이동체의 위치 색인을 위한 궤적 클러스터링 기반의 분할 정책

김진곤[O] 전봉기 홍봉희
부산대학교
{jgkim11[O], bgjun, bhhong}@pusan.ac.kr

Splitting policies based on clustering trajectories for indexing positions of moving objects

Jingon Kim[O] Bonggi Jun Bonghee Hong
Pusan National University

요 약

이동성을 갖는 장치들의 위치 정보를 관리하기 위하여 이동체 데이터베이스에 관한 연구가 필요하게 되었다. 이동체 색인의 검색에서 영역 질의와 궤적 질의는 공간 근접성과 궤적 연결성과 같이 상반된 특징으로 인하여 함께 고려되지 않았다. 이동체 색인에서 영역 질의의 성능개선을 위해서는 노드간의 심한 중복과 사장 공간(Dead space)을 줄여야 하고, 궤적 질의의 성능 개선을 위해서는 이동체의 궤적 보전이 이루어져야 한다. 이와 같은 요구 조건을 만족하기 위해, 이 논문에서는 R-tree를 기반의 색인 구조에서 새로운 분할 정책을 제안한다.

제시하는 색인 구조에서 단말 노드의 엔트리는 궤적이며, 비단말 노드의 엔트리는 자식 노드이다. 단말 노드 분할 정책에서 동일 궤적을 그룹화해서 분할 하는 공간 축 분할 정책과 공간 활용도를 높이는 시간 축 분할 정책을 제안한다. 시간 축 분할 후 사장영역이 클 경우에는 다중 분할을 수행하여 사장 공간을 줄이고 노드간의 중복을 최소화한다. 비 단말 노드 분할 정책에서는 같은 궤적을 저장하는 노드들을 연결 노드(Connected Node)라고 정의하고, 엔트리의 궤적 연결성을 기준으로 분할한다.

1. 서 론

이동체는 시간에 따라 연속적으로 위치 정보가 변경되는 시공간 객체이다. 이동체의 과거 궤적은 3차원 점들을 연결한 선분들(Line Segments)로 표현되며, 이를 저장 관리하기 위해 시공간 색인이 사용된다.

이동체 색인에 있어서 질의 종류로는 크게 영역 질의, 타임스탬프(timestamp) 질의, 궤적 질의, 복합 질의로 나누어진다. 영역 질의는 주어진 시간 간격 동안에 공간 윈도우(spatial window)에 속하는 이동체들을 검색하는 질의이다. 타임스탬프 질의는 특정 시간에 주어진 공간 윈도우에 속하는 이동체들을 검색하는 질의이다. 궤적 질의는 특정한 이동체의 궤적을 검색하는 질의이다. 복합 질의는 " 특정 시간에 주어진 영역을 지나간 객체의 궤적을 검색하라" 와 같이 시공간 도메인의 영역 질의와 궤적 질의가 복합된 질의이다.

이동체 궤적을 3D R-tree에 저장하면 노드간의 중복이 많아 검색 성능이 저하된다. 또한 궤적의 삽입 순서가 시간 축으로 증가하는 특징으로 인하여 공간 활용도가 낮다는 문제점이 있다. 3D R-tree는 궤적의 보전(Trajectory preservation)에 관한 개념을 고려하지 않기 때문에, 궤적 질의를 처리하기 위해서 점 질의(point query)를 이용하여 반복 검색하기 때문에 질의 처리 비용이 크다는 문제점이 있다. 이 논문에서는 R-tree의 색인 구조를 사용하여 노드간의 중복을 최소화하고 궤적 클러스터링을 고려한 노드 분할 정책을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 기술하고, 3장에서 문제 정의를 설명하고, 4장에서 단말 노드의 공간 축 분할 정책, 시간 축 분할 정책, 다중 분할 정책을 제안한다. 5장에서는 궤적 보전을 위한 비단말 노드 분할 정책을 제안한다. 끝으로 결론 및 향후 연구를 기술한다.

