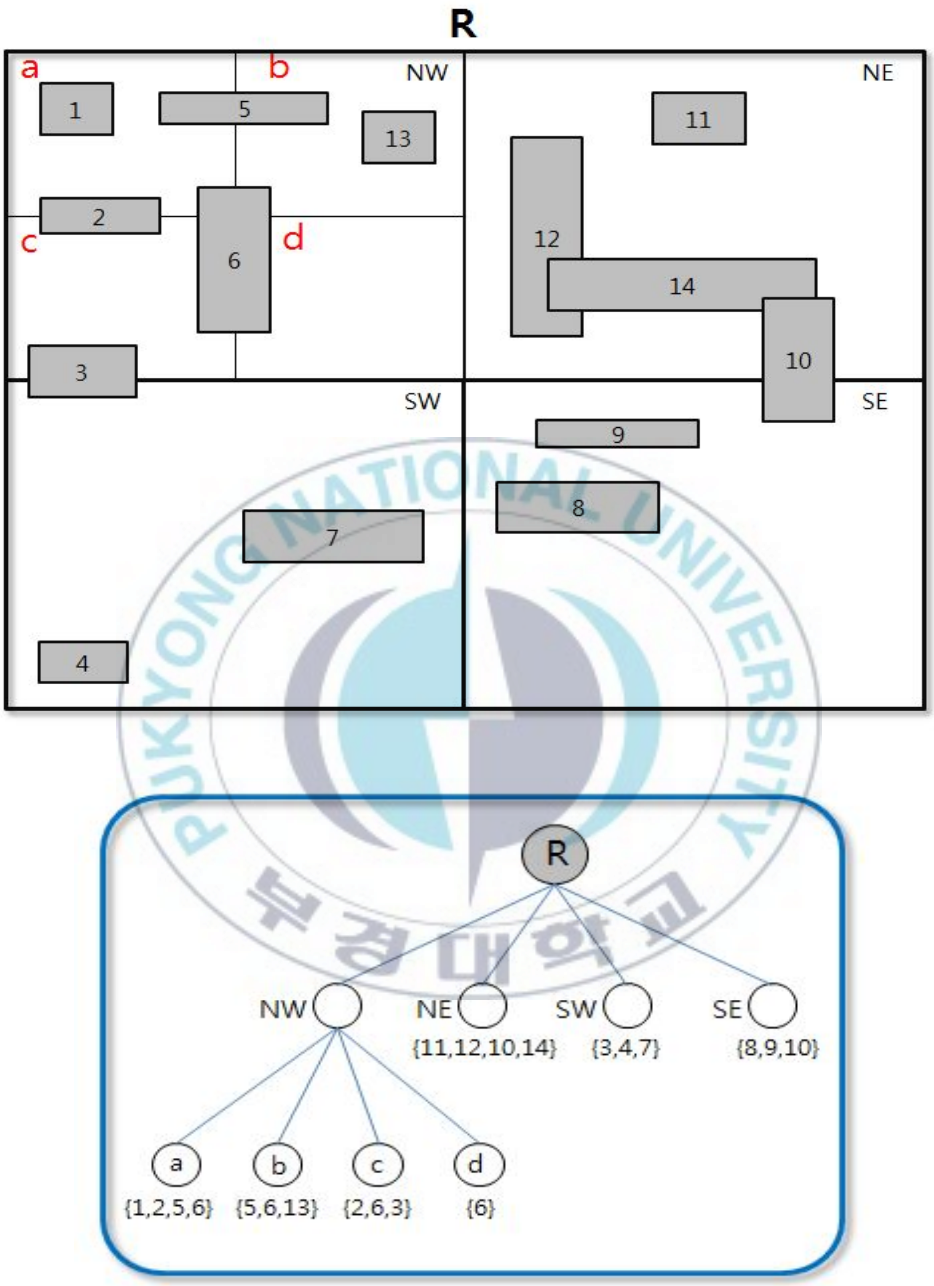

그림 1. 사분트리

Fig. 1. Quadtree

2.1.2 최소경계사각형(MBR)을 이용한 R-트리

R-트리는 각 노드가 하나의 디스크 페이지에 부합하는 깊이-균형 트리(depth-balanced tree)이다[39]. R-트리 계열의 색인 기법은 MBR을 색인으로 트리를 구성한다. 공간 객체들이나 자식 노드의 MBR들을 둘러싸는 MBR을 만들었을 때, MBR의 내부에서 공간 객체가 차지하는 부분을 제외한 나머지 영역을 빈 공간(dead space)라 하며 이러한 빈 공간 비율(dead space ratio)이 높을수록 성능의 저하와 저장의 유용성(storage utilization)을 떨어뜨린다[40,41]. 그림 2는 질의 구역이 주어졌을 때의 빈 공간을 가지는 노드의 빈 공간 비율을 나타낸 것이다. MBR 위주의 R-트리 계열 중에서 이 논문에서는 R-트리를 이용했다.

R-트리의 특성은 루트노드를 제외한 각 노드에서 엔트리들의 개수는 m과 M사이이다. 비단말노드 N에서는 각 엔트리(dr, nodeid)에서 dr은 N의 자식 노드의 directory rectangle이고 이것의 페이지 주소는 nodeid이다. 각 단말 엔트리(mbr, oid)에서 mbr은 주소 oid에 저장된 객체의 공간요소에 대한 MBR이다. 루트 노드는 적어도 두 개의 엔트리를 가지며 모든 단말 노드들은 동일 레벨에 존재한다. 그림 3은 M=3이고 m=2 인 R-트리의 예이다. R-트리 색인은 궤적의 길이가 길어지거나, 많은 이동객체의 이동이 잦은 네트워크의 경우 잦은 삽입과 갱신에 따른 분할과 결합에 많은 비용이 소요된다.

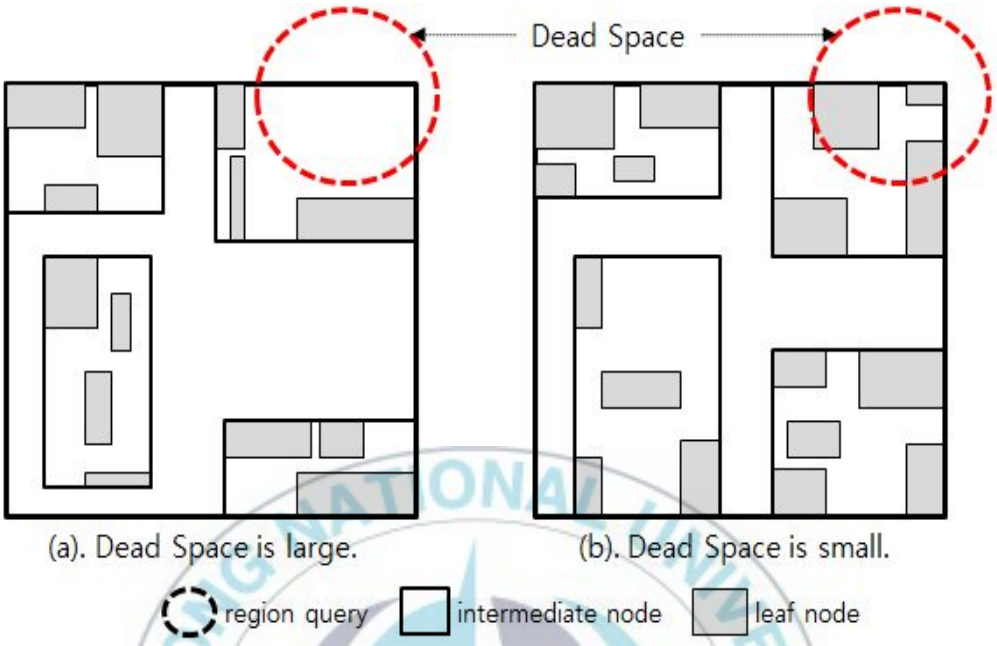

그림 2. 빈 공간을 가지는 노드의 예

Fig. 2. Example of node with dead space

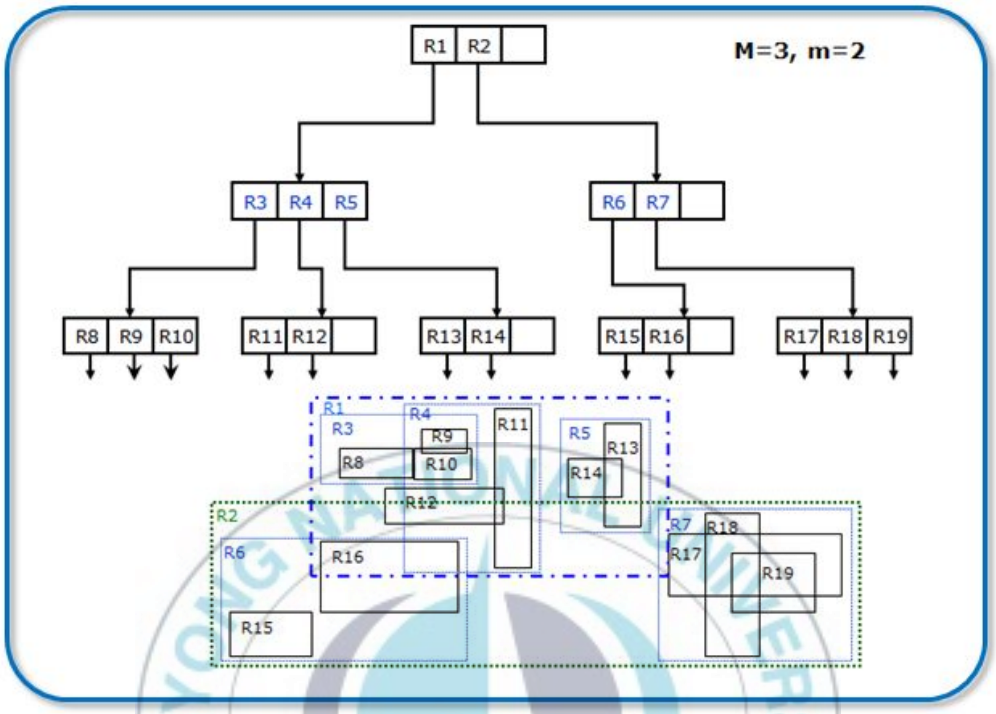

그림 3. 최소경계사각형(MBR)을 이용한 R-트리

Fig. 3. R-tree using minimum bounding rectangle

2.1.3 그리드

이산적인 그리드 표현법을 위해서 전체 작업영역을 같은 크기의 그리드 셀들로 나누고 각 셀에는 x-좌표와 y-좌표의 위치에 따라 라벨을 부여한다. 예를 들면, 가장 왼쪽-아래 그리드 셀의 라벨은 (1,1)이고 가장 오른쪽-위 그리드 셀의 라벨은 (m,n)이다. 여기서 m과 n은 열과 행에서 셀의 총 개수이다. 어떤 그리드 셀 g를 위하여 g.x를 열-라벨로 g.y를 행-라벨로

사용한다. 즉 g=(g.x, g.y)이다. 그림 4는 어떤 이동객체궤적을 그리드 표현으로 나타낸 예이다. 이 예에서 궤적 T를 회색 그리드 셀들의 순서로 나타낸다. 즉 T=(g(1,2), g(2,3), g(2,4), g(3,4), g(3,5), g(4,5), g(5,5), g(6,5), g(7,5), g(8,6), g(9,6), g(9,7), g(10,7), g(10,8))이다. 궤적데이터가 균일하게 분포한다면 효율적이나 편향분포의 경우에는 그리드 셀 데이터가 증가하므로 비효율적이다.

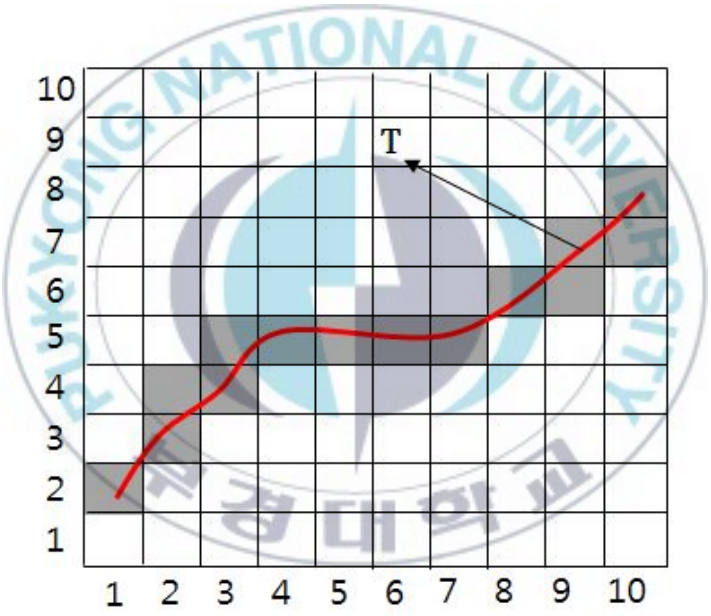

그림 4. 이동객체궤적의 그리드 표현

Fig. 4. Grid representation of moving object trajectory

이러한 방법들은 이동객체궤적에 대한 색인 구성 과정에서 특별한 데이터 구조를 사용하기도 하고 데이터가 포함된 공간 자체를 분할하는 방법 또는 MBR과 같이 실제 데이터에 대한 근사치 영역을 이용해 이동객체궤적 데이터 자체를 분할하는 방법을 사용한다.

