JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 12 : 2919~2925 Dec. 2014

이동객체궤적에 대한 효율적인 최근접이웃검색

김규재 · 박영희 · 조우현*

Efficient Nearest Neighbor Search on Moving Object Trajectories

Gyu-jae Kim · Young-hee Park · Woo-hyun Cho*

Department of Computer Engineering, Pukyong University, Pusan 608-739, Korea

요 약

스마트폰과 같은 이동 통신 매체의 발달과 LTE, NFC, RFID 등 무선통신의 발달로 실시간으로 이동 객체의 위치 데이터를 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 Douglas-Peucker 알고리즘을 응용하여 대용량의 이동객체궤적 데이터를 단순화하여 색인 구조를 생성하고 이 색인 구조를 이용하여 최근접이웃검색 질의를 효율적으로 처리할 수 있는 알고리즘을 제안한다. 제안된 방법으로 대용량의 데이터가 더 적은 양의 데이터로 단순화 되고 얼마나 더 효율적으로 질의를 처리하는지 실험을 통하여 확인하였다.

ABSTRACT

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 a hot issue in database research. Index structure and query processing of such contents are very important for these applications. This paper addressees algorithms that make index structure by using Douglas-Peucker Algorithm and process nearest neighbor search query efficiently on moving objects trajectories. We compare and analyze our algorithms by experiments. Our algorithms make small size of index structure and process the query more efficiently.

키워드 : 이동객체궤적, 색인구조, 최근접이웃검색

Key word : Moving Object Trajectory, Index structure, Nearest Neighbor Search

접수일자 : 2014. 10. 22 심사완료일자 : 2014. 11. 10 게재확정일자 : 2014. 11. 25

* **Corresponding Author** Woo-Hyun Cho(E-mail:whcho@pknu.ac.kr, Tel:+82-051-629-6251)
Department of Computer Engineering, Pukyong University, Pusan 608-739, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.12.2919 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright Ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

이동객체궤적이란 실시간으로 이동객체의 위치데이터를 수집하여 이 위치데이터들이 시간에 따라 연결된 궤적의 데이터를 말한다. 요즘 현대 시대에는 대부분의 사람들이 스마트폰, 패드 또는 테블릿 등의 이동통신매체를 가지고 다녀 여러 방면에서 각 매체의 위치데이터를 다량으로 수집하여 위치기반 서비스를 제공하는 기술이 늘어나고 있다. 이동통신매체 기술뿐만 아니라 LET, RFID 등의 무선통신 기술의 발달로 인해 많은 양의 이동객체궤적 데이터들이 생성되며, 이를 통한 위치기반 서비스는 대량의 데이터를 효율적으로 처리할 수 있는 적절한 색인구조가 필요하고 현재까지 많은 효율적인 색인구조들이 제안되어왔다.

질의를 효율적으로 처리하기 위한 색인구조 방법으로 크게 이동객체궤적을 색인화 하거나 이동객체궤적이 있는 공간자체를 색인화 하여 색인구조를 만드는 방법 등이 있다. 대표적으로는 MBR방식을 이용하여 R-Tree를 구성하거나 이와 유사한 방법을 사용하여 질의를 처리하는 방법들이 있고[1-4], 구형피라미드 기법을 이용하여 색인구조를 생성하는 방법[5], 공간을 그리드로 나누어 색인구조를 생성하는 방법[6, 7] 등이 있으며, 최단 경로 탐색을 통하여 질의를 더 효율적으로 처리하는 방법[8] 등이 제안되었다.

본 논문에서는 Douglas-Peucker 알고리즘을 응용하여 이동객체궤적 데이터에 대하여 트리와 유사한 색인구조를 생성하고, 이 색인구조를 이용하여 주어지는 궤적의 전체 시간에 대해 가장 가까이 있는 궤적을 찾는 최근접이웃검색 질의를 처리하는 알고리즘을 제안한다. 여기서는 궤적들의 전체 시간이 동일하다고 가정한다. 원래 시간(t)과 이차원좌표(x,y) 데이터를 가지는 삼차원의 시공간 데이터를 다루어야 하나, 알고리즘의 단순성을 위하여 시간(t)과 일차원좌표(x) 데이터만을 가지는 이차원 데이터로 축소하여 알고리즘을 설계하였다. 이는 단지 실험의 편의를 위한 것으로, 실제 삼차원 데이터를 처리하는 알고리즘으로 확장하는 것은 면적계산(시간, x)을 체적계산(시간, x, y)으로 변환하면 가능하므로 간단하다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서는 관련 연구를 알아보고, Ⅲ장에서는 본 논문에서 제안하는 색인구조 생성 알고리즘과, 이 색인구조를 이용하여 최근점이웃검색 질의를 처리하는 알고리즘을 제안한다. 4장에서는 실험 및 비교분석을 통하여 제안한 알고리즘의 성능을 평가하였다. Ⅴ장에서는 결론 및 향후 연구내용을 제시한다.