2. 관련연구

이동체의 연속적인 이동을 표현하기 위해, 선분을 저장하는 색인 구조에 관한 기존 연구로서는 3D R-tree[1], STR-tree[2], TB-tree[2] 등이 있다. 3D R-tree는 2차원 공간 색인에 시간 축을 추가한 3차원 색인이다. 이동체의 궤적인 선분들을 3D R-tree에 저장하면 중복이 심하고, 사장 공간이 크다는 문제점이 있다. STR-tree는 R-tree의 변형으로서, 보전 파라미터(Preservation Parameter)를 사용하여 같은 객체의 궤적을 근접한 페이지에 저장하도록 유도한 색인이다. 하지만 STR-tree는 3D R-tree에 비해 영역 질의와 궤적 질의에서 성능 향상이 없다. TB-tree는 궤적 보전을 위해 단말 노드에 하나의 궤적만을 저장함으로써, 궤적 질의에서는 우수한 성능을 보인다. 하지만 단말 노드간의 중복이 심하다는 문제점으로 인하여 영역 질의의 성능이 나쁘다.

노드 간의 중복을 줄여서 색인의 성능을 향상 시키는 기존의 공간 색인에 연구로서는 R*-tree[4], R+-tree[5], X-tree[6]가 있다. R*-tree는 영역(Area) 외에 중복(Overlap)과 자장자리(Perimeter)를 사용하여 노드간의 중복을 줄이고자 하였다. 하지만 이동체와 같이 다차원 데이터 간의 심한 중복(high overlap)이 있는 경우에는 R*-tree의 분할 알고리즘은 노드간의 중복을 줄일 수 없다. R+-tree는 절단 방법(clipping-based scheme)을 사용하는 색인으로서 노드간의 중복을 허용하지 않는다. 하지만 R+-tree는 CPU 비용이 증가하고 절단으로 인한 색인의 크기가 증가하는 문제점이 있다. X-tree는 노드 분할 시에 노드간의 중복 비율을 고려하여, 허용치를 초과하는 분할을 허용하지 않음으로써 노드 간의 중복을 회피하는 정책을 사용한다. 즉, 중복이 심한 노드를 분할할 때는 분할을 수행하지 않고, 슈퍼 노드(Super node)를 사용하여 노드의 최대 허용 엔트리 수를 증가시킨다. 하지만 X-tree

는 슈퍼 노드로 인하여 노드간의 중복을 줄였지만, 4차원 이하의 색인에서 R-tree와 같은 검색 성능을 보인다.

기존의 클러스터링에 관한 연구는 정적인 데이터를 기준으로 수행되었다. 하지만 이동체의 위치 데이터는 삽입 연산을 수반하는 동적 데이터로서 cR-tree와 같이 클러스터링을 기반으로 하는 분할 정책을 사용하는 접근 방법이 적합하다. cR-tree[7]는 노드 분할 시 공간 근접성을 기반으로 하는 k-means 알고리즘을 사용하여 클러스터링을 하며, 이진 분할 및 다중 분할을 한다. 이 연구에서는 궤적 연결성을 기반으로 분할 노드의 엔트리들을 클러스터링하여 분할하는 정책을 제안한다.

3. 문제 정의

3D R-tree에서 복합 질의를 수행하기 위해 두 단계의 처리 과정이 필요하다. 첫 번째 단계는 특정한 영역에 대한 영역 질의(Range query)를 수행해야 하고, 두 번째 단계는 영역 질의 결과에 대한 이동체의 궤적을 검색해야 한다. 동일 궤적을 검색하기 위해서 이미 검색된 선분의 끝점을 이용해서 점 질의(Point query)를 수행해야 한다. 따라서 연결된 모든 궤적을 찾기 위해서 저장되어진 선분 수만큼의 점 질의를 해야 하고, 점 질의는 루트 노드에서 단말 노드로의 검색을 필요로 한다. 기존의 3D R-tree와 STR-tree에서는 이동체의 궤적에 대한 보전이 이루어지지 않기 때문에 궤적 질의에서 노드 방문 횟수가 많은 것이 문제점이다.

TB-tree는 복합 질의를 처리 하기 위한 방법으로써 단말 노드에 동일 궤적을 저장하기 때문에 궤적 질의에는 불필요한 노드 접근을 피할 수 있다. 하지만 동일 궤적만을 단말 노드에 저장을 하기 때문에 노드간의 중복(Overlap)이 심하여 영역 질의에서 노드 방문 횟수가 증가 한다.

이러한 문제점을 해결하기 위해서 영역 질의에서는 노드간의 중복(Overlap)과 사장 공간을 최소화하여야 한다. 또한 색인의 공간 활용도를 높이는 정책이 필요하다. 궤적 질의에서는 궤적을 보전해서 불필요한 노드의 방문을 줄여야 한다.