본 논문에서 응용한 Douglas-Peucker 알고리즘[24]도 MBR과 마찬가지로 실제 위치 데이터에 대한 근사치 영역을 사용하며 실제 점과 직선으로 이루어진 데이터가 있을 때 단순화정도가 주어지면 그 값을 근사치 기준으로 하여 더 단순한 점과 직선들로 근사하는 알고리즘이다. 그림 5는 Douglas-Peucker 알고리즘에 의한 단순화 과정의 간단한 예이다. 기존의 MBR을 기반으로 하는 방법과 색인을 생성하는 과정이 비슷하기 때문에 본 논문에서 제안하는 알고리즘의 성능을 분석하기 위해 MBR방법과 비교 및 분석을 해보기로 한다.

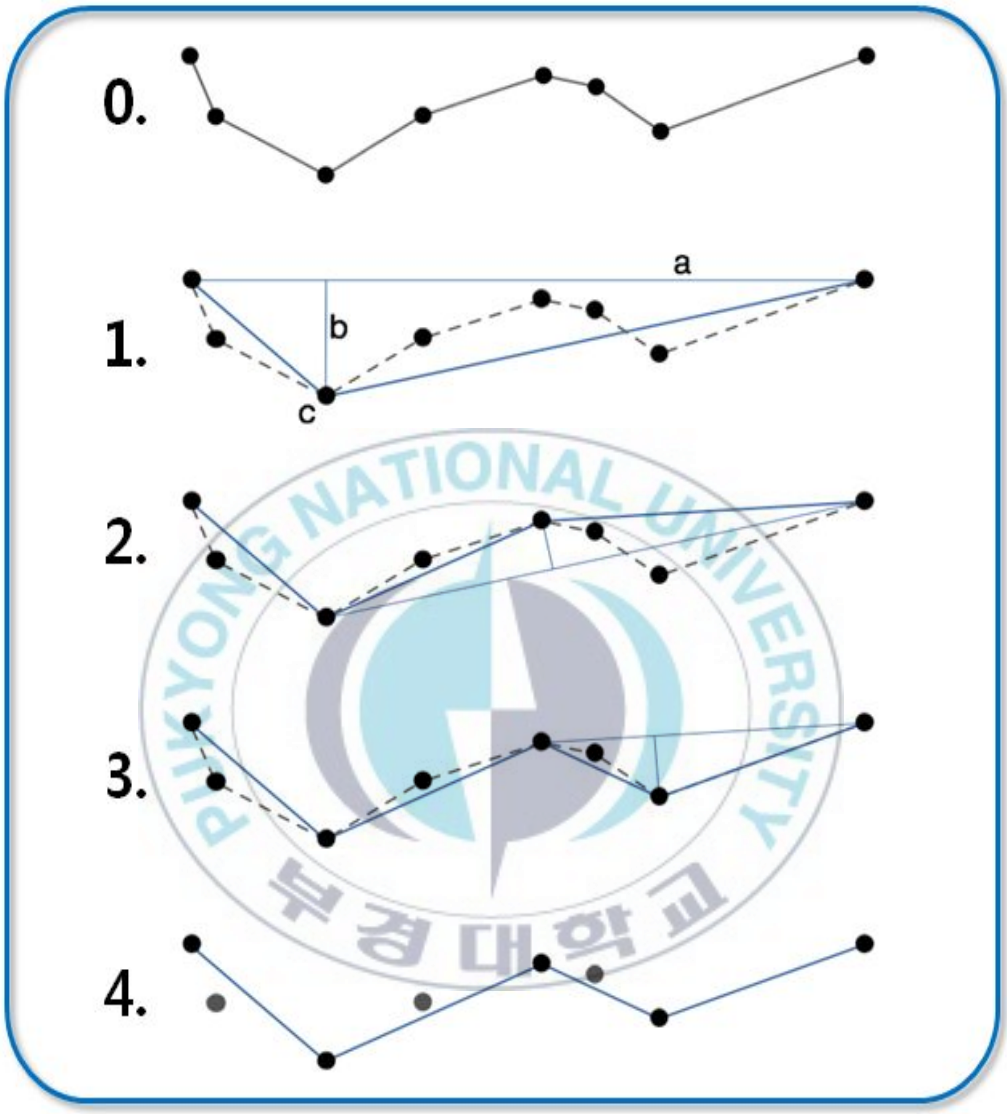

그림 5. Douglas-Peucker 알고리즘 단순화 과정

Fig. 5. Simplification process of Douglas-Peucker algorithm

2.2 범위 질의처리

이동객체궤적에 대한 질의가 주어졌을 경우에 데이터베이스에서 질의에 부합하는 결과를 찾아내는 과정은 방대한 크기의 특성상 테이블 내의 모든 레코드를 읽어 들이기 위해서는 매우 많은 디스크 입출력이 발생하고 CPU의 많은 연산처리를 요구한다. 따라서 연산의 대상이 되는 레코드를 줄임으로써 더욱 효율적인 질의 처리를 수행할 수 있는 범위 질의 처리 방법에 대한 연구가 진행되었다.

범위 질의처리에서 어떤 궤적데이터가 질의 범위에 포함되나 그 궤적을 단순화한 색인데이터는 질의 범위에 포함되지 않게 되는 것을 착오배제(false drop)라고 한다[42,43]. 이 착오배제를 없애기 위해서는 질의 범위를 확장해야 한다[43,44].

MBR기반의 색인구조는 여러 연구나 논문에서 사용되고 있으며 이동객체궤적 데이터를 가장 작은 경계의 사각형들로 분할하여 색인구조를 생성하는 방법으로 아주 효율적으로 범위질의를 처리할 수 있다. 하지만 색인구조는 질의를 효율적으로 처리할 수 있을 뿐만 아니라 업데이트 또한 효율적으로 처리할 수 있어야 하는데 MBR기반의 R-트리의 경우 색인 노드들을 분할하는 과정에서 중복에 의한 오버헤드가 심하여 업데이트 비용이 아주 높다는 단점이 있다[44,45]. 이는 계속해서 움직이는 이동객체와 같이 이동객체궤적의 데이터 변화가 빈번한 응용프로그램에서 사용하기에는 한계가 있음을 나타낸다.

2.3 최근접이웃 질의처리

이동객체궤적 데이터에 대하여 최근접이웃 질의를 처리하기 위한 방법으로는 대표적으로 R-트리를 이용한 방법들이 있다. MBR과 질의점간의 최저길이(MINDIST), 최고탐색길이(MINMAXDIST)를 이용하여 R-트리에서 최근접이웃검색 질의를 효율적으로 처리하는 알고리즘과 질의대상이 점이 아닌 공간개체로 확장하여 질의를 처리하는 방법들이 있다[46,47]. 이 방법들은 건물이나 특정 지역과 같은 시간에 따라 위치가 변하지 않는 객체나 공간객체에 대한 문제 처리를 다루고 있고, 마찬가지로 최단거리 탐색을 통한 질의처리 방법도 정적인 객체에 대한 질의만을 다루고 있다. 이외에도 움직이는 이동객체들 간의 최근접이웃검색 질의를 처리하기 위한 알고리즘도 제안되었다.

R-트리 구조에서 아직 방문하지 않은 모든 노드들 중에서 최소거리의 노드를 선택하는 최적-우선 탐색 알고리즘은 우선순위 큐를 이용한다[48]. 최적우선 알고리즘은 공간 가지치기 관점에서는 Roussopoulos등[47]의 알고리즘보다 우수하나 우선순위 큐가 매우 커질 경우에 알고리즘의 실행시간이 급격하게 증가하는 단점이 있다[49-52].

TPR-트리(time parameterized tree) 구조[37]를 이용하여 이동객체들을 위하여 최근접이웃 질의와 RNN(reverse nearest neighbor)질의에 대한 효율적인 해결책을 제시하였다. RNN질의는 질의객체에 최근접 이웃인 모든 객체들을 반환한다.

R-트리 이외에도 Voronoi 다이어그램을 이용하여 전처리 기법을 통하여 더 빠르게 최근접이웃검색 질의를 처리하는 방법[53]과 그리드를 이용하여 질의를 처리하는 방법[46]등이 제안되었다. Voronoi 다이어그램을 이용한 방법은 미리 주어진 데이터에 대한 Voronoi 다이어그램을 구한 후 각 Voronoi 셀을 MBR로 근사하여 이 MBR을 R^*-tree[54]에 저장한다. 전처리를 통하여 미리 계산을 해놓기 때문에 빠른 질의처리가 가능하지만 Voronoi셀을 MBR로 근사하면서 중첩(overlap) 영역이 많이 발생하고 되고 그 결과로 거리 비교를 해야 하는 후보가 늘어나기 때문에 삽입과 수정이 적은 상태에서만 효율적이다. 그리드를 이용한 방법은 각 궤적을 그리드화 시켜서 궤적간의 그리드 개수를 통하여 질의를 처리하는 방법으로 유사곡선의 궤적을 찾을 때는 효율적이지만 단순히 가까운 궤적을 찾는 데는 효율적이지 않다.

3. 효율적인 범위 및 최근접이웃 질의처리

이 장에서는 효율적인 범위 및 최근접이웃 질의처리를 위한 색인구조를 생성하는 알고리즘을 제안하고 이 색인구조를 이용하여 범위 질의처리 알고리즘과 최근접이웃 질의처리 알고리즘을 제안한다. 그림 6은 제안하는 알고리즘들의 단순화된 흐름도이다.

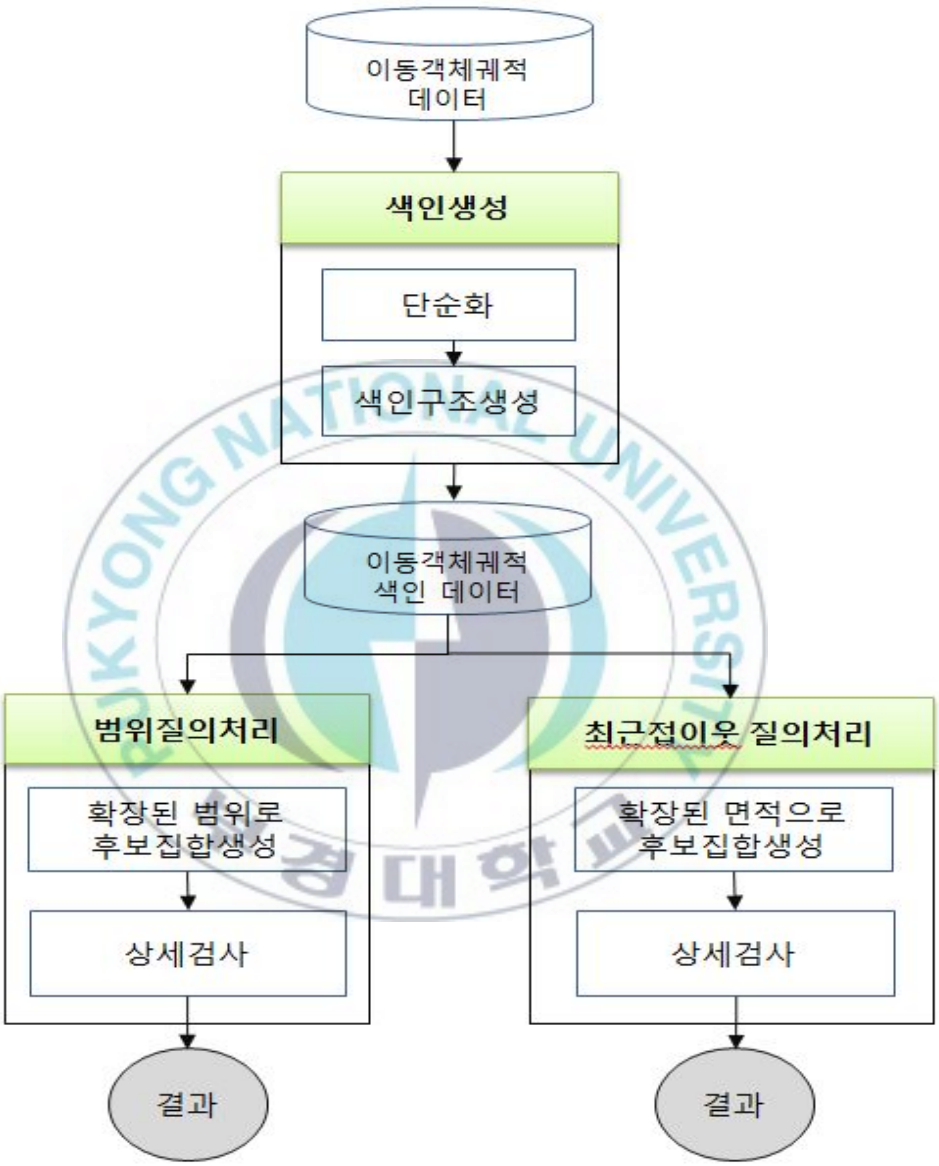

그림 6. 단순화된 흐름도

Fig. 6. Simplified flow diagram

3.1 색인구조 생성

이 절에서는 Douglas-Peucker 알고리즘을 응용하여 이동객체궤적 데이터를 단순화시켜서 색인구조 데이터를 만드는 알고리즘을 설계한다. 먼저 색인구조생성 알고리즘을 단순하게 도식화하면 그림 7과 같다.

