제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

이동 객체의 궤적에 대한 최근접 탐색 기법†

최보윤, 신현호, 지정희, 김상호, 류근호
충북대학교 데이터베이스 연구실
e-mail : {bychoi, shkim, khryu} @dblab.chungbuk.ac.kr

The Method of Nearest Neighbor Search for Trajectory of Moving Objects

Bo Yoon Choi, Hyun Ho Shin, Jeong Hee Chi, Sang Ho Kim, Keun Ho Ryu
Database Laboratory, Chungbuk National University

요 약

이 논문은 질의와 검색 대상 객체가 모두 이동 객체인 경우, 즉 3 차원 폴리라인(polyline) 형태의 경로를 가지는 객체들 간의 연속(continuous) 최근접 질의 처리에 유용한 기법을 제안한다. 질의 경로를 따라 객체를 탐색해가면서 질의에 대한 최근접 정보가 변하는 시점을 찾는 것이 목적인 연속 최근접 질의 처리는 전체 질의 경로에 올바른 최근접 정보 리스트를 제공하지만, 기존의 방법들은 검색 대상 객체가 동적인 경우에 적용되기에는 시간에 따라 움직이는 객체의 위치변화를 처리하지 못하고, 질의 시점과 대상 객체간의 시점을 연관시키기 어렵다는 문제점들을 가지고 있다.

따라서 이 논문에서는 데이터 객체들의 궤적 정보는 STR 트리로 유지하고, 질의 경로 세그먼트와 질의의 시간 인터벌에 포함되는 데이터 객체 세그먼트 모두에 대해 추출시간(sampling time) 선택, 스윕라인(sweep line) 적용, 위치 추정 함수 이용 등의 단계를 처리함으로써, 이 문제를 해결하고 질의 경로 전체에 정확한 최근접 객체 정보 리스트를 제공한다. 제안된 기법은 물류정보시스템, 국방정보시스템, 기상, 교통 등 시공간 이동 객체의 질의를 다루는 시스템에 적용할 수 있다.

1. 서론

효율적인 최근접 탐색 기법의 구현은 상용 어플리케이션에서 자주 연구되는 분야 중 하나이다. 어떤 사용자가 응용 프로그램 상에서 객체나 위치를 선택하고 그 위치에 가장 가까운 객체를 찾고자 한다. 질의 객체와 가장 가까운 거리를 가지는 객체를 결과로 반환하는 것이 최근접 질의이다.

지금까지 연구되어 온 최근접 탐색 기법들은 대부분 공간 데이터베이스에 대한 것이었다[1,2,3]. 이런 공간 데이터베이스 상에서의 최근접 탐색 기법들을 단순히 시공간 데이터베이스 상에서의 문제로 확장시키면, 시공간 객체의 특성으로 인한 많은 문제점들이 발생한다. 시공간 객체는 시간의 흐름에 따라 위치나 크기가 변화하는 이동 객체(moving object)이므로, 이에 대한 최근접 질의를 처리하기 위해서는 시간이 지나면서 변경된 최근접 정보를 갱신해야 하고, 객체의 이동 경로 정보를 유지해야만 한다. 지리적인 용어로 객체의 움직임을 궤적이라 하며, 이 논문에서는 객체의 이동 경로를 궤적으로 표현한다. 또한 이동 객체는 위치만 변하는 객체로 가정한다. 이동 객체는 시간의 흐름에 따라 움직이며 위치가 변하기 때문에 궤적은 3 차원 공간상의 폴리라인으로 표현된다.

대부분의 시공간 최근접 질의 처리 기법들은 특정 타임스탬프에서 공간적으로 질의와 가장 가까운 객체를 찾는다[4,5]. 이를 순간(instantaneous) 최근접 질의라 한다. 이런 정보는 그 시점에서만 유효하고, 계산되지 않은 시점에서 이 정보의 정확성은 보장될 수 없다. 연속 최근접 탐색[6,7]은 어떤 타임스탬프에서 결과를 계산하는 것이 아니라, 질의 궤적을 따라 객체를 탐색해 나가면서 질의에 대한 최근접 정보가 변하는 시점, 즉 분할점(split point) s_i 를 찾아내는 것이 목적이다. s_i 부터 s_{i+1} 까지의 최근접 객체가 a 라는 정보는 <a, $[s_i,s_{i+1}]$> 형태로 리스트에 저장되고, 질의 궤적의 마지막 시점에 도달하면 리스트에 쌓여있는 모든 정보가

† 이 논문은 2002 년도 한국학술진흥재단의 지원(KRF-2002-072-AM1013)에 의하여 연구되었음

반환된다. 따라서 질의 궤적 전체에 대한 모든 시점에서의 최근접 정보를 알 수 있고, 이 정보에 대한 정확성을 보장하게 된다.