Ⅱ. 관련 연구

이동객체궤적에서의 최근접이웃검색 질의를 처리하기 위한 방법으로는 대표적으로 R-Tree를 이용한 방법들이 있다. MBR과 질의점간의 최저길이(MINDIST), 최고탐색길이(MINMAXDIST)를 이용하여 R-Tree에서 최근접이웃검색 질의를 효율적으로 처리하는 알고리즘과 질의대상이 점이 아닌 공간개체로 확장하여 질의를 처리하는 알고리즘 방법들이 있다[2, 3]. 이 방법들은 건물이나 특정 지역과 같은 시간에 따라 위치가 변하지 않는 객체나 공간객체에 대한 문제 처리를 다루고 있고, 마찬가지로 최단거리 탐색을 통한 질의처리 방법[8]도 정적인 객체에 대한 질의만을 다루고 있다. 이 외에도 움직이는 이동객체들 간의 최근접이웃검색 질의를 처리하기 위한 알고리즘도 제안되었다[1, 4].

R-Tree 이외에도 Voronoi 다이어그램을 이용하여 전처리 기법을 통하여 더 빠르게 최근접이웃검색 질의를 처리하는 방법[6]과 그리드를 이용하여 질의를 처리하는 방법[7]등이 제안되었다. Voronoi 다이어그램을 이용한 방법은 전처리를 통하여 미리 계산을 해놓기 때문에 빠른 질의처리가 가능하지만 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.

Ⅲ. 색인구조 및 최근접이웃검색 질의

이 장에서는 Douglas-Peucker 알고리즘을 이용하여 이동객체궤적데이터를 단순화하여 원본 데이터에 대한 색인 구조를 생성하고 이 색인구조 파일을 이용하여 최근접이웃검색 질의를 처리하는 알고리즘을 제안한다.

3.1. 색인구조 생성 알고리즘

Douglas-Peucker 알고리즘은 여러 개의 점과 여러 개의 직선으로 이루어진 곡선이 있을 때, 이를 좀 더 적은 개수의 점과 직선으로 이루어진 유사 곡선으로 만드는 데 이용하는 알고리즘이다.

본 논문에서는 이 알고리즘을 응용하여 이동객체궤적을 단순화하여 색인구조를 생성하는 알고리즘을 제안한다. 색인구조 생성 알고리즘을 단순하게 도식화 하면 아래 그림 1, 2와 같다.

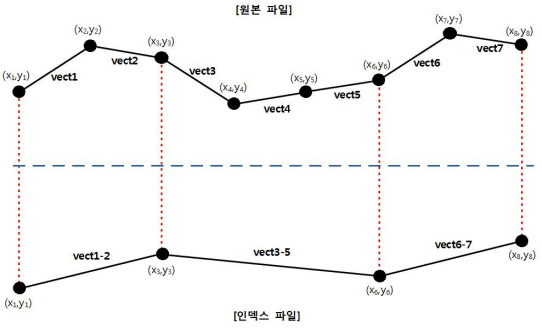

그림 1. 이동객체궤적에 대한 색인구조 생성 도식화
Fig. 1 Diagram of index structure on moving object trajectories

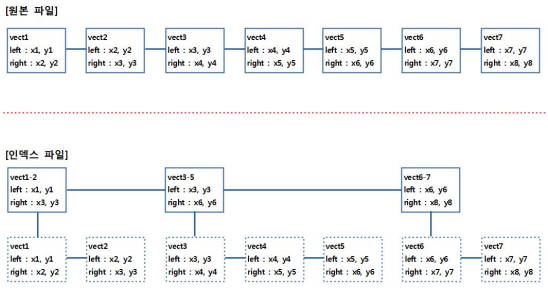

그림 2. 이동객체궤적에 대한 색인구조 생성 데이터 모델
Fig. 2 Data model of index structure on moving object trajectories

Douglas-Peucker 알고리즘은 단순화 범위인 Epsilon 값을 입력받아 이 범위 안으로 궤적을 단순화 시킨다. 마찬가지로 본 논문에서 제안하는 색인구조 생성 알고리즘에서도 Epsilon을 입력받아 원본의 이동객체궤적을 더 적은 데이터로 이루어진 궤적으로 단순화 하고, 단순화된 궤적에 원본 궤적의 데이터를 가리키는 색인을 생성함으로써 색인구조를 생성한다. 단순화 범위의 크기가 커질수록 궤적이 단순화되는 정도가 커지게 되고 이는 더 큰 압축률의 색인구조를 생성하게 된다.