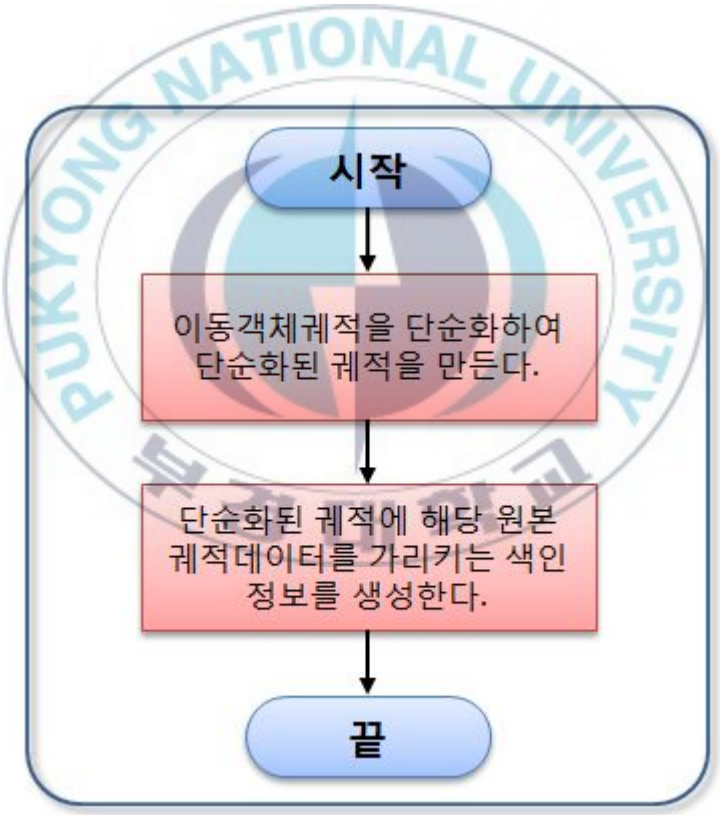

그림 7. 이동객체궤적에 대한 색인구조 생성 도식화

Fig. 7. Diagram of Index structure creation algorithm on moving object trajectories

색인구조생성 알고리즘은 Douglas-Peucker 알고리즘 방식으로 이동객체궤적 데이터를 단순화하고 단순화된 궤적이 원래의 궤적을 참조하도록 색인구조를 생성한다. 이동객체궤적에 대한 색인구조 생성 알고리즘은 그림 8과 같다.

알고리즘 1. 이동객체궤적에 대한 색인구조 생성

입력 : 이동객체궤적데이터(**srcList**), 단순화범위(**Epsilon**), 인덱스(**s**)
출력 : 색인구조데이터(**dugList**)

IndexCreation(srcList, Epsilon, s)

```
1.  FileWrite(궤적데이터파일이름+".dug");
2.  firstPoint = srcList.getFirst();
3.  endPoint = srcList.getLast();
4.  length = srcList.length();
5.  dmax = 0;
6.  index = -1;
    //가장 긴 수선의 길이(dmax)와 위치를 구한다.
7.  for(int i=0; i<length; i++)
8.     distance = vertical(srcList.get(i), firstPoint, endPoint);
9.     if(distance > dmax)
10.            dmax = distance;      index = i;
    //수선의 길이(dmax)가 단순화범위(Epsilon)보다 크면 Split하여 재귀함수 호출을 한다.
11. if(dmax > Epsilon)
12.    lList = IndexCreation(src, devideList(srcList,0,index), Epsilon, s);
13.    rList = IndexCreation(src, devideList(srcList,index, length), Epsilon, s+index);
14.    return (mergeList(lList, rList));
15. else
       //수선의 길이(dmax)가 단순화범위(Epsilon)보다 짧으면 한 궤적으로 병합하고
        색인정보를 추가한다.
16.    FileWrite.write(startPoint);
17.    FileWrite.write(s);
18.    FileWrite.write(length);
19.    return (mergeList(stratPoint, endPoint));
```

그림 8. 이동객체궤적에 대한 색인구조 생성 알고리즘

Fig. 8. Index structure creation algorithm on moving object trajectories

데이터를 어느 정도로 단순화 할 것인지를 나타내는 단순화범위(Epsilon)가 입력되어야 한다. 단순화 과정은 먼저 단계2에서 단계3까지 이동객체궤적 데이터에서 첫 번째 데이터(firstPoint)와 마지막 데이터(endPoint)를 읽고, 단계5에서 단계10까지 이동객체궤적의 모든 위치데이터들을 차례대로 읽으면서 첫 번째와 마지막 위치데이터로 이루어진 직선으로 수선을 그어서 그 거리를 계산하고, 수선들 중에서 길이가 가장 긴 최대수선의 길이(dmax)와 그 값을 가지는 위치데이터의 인덱스(index)를 구한다.

마지막으로 단계11에서 단계19까지 구해진 최대수선의 길이가 단순화범위(Epsilon) 안에 포함이 되면 단순화작업을 수행하여 색인구조결과를 출력하고, 최대수선의 길이가 단순화범위를 벗어나면 단순화 작업을 하지 않고 구해진 색인을 기준으로 이동객체궤적의 위치데이터를 양쪽으로 나누어서 각각 알고리즘을 재귀호출로 반복적으로 수행한다. 더 이상 재귀호출이 발생하지 않고 모든 점들에 대한 단순화작업이 끝나면 알고리즘이 종료된다. 그림 9는 제안된 색인구조 생성 알고리즘의 결과를 간단한 구조로 나타낸 것이다.

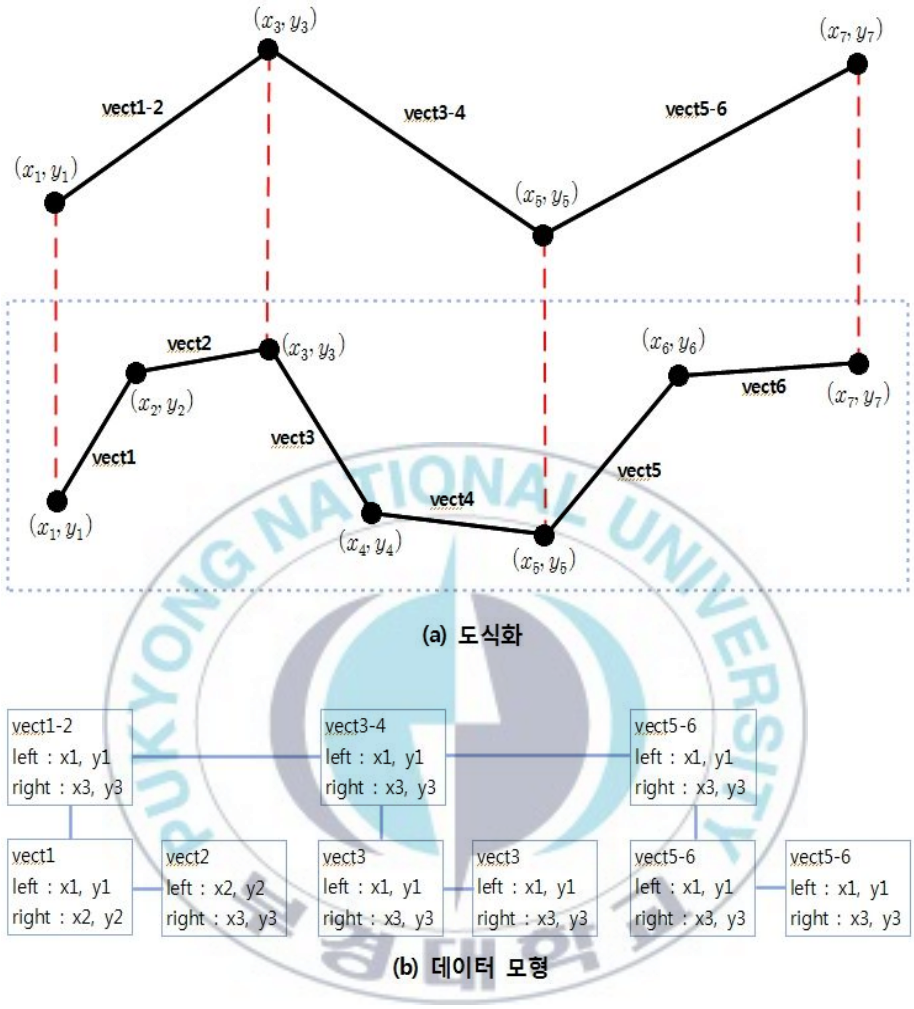

그림 9. 단순화된 색인구조

Fig. 9. Simplified index structure

여러 개(m)의 이동객체궤적 데이터에 대해서 위 알고리즘의 시간 복잡도는 m개의 이동객체궤적의 개수만큼 Douglas-Peucker 알고리즘을 수행한 시간으로 나타낼 수 있다. 하나의 이동객체궤적이 n개의 위치데이터를 가질 때 Douglas-Peucker 알고리즘의 시간복잡도는 $O(n*\log n)$이 되고, m개의 이동객체궤적들을 계산해야 하므로 알고리즘1의 시간복잡도는 $O(m*n*\log n)$이 된다.

3.2 범위 질의처리

이 절에서는 생성된 색인구조를 이용하여 범의 질의처리 알고리즘을 제시한다. 범위 질의가 주어지면 그 결과 질의 영역에 부합하는 객체들이 후보 집합으로 결정되는데 그 후보 집합 중 최종적으로 질의 결과가 되는 객체를 구하기 위해서는 실제 데이터를 디스크로부터 읽어서 질의 영역과 실제로 부합하는 지를 검사해야만 한다. 그 과정에서 착오배제를 없애기 위하여 질의 범위를 확장할 필요가 있다.

먼저 범위질의처리 알고리즘을 단순하게 도식화하면 그림 10과 같다.

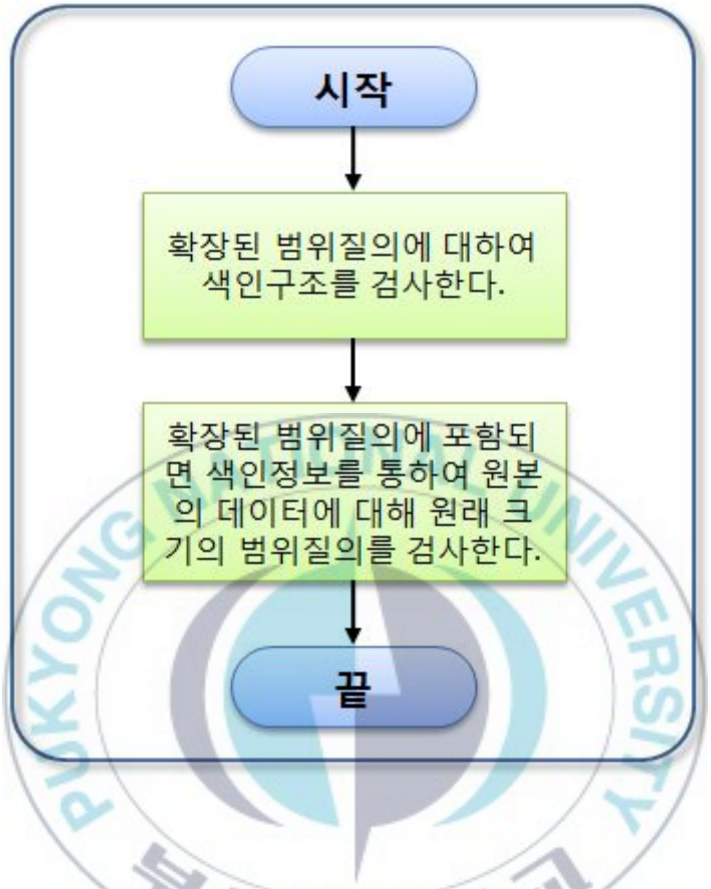

그림 10. 색인구조를 이용한 범위 질의처리 도식화

Fig. 10. Diagram of range query processing using index structure

본 논문에서 제안하는 알고리즘으로 색인구조를 생성하고 이에 대한 범위질의를 처리하기 위해서는 크게 두 단계의 작업이 필요하다. 먼저 색인구조에서는 착오배제를 해결하기 위해서 범위질의의 크기를 확장하여 검사를 하고 확장된 범위질의에 포함이 되면 색인정보를 통해 원본데이터에 접근하여 원래 크기의 범위질의검사를 수행한다.

알고리즘 2. 색인구조를 이용한 범위 질의처리

입력 : 색인데이터(**dug**), 질의범위(**qx, qy, width, height**)

출력 : 질의범위에 포함된 이동객체궤적들(**result**)

DouglasQuery(dug, qx, qy, width, height)

```
1.  FileRead(dug);
    //단순화범위(Epsilon)을 읽어 질의범위의 크기를 확장한다.
2.  Epsilon = FileRead.readInt();
3.  eqx = qx-Epsilon;   eqy = qy-Epsilon;
4.  ewidth = width+(2*Epsilon);   eheight = height+(2*Epsilon);
5.  point1 = FileReader.readPoint();
6.  while(FileReader.available() > 0)
7.     startIndex = FileRead.readInt();
8.     conutPoint = FileRead.readInt();
       //확장된 범위질의에 포함되는 지 검사
9.     point2 = FileRead.readPoint();
10.    if(queryIntersect(eqx, eqy, ewidth, eheight, point1, point2))
11.       if(queryInSrcFile(dug, qx, qy, width, height, firstIndex, countPoint))
          //질의에 포함되면 색인을 통하여 원본파일에 접근하여 질의검사
12.          result = dug;
13.          break;
14.    point1 = point2;
15. return result;
```

그림 11. 색인구조를 이용한 범위 질의처리 알고리즘

Fig. 11. Range query processing algorithm using index structure

그림 11은 범위질의 처리 알고리즘으로, 생성된 색인구조 데이터에 대하여 범위질의를 수행하면 먼저 색인구조 데이터는 단순화된 궤적 데이터이기 때문에 검사하기 전에 단계2에서 단계4까지 착오배제를 해결하기 위해서 범위질의의 크기를 색인구조 데이터의 단순화범위(Epsilon)만큼 확장한다. 이를 그림 12에서 나타내었다.

