이동객체궤적에 대한 효율적인 색인구조

김규재* · 조우현*

*부경대학교

Efficient Indexing structure for Moving Object Trajectoriest

Gyu-Jae Kim* · Woo-hyun Cho*

*Pukoung National University

E-mail : rbwotm22@naver.com

요 약

다차원의 공간상에서 이동객체궤적을 다루는 기존의 방법은 일반적으로 최소 경계 사각형(MBR)의 방법을 사용한다. 이러한 방법은 근사 형태가 부정확하여 최소 경계 사각형에 많은 빈 공간이 발생하고, 같은 공간에 대한 중복이 발생한다. 이러한 문제로 인해 질의를 처리할 때 불필요한 연산을 많이 수행하게 되어 효율성이 저하된다. 여러 연구와 논문에서 이를 해결하기 위해 최소 경계사각형이나 이를 사용하는 트리 구조에 대한 알고리즘들을 논의하거나 대체 방법들을 제안하였다. 본 논문에서는 이동객체궤적 데이터에 대해 Douglas-Peucker 근사화 알고리즘을 응용한 색인구조를 제안한다. 색인구조 생성 알고리즘과 데이터 삽입 및 삭제 알고리즘을 제안하고 기존의 방법보다 얼마나 효율적인지를 비교실험을 통하여 확인한다.

ABSTRACT

In n-dimensional spatial data, Minimum Boundary Rectangle(MBR) was used to handle the moving object trajectories data. But, this method has inaccurate approximation. So, It makes many dead space and performs unnecessary operation when processing a query. In this paper, we offer new index structure using approximation. We developed algorithm that make index strucutre by using Douglas-Peucker Algorithm and had a comparison experiment.

키워드

이동객체궤적, 색인구조, Douglas-Peucker, 편차

Ⅰ. 서 론

위치기반서비스(Location Based Service), 지리정보시스템(Geographic Information System) 등에서는 대량의 데이터인 이동객체 및 공간객체를 효과적으로 저장하고, 갱신, 삽입, 분석할 수 있는 효율적인 자료구조가 필요하다.

지금까지 많은 공간 색인구조가 연구되어 왔으며 크게 점 접근 방법과 공간 접근 방법으로 나누어진다. 대표적으로는 공간을 격자 형태로 나누는 Grid-File을 이용한 방법[1,2,3], 차원변환을 이용하는 k-d 트리, 공간 객체를 이용한 R-Tree기법, 다항식 기법[4,5]등이 있다.

그중에서도 R(+,*)-Tree는 MBR(Minimum Boundary Rectangle)을 이용한 색인 기법으로 공간을 사각형의 객체로 근사시켜 색인구조를 생성하는 방법이다. 초기의 R-Tree는 색인구조의 겹침 허용, 갱신 및 삽입시의 분할 합병에 의한 계산 증가 등의 한계와 문제점이 있었지만 많은 연구를 통해 여러 가지 해결방안들이 제시되어왔다.

본 논문에서는 MBR과 비슷한 방식으로 Douglas-Peucker 알고리즘을 응용하여 이동객체궤적 데이터들을 좀 더 효율적으로 근사화하여 색인구조를 생성하고 범위질의(Range Query)를 처리하는 알고리즘을 제안한다. 또한 제안된 알고리즘이 얼마나 효율적으로 색인구조를 생성하고 질의를 처리하는지 비교실험을 통하여 분석한다.

II. 관련연구

대량의 위치데이터 또는 이동객체궤적 데이터를 좀 더 빠르고 효율적으로 처리하기 위해 Grid[1,2,3], MPTB-tree[6], 3D-Rtree[7,8] 개선된 R-tree[7,8,9]등이 연구되었다.

고정Grid방식[2]은 R-tree의 갱신에 의한 부하 문제를 해결하기 위해 공간을 고정된 Grid로 나누어 R-tree와 혼용하여 색인구조를 생성한다. 공간을 고정된 크기로 나누기 때문에 R-tree처럼 갱신될 때마다 MBR의 크기 변경으로 인한 부하가 줄어들고 저장 공간을 효율적으로 사용할 수 있지만 특정 그리드에 데이터가 집중되면 해당 그리드에 대해 처리하는데 부하가 걸릴 수 있다. 이를 완화시키기 위해서 특정 시간, 특정 지역에 이동객체가 밀집되는 경우 이를 분산시키는 방법들이 제안되었다[1,3].

MPTB-tree(최소전파TB-tree)[6]또한 마찬가지로 EMBR을 이용하여 이동객체의 이동 예상 영역을 설정하여 MBR의 갱신에 의한 부하를 완화시키는 방법을 제안하였다.