하지만 질의 객체와 검색 대상 객체가 모두 이동 객체인 경우, 기존의 방법들을 단순히 3 차원으로 확장하여 적용하면 시간에 따라 객체의 위치가 변했을 때 분할점을 찾기 위한 후보 데이터가 어떤 것인지에 대한 기준이 명확하지 않으며, 질의 시점과 대상 객체 간의 시점을 연관시키는 작업이 어려워진다.

따라서 이 논문에서는 질의 객체와 탐색 객체들이 모두 동적인 경우에 유용한 새로운 최근접 질의 처리 기법을 제안한다. 특히 질의 객체와 데이터 객체의 궤적이 모두 3 차원상의 세그먼트 집합으로 표현되는 경우에 유용하다. 질의 세그먼트와 데이터 객체 세그먼트간의 선형 방정식 계산이나 제한된 인터벌 내에서의 스윕라인 적용, 선분에 대한 대칭 이동 등을 사용하여 질의를 처리한다. 또한 이동 객체들의 궤적 정보는 STR 트리에 유지한다. 논문은 다음과 같이 구성된다. 2 장에서는 이동 객체에 대한 기존의 연속 최근접 질의 처리 방법을 살펴보고, 3 장에서는 기존 연구를 3 차원 공간상으로 확장 시켰을 때, 즉 질의와 대상 객체가 모두 동적인 경우로 적용시켰을 때 발생하는 문제점들을 정의한다. 4 장에서는 이 논문이 제시하는 새로운 최근접 질의 처리 기법을 설명하며 5 장에서 결론 및 향후 연구를 보인다.

2. 관련연구

Yufei Tao[6]는 TP(time-parameterized) 최근접 질의를 제안하였다. 여기서는 검색 대상 객체들의 작용시간(influence time)을 거리 측정에 사용한다. 작용시간이란, 데이터 객체가 질의 객체에 가까워지기 시작하는 시간이다. 최소 작용시간 t 를 가지는 객체는 시간 t 에서 최근접 정보 결과를 변경시킬지도 모르는 후보객체가 된다. 하지만 각 작용시간마다 분할점의 개수만큼 객체들 간의 거리를 계산하기 때문에 심각한 오버헤드를 초래한다.

그는 또한 계산적인 오버헤드를 줄일 수 있는 방법도 제안하였다[7]. s 에서 e 까지의 이동 경로를 가지는 질의 q=[s,e]가 주어졌을 때, 먼저 s 시점에서 모든 데이터 객체들에 대한 거리를 계산하고, 순서대로 정렬한 후 최근접 객체를 찾는다. 발견된 최근접 객체를 a 라고 하면, a 에 대한 정보는 리스트에 {<a, [s,e]>}로 저장된다. 이것은 a 가 s 에서 e 시점까지의 최근접 정보라는 것을 나타낸다. 맨 처음 시점에서 a 를 찾은 후, |s,a| (s 에서 a 까지의 거리)와 |e,a|를 반지름으로 하고 s 와 e 를 각각 중심으로 가지는 근접원(vicinity circle)을 각각 그린다. 맨 처음 시점에서 정렬되었던 객체 순서 중, 두 번째로 가까운 객체가 다음 탐색의 후보 객체가 되고, 이것이 근접원에 포함되지 않으면 고려하지 않는다. 근접원에 포함된 다음 번 최근접 객체가 c 라면, a 와 c 점 사이의 선분에 대한 수직 이등분선을 긋고, 이 선분과 질의 q 의 궤적이 만나는 시점을 분할점 s_1 로 선택한다. 과정은 질의 궤적의 마지막 시점 e 까지 반복되며, 마지막 시점에서 리스트 {<a, [s, s_1], <c, [s_1,e], ...}를 결과로 반환한다. 따라서 질의 궤적 전체에 대한 최근접 객체 정보가 유지된다.