4. 단말 노드 분할 정책

4.1 공간 축 분할 정책

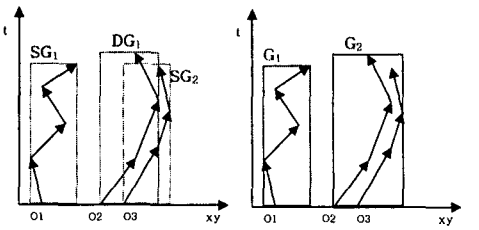
단말 노드의 공간 축 분할은 궤적을 보전하기 위한 분할 정책이다. 새로운 선분의 삽입에 따른 단말 노드의 오버플로우가 발생하면 두개의 노드로 분할을 한다. 분할 과정은 다음과 같이 3단계 과정으로 이루어진다. 1)동일 궤적 그룹화 과정, 2) 그룹들 중 시드 선택 과정, 3)나머지 그룹 분배 과정

첫 단계로 동일 궤적끼리 그룹을 먼저 생성한다. 즉, n을 단말 노드 내에 저장되어진 이동체의 수라고 가정한다면, n만큼의 그룹을 생성한다. 두 번째 단계로, 그룹들 중에 공간적으로 가장 멀리 떨어져 있는 두개의 그룹을 시드로 선택한다. 선택된 두개의 그룹을 시드그룹(SG : Seed Group)이라 정의한다. 세 번째 단계로, 시드그룹을 제외한 나머지 그룹을 분배그룹(DG : Distribution Group)이라고 정의하고, 그룹의 공간 근접성에 따라 각 시드그룹에 분배를 한다. 그림 1(a)는 첫 번째 단계로 같은 궤적끼리 그룹핑한 것이고, 그림 1(b)는 시드그룹에 분배그룹을 분배해서 두개의 노드로 분할 된 결과이다.

정의 1. 분할 노드간의 중복율

궤적 그룹화 과정에서 분할되어 생성되는 두개의 노드를 G_i, G_j 라 한다. 노드내의 궤적을 모두 포함하는 최소경계박스(MBB)를 그룹의 MBB라 하면, 분할 노드간의 중복율은 다음과 같다.

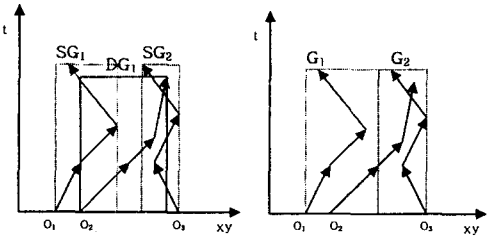
OverlapRate(G_i, G_j) ←
area(G_i.MBB ∩ G_j.MBB) / min(area(G_i.MBB),area(G_j.MBB))



(a) 궤적 그룹화 (b) 두 개의 노드로 분할된 결과

그림 1. 공간 축 분할

분배그룹(DG_i)을 각 시드그룹(SG)에 분배 할 때 분할 노드간의 중복율이 임계값을 넘으면, 분할 노드 간의 심한 중복이 발생한 것이다. 이러한 경우에는 심한 중복이 되지 않도록 분배그룹을 궤적의 구성 요소인 선분으로 분리하여 분배한다. 그림 2(a)는 분배그룹이 각 시드그룹과 심한 중복이 발생한 경우이고, 그림 2(b)는 궤적을 분리하여 중복이 최소화 되도록 두 개의 노드로 분할한 결과이다.

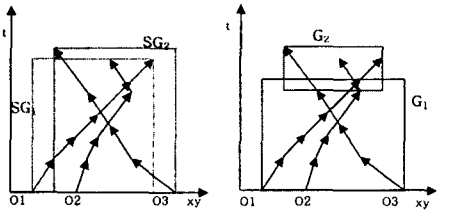


(a) 그룹간의 심한 중복의 예 (b) 궤적을 분리하여 분배하는 예

그림 2. 심한 중복 발생시 분배그룹의 궤적 분리

4.2 시간 축 분할 정책

공간 축 분할일 경우, 첫 번째 단계로서 동일 궤적끼리 그룹을 생성하고 다음 단계로서 시드그룹을 선택 한다. 시드그룹(SG)간에 심한 중복이 발생할 수 있다. 이러한 경우에는 분배그룹(DG)를 분배를 하더라도 심한 중복이 발생하므로 시간 축으로 분할한다.



