이동객체궤적에 대한 효율적인 범위질의

박영희* · 김규재** · 조우현***

*부경대학교

Efficient Range Query on Moving Object Trajectories

Young-Hee Park* · Kyu-Jae Kim** · Woo-Hyun Cho***

*Pu-Kyong National University

E-mail : rbwotm22@naver.com

요 약

이동 통신, RFID, GPS 등의 무선 원거리 통신의 발달로 시간의 흐름에 따라 변하는 이동 객체의 위치 정보들을 수집하여 활용하는 위치 기반의 서비스들이 다방면으로 이용되고 있다. 이에 따라 대용량의 이동 객체의 위치 데이터들을 빠르게 검색하기 위한 효율적인 색인 방법과 질의 처리에 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 분석하였다. 실험 결과로 제안된 방법에서는 색인 데이터 량이 상대적으로 작아지고 색인 및 질의 처리방법이 간단하며 기존의 방법보다 시공간적으로 효율적임을 확인하였다.

ABSTRACT

Location-Based services that collect location information for moving object and utilize in real life are being used in many aspects by the development of wireless network technology. Accordingly, new index structures are required to efficiently retrieve the consecutive location of moving objects. This paper addresses algorithms that make index structure by using Douglas-Peucker Algorithm and process range query efficiently on moving objects trajectories. Our algorithms are going to make smaller size of index structure and process more efficiently.

키워드

색인구조, 이동객체궤적, 범위질의

Ⅰ. 서 론

이동객체란 시간의 변화에 따라 공간적인 위치 및 모양이 연속적으로 변하는 시공간데이터이다. 최근 다양한 위치 기반 서비스의 사용이 증가하면서 이러한 데이터들을 관리하고 사용하는 이동객체 데이터베이스에 대한 관심이 증가하고 있다.

이동 통신의 발달로 현대 사회에서는 대부분의 사람들이 노트북, 스마트폰, 태블릿 등 휴대용 단말기를 하나 혹은 여러 개를 동시에 들고 다니며, 이를 통한 위치 기반 서비스는 대량의 데이터를 효율적으로 처리할 수 있는 적절한 색인구조가 필요하다. 현재까지 B^+-트리, R-트리 또는 사분트리, 최소경계사각형 근사법, 다항식 근사법 등의 여러 가지 색인 방법이 제시되었다. 본 논문에서는 Douglas-Peucker 알고리즘을 이용하여 이동객체궤적을 단순화하고 이를 이용하여 색인구조를 생성하고 이 색인구조에 대하여 범위질의를 처리하는 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 이동객체의 시공간 색인에 대한 관련 연구들을 살펴보고 Ⅲ장에서는 색인구조를 생성하는 알고리즘과 이 색인구조에 대하여 범위질의를 처리하는 알고리즘을 제안한다. Ⅳ장에서는 제안하는 색인구조 알고리즘의 우수성을 입증하기 위해 기존 방법과의 성능 평가를 수행한다. 마지막으로 Ⅴ장

에서는 결론 및 향후 연구에 대해 기술한다.

II. 관련 연구

이동객체궤적에 대한 질의는 범위질의, k-최근접이웃검색[1]등이 있는데 이러한 질의들을 시공간적으로 효율적인 처리를 하기 위해 주로 쓰이는 공간색인 방법으로는 이동객체궤적 데이터들을 최소경계사각형으로 분할하여 R-트리 기반색인구조를 생성하는 방법[2]과 B^+-트리를 기반으로 하는 방법[3] 그리고 공간을 몇몇의 섹터로 나누어 처리하는 사분트리와 해싱을 이용한 방법[4]과 이동객체궤적을 다항식으로 근사하여 색인함으로써 검색공간을 줄이는 방법[5,6] 등이 고안되었다.

이러한 방법들은 이동객체궤적에 대한 색인 구성 과정에서 효율적인 데이터 구조를 사용하거나 MBR과 같은 실제 데이터에 대한 근사치 영역을 사용한다. 어떠한 데이터 구조와 근사 방법을 사용하는지는 색인 구성과정에서 검색 효율을 위해 중요한 요소가 된다.

본 논문에서 응용한 Douglas-Peucker 알고리즘[7]도 실제 점과 직선으로 이루어진 데이터가 있을 때 단순화정도가 주어지면 그 값을 근사치 기준으로 하여 더 단순한 점과 직선들로 근사하는 알고리즘이다.

