시공간 이동객체들을 위한 효율적인 색인 기법

강소영°, 양성봉
{milkyway, yang}@cs.yonsei.ac.kr
연세대학교 컴퓨터과학과

An Efficient Indexing Technique for Time-evolving Moving Spatial Data

Soyoung Kang°, Sung-Bong Yang
Department of Computer Science, Yonsei University

요 약

무선 네트워크의 기술 발달과 이동기기들의 많은 보급에 따라 GIS, LBS 등의 위치기반 서비스 등이 널리 사용되고 있다. 이러한 서비스를 수행하기위하여 방대한 양의 이동 객체들의 효율적인 관리, 검색에 대한 연구가 많이 이루어지고 있으며 이동 객체들의 특성과 사용목적에 따라 다양한 연구결과가 제시되고 있다. 본 논문에서는 시간에 따라 다양한 속도와 방향으로 이동하는 다차원 이동 공간 데이터를 위한 효율적인 색인 기법을 제시한다. 제안된 방법은 KDB-Tree와 QSF-Tree에 기반을 두고 있으며 색인 구조의 검색 성능을 향상시키기 위한 분할, 갱신 방법을 제시하여 시간에 따라 현재 속도와 방향이 다양하게 변화하는 데이터를 신속히 데이터를 색인하고 영역 질의 뿐만 아니라 미래 예측 질의에 대하여 현재 보유한 데이터 정보를 기반으로 미래 예측 결과를 신속히 산출할 수 있도록 한다.

1. 서 론

무선인터넷 기술의 발달과 휴대폰이나 PDA같은 무선 이동기기의 보급과 더불어 이를 이용한 다양한 위치 기반 서비스 사용이 활성화 되고 있다. 이에 따라 이들 서비스를 위한 공간 이동객체 데이터를 관리하는 방법은 여러 응용 서비스에서 빼 놓을 수 없는 중요한 역할을 하고 있다. 시간에 따라 이동하는 공간 이동객체의 위치정보는 그 크기가 매우 방대하다. 시시각각 변하는 데이터 정보를 신속히 갱신하고 이들 데이터를 검색하는 방법은 이러한 서비스를 제공하는데 매우 중요한 역할을 하므로 다양한 시공간 색인 방법들이 연구되고 있다.[4,13]

이동객체에 대한 질의는 위치 영역에 포함되는 이동객체에 관한 영역 질의와 과거로부터 현재 혹은 현재부터 가까운 미래 시간까지 등의 시간 범위에 관련된 이동객체를 찾는 시간관련 질의로 나눌 수 있다. 기존의 위치 영역 질의 결과 산출과 달리 시간관련 질의일 경우 미래 시간이 포함된 질의 결과를 위해서는 지금까지 위치정보를 기반으로 해서 미래 예측이 가능한 질의 결과를 산출할 수 있어야 한다.

본 논문에서는 시간에 따라 위치정보와 속도가 변화하는 공간 이동객체를 신속히 색인하고 미래 예측 질의에 대하여 빠르게 결과를 구할 수 있는 QSF-Tree를 기반으로 하는 색인기법을 실험한다. 이 방법은 고정된 크기를 가진 이동객체 뿐만 아니라 크기가 변화하는 이동객체에 대해서도 기존의 TPR-Tree 기반의 데이터구조보다 훨씬 우수한 색인기능을 보여주리라 생각된다.

2. 관련연구

기존의 공간데이터를 색인하는 방법은 데이터의 형태에 따라 포인트 객체를 다루는 PAM(point access method)과 여러 포인트로 이루어진 공간데이터를 다루는 SAM(spatial access method)로 분류할 수 있다.