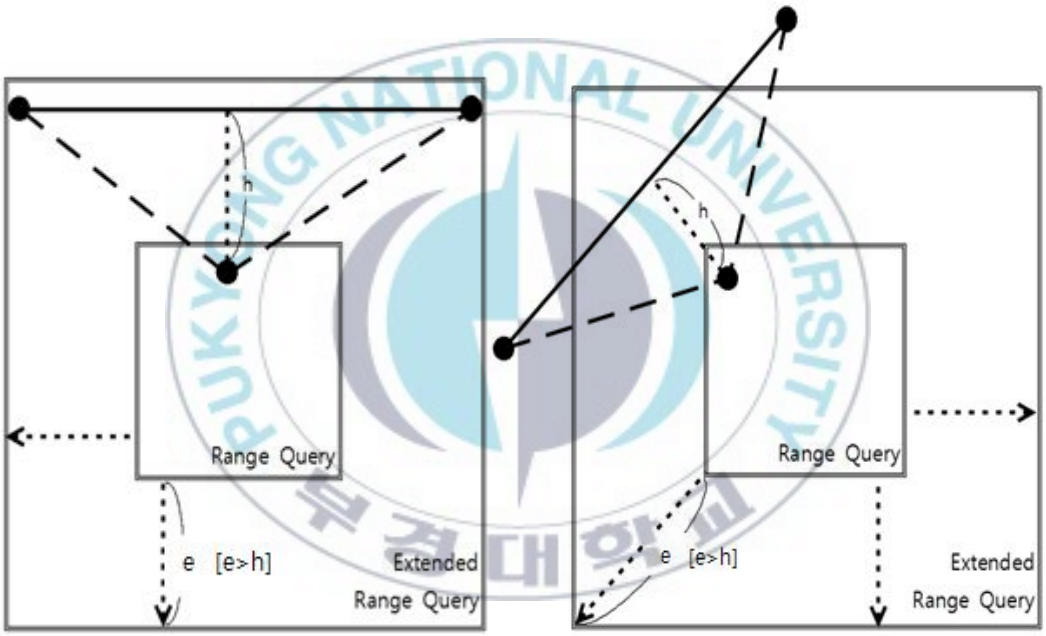

그림 12. 착오배제를 해결하기 위한 범위질의 크기 확장

Fig. 12. Extending size for range query to solve false drop

그 다음 단계5에서 10까지 색인구조 데이터로부터 차례대로 데이터를 읽으면서 확장된 범위질의에 속하는지 검사한다. 만약 확장된 범위질의에 포함된다면, 해당 색인구조 데이터의 이동객체궤적은 후보 이동객체궤적이 되고, 단계11에서 13까지 이 후보 이동객체궤적에 대하여 그림 13과 같이 원래 크기의 범위질의에 대한 상세검사를 수행하여 최종 결과를 출력한다.

알고리즘2의 시간복잡도는 크게 두 단계로 n개의 위치데이터를 가지는 m개의 이동객체궤적을 압축률 C로 색인구조 데이터를 만들어 확장된 범위질의를 처리하고, k개의 결과 후보이동객체궤적에 대하여 원래 크기의 범위질의를 계산해야 하므로 $O(n*(C*m+k))$이 된다.

알고리즘 3. 후보 이동객체궤적에 대한 상세 범위 질의처리

입력 : 후보 이동객체 궤적데이터(**candidate**),
질의범위(**qx, qy, width, height**),
색인정보(**startIndex, count**)

출력 : 상세검사 결과(**true 혹은 false**)

queryInSrcFile(candidate, qx, qy, width, height, startIndex, count)

```
1.  FileRead(candidate.getName()+".src");
    //색인정보를 통해 원하는 데이터 위치에 접근
2.  FileRead.skip(startIndex);
3.  point1 = FileRead.readPoint();
    //범위질의에 포함되는지를 차례대로 검사
4.  for(int i=0; i<count-1; i++)
5.     point2 = FileRead.readPoint();
6.     if(queryIntersect(qx, qy, ewidth, eheight, point1, point2))
7.           return true;          //질의에 포함되면 TRUE
8.     point1 = point2;
9.  return false;                  //질의에 포함되지 않으면 FALSE
```

그림 13. 후보 이동객체궤적에 대한 상세 범위 질의처리 알고리즘

Fig. 13. Refinement range query processing algorithm on candidate moving object trajectories

원본 파일에 대한 상세검사 과정은 후보 이동객체궤적 데이터에서의 입력된 색인정보가 가리키는 위치와 범위의 데이터들을 차례대로 범위질의에 대한 검사를 수행하고 만약 데이터가 범위질의에 포함되면 TRUE를 결과로 출력하고, 검사받는 모든 데이터가 범위질의에 포함되지 않으면 FALSE를 결과로 출력한다.

3.3 최근접이웃 질의처리

이 절에서는 생성된 색인구조를 이용하는 최근접이웃 질의처리 알고리즘을 제시한다. 먼저 최근접이웃 질의처리 알고리즘을 단순하게 도식화하면 그림 14와 같다.

그림 14를 살펴보면 본 논문에서 제안하는 색인구조 생성 알고리즘으로 색인구조 파일들을 생성하고 이에 대하여 최근접이웃 질의를 처리하기 위해서는 크게 두 가지 단계가 필요하다. 먼저 색인구조 파일은 원본궤적을 단순화 하여 만들어진 이동객체궤적 데이터를 가지고 있으므로 질의에 대한 결과가 원본 데이터에 대한 질의 결과와 다를 수가 있다. 때문에 색인구조 파일들에 대하여 먼저 확장된 질의를 수행하여 질의에 부합할 가능성이 있는 이동객체궤적들을 선택(후보자 선택)하고, 이 후보자들에 대해 원본파일을 찾아가서 한 번 더 질의를 수행함으로써 부합하는 결과를 찾을 수 있다.

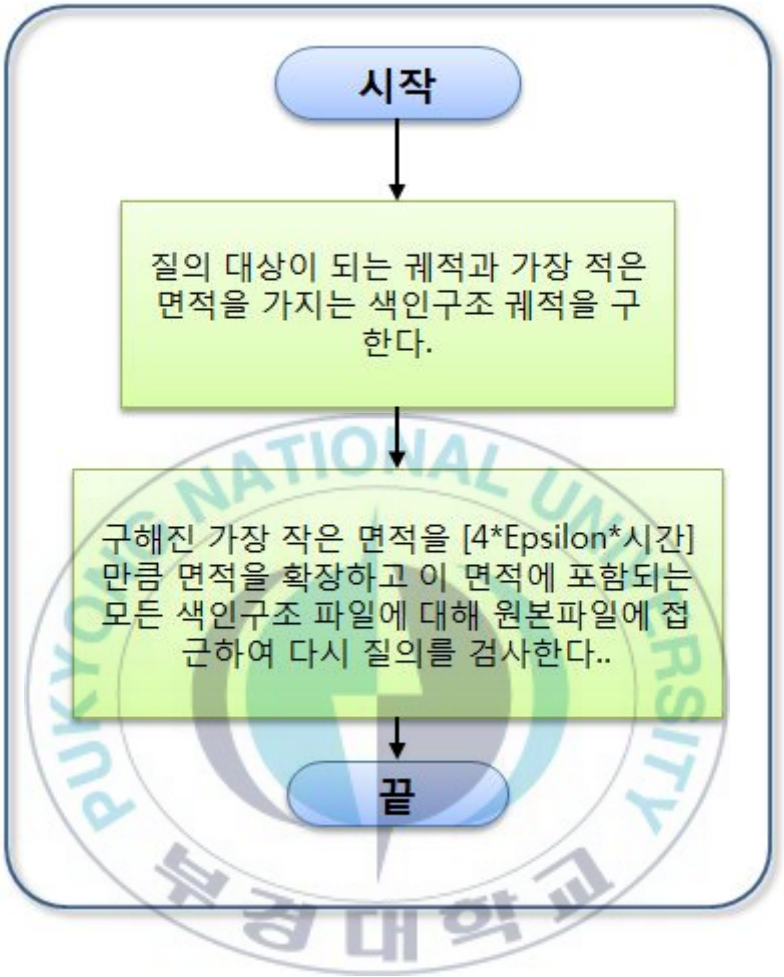

그림 14. 색인구조를 이용한 최근접이웃 질의처리 도식화

Fig. 14. Diagram of nearest neighbor query processing using index structure

그림 15는 본 논문에서 제안한 색인구조 파일들에 대하여 최근접이웃 검색질의를 처리하는 알고리즘이다. 알고리즘은 크게 두 단계로 먼저 색인 구조 파일들에 대하여 질의에 대하여 검사한 다음 그 중에 질의의 결과가 될 가능성이 있는 궤적들을 색출하고 이에 대하여 원본파일에 접근하여 검사함으로써 결과를 도출한다.

알고리즘 4. 색인구조를 이용한 최근접이웃 질의처리
입력 : 색인데이터파일 리스트(**file**), 질의궤적(**base**)
출력 : 최근접질의에 포함된 이동객체궤적들(**result**)

IndexNearestNeighborQuery(file, base)

```
1.  FileRead(base);
    //Douglas-Peucker 단순화 범위
2.  Epsilon = FileRead.readInt();
3.  PointList baseList = FileReader.readPointList();
4.  MinArea = Double.MAXIMUM;  DoubleList tempList;
    //파일들중 가장 작은 면적을 가지는 파일을 구한다
5.  for(int i=0; i<file.size(); i++)
6.     FileRead(file.get(i));
7.     PointList searchList = FileReader.readPointList();
8.     bindex = 0;  sindex = 0;  areaSum = 0;
9.     while(bindex < baseList.size && sindex < searchList.size)
           //순서대로 위치데이터를 읽어 면적을 측정
10.        basePoint1 = baseList.get(bindex);
11.        basePoint2 = baseList.get(bindex+1);
12.        searchPoint1 = searchList.get(sindex);
13.        searchPoint2 = searchList.get(sindex+1);
14.        if(basePoint2.getX() < searchPoint2.getX())
15.           areaSum += getArea(basePoint1, searchPoint1, basePoint2);
16.           bindex++;
17.        else  areaSum += getArea(basePoit1, searchPoint1, searchPoint2);
18.           sindex++;
19.    tempList.add(areaSum)
20.    if(areaSum < MinArea)
21.        MinArea = areaSum;
    //MinaArea확장
22. ExtendMinArea = MinArea + (Epsilon * baseList.getLastX() * 4);
    //확장된 MinArea에 포함되는 후보자 선택
23. for(int i=0; i<tempList.size(); i++)
24.    if(tempList.get(i) < ExtendMinArea)
25.       tempResult.add(file.get(i));
    //후보자들에 대하여 원본파일에 대해 검사
26. result = NearestNeighborQuery(tempResult, base.getSourceFile());
27. return result;
```

그림 15. 색인구조를 이용한 최근접이웃 질의처리 알고리즘

Fig. 15. Nearest neighbor query processing algorithm using index structure

단계5에서 단계22까지 색인구조 파일 중에서 질의대상이 되는 궤적과 가장 가까운 위치에 있는 궤적을 찾는다. 좀 더 자세하게 살펴보면 단계9에서 단계19까지 질의대상이 되는 궤적(baseList)과 검사 대상이 되는 궤적(searchList)을 시간 순으로 각각 하나씩 위치데이터를 읽어가면서 각 위치데이터들이 이루는 면적을 구하고 이를 다 더함으로써 궤적간의 거리를 구한다. 단계20에서 단계22까지 이렇게 구한 면적들 중에서 가장 작은 면적(MinArea)을 가지는 궤적이 질의대상이 되는 궤적과 가장 가까이 있는 궤적이라고 가정한다. 그래서 질의대상인 이동객체궤적(T2)에 최근접이웃 이동객체궤적(T3)은 그림 17과 같다.