하지만 이 기법은 데이터 객체가 동적인 경우에 대한 평가가 이루어지지 않았다. 질의 객체와 데이터 객체가 모두 동적인 경우에 이를 적용하면, 시간의 흐름에 따른 객체들의 위치 변경을 고려해야 하므로 최근접 객체 정보가 그 시점에서 유효한지를 판단하는 작업에서 몇 가지 문제점들이 발생한다. 다음 장에서는 기존 방법을 이동 객체에 적용시켰을 때 나타날 수 있는 문제점들을 정의해본다.

3. 문제 정의

이 논문에서 다루고자 하는 주요 질의는, 질의 궤적과 가장 가까운 위치에서 움직이고 있는 객체를, 전체 질의 궤적에 대해 어떤 시점에서도 정확성이 높게 보장될 수 있도록 연속적으로 찾는 것이다. 위의 문제를 해결하기 위해서 기존의 연속 최근접 질의 처리 방식을 단순히 3 차원으로 확장하여 사용하면 다음과 문제점들이 발생할 수 있다.

기존 방법에서는, 시작 시점에서 전체 데이터 객체들이 이 점과의 거리 순서대로 정렬되고 그 중 가장 가까운 거리를 가지는 객체가 첫 시점과 마지막 시점 사이의 최근접 객체로 초기 설정된다. 맨 처음에 정렬되었던 순서에 따라 탐색 후보 키로 설정되면서, 그때마다 근접원에 포함되었는지를 판별하여, 포함된다면 최근접 객체라 결정하고 분할점 계산을 진행한다. 하지만, 데이터 객체가 시간의 흐름에 따라 움직이는 경우, 시작 시점에서 정렬되었던 객체들의 순서 정보는 더 이상 참이 되지 않는다. 그리고 기존의 객체 점들이 3 차원 폴리라인으로 바뀌기 때문에 질의 시점과 대상 객체 시점간의 연관성 판단 여부가 어려워진다.

최근접 객체를 찾고, 분할점을 설정할 수 있었다 하더라도, 이 객체가 시간이 흐르면서 질의 궤적에서 멀어지는 위치로 움직이는지, 계속 가까운 위치에 머물러 있는지에 대한 판별이 어렵다. 즉, 최근접 객체 정보에 대한 참을 보장할 수 없다.

객체들이 3 차원 폴리라인으로 표현되기 때문에, 근접원은 3 차원 공간 상의 근접구(vicinity sphere)로 확장된다. 구의 계산은 오버헤드를 발생시킬 수 있다.

위의 여러 불확실한 문제들을 해결하고 정확도가 높은 최근접 질의 처리를 하기 위해서는 3 차원 공간 상에서의 새로운 기법이 제시되어야만 한다. 4 장에서는 이 논문이 제안하는 처리 기법을 보인다.

4. 이동 객체들간의 최근접 질의 처리 기법

질의 객체를 q, 검색 대상 객체의 집합을 S, 계산된 최근접 질의 객체를 p 라고 하자. 여기서 q 와 S 의 이동 경로는 미리 알려져 있고, 각 객체는 일정한 속도를 가지고 움직인다고 가정한다. q 와 S 는 그림 1(a)처럼, 모두 궤적을 가지는 이동 객체이다. 궤적은 3 차원 (x, y, t) 공간 상의 선분 세그먼트 집합으로 표현되고, 그림 2 처럼 모두 STR 트리[8]로 유지 된다. STR 트리는 공간 속성에 따라 객체를 인덱스할 뿐만 아니라

그들이 속해 있는 궤적을 따르는 세그먼트들을 그룹화하는 트리이다. 객체의 궤적을 이루는 선분 세그먼트들은 각각 하나의 객체 점으로 표현되어 단말 엔트리에 저장된다.

단말 엔트리의 객체 점은 *seg* = <id, MBB, orient>로 정의한다. 이 때 id 는 객체 식별자이고, MBB 는 한 객체의 한 선분 세그먼트를 대각선으로 하는 직육면체를 말하며, orient 는 MBB 안에서 표현되는 선분 세그먼트의 방향이다. 방향은 그림 1(b)와 같이, 네 개의 방향으로 표현되며, 각각은 {1, 2, 3, 4}의 값 중 하나로 나타난다. MBB 는 두 속성 <T, R>로 표현되며, 이 때 T 는 선분 세그먼트의 유효시간을 나타내는 타임스탬프이고, R 은 세그먼트의 (x, y) 좌표 값을 나타낸다. T 는 <t_s, t_e>로 정의하고, 각각은 세그먼트의 시작 시점과 끝 시점을 나타낸다. R 은 <(x_s, y_s), (x_e, y_e)>로 정의하고, 각각은 시작 시점에서의 공간 좌표, 끝 시점에서의 공간 좌표를 나타낸다.