다차원 포인트 데이터의 경우 데이터의 색인을 위한 영역 분할시 오버랩(overlap) 영역이 생기지 않으며 영역을 분할하는 방식에 따라 차원이 증가 할수록 데이터의 분산도에 따라서 빈 페이지(page)가 증가하거나 트리의 높이가 증가하여 그 성능을 떨어뜨릴 수 있다. 공간데이터(spatial data)의 경우 가장 대표적인 방법으로 R-tree계열을 들 수 있으며 데이터가 면적을 가지고 있는 특성상 이들 데이터를 어떻게 효율적으로 영역의 오버랩을 줄이며 이들을 그룹지어 색인하는 여러 방법들이 제시되었다.[1,2,3,5,14,15]

2.1 KDB-Tree

대표적인 PAM방법으로 KD-Tree[6], KDB-Tree[8] GRID-File[7], QUAD-Tree[9]등이 있다. 이중 KDB-Tree는 KD-Tree와 B-Tree가 결합된 형태로 높이 균형 트리이다. 데이터의 분포에 따라 한 축을 기준으로 공간을 분할하며 색인하는 기법으로서 다차원 데이터 색인에 좋은 성능을 보인다.

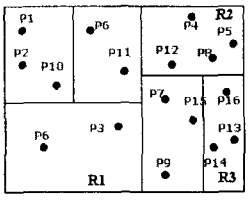

그림 1 KDB-Tree

포인트 페이지와 영역 페이지를 정의하여, 포인트 페이지에는 실제 데이터의 정보를 저장하고, 영역 페이지는 하위 영역에 대한 정보와 하위 정보를 가리키는 포인터를 저장하며 분할하는 축을 선택하는 분할 정책에 따라 색인 결과가 달라질 수 있다.

2.2 QSF-Tree

QSF-Tree[10,11,12]는 KDB-Tree를 근간으로 하는 공간데이터 색인기법이다. 색인구조 생성 시 공간데이터의 각 축별 최소값만을 기준으로 생성하기 때문에 색인 생성시간이 MBR(minimum bounding rectangle)을 고려해야 하는 R*-Tree[3]에 비해 빠르다. 본 논문에서는 공간데이터를 포인트 데이터 색인기법에 적용한 이유는 R-Tree 기반의 색인구조는 MBR들의 오버랩으로 인하여 탐색시 여러 검색경로가 생성되며 차원이 높아질수록 이러한 문제는 더욱더 심각해진다. 또한 데이터의 추가와 삭제 시 이를 묶는 MBR을 계산하는데 많은 비용이 소비되므로 영역 페이지에 오버랩이 발생하지 않는 포인트 색인기법을 사용한 이유이다.

실제 데이터정보는 KDB-Tree와 마찬가지로 포인트 노드에서 저장하며 영역노드에 하위 영역노드 정보와 포함하고 있는 객체들 중 축별 가장 긴 객체 값 M_i 와 작은 값 m_i 를 저장한다. 이는 검색 시 검색 윈도우와 객체의 관계를 계산할 때 검색성능을 향상시키는데 이용된다. 다양한 차원과 크기를 가진 공간데이터를 R*-Tree와 비교해 실험한 결과 많은 향상을 보인다.

2.3 R-Tree

R-Tree[1]는 B-Tree와 유사한 인덱스 레코드들로 구성되는 높이 균형 트리로서 가장 대표적인 다차원 공간데이터 색인 방법으로서 MBR로 객체들을 표현한다. 객체 자체를 저장하지 않고, 객체를 둘러싸는 MBR과 객체를 가리키는 포인터를 저장한다.

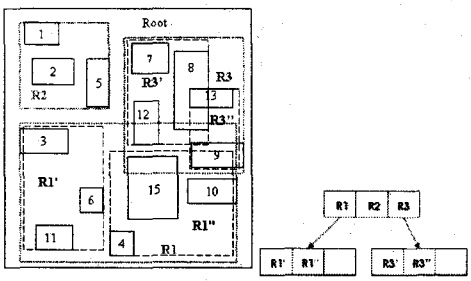

그림 2. R-Tree의 예

특징

① 각 노드는 지정한 갯수 사이의 객체를 가진다.