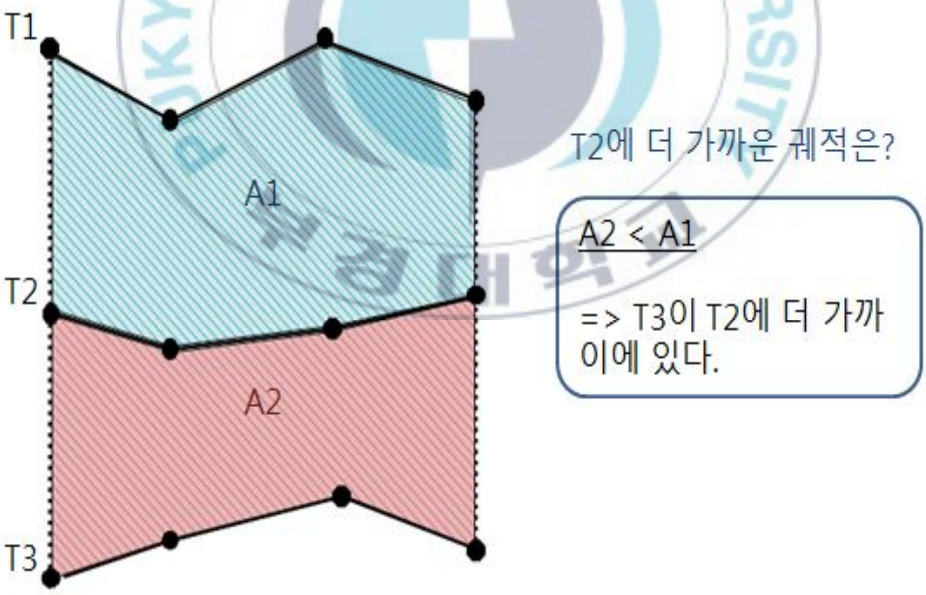

그림 16. 질의대상인 이동객체궤적(T2)에 최근접이웃 이동객체궤적(T3)

Fig. 16. Nearest neighbor moving object trajectory(T3) on query object(T2)

질의의 결과가 될 가능성이 있는 궤적을 찾기 위해서 단계23에서 단계27까지 구해진 가장 작은 면적을 단순화 범위(Epsilon:e)로 인하여 발생할 수 있는 오차 범위를 계산하여 확장된 면적(ExtendMinArea)을 구한다. 이것은 착오배제를 방지하기 위함이다. 확장 범위는 최악의 경우를 고려하면 그림 17과 같다. 실제 궤적B에 가까운 궤적은 T2이나 단순화된 궤적에서는 T1이 더 가깝다. 그러므로 그림 18과 같이 확장해야하는 질의범위는 최소면적 A1에 4*e*t를 더한 면적이어야 한다.

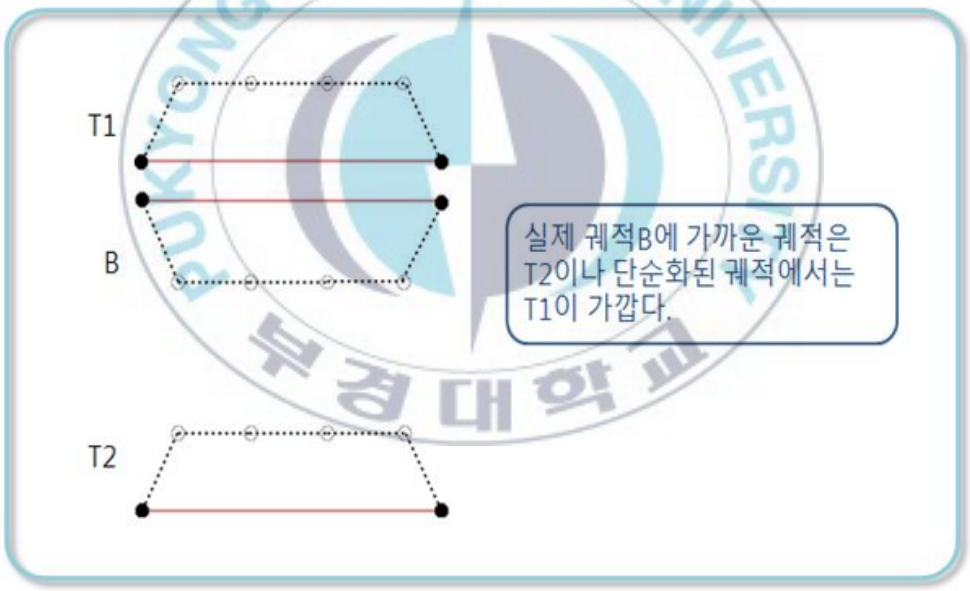

그림 17. 최근접이웃 질의처리에서 착오배제

Fig. 17. False drop in nearest neighbor query processing

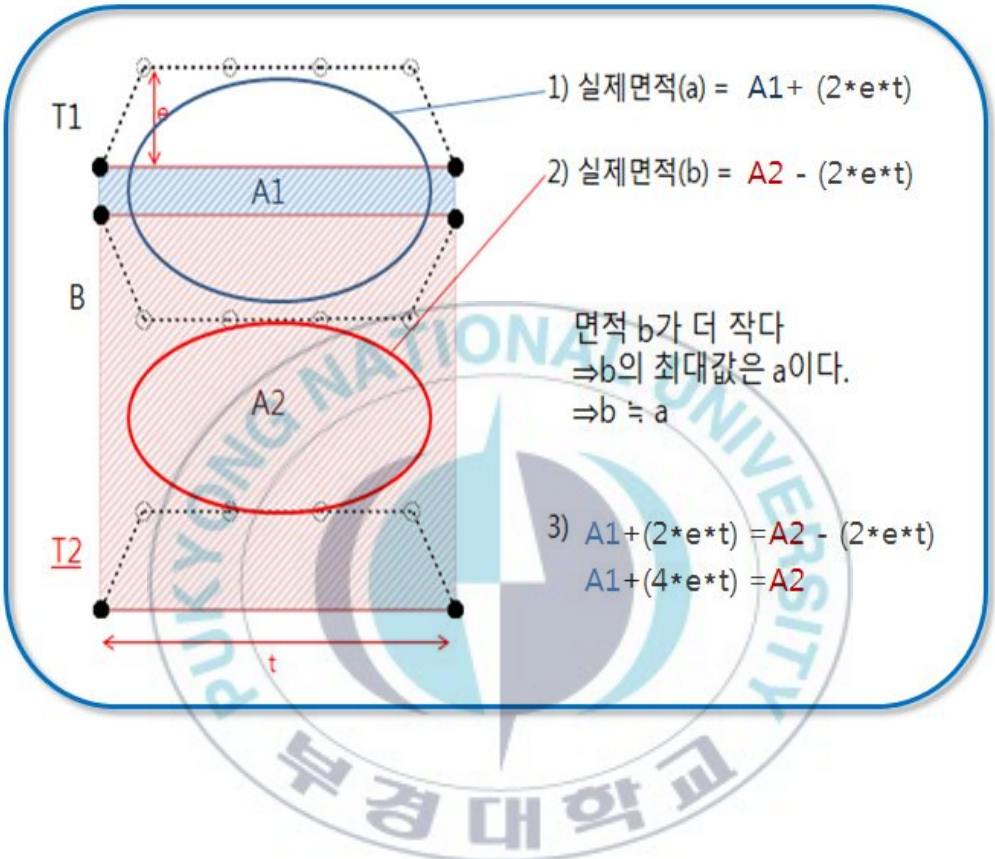

그림 18. 착오배제를 해결하기 위한 질의 영역 확장

Fig. 18. Extending query area to solve false drop

이 범위에 포함되는 모든 색인구조 파일에 대하여 원본파일에 접근하여 최근접이웃 검색 질의를 검사하여 그 결과(result)를 출력하고 알고리즘을 종료한다. 그림 19는 후보 이동객체궤적에 대한 상세 최근접이웃 질의처리 도식화이다.

그림 19. 후보 이동객체궤적에 대한 상세 최근접이웃 질의처리 도식화

Fig. 19. Diagram of refinement nearest neighbor query processing on candidate moving object trajectories

이 알고리즘의 시간복잡도는 크게 두 단계로, 먼저 n개의 위치데이터를 가지는 m개의 원본의 이동객체궤적을 압축률 C로 단순화하여 만든 색인 구조 파일에 대하여 최근접이웃 검색 질의를 처리하는데 $O(m*(n*C))$만큼의 시간이 걸리고, 오차범위에 포함되는 k개의 이동객체궤적에 대하여 원본파일에 대하여 다시 검사해야함으로 알고리즘의 총 시간복잡도는 $O(n*(C*m+k))$가 된다.

알고리즘 5. 후보 이동객체궤적에 대한 상세 최근접이웃 질의처리

입력 : 원본데이터파일 리스트(**file**), 질의궤적(**base**)
출력 : 최근접질의에 포함된 이동객체궤적들(**result**)

NearestNeighborQuery(file, base)

```
1.  FileRead(base);
2.  PointList baseList = FileReader.readPointList();
    //파일들중 가장 작은 면적을 가지는 파일을 구한다
3.  for(int i=0; i<file.size(); i++)
4.     FileRead(file.get(i));
5.     PointList searchList = FileReader.readPointList();
6.     bindex = 0;  sindex = 0;  areaSum = 0;
       while(bindex < baseList.size && sindex < searchList.size)
7.     //순서대로 위치데이터를 읽어 면적을 측정
8.          basePoint1 = baseList.get(bindex);
9.          basePoint2 = baseList.get(bindex+1);
10.         searchPoint1 = searchList.get(sindex);
11.         searchPoint2 = searchList.get(sindex+1);
12.         if(basePoint2.getX() < searchPoint2.getX())
13.            areaSum += getArea(basePoint1, searchPoint1, basePoint2);
14.            bindex++;
15.         else  areaSum += getArea(basePoit1, searchPoint1, searchPoint2);
16.               sindex++;
17.    if(areaSum < MinArea)
         //가장 작은 면적을 가지는 파일을 구한다
18.       MinArea = areaSum;
19.       result = file.get(i);
20. return result;
```

그림 20. 후보 이동객체궤적에 대한 상세 최근접이웃 질의처리 알고리즘

Fig. 20. Refinement nearest neighbor processing algorithm on candidate moving object trajectories

그림 20은 색인구조를 생성하지 않고 원본파일에서 직접 최근접이웃 질의를 처리하는 알고리즘이다. 원본파일에서 최근접이웃 질의를 처리하는 경우에도 색인구조 파일에 대한 질의처리 알고리즘과 동일하게 가장 작은 면적을 가지는 궤적을 구하여 결과로 출력한다. 원본파일에 대한 검사이므로 단순화로 인한 오류가 발생하지 않아 따로 추가적인 검사가 필요 없다.

4. 실험 및 비교분석

이 장에서는 먼저 범위 질의처리 알고리즘에 대한 실험 및 비교분석을 한 후에 최근접이웃 질의처리 알고리즘에 대한 실험 및 비교분석을 수행한다.

4.1 범위 질의처리 알고리즘에 대한 실험 및 비교분석

본 논문에서 제안하는 알고리즘(DGS)의 성능을 분석하기 위해서 기존의 이동객체궤적 색인방법인 MBR방법과 비교 실험을 해보았다. MBR방법은 실제 위치 데이터에 대한 근사치 영역을 이용한 방법이다. 본 논문의 방법도 실제 이동객체궤적의 데이터를 더 단순한 궤적데이터로 만들어서 색인구조를 생성하므로 실제 위치 데이터에 대한 근사치 영역을 사용하며 기존의 MBR방법과 아주 유사한 방법으로 색인구조를 생성한다.

이 비교 실험은 Intel(R) Core i5-2400 CPU 3.10GHz 프로세서와 메모리 3.24Gbyte, Windows 운영체제를 사용하는 시스템 상에서 수행되었으며, 알고리즘 구현을 위해서 Java 언어를 사용하였다. 원래 궤적의 위치 정보는 2차원 좌표로 표현되고, 시간을 고려하여 3차원으로 궤적을 나타낸다.

그러나 본 논문에서는 간단히 하기 위해 위치정보를 1차원 좌표로 표현하고 시간정보를 고려하여 궤적을 2차원 공간으로 나타낸다. 각 실험은 총 120개의 일정한 시간에 따른 위치데이터를 가지는 100개의 동일한 파일들을 MBR방법과 본 논문의 알고리즘 방법으로 색인화 하였고, 압축 비율은 1/10로 하여 색인구조 데이터에서 12개의 위치데이터를 가지도록 하였다.

비교 항목으로는 범위질의에 대하여 각 방법으로 질의를 처리할 때, 알고리즘의 정확도 그리고 질의를 처리하는데 걸리는 시간을 비교하였다. 여기서 정확도는 질의 검사를 하는데 있어 최종 결과 이동객체궤적과 후보이동객체궤적의 비율을 말한다. 즉, 결과를 도출하는데 있어 얼마만큼의 후보가 추출되었는지를 나타낸다. 예를 들면, 범위질의 검사에 대하여 후보 이동객체궤적이 100개가 추출되고, 결과로서 이동객체궤적이 80개가 질의에 포함되면 80/100으로 80%의 정확도를 가진다.

아래 그림 21과 그림 22는 범위질의의 크기에 따라서 두 방법의 수행속도와 정확도가 어떻게 달라지는지를 실험한 결과이다. 각 실험에서의 정확도 값과 수행시간 값은 한 번의 범위질의에 대하여 100개의 모든 파일을 검사할 때 걸리는 시간과 그에 따른 정확도이며, 500여 차례 이상의 반복실험을 통하여 평균을 내었다. 실험 결과 그래프에서 색인구조를 이용하여 질의를 처리한 결과 선은 DGS, 원본파일만으로 질의를 처리한 결과 선은 Source로 표기한다.