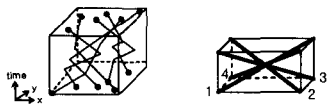

그림 1. (a) 이동 객체의 궤적, (b) 세그먼트의 방향

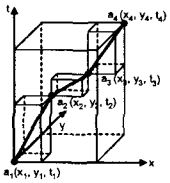

그림 2. MBB 로 표현된 객체의 궤적

이동 객체의 궤적에 대한 최근접 질의 예로 다음을 들 수 있다. 항해를 시작하려는 배 A 가 있다. 이 배는 출발하기 전에, 자신의 이동 경로 전체에 대해 가장 가까운 위치에서 움직이는 급유선의 정보를 알고자 한다. 질의는 다음과 같이 나타난다. "나는 1 시에서 2 시 사이에 i j, 2 시에서 4 시 사이에 j k 경로를 따라 운행 할 예정이다. 나의 경로와 가장 가까운 위치에서 움직이는 급유선을 찾아라". 모든 급유선들의 이동 경로 정보는 중계기를 통해서 미리 알 수 있다고 가정한다. 중계기로부터 받은 급유선들의 궤적 정보를 가지고 다음과 같은 응답을 얻을 수 있다. "1 시에서 2 시까지는 1 번 급유선이 가장 가까이 위치하고, 2 시에서 2 시 30 분까지는 2 번 급유선이, 2 시 30 분에서 4 시까지는 3 번 급유선이 가장 가까이 있다".

이 응답을 얻기 위한 질의 처리 과정은 다음과 같다. 먼저, 질의 궤적 q=[t_s, t_e]의 전체 시간 인터벌 (t_s, t_e)에 해당하는 노드를 접근하여, 그 노드가 포함하는 단말 노드 안의 모든 선분 세그먼트 정보를 가져온다. 이때 각 객체들의 궤적은 그림 3 과 같이 보여진다.

질의 시작 시점 t_s 부터 질의 끝 시간 t_e 까지의 인터벌 내에서, 그림 3 의 점선과 같이 그 안에 있는 모든 선분 세그먼트들(질의 궤적도 포함)의 시작 시점들을 추출시간이라 부른다. 각 추출시간에서 질의 객체의 공간 좌표와 데이터 객체들의 공간 좌표간의 거리를 계산하여, 가장 가까운 거리의 데이터 객체를 최근접 객체로 선택하고 리스트에 저장한다. 예를 들어 그림 3 에서, 질의 시작 시간 t1 에서 최근접 객체가 a 라면 이는 t2 시간까지 질의에 가장 가까운 위치에서 움직이는 객체가 되고, 리스트에 <a, [t1,t2]>로 저장된다.

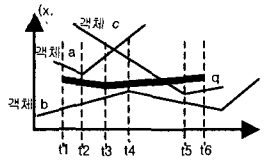

그림 3. 탐색되는 선분 세그먼트들과 추출시간

이 때, 그림 3 에서 t1 의 객체 a, b 나 t2 의 객체 b, q 는 인덱스에 저장되어 있지 않은 공간 좌표 값을 가진다. 이 시점의 좌표 값은 시간에 대한 선형 함수를 사용함으로써 추정할 수 있다. 선형 함수 $f(t)$는 추출되지 않은 시간에서 이동 점 객체의 위치 정보를 반환하는 시간 종속 위치 추정 함수이다.

$$if \quad (t_{i-1} < t < t_i) \ then$$

$$f(t) = \left(\frac{x_i - x_{i-1}}{t_i - t_{i-1}}(t - t_{i-1}) + x_{i-1}, \frac{y_i - y_{i-1}}{t_i - t_{i-1}}(t - t_{i-1}) + y_{i-1} \right)$$

하지만 각 추출시간에서 찾은 최근접 객체 정보는 순간 최근접 처리 기법처럼 사용되므로, 정보의 정확성은 보장될 수 없다. 따라서 다음의 작업들이 추가로 요구된다.

추출시간 t4 에서 t5 까지 객체 b 가 가장 가까운 객체라는 정보를 얻었지만 정확하지는 않다. *t4* 부터 객체 c 와 q 가 교차하는 시점까지는 객체 b 가 가장 가까운 이웃이 될 수 있지만, 교차 시점부터 t5 까지는 오히려 객체 c 가 더 가까이서 움직이는 객체이기 때문이다.