② 단말노드(leaf node)에는 객체 자체의 MBR과 객체를 가리키는 포인터를 가진다. 비단말노드들은 자식노드들을 둘러싸는 MBR과 자식노드를 가리키는 포인터를 가진다. 루트(root)는 최소 2개의 하위 노드를 가질 수 있다.

③ 트리의 높이는 ($\log_m N$)-1으로 모든 단말노드들은 같은 높이를 가진다.

④ 부모노드는 모든 자손노드들을 포함한다. 같은 높이에 있는 형제노드 간에는 오버랩이 존재 할 수 있다.

이 후 R-Tree의 오버랩 단점을 극복하기 위해 오버랩을 없앤 R+-Tree[2]가 있으며 R-Tree의 여백과 오버랩 현상을 최소화하기 위한 알고리즘이 첨가 된 R*-Tree[3]가 있다. 대부분 R-Tree를 기반한 인덱스구조인 경우 R*-Tree의 정책을 따르며 이러한 트리에는 TV-Tree[5], X-Tree[14], TPR*-Tree[15] 등이 있다.

3. 연구내용

많은 공간객체를 다루는 연구를 보면 R-Tree를 그 근간으로 하는 것이 많다. 하지만 위에 언급했듯이 R-Tree를 생성하는데 있어 야기되는 노드 간의 오버랩은 검색 시 여러 경로를 생성해야하는 단점이 있다. 이동 데이터를 대상으로 할 때 시간에 따라 변화하는 위치정보를 갱신하는데 많은 삽입과 삭제가 발생하는 관계로 MBR 정보를 갱신하는 데에는 많은 갱신 비용이 발생되며, 이동객체의 움직이는 영역이 크면 더욱 그러하다. 본 논문에서는 계속해서 들어오는 객체 정보를 신속하게 색인하고 검색 성능을 향상하기 위하여 PAM 방식인 KDB-Tree와 QSF-Tree를 채택하여 공간 이동객체를 색인하는 기법을 제시한다.

3.1 트리의 생성

QSF-Tree에서와 같이 공간데이터의 각 축별 최소값을 기준으로 트리를 생성한다. 여기서 생성하는 기본 방법은 KDB-Tree와 같다.

① 객체의 각 축별 최소값을 기준으로 트리를 생성하고 영역을 분할한다. 이때 포함하는 객체의 개수는 최대 M이다.

② 영역 페이지나 포인트 페이지의 자식의 수가 M개가 넘으면 분할한다.

이때 영역페이지의 분할은 KDB-Tree에서 제시된 FS(forced splittig)방식과 FD(first division splitting)방식이 있으나 여러 논문에서 증명된 것과 같이 연속 분할을 방지하는 FD 방식을 따르도록 한다.[8,16]

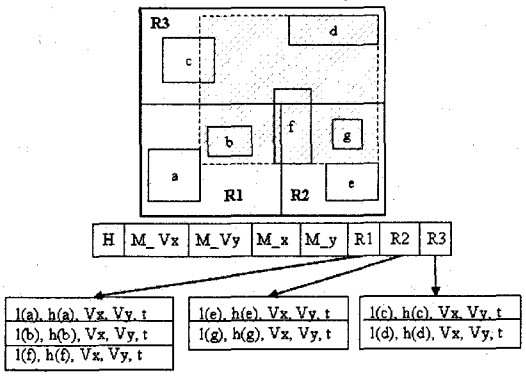

그림 3 QSF-Tree를 기반한 이동객체 데이터구조

③ 영역노드에서 부모노드는 자식노드의 영역정보와 id를 보유하며, 포함하고 있는 객체 중 최대 최소 vector 값 정보인 V_{max}, V_{min}과 축별 최장 길이값 M_i와 최단 길이값 m_i를 가지고 있다. 이 정보는 트리 생성 후 미래 예측 질의 및 데이터 관계 질의 시 검색범위 설정과 객체간의 혹은 객체와 검색 창간의 관계 검색 시 사용된다.