III. 색인구조 및 범위질의

Algorithm 1. 이동객체궤적에 대한 색인 구조 생성
Input : 이동객체궤적데이터(**srcList**), 단순화범위(**Epsilon**), 인덱스(s)
Output : 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;
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;
11. if(dmax > Epsilon)
12.      lList = IndexCreation(src, devideList(srcList,0,index), Epsilon, s);
13.      rList = IndexCreation(src, devideList(srcList,index, length), Epsilon, s);
14.      return (mergeList(lList, rList));
15. else
16.      FileWrite.write(startPoint);
17.      FileWrite.write(s);
18.      FileWrite.write(length);
19.      return (mergeList(stratPoint, endPoint));
```

그림 1. 이동객체궤적에 대한 색인구조 생성 알고리즘

본 논문에서 제안하는 색인구조 생성 알고리즘은 Douglas-Peucker 알고리즘을 응용하여 이동객체궤적 데이터를 단순화시켜서 색인구조데이터를 만드는 알고리즘이다. 데이터를 어느 정도로 단순화 할 것인지를 나타내는 단순화범위가 입력되어야 한다. 단순화 과정은 먼저 단계2에서 3까지 이동객체궤적 데이터에서 첫 번째 데이터(firstPoint)와 마지막 데이터(endPoint)를 읽고, 단계5에서 10까지 첫 번째와 마지막 데이터로 이루어진 직선으로 이동객체궤적의 모든 데이터에서 수선을 그어서 그 거리를 계산하고, 최대수선의 길이(dmax)와 그 값을 가지는 데이터의 인덱스(index)를 구한다.

마지막으로 단계11에서 19까지 구해진 최대수선의 길이가 단순화범위 안에 포함이 되어야 단순화작업을 수행하여 색인구조결과를 출력하고, 최대수선의 길이가 단순화범위를 벗어나면 단순화 작업을 하지 않고 구해진 인덱스를 기준으로 이동객체궤적 데이터를 양쪽으로 나누어서 알고리즘을 반복적으로 수행한다.

Algorithm 2. 범위질의에 대한 이동객체궤적 검색
Input : 색인데이터(**dug**), 질의범위(**qx, qy, width, height**)
Output : 질의범위에 포함된 이동객체궤적들(**result**)

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

```
1.  FileRead(dug);
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();
9.       point2 = FileRead.readPoint();
10.      if(queryIntersect(eqx, eqy, ewidth, eheight, point1, point2))
11.           if(queryInSrcFile(dug, qx, qy, width, height, firstIndex, countPoint))
12.                result = dug;
13.                break;
14.      point1 = point2;
15. return result;
```

그림 2. 이동객체궤적에 대한 범위질의 처리 알고리즘

[그림2]는 범위질의 처리 알고리즘으로, 생성된 색인구조데이터에 대하여 범위질의를 수행하면 먼저 색인구조데이터를 검사하기 전에 단계2에서 4까지 착오배제(False Drop)를 해결하기 위해서 범위질의의 크기를 색인구조데이터의 단순화범위(Epsilon)만큼 확장한다.

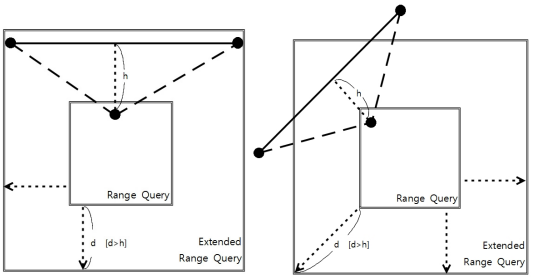

그림 3. 범위질의 크기 확장

그 다음 단계5에서 10까지 색인구조데이터로부터 차례대로 데이터를 읽으면서 확장된 범위질의에 속하는지 검사한다. 만약 확장된 범위질의에 포함된다면, 해당 색인구조데이터의 이동객체궤적은 후보 이동객체궤적이 되고, 단계11에서 13까지 이 후보 이동객체궤적에 대하여 원래 크기의 범위질의에 대한 상세검사(알고리즘3)를 수행하여 최종 결과를 출력한다.

```
Algorithm 3. 후보 이동객체궤적에 대한 상세검사
Input  : 후보 이동객체 궤적데이터(candidate), 질의범위(qx, qy, width, height),
         색인정보(startIndex, count)
Output : 상세검사 결과(true 혹은 false)