갱신에 의한 부하를 줄이는 방법 이외에도 이동객체궤적을 다항식으로 근사화하여 색인함으로써 검색공간을 줄이는 방법[4,5], 3D-Rtree에 now 태그를 추가하여 현재 시점에서의 이동객체의 위치를 효율적으로 처리하는 방법[8]과 MBR자체의 공간을 효율적으로 분할하기 위하여 범위질의의 크기를 통해 질의 예상영역을 설정하고 이를 반영하여 MBR을 분할하는 방법, MBR에 각도값을 추가하여 MBR을 회전시킴으로써 Dead Space를 줄여 좀 더 효율적으로 질의를 처리할 수 방법[9] 등이 제안되었다.

MBR을 좀 더 효율적으로 분할하는 방법[9,10]들은 교차영역과 Dead Space영역을 줄임으로써 질의의 효율이 향상되지만, 색인구조 생성, 질의처리, 데이터 입력 및 갱신 등의 작업을 처리하는데 추가적인 계산이 필요하다는 부하가 생긴다. 본 논문에서는 이와 비슷한 개념으로 색인구조를 더 정확하고 효율적으로 근사화 할 수 있는 방법을 제안하고자 한다.

III. 색인구조 생성 및 범위질의 처리

Douglas-Peucker알고리즘은 여러 개의 점으로 이루어진 곡선이 있을 때 수선을 이용하여 더 적은 개수의 점으로 이루어진 유사 곡선으로 근사화하는 알고리즘이다. 본 논문에서 제안하는 색인구조 생성 알고리즘은 이 Douglas-Peucker 알고리즘을 응용하여 더 효율적인 근사화를 통해 색인구조를 생성한다.

· 효과적인 처리 범위

아래 그림[1, 2]는 본 논문에서 제안하는 색인구조 알고리즘을 도식화한 것이다.

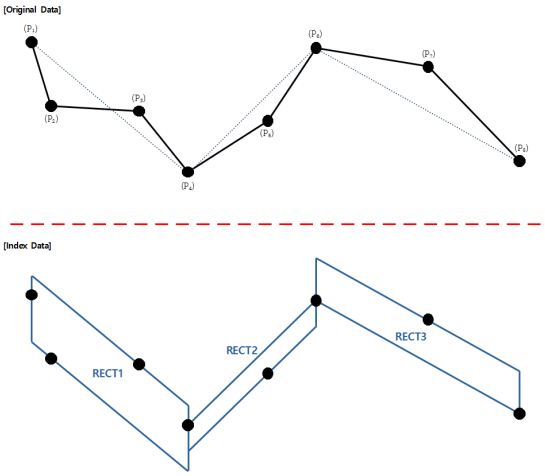

그림 1. 색인구조 생성 알고리즘 도식화

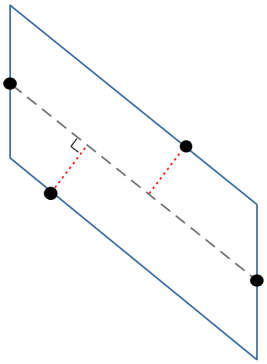

그림 2. 색인구조 데이터 도식화

Douglas-Peucker 알고리즘과 같이 수선을 이용하여 이동객체궤적을 근사화 하고 사각 형태를 만들어 원본데이터에 대한 색인을 추가함으로써 색인구조를 생성한다.

알고리즘 1. 색인데이터 생성
입력 : 색인대상 원본파일(**srcList**)
출력 : 색인 데이터(**result**)

IndexRectangle(file, base)

```
1.  maxVertical = upVerticla = downVertical = 0;
2.  maxVerticalIndex = -1;
3.  firstPoint = srcList.getFirst();
4.  lastPoint = srcList.getLase();
    //첫점과 끝점을 잇는 직선에 각 점들로부터의 수선의 길이를 구한다.
5.  for(int i=0; i<srcList.size(); i++)
6.      distance = srcList.get(i).verticalDistance(firstPoint, lastPoint);
7.      if(절대값(distance) > maxVertical) //최대 수선길이와 위치를 구한다.
8.          maxVertical = 절대값(distance);
9.          maxVerticalIndex = i;
10.     if(distance >= 0) //위쪽 방향에 대한 최대수선의 길이를 구한다.
11.         if(절대값(distance) > upVertical)
12.             upVertical = 절대값(distance);
13.     else //아래쪽 방향에 대한 최대수선의 길이를 구한다.
14.         if(절대값(distance) > downVertiacl)
15.             downVertical = 절대값(distance);
16. deviation = upVertical - downVertical; //방향에 대한 편차를 구한다.
17. return new index(srcList, maxVertical, maxVerticalIndex, deviation);
```

그림 3. 색인데이터 생성 알고리즘

제안된 색인구조에서 하나의 색인 데이터[그림 1:RECT]는 MBR처럼 사각형모양의 데이터이지만 기울어진 모양을 취한다. 사각형의 크기는 [그림 3]을 알고리즘의 단계5에서 단계16까지 원본데이터의 양 끝점을 잇는 직선에 각 점들로부터 수선의 길이를 구하고, 가장 길이가 긴 수선의 길이(maxVertical)와 수선의 방향에 의한 편차(deviation)값에 의해 결정된다. 하나의 색인 데이터는 사각형의 시작 위치점(x,y)과 색인정보, 최대 수선의 길이, 편차 정보를 가진다.