④ 포인트 노드에는 실제 데이터 정보 및 갱신 시점 시간이 기록되며, 포함하는 객체의 각 축별 상한값을 포함하는 영역의 정보 H를 기록한다.

3.1.1 데이터 삽입 시 분할정책

데이터를 추가하거나 정보 변경으로 인한 갱신 시 데이터의 이동에 따른 위치 변경에 따라 다른 영역노드로 이동될 수 있다. 이때 객체의 벡터값이 변경되었거나 이동된 영역노드에서의 최대 최소 vector값과 축별 최장 길이값을 비교하여 갱신한다.

새로운 데이터의 삽입으로 인하여 포인트 노드에 처음 오버플로우가 발생하는 경우 무조건 노드를 분할하는 것이 아니라 현재 노드의 형제노드(분할되기 전 같은 노드에 있었던)를 검색하여 포함하는 자식, 객체의 수가 최대 갯수의 2/3이면 형제노드를 합친 후 다시 영역을 재분할하도록 한다.

분할하는 축을 선택하는 방법은 객체의 각 축의 최대값을 포함하는 영역이 되도록 현재 객체를 포함하는 영역을 많이 벗어나지 않는 쪽을 택한다.

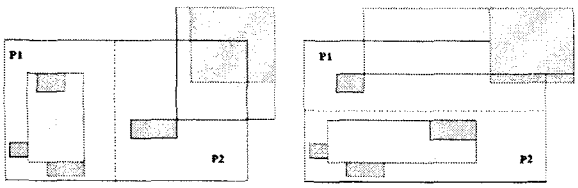

그림 4. 페이지 노드의 분할

3.1.2 데이터 삭제

객체의 이동으로 현재 노드 영역을 벗어나거나 삭제되어 언더플로우가 발생되는 경우에는 형제노드와 합친다. 이때 오버플로우가 발생하는 경우 재분할을 시도한다. 이 방법은 KDB-Tree와 같다.

3.1.3 공간객체의 관계

공간데이터의 특성상 개체간의 위치에 따라서 다른 개체와의 관계를 정의할 수 있다. 객체가 만나는 형태에 따라 *오버랩*, *포함*(cover), *일치*(equal), *불일치*(disjoint)로 나뉠 수 있다. 객체간의 관계는 QSF-Tree에서와 마찬가지로 객체의 축별 하한값과 영역노드에 저장하는 객체 중 최장, 최단 길이값, 최대 최소 벡터값 정보, 객체의 상한값을 모두 포함하는 MBR 영역정보를 바탕으로 쉽게 객체간의 관계를 정의할 수 있다.

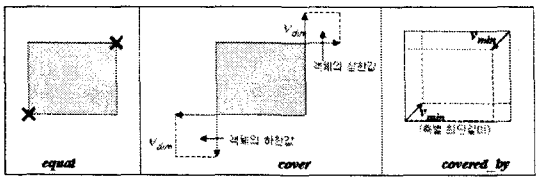

그림 5. 객체간의 관계

여기서 V_{dim}은 영역노드의 객체 최장 길이값에서 질의영역 q의 축별 길이값을 뺀 $M_i - q_i$ 이며 v_{min}은 영역노드의 축별 최단 길이값이다. 객체의 축별 하한값과 상한값이 질의영역의 어떤 부분에 있는가에 따라 쉽게 이를 계산할 수 있다.

3.2 데이터 검색

검색질의의 종류를 검색내용과 목적에 따라 3가지로 분류한다.

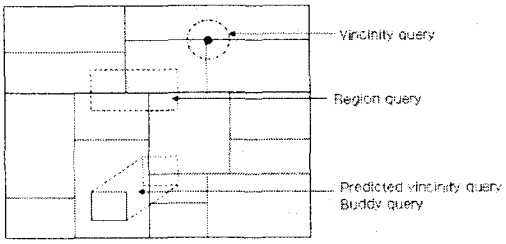

그림 6. 검색 종류에 따른 질의영역

3.2.1 영역 검색(region query)

지정한 영역범위에 연관되는 데이터를 검색하는 방법이다. 질의영역 R에 겹쳐지는 객체를 검색하는 방법으로 질의영역 R이 어떤 노드에 겹쳐지는지 대상 노드를 선택한다.