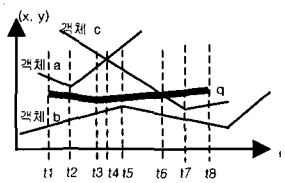

그림 4. 선분 세그먼트들 간의 교차 처리

따라서 그림 4 와 같이 선분 세그먼트가 교차하는 시점도 추출시점으로 선택하고, 최근접 객체를 계산한다. 세그먼트의 교차점을 찾기 위해서, 스윕라인 기법을 사용한다. 먼저, (x, t) 평면에 대해 스캔 하면서 교차점을 찾고, 그 교차점을 가지는 세그먼트들에 대해서만 (y, t) 평면에 대해 교차점을 찾는다. 이렇게 함으로써 3 차원 상에서 교차하는 선분 세그먼트들의 교차점을 얻을 수 있다.

교차점과 세그먼트의 시작 점, 그리고 질의 세그먼트의 끝 점을 추출시간으로 설정하고 최근접 객체 정보를 얻었을 때, 예를 들어 [t2, t3]에서는 최근접 객체가 a 라는 정보를 얻었을 때, 객체 a 는 t2 에서는 가깝지만 질의 객체와 급격히 멀어지고 객체 b 는 t2 에서는 a 보다 멀리 있지만 원만한 기울기를 가지고 질의에 가까워 진다. 이때의 최근접 객체 정보도 정확도가 떨어진다. 이를 정확하게 만들기 위해서 우리는 질의

객체 선분에 대한 대칭 이동을 수행한다. 즉, 질의 객체 선분의 위쪽으로 모든 선분 세그먼트들을 그림 5와 같이 대칭이동 함으로써, 더 정확한 최근접 객체 정보를 얻을 수 있다.

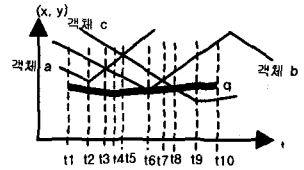

그림 5. 질의 선분에 대한 대칭 이동

같은 추출 시간에서 같은 공간 좌표를 가지는 객체가 여러 개 있을 경우에, 이 객체 중 하나가 최근접 객체가 되어야 한다면, 세그먼트의 방향을 고려한다. 최대한 질의와 가까운 방향으로 움직이는 객체를 선택하고, 방향도 똑같은 경우에는 세그먼트의 기울기를 비교하여 더 완만한 기울기를 가지고 움직이는 객체를 선택할 수 있다. 최종적으로, 각각의 추출시간 인터벌에서의 최근접 정보는 그 정보를 얻을 때마다 리스트에 저장되고, 질의의 마지막 시점에 도달하면 전체 리스트 내의 객체 쌍들이 결과 값으로 반환된다.

그림 6 은 질의와 대상 객체가 모두 이동 객체인 경우, 즉 3 차원 폴리라인으로 경로를 나타낼 수 있는 객체인 경우에 적용 가능한 연속 최근접 질의 탐색 과정을 정리해서 보인다. 이 알고리즘에서 Q[Ts,Te]는 질의 궤적의 전체 시간 인터벌, $\{Q_i(x,y,t)\}_{Ts}^{Te}$ 는 질의 궤적을 구성하는 각 세그먼트의 좌표 값이다. *SL* 은 두 추출시간 사이 $[ST_i, ST_{i+1}]$의 최근접 객체 정보 (NN.id 로 표현된다) 쌍을 모아놓은 리스트이다. $S_i < (x_s, y_s, t_s), (x_e, y_e, t_e) >$는 세그먼트 i 의 시작 좌표와 끝 좌표를 나타낸다.

```
function findNN(List SL)
input : Q[Ts,Te], {Qi(x,y,t)}Ts^Te
output : List SL
method :
1. 트리의 [Ts, Te] 인터벌 안에서
   get Si<(xs, ys, ts), (xe, ye, te)>
2. 질의 세그먼트 위쪽으로, 모든 세그먼트 대칭이동
3. 추출시간 ST 선택 (질의와 모든 객체에 대해)
   1) 전체 Si (xs, ys, ts) 와 Q(Te)의 t 값
   2) 세그먼트가 교차하는 지점의 t 값
      SweepLine (x, t) → return 교차점 집합 C의 좌표
        /* C 를 가지는 세그먼트에 대해서만 */
        SweepLine(y, t) → return 교차점의 시간 값
4. 각 ST 에서, 질의와 객체 간의 거리계산
   2 차원(x, y) 상의 두 점간의 거리 계산
      → return 최근접 좌표를 가지는 객체 id
   if 같은 좌표를 가지는 객체가 여러 개라면, then
      → return 질의와 같은 방향을 가지는 객체 id
      if 방향도 같으면, then
      → return 더 작은 기울기 가지는 객체 id
5. 리스트에 저장
   List SL ← < NN.id, [STi, STi+1] >
```