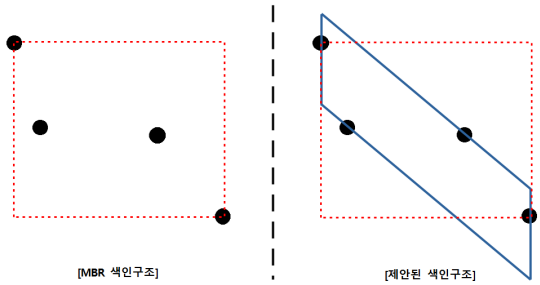

그림 4. MBR과 제안된 색인구조 형태

기존 MBR과의 차이점은 색인데이터의 구조가 단순히 가로세로의 사각형 형태로 나타나지 않고 위치데이터의 분포에 따라 기울어진 형태로 나타난다는 것이다. 이는 위치데이터가 위아래로 급격히 변할 때 좀 더 적은 DeadSpace를 생성할 수 있다는 이점이 생긴다.

· 색인구조 생성 알고리즘

알고리즘 2. 이동객체궤적에 대한 색인 구조 생성
입력 : 이동객체궤적데이터(**srcList**)
출력 : 색인구조데이터(**indexList**)

```
IndexCreation(srcList)
1.  tmpIndex = indexRectangle(srcList)
    //tmpIndex=하나의 색인. 원본파일의 색인, 최대수직선길이(dmax), 최대수선위치(index), 편차(deviation)를 가진다.
2.  indexList.add(tmpIndex);
3.  while(색인구조 생성 완료까지)
4.      indexList.sort(desc);
5.      for(int i=0; i<indexList.size(); i++)
        //현재 색인데이터의 최대수선의위치를 기준으로 양쪽에 대해 색인구조를 구한다.
6.          devIndex = indexList.get(i).index;
7.          LeftChild = indexRectangle(indexList.get(i).subList(0,devIndex));
8.          RightChild = indexRectangle(indexList.get(i).subLict(devIndex+1));
            //나누어진 양쪽의 색인구조 영역의 합이 기존의 색인구조 영역보다 적으면 분할하여 저장한다.
9.          if(LeftChild.getArea() +RightChiled.getArea()<= indexList.get(i).getArea())
10.             indexList.remove(i);
11.             indexList.add(LeftChild);
12.             indexList.add(RightChiled);
13.             break;
        //나누어진 양쪽영역의 합이 기존 색인구조 영역보다 적은 색인이 없으면 가장 큰 색인영역을 가지는 색인구조 데이터를 분할한다.
14.     devIndex = indexList.get(0).index;
15.     LeftChild = indexRectangle(indexList.get(0).subList(0,devIndex));
16.     RightChild = indexRectangle(indexList.get(0).subList(devIndex+1));
17.     indexList.remove(0);
18.     indexList.add(LeftChild);
19.     indexList.add(RightChild);
20. return indexList;
```

그림 5. 색인구조 생성 알고리즘

[그림5]는 원본데이터를 입력받아 색인구조를 생성하는 알고리즘이다. 원본데이터를 근사화 하여 [그림1]과 [그림4]에서의 형태로 이어지는 색인구조 데이터가 생성된다. 색인구조 생성 방법은 먼저 단계1에서 단계2까지 전체 원본데이터를 가리키는 색인데이터를 생성하여 색인결과리스트에 추가시킨다. 단계4에서 색인결과리스트를 색인구조의 크기(사각형의 넓이)에 따라 내림차순으로 정렬한다. 단계5에서 단계13까지 정렬된 색인결과리스트를 차례대로 읽으면서 색인데이터를 분할하였을 때 분할된 영역들의 합이 기존의 영역보다 크기가 작으면 분할을 하고 그렇지 않으면 다음 색인데이터를 읽는다. 만약 분할할 수 있는 영역이 없으면 단계14에서 단계19까지 가장 큰 영역크기를 가지는 색인데이터를 분할하여 저장한다. 이와 같은 과정을 색인구조가 생성이 완료 될 때까지 반복함으로써 색인구조가 생성된다.

Ⅳ. 실험 및 비교분석

실험은 파일시스템에서 난수발생으로 이동객체궤적데이터들을 생성하고 동일한 원본데이터에 대해 MBR과 본 논문에서 제안하는 방법으로 색인구조를 생성하였다. 색인구조 레코드 하나의 크기는 MBR과 본 논문의 방법 모두 24바이트로 동일하고, 두 방법 모두 1/10 압축률로 색인구조를 생성하였다.