(a) 시드그룹간의 심한 중복 발생 (b) UC 그룹의 예

그림 3. 시간 축 분할

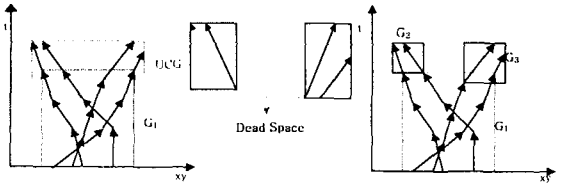
이동체가 가장 최근에 보고한 위치를 포함하는 선분을 UC 선분(UC line segment)이라 한다. 시간 축 분할은 그림 3(b)와 같이 UC선분만을 포함하는 UC그룹과 나머지 선분을 포함하는 그룹으로 분할한다. 이동체의 위치 정보는 시간 축으로 증가하는 특징을 가지므로 UC그룹의 엔트리 수는 노드의 최소 엔트리 수(m)보다 작은 경우를 허용한다.

정의 2. UC 그룹의 사장 공간율

시간 축 분할에서 UC선분을 포함하는 그룹을 UC그룹이라 할 때, UC그룹을 G라 하고, G의 엔트리들을 E_1,... E_n 라 하면, G의 사장 공간율은 다음과 같다.

DeadSpaceRate(G) ←
area(E_1.MBB ∪ E_2.MBB ∪··· ∪E_n.MBB) /area(G.MBB)

시간 축 분할 후에 UC그룹은 큰 사장 공간(Dead Space)을 가질 수 있다. 따라서, UC그룹에 두개 이상의 선분이 존재하고, 사장 공간율이 임계값을 넘으면, UC 그룹을 두개의 그룹으로 분할함으로써 사장 공간을 줄인다. 그림 4에서 시간 축 분할 후 UC그룹의 사장 공간율이 임계값을 넘어 재 분할한 것을 보여 주고 있다.



(a) 시간 축 분할 (b) UC그룹의 사장영역 (c) UC그룹 재 분할

그림 4. 시간 축 분할 후 공간 분할(다중 분할)

5. 비단말 노드 정책

5.1 비단말 노드의 자료구조

이동체의 궤적은 서로 다른 단말 노드에 저장 될 수 있다. 동일한 궤적을 저장하는 서로 단말 노드를 연결 노드(Connected Node)라 한다. 연결 노드에 대한 정보를 표현하기 위해 같은 부모를 가진 자식 노드 간의 궤적 연결성을 그림 5와 같이 CNInfo(Connected Node Information)에 저장한다. CNInfo는 자식 노드만큼의 비트가 필요하고, 하나의 선분이 삽입될 때 갱신된다. 단말 노드들의 궤적 연결성을 저장함으로써 비단말 노드 분할에서 궤적 클러스터링을 가능하게 한다. 궤적 질의에서 너비우선 탐색(Breadth first search)을 수행함으로써 루트 노드에서 단말 노드까지 반복 검색하는 것을 최소화하여 불필요한 노드 방문을 줄일 수 있다. 비단말 노드의 구조는 그림 5와 같다.

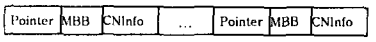


그림 5. 비단말 노드 자료 구조

그림 6에서 노드의 최대 엔트리 수가 3일 때, CNInfo의 비트 표기와 단말 노드 LN_1과 LN_2가 연결 노드로 표현되어 있는 것을 보여준다.

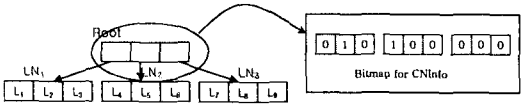
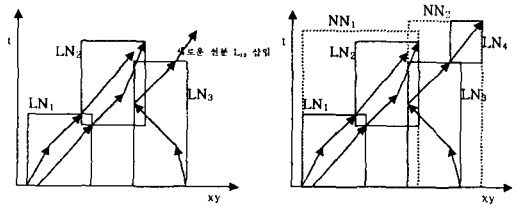


그림 6. 연결된 노드 정보(CNInfo)

5.2 비단말 노드 분할 정책

비단말 노드의 분할은 공간적으로 가장 멀리 떨어진 두개의 단말 노드를 시드로 선택하고, 시드가 되는 엔트리에서 CNInfo의 비트 중에 1인 엔트리들을 우선적으로 시드그룹에 분배한다. 분배 과정에서 엔트리가 두 시드그룹과 연결 노드인 경우에는 중복이 최소화 되는 시드그룹으로 분배한다. 이렇게 함으로써 단말 노드의 동일 궤적을 보전하는 효과를 볼 수 있다.