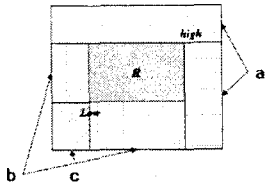

그림 7. 질의영역 R과 disjoint 조건

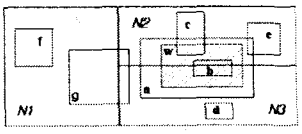

그림 8. 질의영역과 객체의 관계

그림 7을 보면 검색 대상에서 제외되는 노드를 판별할 수 있다.

① 대상 노드 검색 후 노드영역에 있는 객체들 중 질의영역의 각 축을 비교하여 질의영역 축 별 상한값 보다 큰 하한값을 가지는 객체는 *불일치관계*: 영역 a

② 객체의 각 축 하한값이 질의영역의 축 하한값 보다 같거나 큰 경우 객체의 다른 축의 상한값이 질의영역의 하한값 보다 작으면 이는 *불일치관계*: 영역 b

③ 객체의 각 축의 하한값이 질의영역의 하한값 보다 작은 경우 그 축의 상한값이 질의영역의 하한값 보다 작거나, 같거나 큰 경우 다른 축의 상한값이 질의영역의 하한값보다 작으면 *불일치관계*: 영역 c

이 외의 경우에는 모두 겹침(일치, 포함)관계라 볼 수 있으며 그림 8에서 보여주는 다양한 질의영역과 객체의 관계는 그림 5에서처럼 두 객체의 상한값과 하한값을 비교하여 다른 객체의 상한값과 하한값에 완전히 포함하는 경우 *포함관계*이고 이외에 각 축의 끝값이 질의영역내에 존재 하는 경우 *겹침관계*, 상한값과 하한값이 일치하는 경우 *일치관계*라 할 수 있다.

3.2.2 근접 검색 (vicinity query)

근접 검색은 질의가 있는 시점에서 기존의 객체정보를 현재 시점으로 갱신한 후 현재 위치에 맞추어 주변 객체와의 관계에 따라 결과를 얻는다.

3.2.3 미래 예측 근접 질의(predicted vicinity query)

현재 시점부터 정해진 period 시간동안 이동객체가 움직이는 방향과 속도에 따라서 검색되는 객체를 예측한다. 이동객체의 현재 위치와 period 경과 후의 이동객체 위치를 예측하여 객체의 이동범위 안에 포함되는 각 단위 시간별 해당 객체들을 검색하도록 한다.

period가 증가함에 따라서 축별 벡터값과 period를 계산하여 얻어진 이동객체 영역과 변경된 위치정보를 갖게 되는 다른 객체와의 관계를 추측한다. 이 경우 period 시간이 길면 길수록 중간에 이동객체의 방향 값이나 속도값이 현재와 달리 변화 될 수 있으므로 실제 결과와 예측 결과가 달라지는 폭이 커질 수 있다.

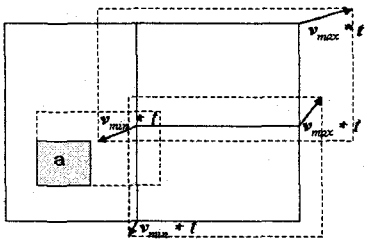

그림 9. 미래 예측 검색 대상 노드 설정

지정한 period에 따라 확장된 질의영역과 v_{min}, v_{max}와 미래 시간만큼 확장된 주변 노드영역을 계산하여 오버랩이 발생하는 노드를 대상으로 위 절에서 언급한 객체 관계에서 정의한 방법으로 해당 노드에 포함된 객체 중 미래에 만날 수 있는 다른 이동객체 대상을 검색할 수 있다.