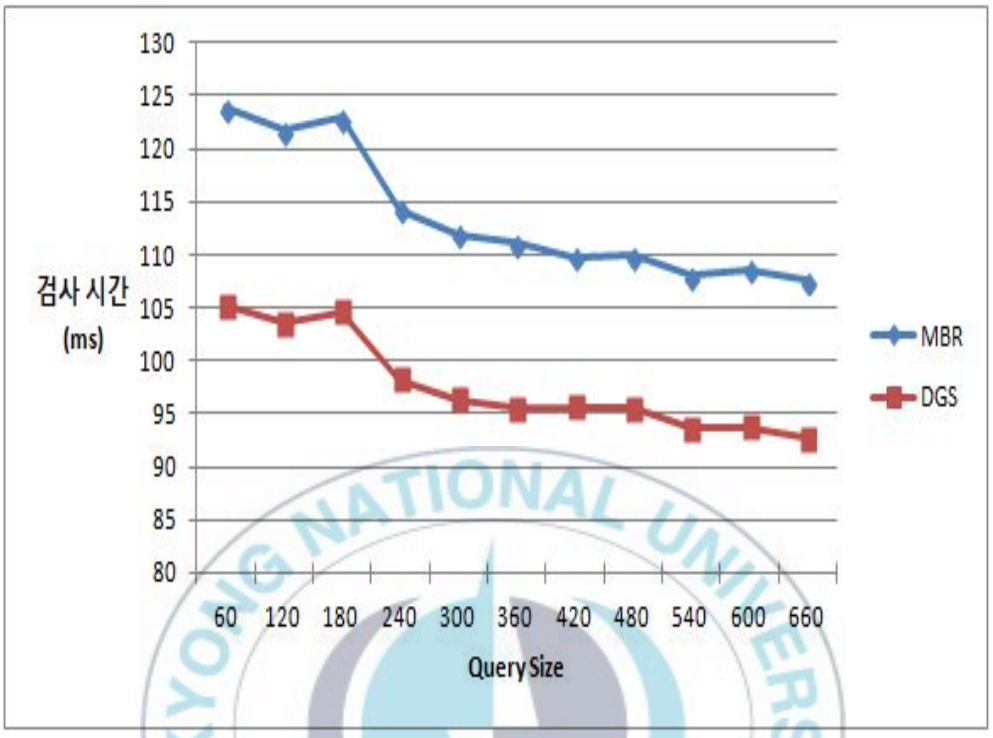

그림 21. 범위질의 크기에 따른 알고리즘 수행시간

Fig. 21. Algorithm execution times for various query sizes

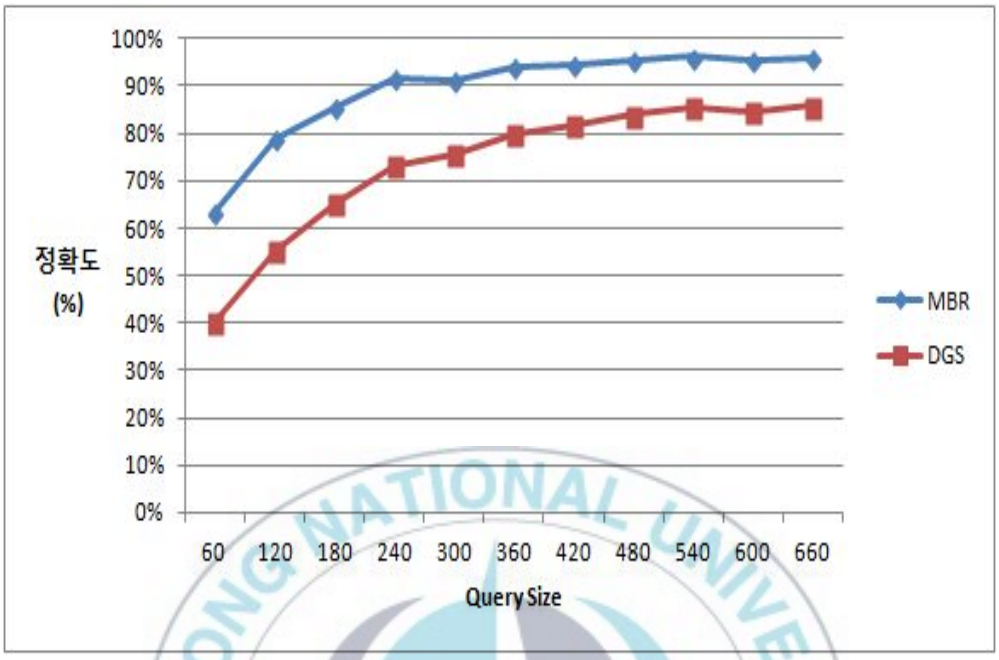

그림 22. 범위질의 크기에 따른 알고리즘 정확도

Fig. 22. Algorithm accuracy for various query sizes

정확도 부분에서는 MBR방법이 본 논문의 알고리즘 방법보다 대체적으로 높은 평균값이 나왔지만, 알고리즘 수행시간 그래프를 보면 MBR방법보다 제안한 알고리즘의 방법이 수행시간이 더 적게 걸린다는 것을 알 수 있다. 범위질의의 크기가 증가할수록 두 방법 모두 수행시간이 줄어들었으나 두 방법 간의 수행시간의 차이 변화는 없었고 모든 질의 크기에서 본 논문에서의 방법이 더 적은 시간 안에 범위질의를 처리하였다.

이는 본 논문의 방법이 더 많은 위치데이터를 검사하지만 전체 색인구조 데이터의 크기가 상대적으로 MBR방법보다 평균 1.8배 빠르다는 것을 알 수 있다.

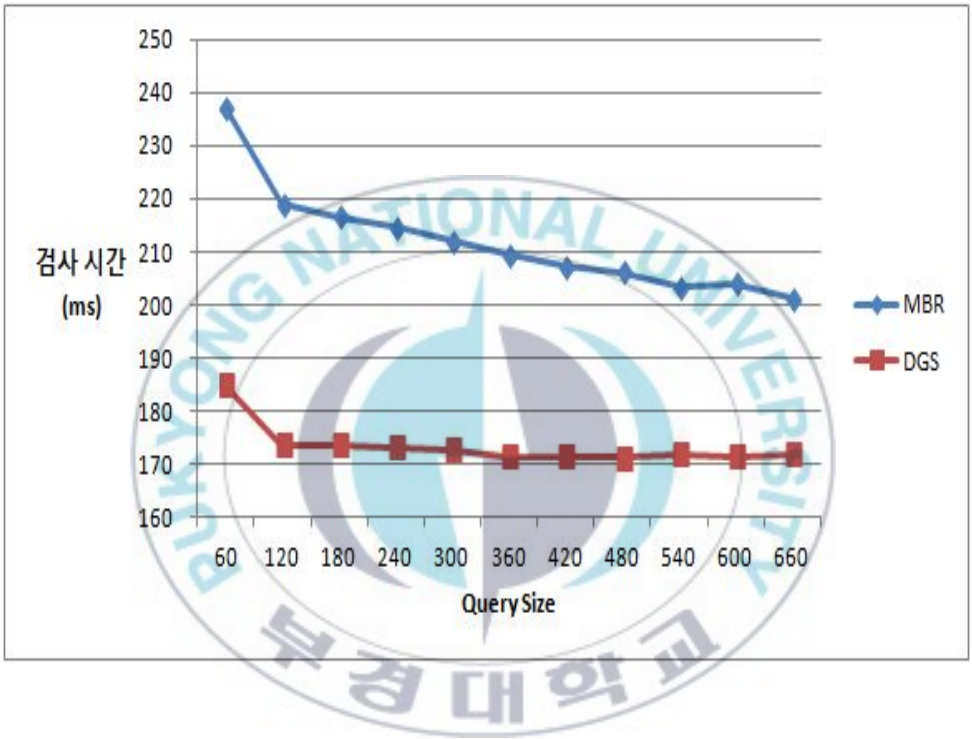

그림 23. 범위질의 크기에 따른 알고리즘 수행시간 (압축비율 : 1/5)

Fig. 23. Algorithm execution times for various range query sizes (compression ratio : 1/5)

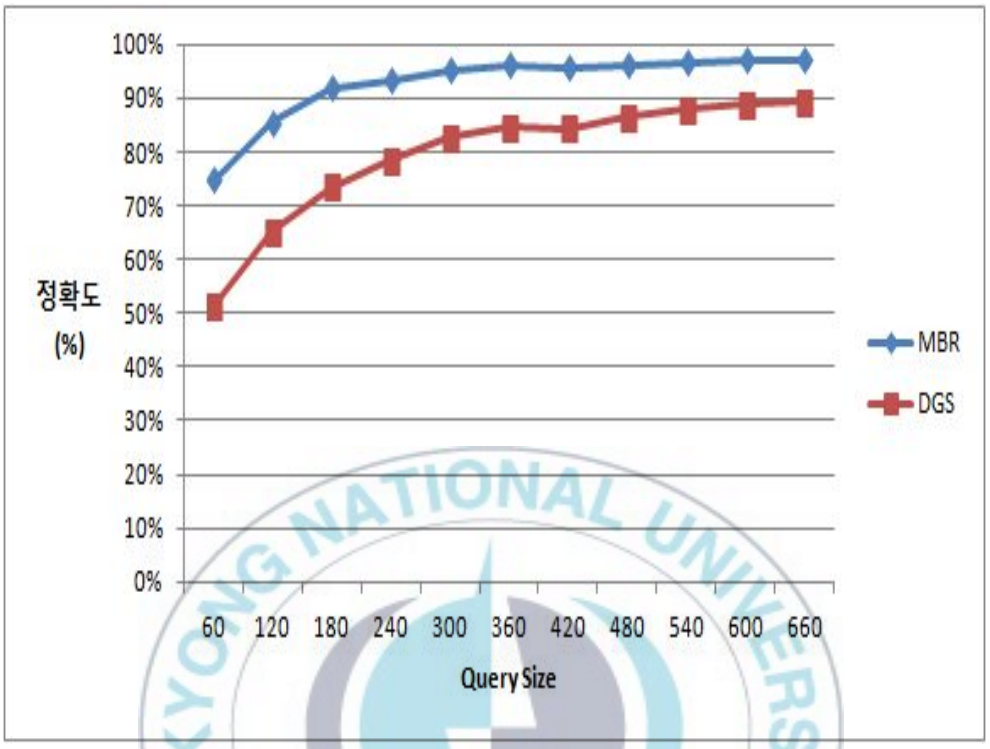

그림 24. 범위질의 크기에 따른 알고리즘 정확도 (압축비율 : 1/5)

Fig. 24. Algorithm accuracy for various range query sizes (compression ratio : 1/5)

그림 23과 그림 24의 그래프는 이전 실험과 동일한 데이터를 가지고 압출 비율을 조정하여 실험한 결과로서 마찬가지로 제안한 방법이 범위질의를 처리하는데 걸리는 시간은 기존의 방법보다 더 적게 걸리는 것을 볼 수 있다. 위의 실험들 이외에도 시간에 따른 위치 변화도를 조정해보거나 데이터의 압축 비율 등을 조정해보면서 여러 방면으로 실험을 해보았고 실험

결과로는 모두 비슷한 결과가 나왔다.

본 논문에서 제안한 알고리즘의 방법은 범위질의를 처리하는데 있어서 질의의 크기를 확장하기 때문에 알고리즘의 정확도는 기존의 MBR방법보다 부족하지만, 직선형 데이터를 사용하기 때문에 사각형 데이터를 사용하는 기존의 색인구조 방법보다 좀 더 가볍고 단순한 구조의 색인구조를 생성하고 데이터의 크기 또한 줄어들어서 범위질의를 처리하는데 있어 전체적인 처리 시간을 절약할 수 있다.

4.2 최근접이웃 질의처리 알고리즘에 대한 실험 및 비교분석

본 논문에서 제안하는 알고리즘이 얼마나 효율적으로 데이터를 처리하는지 확인하기 위하여 단순히 원본파일들만으로 최근접이웃검색 질의를 처리하는 것과 제안된 색인구조 파일을 통하여 최근접이웃검색 질의를 처리하는 것을 비교 실험을 해보았다.

실험은 Windows 7 Home Premium K 32bit 운영체제, Intel(R) Core(TM) i5-2400 CPU 3.10GHz 프로세서, 4.00GB 메모리 환경에서 수행하였고, 두 실험 모두 같은 환경에서 수행하였다. 실험의 복잡성을 줄이기 위하여 원래 시간(t)과 좌표(x,y) 데이터를 가지는 3차원의 시공간 데이터를 시간(t)과 좌표(x) 데이터만을 가지는 2차원 데이터로 축소하여 실험을 수행하였

고, 실험 데이터의 크기는 5,000개의 위치데이터를 가지는 200개의 이동객체궤적 데이터로 총 100만개의 위치데이터로 실험을 진행하였다.