(a) 비단말 노드 분할 전 모습 (b) 비단말 노드 분할 후 모습

그림 7. 비단말 노드 분할의 예

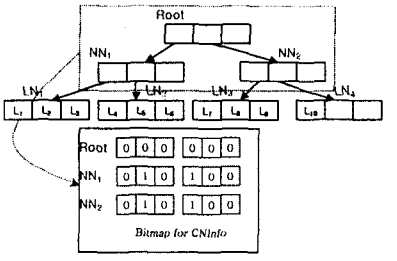


그림 8. 분할 후 노드 구조

그림 6에서 새로운 선분 L_{10}이 LN_3에 삽입되면 단말 노드가 분할 되고, 단말 노드 분할에 따라 비단말 노드도 분할 된다. 비단말 노드 분할 할 때 CNInfo에 저장된 단말 노드 연결성을 고려해서 분할 한다. 즉, 단말 노드 LN_1과 LN_2가 연결된 노드이다. 따라서 그림 7과 같이 루트 노드와 비단말 노드 NN_1과 NN_2가 생성된 결과이다.

노드의 페이지 크기가 1024 바이트이고, 노드의 헤더를 20 바이트로 가정한다. 3D R-tree는 최대 엔트리수는 (1024-20)/28 = 35이다. 비단말 노드에 부가 정보(CNInfo)를 추가할 경우에 엔트리당 비트 수는 최대 엔트리 수(M)만큼의 비트가 필요하다. 최대 엔트리수(M)는 (1024-20)/(28+⌈M/8⌉) = 31이다. 단말노드의 최대 엔트리 수는 3D R-tree와 동일하다.

6. 결론 및 향후 연구

이동체의 연속적인 움직임을 저장하는 이동체 색인에서의 문제점을 기술하고, 궤적 클러스터링 기반의 분할 기법을 제안하였다. 단말 노드 분할 정책으로 궤적 보전을 위한 공간 축 분할 정책과 공간 활용도를 높이는 시간 축 분할 정책을 제시하였다. 또한 시간 축 분할에서 사장 공간을 최소화하는 다중 분할 정책을 소개하였다. 비단말 노드 정책으로 연결 노드 정보를 이용하여 이동체의 궤적을 보전하기 위한 궤적 연결성을 기반으로 하는 정책을 제시 하였다.

향후 연구로서, 제한한 분할 정책들의 구현 및 실험을 통한 성능평가가 필요하다. 또한, 그룹간의 중복율과 그룹내의 사장 공간율을 실험을 통해 최적의 임계값을 정의할 필요성이 있다.

7. 참고문헌

[1] Yannis Theodoridis, Michael Vazirgiannis, Timos Sellis, " Spatio-Temporal Indexing for Large Multimedia Applications" , In International Conf, on Multimedia Computing and Systems, pp. 441-448, 1996

[2] Dieter Pfoser, Christian S. Jensen, Yannis Theodoridis, " Novel Approaches to the Indexing of Moving Objects." , In Proc. of the 26th VLDB Conf, pp. 395-406, 2000.

[3] Antonm Guttman, " R-Trees: A Dynamic Index Structure for Spatial Searching" , In Proc. of ACM-SIGMOD Conf. on the Management of Data, pp. 47-57, 1984.

[4] N. Beckmann and H. P. Kriegel, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", In Proc. ACM SIGMOD, p332-331, 1990.

[5] T. K. Sellis, N. Roussopoulos and C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects", Proceedings of the 13th VLDB Conference, p507-518, 1987.

[6] S. Berchtold, D. A. Keim, H. P. Kriegel, " The X-tree : An Index Structure for High-Dimensional Data" , International Conference on Very Large Data Bases, p28-39, 1996.

[7] S. Brakatsoulas, D. Pfoser, and Y. Theodoridis. Revisiting R-tree construction principles. Technical report, Computer Technology Institute, Patras, Greece, 2002. http://dias.cti.gr/~pfoser/clustering.pdf.