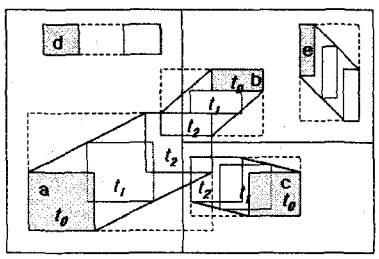

그림 10. 간변화에 따른 객체의 이동

그림 10에서 볼 수 있듯이 객체 a가 2 단위시간 동안 움직이는 동안 시점 1에는 다른 객체와 만나지 않으나 2인 시점에는 객체 b와 c를 만나게 된다. 이와 같이 공간객체의 하나의 포인트 값을 기준으로 색인하는 경우 같은 영역내의 이동은 영역노드의 상한값 정보에만 영향을 줄뿐 영역 재설정을 위한 갱신 비용이 들지 않으며 정적인 공간데이터를 대상으로 한 성능 비교에서 보이듯이 다른 영역으로 이동시에도 삽입 삭제가 R*-tree에 비해 신속히 이루어질 수 있다.

4. 실험결과

성능평가를 위하여 펜티엄 IV-300 프로세서에 512 MByte의 메모리를 가진 윈도우 XP 운영체제 실험 환경에서 수행되었다.

조건	값
영역의 크기	0 ~ 100000
데이터의 갯수	10000 ~ 100000
갱신시간	0 ~ 10
갱신 데이터 갯수	시간당 데이터 갯수의 20%
데이터 최대/최소 이동속도	100/-100
데이터 최대 크기	1%, 3%, 10%
질의영역 크기	0.1 ~ 5
노드의 크기	4Kb

표 1. 실험 공간객체 생성 조건

위와 같은 조건으로 다양한 크기의 이동객체를 생성하였으며 이들 객체마다 이동 벡터값을 지정하고 갱신 시간마다 벡터값 정보를 변경해 주었다.

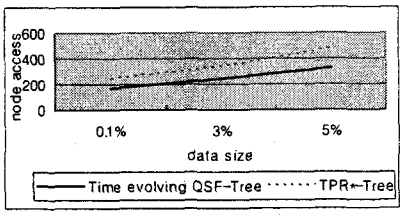

그림 11. 데이터 삽입 노드 검색수

그림 11의 그래프에 나타나듯이 데이터의 삽입시 기존의 TPR*-Tree에 비해 MBR 정보 비용 등이 없으므로 간단한 삽입 방법인 Time-evloving QSF-Tree가 노드 검색수가 적어 색인시간이 빠른 것을 알 수 있다.

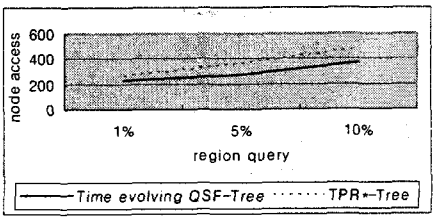

그림 12. 영역 질의 노드 검색수

그림 12는 다양한 사이즈의 영역 질의시 검색되는 노드의 수를 나타낸 것이다. 검색 영역의 사이즈가 커질수록 검색 노드 대상이 훨씬 증가한다. 또한 대상 객체의 차원이 높아질수록 위의 실험 결과의 폭은 더욱 커질 것이라 생각된다.

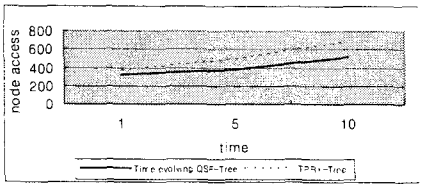

그림 13. 미래 예측 근접 질의 노드 검색수

그림 13은 미래예측 질의를 성능 평가한 것으로 시간값이 커질수록 질의영역과 주위 노드의 영역이 크게 증가하므로 많은 노드 검색이 요구된다.