알고리즘 1. 이동객체궤적에 대한 색인 구조 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));
    //수선의 길이(dmax)가 단순화범위(Epsilon)보다 짧으면
    한 궤적으로 병합하고 색인정보를 추가한다.
15. else
16.     FileWrite.write(startPoint);
17.     FileWrite.write(s);
18.     FileWrite.write(length);
19.     return (mergeList(stratPoint, endPoint));
```

그림 3. 색인구조 생성 알고리즘
Fig. 3 Indexing algorithm on moving object trajectories

알고리즘은 분할과 병합작업을 반복하면서 진행된다. 먼저 그림 3의 단계7에서 단계10까지 궤적에서 첫 번째 위치 점과 마지막 위치 점을 연결하여 직선을 만들고 이 직선에 나머지 위치 점들로부터 수선을 내려서 가장 긴 수선의 길이와 해당 위치 점을 구한다. 단계11에서 단계19까지 구해진 가장 긴 수선의 길이가 단순화범위(Epsilon)보다 길면 해당 위치 점을 기준으로 궤적을 양분하여 각각에 대해 알고리즘을 재귀 호출한다. 반대로 수선의 길이가 단순화범위(Epsilon)보다 짧으면 첫 번째 위치 점과 마지막 위치 점을 연결한 직선을 단순화된 궤적으로 하고 나머지 점들을 가리키는 색인정보를 추가하여 결과로 출력한다. 모든 재귀호출이 끝나고 더 이상 궤적이 단순화되지 않으면 알고리즘은 종료된다.

여러 개(m)의 이동객체궤적에 대해서 알고리즘을 수행할 때의 시간 복잡도는 m개의 이동객체궤적에 대해 Douglas-Peucker 알고리즘을 수행한 시간으로 나타낼 수 있다. 하나의 이동객체궤적이 n개의 위치데이터를 가질 때 알고리즘이 한번 수행되면 n번의 데이터를 읽고, 이를 절반으로 나누어 재귀호출을 함으로서 시간

복잡도는 $O(n*\log n)$이 되고, m개의 이동객체궤적에 대해 알고리즘을 수행해야 함으로 알고리즘의 총 수행시간 복잡도는 $O(m*n*\log n)$이 된다.

3.2. 최근접이웃검색 질의처리 알고리즘

알고리즘 2. 색인구조에 대한 최근접이웃검색 질의처리
입력 : 색인데이터파일 리스트(**file**), 질의궤적(**base**)
출력 : 최근접질의에 포함된 이동객체궤적들(**result**)

```
IndexNearestNeighborQuery(file, base)

 1. FileRead(base);
 2. Epsilon = FileRead.readInt(); //Douglas-Peucker 단순화 범위
 3. PointList baseList = FileReader.readPointList();
 4. MinArea = Double.MAXIMUM;  DoubleList tempList;
 5. for(int i=0; i<file.size(); i++) //파일들중 가장 작은 면적을 가지는 파일을 구한다
 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;
    //MinArea확장
22. ExtendMinArea = MinArea + (Epsilon * baseList.getLastX() * 4);
23. for(int i=0; i<tempList.size(); i++) //확장된 MinArea에 포함되는 후보자 선택
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 4. 색인구조에 대한 최근접이웃검색 질의처리 알고리즘
Fig. 4 Nearest neighbor search algorithm on index structure

본 논문에서 제안하는 색인구조 생성 알고리즘으로 색인구조 파일들을 생성하고 이에 대하여 최근접이웃검색 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 4는 본 논문에서 제안한 색인구조 파일들에 대하여 최근접이웃검색 질의를 처리하는 알고리즘이다. 알고리즘은 앞에 언급한대로 두 단계이며, 먼저 색인구조 파일들에 대하여 질의에 대하여 검사한 다음 그 중에 질의의 결과가 될 가능성이 있는 궤적들을 색출하고 이에 대하여 원본파일에 접근하여 검사함으로써 결과를 도출한다.