그림 6. 이동 객체들에 대한 연속 최근접 탐색 알고리즘

5. 결론 및 향후 연구

이동 객체에 대한 최근접 질의를 처리하기 위한 기존의 기법들은 특정 타임스탬프에서 공간 좌표를 비교하여 가장 가까운 객체를 찾는 것이 대부분이었다. 하지만 질의 객체가 동적인 경우, 즉 선분 세그먼트로 주어지는 경우, 특정 시간에서의 최근접 정보는 그 시간이 지나면 잘못된 정보가 될 수 있기 때문에 적합하지 않다. 질의 세그먼트에 대한 최근접 정보는 연속 최근접 질의 처리 방법으로 처리하는 것이 더 정확한 결과를 얻을 수 있다. 하지만 기존에 나와있는 연속 최근접 연구들은 질의의 동적 속성은 고려했지만 데이터 객체들도 동적인 경우는 고려하지 않았다.

따라서 이 논문에서는 질의와 데이터 객체가 모두 3 차원 상에서 궤적을 가지는 경우에 유용한 최근접 질의 처리 기법을 제안하였다. 데이터 객체들의 궤적은 선분 세그먼트 단위로 나뉘어 STR 트리에 점 { id, $<t_s, (x_s, y_s)>$, $<t_e, (x_e, y_e)>$, *orient* }로 저장되고, 질의가 주어졌을 때 질의 궤적의 전체 인터벌 시간에 해당되는 객체 점들만 선택되어 추출시간 선택, 교차지점 발견을 위한 스윕라인, 같은 추출 시점에서의 두 점간의 거리 계산, 방향 고려, 기울기 고려, 리스트 안에 저장되는 순서로 처리 된다.

이 논문에서 제안한 모든 계산 과정은 선형 함수 계산이고, 재계산 작업이 덜 이루어진다. 그리고 모든 단계를 다 고려한다면 정확하고 세밀하게 연속적인 최근접 객체 정보를 얻을 수 있지만, 정확도가 떨어지더라도 적은 계산 시간을 가지고 싶다면 단계를 줄일 수도 있다. 즉, 정확도와 복잡도 사이에 trade-off 가 존재한다. 이 논문에서 제안한 기법은 물류정보시스템, 국방정보시스템, 기상, 교통 등 시공간 이동 객체의 질의를 다루는 시스템에 적용할 수 있으며, 향후 연구로는 성능 평가와 최적화 작업이 남아있다.

참고문헌

[1] Nick Roussopoulos, Stephen Kelley, Fredeic Vincent, "Nearest Neighbor Queries", *SIGMOD Conference* 1995, pp.71~79

[2] Suni Araym David M. Mount, Nathan S. Netanyahu, Ruth Silverman, Angela Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions", Journal of the ACM 1994, pp.573~582

[3] Cui Yu, Beng Chin Ooi, Kian-Lee Tan, H. V. Jagadish, "Indexing the Distance : An Efficient Method to KNN Processing", VLDB 2001, pp.421~430

[4] George Kollios, Dimitrios Gunopulos, Vassilis J. Tsotras, "Nearest Neighbor Queries in a Mobile Environment", Spatio-Temporal Database Management 1999, pp.119~134

[5] Zhexuan Song, Nick Roussopoulos, "k-Nearest Neighbor Search for Moving Query Point", SSTD 2001, pp.79~96

[6] Yufei Tao, Dimitris Papadias, "Time-Parameterized Queries in Spatio-Temporal Database", *SIGMOD Conference*2002, pp.334~345

[7] Yufei Tao, Dimitris Papadias, Qiongmao Shen, "Continuous Nearest Neighbor Search", *VLDB 2002*, pp.287~298

[8] Dieter Pfoser, Christian S. Jensen, Yannis Theodoridis, "Novel Approaches in Query Processing for Moving Objects Trajectories", VLDB 2000, pp.395~406