5. 결론 및 향후 연구

현재 이동 공간데이터의 색인에 대한 연구는 계속 이어져 오고 있다. 하지만 시공간 질의나 궤적 질의, 빠른 색인 생성 등 모든 경우를 효율적으로 지원할 수 있는 색인기법은 거의 없다. 많은 사용자와 더욱 다양한 이동기기 서비스 증가가 예상되는 미래에 더 많은 형태의 다양한 데이터들을 대상으로 하는 서비스가 예상된다.

지금까지 대부분의 연구에서 실험 한 공간데이터는 고정된 크기를 가지고 있다. 향후에는 이런 데이터의 크기가 시간에 따라 변하는 shrink spatial data를 대상으로 연구하고자 한다. 크기가 변하는 데이터는 일정한 패턴으로 움직이거나 일정 반경 내에 움직이는 데이터의 그룹이나 크기가 변하는 이미지 등을 표현하고 색인하는 예가 될 것이다. 이러한 그룹 성격을 이용하여 갱신하고자 하는 많은 데이터의 갱신 횟수를 줄여 빠른 색인이 될 것이며, 이는 다양한 데이터나 응용프로그램에서 활용될 수 있을 것이다.

6. 참고문헌

[1] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching," *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pp. 47-54, 1984.

[2] Sellis, T., Roussopoulos, N., & Faloutsos, C., "The R+-tree : A Dynamic Index for Multidimensional Objects," *Proceedings of the 13th Int'l Conference on VLDB*, pp. 507-518, 1987.

[3] Beckmann, N., Kriegel, H., Schneider, R., & Seeger, B., "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, pp. 322-331, 1990.

[4] Gaede, V. & Gunther, O., "Multidimensional Access Methods," *ACM Computing Surveys*, vol. 30, no. 2, pp. 170-23, 1998.

[5] Lin, K., Jagadish, H., & Faloutsos, C., "The TV-tree: An Index Structure for High-dimensional Data," *VLDB Journal*, vol. 3, pp. 517-542, 1995.

[6] L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, pp. 509-517, 1975.

[7] Nievergelt, J. et al., "The Grid File: An Adaptable, Symmetric Multikey File Structure," *ACM Transactions on Database Systems*, 1984.

[8] Robinson, J. T., "The KDB-Tree: A Search Structure for Large Multidimensional Dynamic Indexes," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 10-18, 1981.

[9] Tayeb, J., Ulusoy, O., Wolfson, O., "A Quadtree-Based Dynamic Attribute Indexing Method," *The Computer Journal*, vol. 41, no.3, pp. 185-200, 1998.

[10] Yu, B., Orlandic, R., & Evens, M., "Simple QSF-trees: An Efficient and Scalable Spatial Access Method," *Proceedings of the 8th Int'l Conference on Information and Knowledge Management*, pp. 5-14, 1999.

[11] Orlandic, R. & Yu, B., "Implementing KDB-trees to Support High-dimensional Data," *Proceedings of the Int'l Database Engineering and Applications Symposium*, pp. 58-67. 2001.

[12] Orlandic, R, Yu, B., "Scalable QSF-Trees: Retrieving Regional Objects in High-Dimensional Spaces," *J. of Database Management*, vol. 15, 2004.

[13] P. K. Agarwal, L. Arge, and J. Erickson., "Indexing Moving Points," *Proceedings of the ACM Symposium. on Principles of Database Systems*, 2000.

[14] Berchtold, S., Keim, D.A., & Kriegel, H., "The X-tree: An Index Structure for High-dimensional Data," *Proceedings of the 22th Int'l Conference on VLDB*, pp. 28-39, 1996.

[15]Tao, Y., Papadias, D., "The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries," *Proceedings of the 2zth VLDB Conference*, 2003.

[16] Orlandic, R. & Yu, B., "Implementing KDB-Trees to Support High-Dimensional Data" *IDEAS* pp.58-67, 2001