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. 질의의 결과가 될 가능성이 있는 궤적을 찾기 위해서 단계23에서 단계27까지 구해진 가장 작은 면적을 단순화 범위(Epsilon)로 인하여 발생할 수 있는 오차 범위를 계산하여 확장된 가장 작은 면적(ExtendMinArea)을 구하고 이 범위에 포함되는 모든 색인구조 파일에 대하여 단계26에서 원본파일에 접근하여 최근접이웃검색 질의를 처리하는 알고리즘을 호출하여 그 결과(result)를 출력하고 알고리즘을 종료한다.

이 알고리즘의 시간복잡도는 크게 두 단계로, 먼저 m개의 원본 이동객체궤적이 있고 모든 이동객체궤적이 n개의 위치데이터를 가진다고 가정할 때, 원본의 이동객체궤적을 압축률 C로 단순화하여 만든 색인구조 파일에 대하여 최근접이웃검색 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.

Ⅳ. 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, 실험은 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 수행하였다. 실험 데이터는 무작위로 시작위치와 위치변화도를 주어 데이터를 생성하였으며, 데이터의 크기는 5000개의 위치데이터(40Kbyte)를 가지는 200개의 이동객체궤적 데이터로 총 100만개의 위치데이터(8000Kbyte)로 실험을 진행하였다. 실험 결과 그래프에서 색인구조를 이용하여 질의를 처리한 결과 선은 DGS, 원본파일만으로 질의를 처리한 결과 선은 Source로 표기한다.

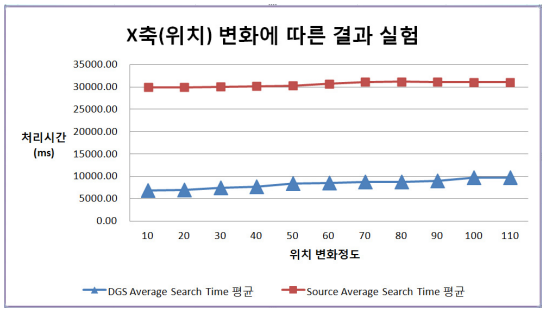

그림 5. X축(위치) 변화정도에 따른 실험 결과
Fig. 5 Algorithm execution times about the location change

그림 5는 X축(위치)의 변화되는 정도에 따라 알고리즘을 처리하는 속도를 실험한 결과이다. 압축률은 1/10로 설정하였고 색인구조 파일에는 10만개의 위치데이터와 추가적인 색인정보 데이터(1620Kbyte)를 가지고 있다. 세로축은 최근접이웃검색 질의를 처리하는데 걸리는 시간을 나타내고, 가로축은 위치 값 변화정도의 크기를 나타낸다.

위 실험에서 위치 변화정도가 가장 작은 값(가장좌측)은 10으로 원본데이터를 1/10로 압축하기 위해서 Epsilon의 값이 12가 된다. 위치 변화정도가 가장 큰 값(가장우측)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 재검사를 수행하였고 걸리는 시간은 평균적으로 6800ms가 걸렸다. 위치 변화정도가 110일 때는 200개의 파일 중 평균적으로 22.4개의 후보자(896Kbyte)가 선택되어 재검사를 수행하였고 걸리는 시간은 평균적으로 9700ms가 걸렸다. 원본파일만으로 질의를 처리하는 경우에는 평균적으로 30000ms의 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(DGS) 시간이 점점 길어지는 상향곡선을 타는 것을 볼 수 있다. 하지만 위치변화의 정도가 커지더라도 그 영향이 적어 원본파일을 이용한 질의처리(Source)보다는 평균적으로 3~4배로 빠르게 질의를 처리하는 것을 볼 수 있다.

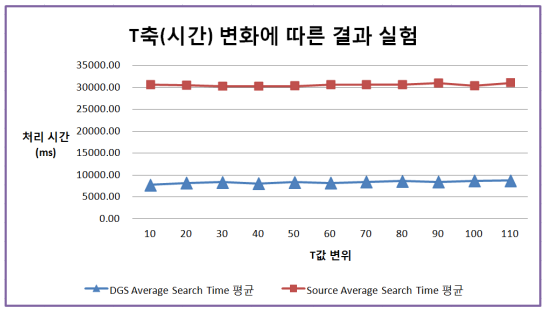

그림 6. T축(시간) 변화정도에 따른 실험 결과
Fig. 6 Algorithm execution times about time interval change

그림 6은 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 위치데이터와 추가적인 색인정보(1620Kbyte)를 포함하고 있다. 세로축은 최근접이웃검색 질의를 처리하는데 걸리는 시간을 나타내고, 가로축은 시간 측정 간격의 변화정도 크기를 나타낸다.