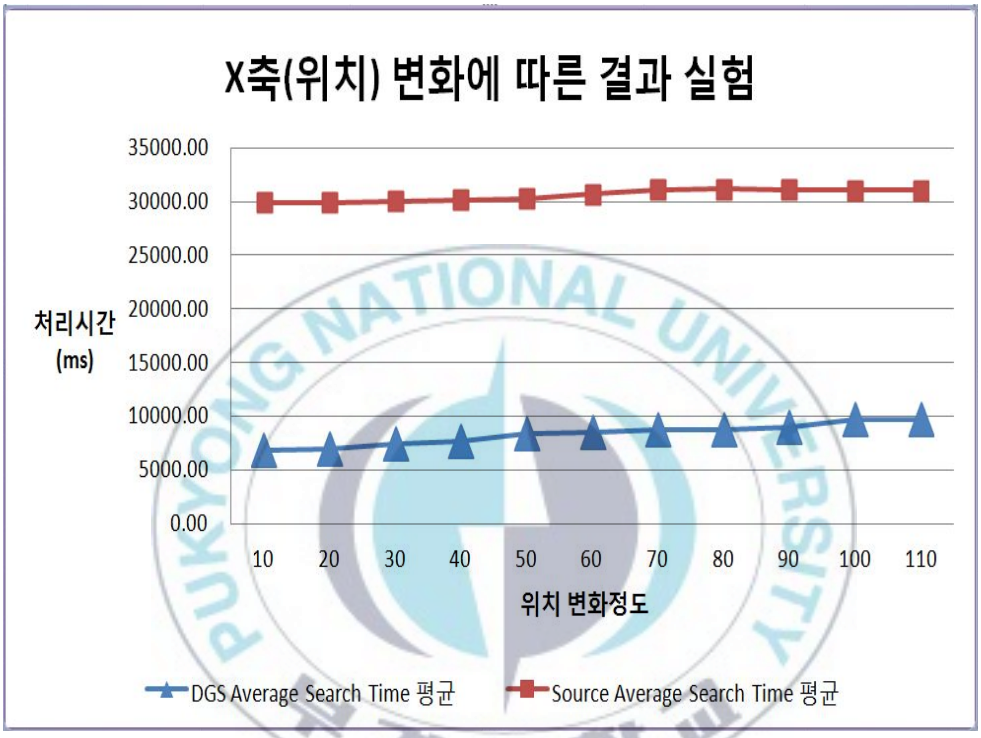

그림 25. X축(위치) 변화정도에 따른 실험 결과

Fig. 25. Algorithm execution times about the location change

그림25는 X축(위치)의 변화되는 정도에 따라 알고리즘을 처리하는 속도를 실험한 결과이다. 압축률은 1/10로 설정하였고 색인구조 파일에는 10만개의 위치 데이터와 추가적인 색인정보 데이터(1,620Kbyte)를 가지고 있다. 세로축은 최근접이웃 검색질의를 처리하는데 걸리는 시간을 나타내고, 가로축은 위치 값 변화정도의 크기를 나타낸다.

위 실험에서 위치 변화정도가 가장 작은 값(가장 좌측)은 10으로 원본데이터를 1/10로 압축하기 위해서 Epsilon의 값이 12가 된다. 위치 변화정도가 가장 큰 값(가장 우측)은 110으로 원본데이터를 1/10로 압축하기 위해서 Epsilon의 값은 84가 된다. 위치 변화정도가 10일 때 색인구조를 이용하여 질의를 처리하면 200개의 파일 중 평균적으로 5.2개의 후보자(208Kbyte)가 선택되어 상세검사를 수행하였고, 걸리는 시간은 평균적으로 6,800ms가 걸렸다. 위치 변화정도가 110일 때는 200개의 파일 중 평균적으로 22.4개의 후보자(896Kbyte)가 선택되어 상세검사를 수행하였고 걸리는 시간은 평균적으로 9,700ms가 걸렸다. 원본 파일만으로 질의를 처리하는 경우에는 평균적으로 30,000ms의 시간이 걸린다.

위치의 변화되는 정도가 커질수록 궤적의 굴곡이 심해지고 이에 따라 단순화범위인 Epsilon의 크기가 증가하게 된다. 즉, Epsilon의 크기가 커져 알고리즘2에서의 오차범위 해결을 위한 범위확장의 크기가 커지게 되어 검사해야하는 원본데이터의 양이 증가하게 된다. 그 결과, 위치 변화정도가 커질수록 색인구조를 통한 질의처리 시간이 점점 길어지는 상향곡선을 타는 것을 볼 수 있다. 하지만 위치변화의 정도가 커지더라도 그 영향이 적어 원본파일을 이용한 질의처리(Source)보다는 본 논문에서 제안한 색인구조를 이용하여 최근접이웃 질의처리 하는 것이 평균 3.5 배 빠르다는 것을 볼 수 있다.

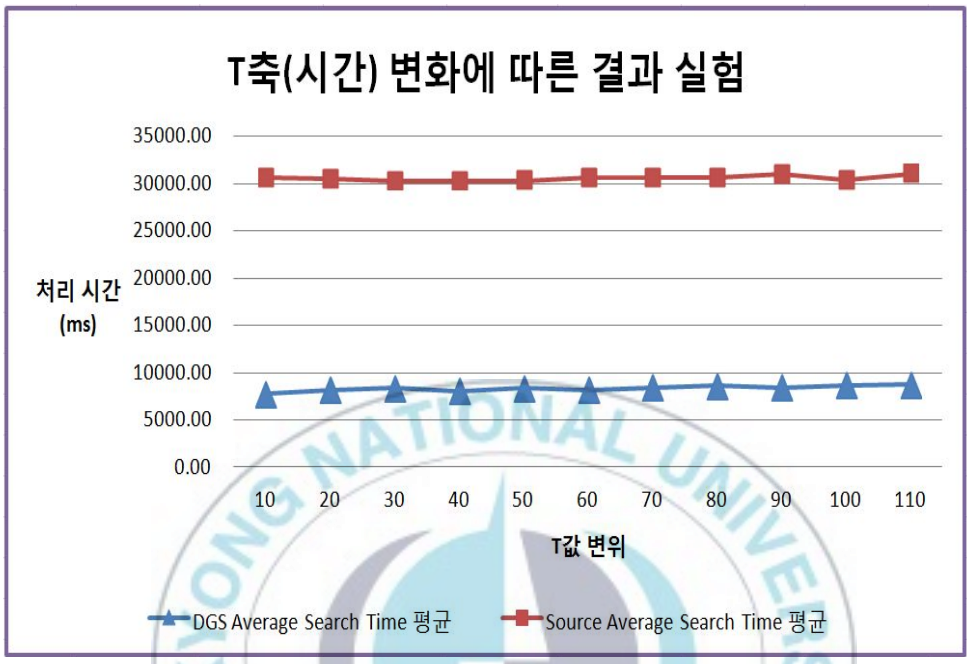

그림 26. T축(시간) 변화정도에 따른 실험 결과

Fig. 26. Algorithm execution times about time interval change

그림 26은 위치의 변화정도는 40으로 동일하게 하고, T축(시간)의 측정 간격에 따라 알고리즘을 처리하는 속도를 실험한 결과이다. 위의 실험과 마찬가지로 압축률은 1/10로 설정하였고 색인구조 파일에는 10만개의 위치 데이터와 추가적인 색인정보(1,620Kbyte)를 포함하고 있다. 세로축은 최근접이웃검색 질의를 처리하는데 걸리는 시간을 나타내고, 가로축은 시간 측정 간격의 변화정도 크기를 나타낸다.

위 실험에서 측정 시간간격이 가장 작은 값(가장 좌측)은 10으로 원본데이터를 1/10로 압축하기 위해서 Epsilon값은 45가 되고, 가장 큰 값(가장 우측)은 110으로 원본데이터를 1/10로 압축하기 위해서 Epsilon값은 47이 된다. 측정 시간간격이 10일 때 색인구조를 이용하여 질의를 처리하는데 걸리는 시간은 평균적으로 8,200ms이고, 110일 때는 8,700ms이다. 원본파일만으로 질의를 처리하는 경우에는 대략 30,000ms의 시간이 걸린다.

위치가 측정되는 시간 간격은 그 변화정도가 바뀌어도 위치데이터 간의 시간간격이 변할 뿐, 수선의 길이와 같은 Epsilon의 크기에 큰 영향을 주지 않았다. 따라서 시간의 변화정도에 따라서는 큰 변화가 없이 본 논문에서 제안하는 색인구조(DGS)가 원본파일을 통한 질의처리방법(Source) 보다 평균 3.5 배 정도 빠르게 질의를 처리하였다.

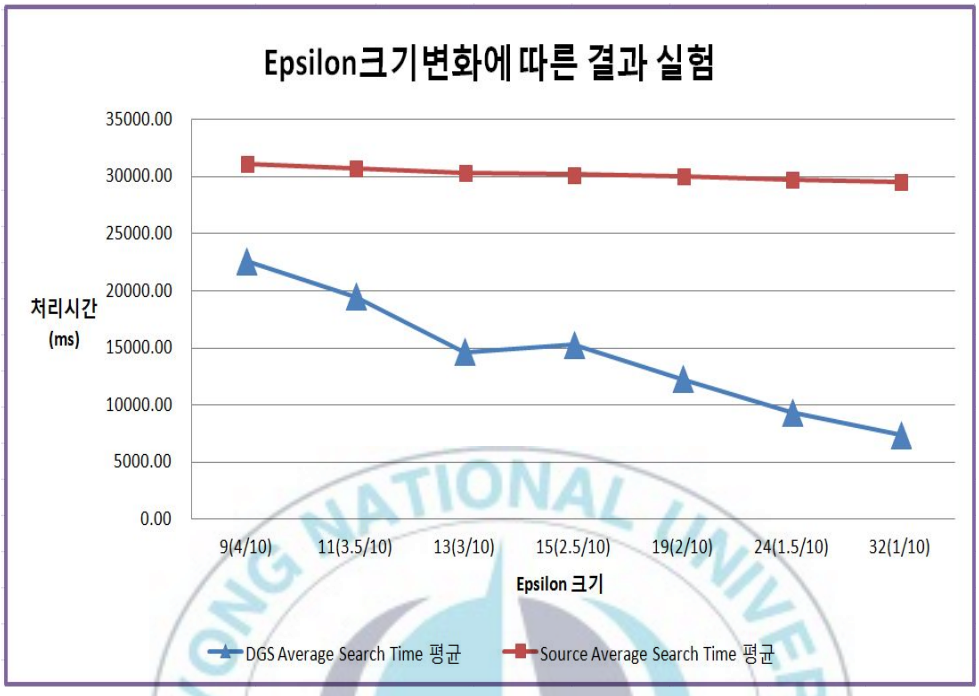

그림 27. Epsilon 크기 변화에 따른 실험 결과

Fig. 27. Algorithm execution times about Epsilon size

그림 27은 동일한 X축(위치)과 T축(시간)을 가지는 파일들에 단순화범위(Epsilon)의 크기 변화에 따라 처리시간이 어떻게 되는지를 실험한 결과이다. Epsilon의 크기가 커질수록 오차범위 해결을 위한 범위확장이 커지게 된다. 위 실험에서 가장 작은 Epsilon의 크기 값(가장 좌측)은 9로 압축률이 대략 4/10정도이며, 그 결과로 색인구조의 데이터 크기는 색인정보를 포함하여 대략 6,250Kbyte가 되고, 색인구조를 통하여 질의를 처리할 때 200개의 파일 중 평균적으로 2개의 후보자(80Kbyte)만 선택되어 원본데이

터를 상세 검사하였다. 가장 큰 Epsilon의 크기 값(가장 우측)은 32로 압축률이 대략 1/10정도이며, 그 결과로 색인구조의 데이터 크기는 색인정보를 포함하여 대략 1,620Kbyte가 된다. 색인구조를 통하여 질의를 처리할 때 200개의 파일 중 평균적으로 11개의 후보자(440Kbyte)가 선택되어 원본데이터를 상세 검사하였다.

위 실험을 통하여 Epsilon의 크기가 커질수록 오차범위 해결을 위한 범위확장이 커지게 되어 후보자 개수가 증가하고, 상세 검사하는 파일의 크기가 커지지만, Epsilon크기가 커질수록 압축률이 높아져서 색인구조 파일이 가지는 위치데이터와 색인정보의 크기가 줄어들어 전체적인 데이터 크기를 줄여 더 빠르게 질의를 처리하는 것을 확인할 수 있었다.

5. 결론

본 논문에서는 Douglas-Peucker 알고리즘을 이용하여 이동객체궤적을 단순화하고 이를 이용하여 색인구조를 생성하는 알고리즘을 제시하였고 이 생성된 색인구조에 대하여 범위질의를 처리하는 알고리즘을 고안하였다. 지금까지 널리 연구된 MBR 방식과 실험을 통하여 그 성능을 비교 및 분석하여 본 논문에서 제안된 방법이 더 단순하고 적은 양의 색인 데이터를 생성하며 범위질의 처리속도가 빠른 이점이 있음을 보였다. 실험을 통하여 본 논문에서 제안하는 방법이 MBR방식보다 평균 1.8배 빠르다는 것을 확인하였다. 또한 이 색인구조를 이용하여 이동객체궤적 데이터에 대한 최근접이웃 질의를 처리하는 알고리즘도 제안하였다. 제안된 최근접이웃 질의 처리 알고리즘이 얼마나 더 효율적으로 처리하는지 실험을 통하여 비교 및 분석하였다. 본 논문에서 제안한 단순화된 색인구조를 이용하여 최근접이웃 질의를 처리함으로써 질의처리 속도가 색인구조를 이용하지 않는 경우보다 평균 3.5 배 빠르다는 것을 확인하였다.

이동객체궤적 데이터를 단순화하여 색인구조를 만들고 이 색인구조에 대하여 범위 질의를 처리하기 때문에 착오배제 문제가 발생하였다. 이 착오배제를 방지하기 위해서는 범위 질의 시에는 범위를 확장함으로써 해결하였다. 최근접이웃 질의 시에도 단순화된 색인구조에 대하여 최근접이웃 질의를 처리하기 때문에 역시 여기서도 착오배제 문제가 발생하였다. 이 착오배제를 방지하기 위해서는 질의 면적의 확장을 통하여 해결하였다. 질

의 범위나 면적의 확장은 착오배제 문제를 해결하나 질의 결과를 상세조사해야 하는 후보자의 개수가 증가한다. 그러나 색인 데이터를 만들기 위한 단순화 범위의 크기, 위치의 변화정도, 시간의 변화정도에 따라 질의를 처리하는데 실험을 통하여 본 논문에서 제안하는 알고리즘이 얼마나 효율적인지를 확인하였다. 이동객체궤적에서 위치 또는 시간의 변화정도는 질의를 처리하는데 크게 영향을 끼치지 않았다. 단순화 범위의 크기가 커질수록 압축률이 더 좋아져서 질의를 더 빠르게 처리하는 것을 확인할 수 있었다.