    queryInSrcFile(candidate, qx, qy, width, height, startIndex, count)
 1. FileRead(candidate.getName( )+".src");
 2. FileRead.skip(startIndex);
 3. point1 = FileRead.readPoint( );
 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;
 8.       point1 = point2;
 9. return false;
```

그림 4. 범위질의에 대한 후보이동객체궤적 상세검사

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.

Ⅳ. 실험 및 비교분석

본 논문에서 제안하는 알고리즘의 성능을 분석하기 위해서 기존의 이동객체궤적 색인방법인 MBR방법과 비교 실험을 해보았다. 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 총120개의 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 나타내며, 예를 들면 범위질의 검사에 대하여 후보 이동객체궤적이 100개가 추출되고, 결과로서 이동객체궤적이 80개가 질의에 포함되면 80/100으로 80%의 정확도를 가진다.

아래 [그림5]과 [그림6]은 범위질의의 크기에 따라서 두 방법의 정확도와 수행속도가 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.

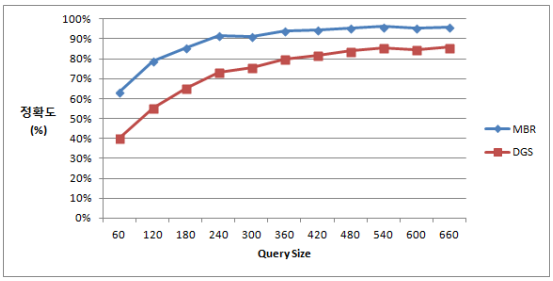

그림 5. 범위질의 크기에 따른 알고리즘 정확도

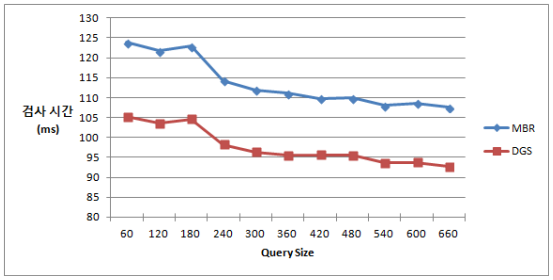

그림 6. 범위질의 크기에 따른 알고리즘 수행시간

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.

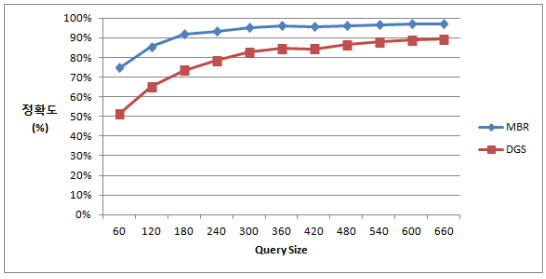

그림 7. 범위질의 크기에 따른 알고리즘 정확도 (압축 비율 1/5)

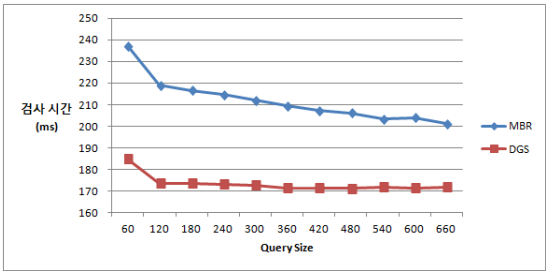

그림 8. 범위질의 크기에 따른 알고리즘 수행시간 (압축 비율 1/5)

[그림7]과 [그림8]의 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 실험결과로는 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.

Ⅴ. 결 론

본 논문에서는 이동객체궤적에 대한 색인구조 생성과 범위질의를 처리하는 알고리즘을 제안하였다. 제안하는 알고리즘들은 Douglas-Peucker 알고리즘을 응용하여 단순한 색인구조를 만들고 이를 이용하여 범위질의를 처리한다. 기존의 MBR 방법과의 성능 비교 실험 결과 제안하는 알고리즘의 방법이 더 적은 데이터양의 색인구조를 생성하고 범위질의를 처리하는데 걸리는 시간이 더 적게 걸리는 결과가 나왔다. 이는 제안하는 알고리즘을 이용하여 이동객체궤적 데이터를 색인구조로 만들어 두면 범위질의를 처리하는데 있어 처리하는 데이터의 양이 감소되어 성능 향상을 이룰 수가 있다는 것을 알 수 있다.

본 논문의 내용은 단순히 두 방법의 알고리즘만을 비교한 내용으로 향후 더욱 정확한 비용모델을 제시하기 위해서 이동객체궤적에 대하여 좀 더 구체화된 색인구조 구현을 통한 실험이 필요하다.

참고문헌

[1] Hans-Peter Kriegel, Peer Kroger, and Matthias Renz. Techniques for Efficiently Searching in Spatial, Temporal, Spatio-temporal, and Multimedia Database. 14th ICDE, DASFAA'09, Brisbane, Australia, pp. 780-783, 2009.

[2] Marios Hadjieleftheriou, George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. Efficient Indexing of Spatio temporal Objects. 8th International Conference on Extending Database Technology Prague, Czech Republic, 2002.

[3] Christian S. Jensen, Dan Lin, and Beng Chin Ooi. Query and Update Efficient B^{+}-Tree Based Indexing of Moving Objects. Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004.

[4] Dong Li, Yu-hui Peng, and Jiang-long 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.

[5] Elena Braynova. Indexing Spatio-Temporal Trajectories with Orthogonal Polynomials. Conference on Data Mining - DMIN, 2006.

[6] Jinfeng Ni and Chynya V. Ravishankar, Senior Member, IEEE. Indexing Spatio-Temporal Trajectories with Efficient Polynomial Approximations. IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 5, May 2007.

[7] 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.