위 실험에서 측정 시간간격이 가장 작은 값(가장좌측)은 10으로 원본데이터를 1/10로 압축하기 위해서 Epsilon값은 45가 되고, 가장 큰 값(가장우측)은 110으로 원본데이터를 1/10로 압축하기 위해서 Epsilon값은 47이 된다. 측정 시간간격이 10일 때 색인구조를 이용

하여 질의를 처리하는데 걸리는 시간은 평균적으로 8200ms이고, 110일 때는 8700ms이다. 원본파일만으로 질의를 처리하는 경우에는 대략 30000ms의 시간이 걸린다.

위치가 측정되는 시간 간격은 그 변화정도가 바뀌어도 위치데이터 간의 시간간격이 변할 뿐, 수선의 길이와 같은 Douglas-Peucker 알고리즘에 영향을 미치는 요소에는 영향을 주지 않는다. 때문에 Epsilon의 크기에 큰 영향을 주지 않았고, 시간의 변화정도에 따라서는 큰 변화가 없이 본 논문에서 제안하는 색인구조(DGS)가 원본파일을 통한 질의처리방법(Source) 보다 평균 3~4배정도 빠르게 질의를 처리하였다.

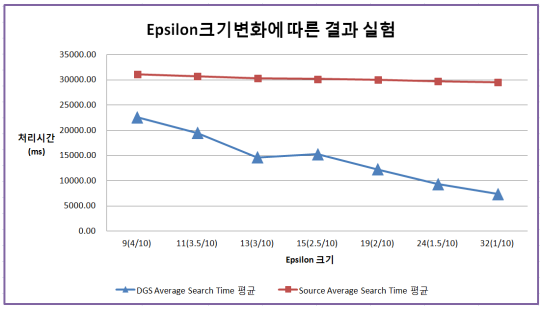

그림 7. Epsilon크기 변화에 따른 실험 결과
Fig. 7 Algorithm execution times about Epsilon size

그림 7은 동일한 X축(위치)과 T축(시간)을 가지는 파일들에 단순화범위(Epsilon)의 크기 변화에 따라 처리시간이 어떻게 되는지를 실험한 결과이다. 본 논문에서 제안하는 알고리즘으로 색인구조를 생성하면 Epsilon의 값이 커질수록 압축률이 높아져 색인구조의 데이터 크기가 줄어들지만, 질의를 처리하는데 오차범위 해결을 위함 범위확장이 커지게 된다.

위 실험에서 가장 작은 Epsilon의 크기 값(가장좌측)은 9로 압축률이 대략 4/10정도이며, 그 결과로 색인구조의 데이터 크기는 색인정보를 포함하여 대략 6250Kbyte가 되고, 색인구조를 통하여 질의를 처리할 때 200개의 파일 중 평균적으로 2개의 후보자(80Kbyte)만 선택되어 원본데이터를 재검사 하였다. 가장 큰 Epsilon의 크기 값(가장우측)은 32로 압축률이 대략 1/10정도이며, 그 결과로 색인구조의 데이터 크기는 색인정보를 포함하여 대략 1620Kbyte가 되고, 색인구조를 통하여 질의를 처리할 때 200개의 파일 중 평균적으로 11개의 후보자(440Kbyte)가 선택되어 원본데이터를 재검사 하였다.

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

Ⅴ. 결 론

본 논문에서는 Douglas-Peucker 알고리즘을 이용하여 이동객체궤적에 대한 새로운 색인구조를 생성하는 알고리즘과 이 색인구조에 대한 최근접이웃검색 질의를 처리하는 알고리즘을 제안하였다. 데이터를 단순화시켜서 색인구조를 만드는 방식이기 때문에 색인구조만으로 질의를 처리하면 오류가 발생하였고, 이러한 오류 발생을 해결하면서 최근접이웃검색 질의를 처리하기 위해서 질의에 대해 구해진 면적을 확장하여 후보자를 색출하고, 다시 한 번 원본데이터에 접근하여 질의를 처리하는 두 단계의 검사가 필요했다.