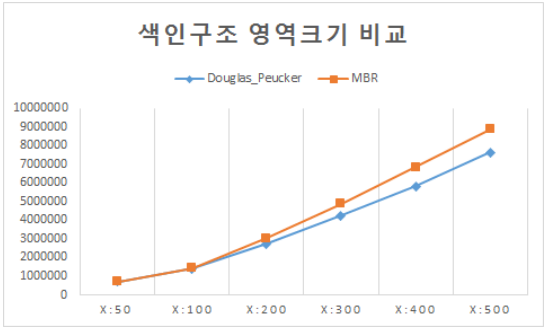

그림 6. 색인구조 영역크기 비교 그래프

[그림6]은 두 방법으로 색인구조를 생성하였을 때 생기는 면적의 크기를 비교한 그래프이다. 가로축은 이동객체궤적 원본데이터의 위치변화 정도를 나타내며 세로축은 총 면적의 크기를 나타낸다.

위치변화 정도가 크지 않으면 두 방법이 비슷한 크기의 면적으로 색인구조를 생성하였으나 위치변화 정도가 커질수록 본 논문에서 제안하는 방법이 기존 MBR방법보다 적은 면적의 색인구조를 생성하는 것을 볼 수 있었다. 위치변하가 무한정 커지더라도 일정 한계 이상의 차이는 발생하지 않았으며 최대 15%정도까지 면적차이가 발생하였다.

이를 통해 동일한 질의를 처리할 때, 본 논문에서 제안하는 방법이 면적을 적게 생성하여 더 효율적으로 질의를 처리할 수 있다는 것을 알 수 있다.

Ⅴ. 결 론

본 논문에서는 Douglas-Peucker 알고리즘을 응용하여 수선과 편차를 이용해 기울어진 사각형 모양으로 색인구조를 생성하는 방법을 제안하였다. 4장에서의 실험을 통해 기존의 색인구조보다 더 적은 면적의 색인구조를 생성하여 더 효율적인 것을 확인하였다.

이동객체궤적의 위치데이터가 특정 방향으로 기울어져 있거나, 위치변화의 정도가 크지 않으면 기존의 방법과 큰 차이가 없지만 시간과 2차원공간에서 3차원데이터에 대해 수선을 이용하여 색인구조를 생성하면 직육면체 모양이 아닌 원기둥 모양으로 더 효과적인 색인구조를 생성할 수 있다는 향후 가능성을 제시할 수 있다.

본 논문에서 수행한 실험은 가상으로 난수를 이용해 생성한 데이터이기 때문에 실제 현실에서의 데이터를 사용하면 결과가 다를 수 있다. 실제 데이터를 이용하여 실험을 수행하고 어떠한 상황에서 제안된 알고리즘이 더 효과적인지를 연구하고 이를 보완할 필요가 있다.

향후 더 많은 연구를 통해서 더 효과적인 편차를 구하는 방법과 분할 방법을 고안하여 질의를 좀 더 빠르고 효율적으로 처리할 수 있는 색인구조를 개발해야 한다.

참고문헌

[1] Segil Jeon, Yunmook Nah. Range Query Processing using Space and Time Filtering in Fiexed Grid Indexing. Korea Information Processing Society Conference. Vol. 11-D, No. 4, pp. 835-844. Aug 2004.

[2] 이양구, 이용대, 류근호. 고정 그리드를 이용한 이동객체의 위치 색인 기법. 한국공간정보시스템학회 국내 LBS 기술개발 및 표준화 동향 세미나. 2004년

[3] Seung-Tae Hong, Young-Chang Kim, Jae-Woo Chang. Design of Distributed Grid Sceme for Moving Objects. Koria GIS 2008 conference. pp. 188-194. Oct 2008.

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

[5] 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.

[6] 고주일. 이동객체궤적 색인을 위한 최소 전파 TB-Tree. 인하대학교 전자계산공학 학위논문(석사). 2004.

[7] Ren XiangChao, Kee-Wook Rim, Nam ji Yeun, Lee KyungoH. Rend 3D R-tree: An Improved Index Structure in Moving Object database Based on 3D R-tree. Korea Information Processing Society Conference. Vol. 15, No. 02, pp. 0878~0881. Nov 2008.

[8] Bong-Gi Jun. A Study on Indexing Moving Objects using the 3D R-tree. Journal of the KSCI. Vol. 10, No. 4, pp. 65-75. Sep 2006.

[9] Jung-Ho Son, Hee-Chul Kim, Kirack Sohn. RR-tree: Spatial Indexing Method using Rotated Minimum Bounding Rectangles. Vol. 4, No. 2, pp. 0487-0492. Oct 1997.