본 논문에서는 이동객체궤적에 대하여 하나의 단순화된 색인구조를 생성하였으나 실제 궤적데이터에는 다단계 색인구조가 필요하다. 이에 다단계 색인구조에 대하여 범위 및 최근접이웃 질의처리 알고리즘을 확장할 필요가 있다. 또한 제안된 색인구조를 이용한 점 질의, 위상 질의, 항해 질의, 미래 질의 등의 알고리즘들을 연구할 필요가 있다.

참고 문헌

[1] J. F. Roddick, K. Hornsby and M. Spiliopoulou, "An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining Research," TSDM 2000, LNAI2007, pp. 147-163, 2001.

[2] D. Barbara, "Mobile computing and databases-a survey," IEEE TKDE, Vol.11 No.1, pp. 108-117, Jan. 1999.

[3] 한국데이터베이스진흥원, "이동객체 데이터베이스 기술 동향," 데이터베이스 백서, 2008.

[4] C. S. Jensen, D. Lin and B. C. Ooi, "Query and Update Efficient B+-Tree Based Indexing of Moving Objects," Proceedings of the 30th VLDB Conference, Toronto, Canada, pp. 768-779, 2004.

[5] M. Hadjieleftheriou, G. Kollios, D. Gunopulos and V. J. Tsotras, "Efficient Indexing of Spatio temporal Objects," 8th International Conference on Extending Database Technology Prague, Czech Republic. pp. 251-268, 2002.

[6] J. Ni and C. V. Ravishankar, "Indexing Spatio-Temporal Trajectories with Efficient Polynomial Approximations," IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 5, pp. 663-678, May. 2007.

[7] E. Braynova, "Indexing Spatio-Temporal Trajectories with

Orthogonal Polynomials," Conference on Data Mining - DMIN, 2006.

[8] D. Li, Y. Peng and J. Yin, "Quadtree and Hash Table Based Index Structure for Indexing the Past, Present and Future Positions of Moving Objects. Computer Science and its Applications," CSA '08. International Symposium on 2008.

[9] A. Guttman, "R-trees:A Dynamic Index Structure for Spatial Searching," ACM Inernational Conference on Management of Data, pp. 47-57, 1984.

[10] E. K. Frentzos , "Trajectory Data Management in Moving Object Databases," Jul. 2008.

[11] R. Zhang, H. V. Jagadish, B. T. Dai and K. Ramamohanarao, "Optimized Algorithms for Predictive Range and KNN Queries on Moving Objects," Information Systems, 35(8): 911-932, 2010.

[12] S. Sioutas, E. Magkos, I. Karydis and V. S. Verykios, "Uncertainty for Anonymity and 2-Dimensional Range Query Distortion," S. Sioutas et al. 2011.

[13] D. Yung, E. Lo and M. L. Yiu, "Authentication of Moving Range Queries," 2012.

[14] D. V. Kalashnikov, S. Prabhakar and S. E. Hambrusch, "Efficient Evaluation of Continuous Range Queries on Moving Objects," Appeared in Proceedings of DEXA Conference. Sep. 2-6 2002.

[15] M. McCarthy, Z. He and X. S. Wang, “Evaluation of Range Queries with Predicates on Moving Objects,” Jun. 2013.

[16] 박영희, 김규재, 조우현, “이동객체궤적에 대한 효율적인 범위 질의,” 한국정보통신학회논문지, Vol. 18, No. 2, pp. 364-370, Feb. 2014.

[17] 김규재, 박영희, 조우현, “이동객체궤적에 대한 효율적인 최근접 이웃 검색,” 한국정보통신학회 추계 학술대회, 2014.

[18] G. S. Iwerks, H. Samet and K. Smith, “Continuous K-nearest neighbor queries for continuously moving points with updates,” in Proceedings of VLDB. pp. 512-523, 2003.

[19] R. Benetis, C. Jensen, G. Karciauskas and S. Saltenis, “Nearest neighbor and reverse nearest neighbor queries for moving objects,” IDEAS, pp. 44-53, 2002.

[20] K. Mouratidis, M. Hadjieleftheriou and D. Papadias, “Conceptual partitioning:An efficient method for continuous nearest neighbor monitoring,” ACM SIGMOD, pp. 634-645, 2005.

[21] Y. Tao and D. Papadias, “Time parameterized queries in spatio-temporal databases,” ACM SIGMOD, pp. 334-345, 2002.

[22] X. Yu, K. Pu and N. Koudas, “Monitoring k-nearest neighbor queries over moving objects,” ICDE, pp. 631-642, 2005.

[23] X. Xiong, M. Mokbel and W. Aref, “SEA-CNN:Scalable processing of continuous K-nearest neighbor queries in spatio-temporal databases,” ICDE, pp. 643-654, Apr. 2005.

[24] D. Douglas and T. Peucker, “Algorithms for the reduction of the number of points required to represent a line or its character,” The American Cartographer, 10(42):112-123, 1973.

[25] N. Roussopoulos, S. Kelly and F. Vincent., “Nearest Neighbor Queries,” SIGMOD Conference, pp. 47-57, 1984.

[26] D. Papadias, Q. Shen, Y. Tao and K. Mouratidis, “Group nearest neighbor queries,” ICDE, pp. 301-312, Apr. 2004.

[27] H. Hu and D. L. Lee, “Range Nearest-Neighbor Query,” IEEE IKDE, pp. 78-91, Jan. 2006.

[28] K. L. Cheung and A. W. Fu, “Enhanced Nearest Neighbor Search on the R-Tree,” SIGMOD Record, vol. 27, no. 3, pp. 16-21, Sep. 1998.

[29] G. Hjaltason and H. Samet, “Distance Browsing in Spatial Databases,” TODS, pp. 265-318, Jun. 1999.

[30] Y. Tao, D. Papadias and J. Sun, “The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries,” VLDB, vol. 29, pp. 790-801, 2003.

[31] Y. Cai, K. A. Hua, G. Cao and T. Xu, “Real-Time Processing of Range-Monitoring Queries in Heterogeneous Mobile Databases,” IEEE Transacions on Mobile Computing, pp. 931-942. Jul. 2005.

[32] B. Gedik and L. Liu, “Mobieyes: Disributed Processing of Continuously Moving Queries on Moving Objects in a Mobile

System," Proceedings EDBT Conference, pp. 67-87, 2004.

[33] M. Mokbel, X. Xiong and W. Aref, "SINA:Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases," Proceedings of ACM SIGMOD Conference, pp. 623-634, 2004.

[34] X. Xiong., M. Mokbel and W. Aref, "SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases," Proceedings of ICDE Conference, pp. 643-654. 2005.

[35] N. Beckmann, H. Kriegel, R. Schneider and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," In Proc. ACM SIGMOD, pp. 322-331. 1990.

[36] D. V. Kalashnikov, S. Prabhakar and S. E. Hambrusch, "Main memory evaluation of monitoring queries over moving objects," Distrib. Parallel Databases, 15(2):117-135, 2004.

[37] S. Saltenis, C. S. Jensen, S. T. Leutenegger and M. A. Lopez, "Indexing the positions of continuously moving objects," In SIGMOD Conference, 2000.

[38] Y. Xiao, "Set Nearest Neighbor Query for Trajectory of Moving Objects," Sixth International Conference on Fuzzy Systems and Knowledge Discovery, pp. 211-214, 2009.

[39] P. Rigaux, M. Scholl and A. Voisard, "Spatial Database," 2002.

[40] T. Brinkhoff, H. P. Kriegel and B. Seeger, "Efficient Processing of

Spatial joins Using R-trees," Proc SIGMOD, pp. 237-246. 1993.

[41] 조문정, 김민수, 이기준, "MBR 특성을 이용한 R-tree계열의 공간 색인 방법 성능 비교," 한국정보과학회 가을 학술발표논문집, Vol. 22, No. 2, 1995.

[42] T. Brinkhoff, H. P. Kriegel and R. Schneider, "Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Spatial Database Systems," Proceedings Ninth Intl. Conf. on DATA ENGINEERING, pp. 40-49, Apr. 1993.

[43] H. P. Kriegel, H. Horn and M. Schiwietz, "The Performance of Object Decomposition Techniques for Spatial Query Processing, Data Structures and Efficient Algorithms," Lecture Notes in Computer Science 594, Springer-Verlag, pp. 104-123, 1992.

[44] Y. J. Lee, H. H. Park, N. H. Hong and C. W. Chung, "Controlled Decomposition Strategy for Complex Spatial Objects," 7th International Conference on Database and Expert System Applications, Lecture Notes in Computer Science 1134, Springer-Velag, pp. 321-329, 1994.

[45] E. G. Hoel and H. Samet, "A Qualitative Comparison Stydy of Data Structures for Large Line Segment Databases," Proc. Of the ACM SIGMOD, pp. 205-214, Jun. 1992.

[46] 이동만, 이용주, 정진완, "R-트리를 이용한 최근접 질의 처리에 관한 연구," 한국정보과학회 가을 학술발표논문집, 제23권 제2호(A), pp.

35-38, 1996.

[47] N. Roussopoulos, S. Kelley and F. Vincent, “Nearest neighbor queries,” Proc. ACM SIGMOD Conf, pp. 71-79, May. 1995.

[48] G. Hjaltason and H. Samet, “Distance browsing in spatial databases,” ACM Transactions on Database Systems, Vol. 24(2):265-318, Jun. 1999.

[49] E. Frentzos, K. Gratsias, N. Pelekis and Y. Theodoridis, “Algorithms for Nearest Neighbor Search on Moving Object Trajectories,” GeoInformatica, Volume 11, Number 2, pp. 159-193, Jun. 2007.

[50] 이동호, 김형주, “구형 피라미드 기법을 이용한 최근접 질의 처리 기법,” 정보과학회논문지(B), 제28권 제1호, pp. 86-94, Mar. 1999.

[51] Z. Song and N. Roussopoulos, “K-nearest neighbor search for moving query point,” in Proceedings of SSTD, pp. 79-96, Jul. 2001.

[52] Y. Tao, D. Papadias and Q. Shen, "Continuous nearest neighbor search," Proceedings of VLDB, pp. 287-298, 2002.

[53] 권동섭, “효율적인 최근접 질의 처리를 위한 Voronoi 다이어그램 기반 그리드 검색 구조,” 한국컴퓨터정보학회논문지, 제13권 제1호, pp. 11-20, 2008.

[54] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seegel, “The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles,” Proc. Of ACM SIGMOD, pp. 322-331, Jun. 1990.

감사의 글

조용히 눈을 감고 돌이켜 생각해 보니 수많은 생각과 추억이 스쳐갑니다. 결혼 후 뒤늦게 시작한 학업이었기에 조금 더 힘들었었고 울산에서 통학하며 폭설이 쏟아진 날에도 발 동동거리며 수업에 임한 날들이 있었기에 더욱 더 소중한 시간이 아니었나 싶습니다. 학업 중 아이가 생겼고 그 아이들은 벌써 10살이 되었습니다. 인생 전체를 보면 아주 짧은 시간이지만, 또한 힘들고도 긴 시간이었습니다. 긴 터널을 지나는 중에 많은 분들의 도움과 지원 그리고 큰 가르침이 없었다면 오늘과 같은 결과는 만들어지지 않았을 것입니다.

먼저 부족한 저를 제자로 받아주신 영원한 스승님이자 존경하옵는 조우현 교수님께 진심으로 감사를 드립니다. 교수님의 가르침을 평생 가슴에 새기며 한없이 부족한 제가 그 부족함을 하나하나 채워나가는 모습으로 큰 가르침에 답하겠습니다. 더불어 논문을 심사하는 과정에서 아낌없는 조언을 해주신 정목동 교수님, 윤성대 교수님, 김종진 교수님, 류시국 교수님께 깊은 감사를 표합니다. 박사 과정 동안 많이 도와주신 연구실 김규재 후배님, 박희숙 선배님께도 감사의 말씀을 드립니다. 이 길을 선택하게 하고 포기하고 싶을 때 항상 옆에서 힘이 되어준 친구 김재천 석·박사과정 동기에게 깊은 감사를 드리며 항상 든든하게 나를 지탱해주는 ㈜나라시스템의 모든 가족들에게도 감사드립니다. 진심으로 축하해주시는 한국폴리텍대학 울산캠퍼스 박광일 학장님 이하 정보통신시스템과 전 교수님들께도 깊은 감사를 드립니다.

이 외에 제가 미처 언급하지 못한 고마운 분들이 너무나 많습니다. 그 분들의 이름을 하나하나 되새기지 못함을 죄송하게 생각하며, 과정 중에 지혜와 마음을 나누어주었던 모든 분들께 감사드립니다. 마지막으로 항상 엄마와 아내의 빈자리를 채우며 묵묵하게 믿고 기다려주며 격려해준 사랑하는 남편 김일윤씨와 쌍둥이 아들 민준, 민규, 그리고 맏언니 같지 않은 언니를 믿고 따라주는 사랑하는 여동생들과 평생 갚아도 못 갚을 한결같은 사랑을 베푸시는 양가 부모님께 이 논문을 바칩니다.