단순화범위(Epsilon)의 크기, X축(위치)의 변화정도, T축(시간)의 변화정도에 따라 질의를 처리하는데 영향을 얼마나 끼치는지 실험을 통하여 본 논문에서 제안하는 알고리즘이 얼마나 효율적인지를 확인하였다. 위치 또는 시간의 변화정도는 질의를 처리하는데 크게 영향을 끼치지 않았고, Epsilon의 크기 변화정도에 따라서는 Epsilon이 커질수록 압축률이 더 좋아져서 질의를 더 빠르게 처리하는 모습을 확인할 수 있었다.

본 논문의 내용은 이동객체궤적을 단순한 데이터의 색인구조로 만드는 간단한 알고리즘을 제안하는 내용으로 향후 더욱 정확한 비용모델을 제안하기 위해 좀 더 구체적인 색인구조 시스템 구현이 필요하고, 인위적으로 생성한 데이터가 아니라 여러 다양한 상황과 환경에서 실제 측정으로 발생하는 데이터들을 통하여 실험을 해볼 필요가 있다. 또한 기존의 방법들과의 비교를 통하여 본 논문의 알고리즘의 가능성을 확인해볼 수 있을 것이다.

향 후 Douglas-Peucker 알고리즘을 더 효율적인 알고

리즘으로 적용시킬 수 있는 방법이나 다른 방식으로 접근하여 질의를 더 빠르게 처리할 수 있는 알고리즘을 연구해볼 예정이다.

감사의 글

이 논문은 부경대학교 자율창의학술연구비(2014년)에 의하여 연구 되었습니다.

REFERENCES

[1] Elias Frentzos, Kostas Gratsias, Nikos Pelekis, and Yannis Theodoridis, “Algorithms for Nearest Neighbor Search on Moving Object Trajectories,” *Joutnal of GeoInformatica*, Volume 11, Number 2, pp. 159-193, 2007.

[2] Dong-Man Lee, Yong-Ju Lee, and Chin-Wan Cung. “Research on the Nearest Neighbor Query Processing using R-trees,” in *Journal of KIISE : database*, Vol. 23, No. 2(A), pp. 35-38, 1996.

[3] Nick Roussopoulos, Stephen Kelley, and Frederic Vincent, “Nearest Neighbor Queries,” in *Proc. ACM SIGMOD Conf*, pp. 71-79, 1995.

[4] R. Benetis, C. Jensen, G. Karciauskas, and S. Saltenis. “Nearest neighbor and reverse nearest neighbor queries for moving objeccts,” in *Proceedings of IDEAS*, 2002.

[5] Dong-Ho Lee, and Hyoung-Joo Kim. “Nearest Neighbor Query Processing using the Spherical Pyramid Technique,” in *Journal of KIISE : database*, vol. 28, No. 1, Mar. 2001.

[6] Dongseop Kwon. “A Voronoi Diagram-Based Grid Structure for Efficient Nearest Neighbor Query Processing,” in *Journal of KSCI*, vol. 13, No. 1, pp. 11-29, Jan. 2008.

[7] Yingyuan Xiao. “Set Nearest Neighbor Query for Trajectory of Moving Objects,” *Fuzzy Systems and Knowledge Discovery*, Volume. 5, pp. 211 - 214, Aug. 2009.

[8] Sung-Hyun Shin, Sang-Chul Lee, Sang-Wook Kim, Junghoon Lee, and Eul-Gyu Im. “Efficient Path Finding Based on the A* algorithm for Processing k-Nearest Neighbor Queries in Road Network Databases,” in *Journal of KIISE : database*, Vol. 36, No. 5, Oct. 2009.

김규재(Gyu-jae Kim)

부경대학교 컴퓨터공학과 학사 졸업
2013년 ~ 현재 부경대학교 컴퓨터공학과 석사과정
※관심분야 : 데이터베이스, 데이터 마이닝

박영희(Young-hee Park)

부경대학교 컴퓨터공학과 박사과정 수료
2014년 ~ 현재 한국폴리텍 VII대학 정보통신시스템과 초빙교수
※관심분야 : 데이터베이스, 디지털 포렌식

조우현(Woo-hyun Cho)

경북대학교 컴퓨터공학 박사
1989년 ~ 현재 부경대학교 공과대학 컴퓨터공학과 교수
※관심분야 : 지능형 데이터베이스, 멀티미디어 인덱싱, 객체 데이터베이스 관리 기술