슬랩을 이용한 효율적인 연속적 최근접 이웃 탐색기법

한 석[O] 박광진* 김종완* 황종선*
고려대학교 컴퓨터학과
{seokh[O], kjpark, wany, hwang}@disys.korea.ac.kr

An Efficient Continuous Nearest Neighbor Search Scheme Using the Slab

Seok Han[O] Kwang-Jin Park* Jong-Wan Kim* Chong-Sun Hwang*
Dept of Computer Science and Engineering, Korea University

요 약

최근에 이동객체의 위치정보를 활용한 위치기반서비스(LBS, Location Based Services)에 대한 관심이 증가하고 있다. 전통적으로 정적인 위치정보를 갖는 공간 객체는 GIS(Geographic Information System) 서버에 저장, 관리되었다. 이동객체는 시간에 따라 위치의 변화가 매우 빈번하여 위치 정보가 계속 갱신되기 때문에, 전통적인 GIS 서버로는 관리가 어렵다. 본 논문에서는 기존의 연속적인 최근접 이웃 탐색 기법에서 데이터의 처리 순서에 따라 탐색공간과 계산비용이 증가하는 문제점을 슬랩을 사용하여 해결한다. 최근접 이웃의 수직연장선 사이의 공간인 슬랩 내부영역에 대해서만 탐색하도록 하여 탐색영역을 줄이고, 그 내부에 있는 점들에 대해서만 처리하여 계산비용을 줄인다.

1. 서 론

우선 환경에서 사용 가능한 이동 단말의 보급과 GPS로 대표되는 측위 기술의 발달에 따라 다양한 종류의 위치기반 응용서비스에 대한 관심이 증가하고 있다. 위치기반 응용서비스를 제공하기 위해 지속적으로 위치가 변하는 객체를 저장, 처리하기 위한 효과적인 방법이 요구된다. 예를 들어, "현재의 경로를 유지했을 때 가장 가까운 버스 정류장을 찾아라." 와 같이, 질의점이 계속적으로 움직여서 시간에 따라 질의의 위치가 달라지는 경우에 질의와 질의 대상과의 거리가 계속적으로 변하기 때문에 최근접 이웃이 계속해서 갱신되어야 한다. 이러한 객체에 대한 질의는 현재까지의 상용 데이터베이스 및 GIS 시스템을 이용해서는 효율적으로 처리할 수 없다[1][2][3].

직선 세그먼트 상에서 질의점이 계속해서 이동하는 경우, 최근접 이웃의 계속적인 갱신을 처리하기 위해 기존 기법들에서는 단순한 최근접 이웃 탐색 알고리즘의 반복 응용에 기반한다. 기존의 시간 매개화(Time-Parameterized) 질의에서는 질의 세그먼트 상의 모든 영향점을 탐색하기 위해 최근접 이웃 질의가 세그먼트의 모든 점에 대해 반복적으로 수행되어야 한다. 질의의 대상이 되는 점집합 P의 각 점이 현재의 최근접 이웃보다 직선 세그먼트에 더 가깝게 되는 세그먼트 상의 점을 영향점이라고 한다. 이 기법은 최근접 이웃의 계산비용이 영향점의 개수에 비례하여 증가하기 때문에 대규모의 질의와 데이터 집합에서 연속적인 최근접 이웃 탐색 시 매우 심각한 CPU 부하를 유발하는 문제점을 발생시킨다[4].

한편, 기존의 연속적 최근접 이웃 탐색기법에서는 전체 질의 세그먼트에 대해 단일 질의를 수행하여 이러한 문제점을 해결하고자 하였다. 그러나 객체의 처리 순서가 탐색성능에 큰 영향을 미치는 문제점이 있다. 예로, 최초로 처리되는 객체가 질의 세그먼트에서 원거리에 위치한 경우, 탐색해야 할 공간이 증가하여 계산해야 할 객체의 수가 증가하게 된다.

본 논문에서는 기존의 연속적인 최근접 이웃 탐색기법이 갖는 반복수행의 문제점을 해결하기 위해 질의 세그먼트의 시작점과 끝점으로부터 각각 최단거리를 갖는 최근접 이웃을 탐색하고, 슬랩을 사용한 단일 질의를 수행하여 반복수행의 문제를 해결한다. 슬랩은 최근접 이웃의 수직연장선이다. 본 논문에서는 질의 처리 시에 탐색공간을 시작점과 끝점의 최근접 이웃의 슬랩 사이의 공간으로 부터 점진적으로 줄여나감으로써 계산 비용을 감소시키는 연속적 최근접 이웃 탐색 기법을 제안한다.

2. 관련연구

2.1 시간 매개화(Time-Parameterized) 질의

시간 매개화 질의는 <R, T, C>의 형태를 가진다[4]. R은 질의의 현재 결과, T는 R의 유효 기간, C는 T의 마지막 시점에서 R에 영향을 주는 객체들의 집합이다. R로부터 C까지 계속적으로 다음 결과를 계산한다.

초기에 시작점 s_0에서 최초의 최근접 이웃을 검색하기 위해 최근접 점 질의(point-NN query)가 수행된다. 그리고 현재의 최근접 이웃보다 직선 세그먼트에 더 가깝게 되는 데이터 집합 P의 객체 x의 영향점 s_x가 계산된다. 이러한 영향점들을 식별하는 것은 최소 dist(s, s_x)를 갖는 점 x를 찾는 전통적인 최근접 이웃 질의와 같다[5]. 그림1은 연속적인 최근접 이웃을 탐색하는 시간 매개화 질의처리 과정을 보여준다.

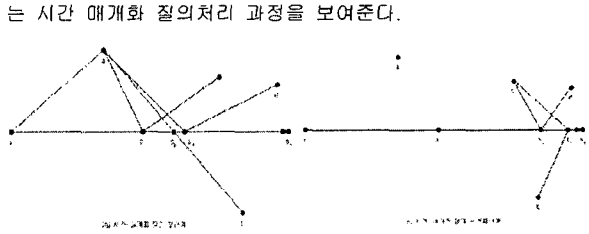

그림 1. 시간 매개화 질의에서의 연속적인 최근접 이웃 탐색

그림1의 (a)는 시작점 s_0에서 첫번째 최근접 이웃 a를 검색한

후의 영향점들을 보여준다. s_c가 첫 번째 영향점 s_1이 되고, c가 다음 최근접 이웃이 된다. 그림 1의 (b)에서는 최종 결과를 얻기 위해 반복적인 시간 매개화 질의가 수행되는 것을 보여준다. 이 기법은 모든 영향점들 각각에서 다음 최근접 이웃을 검색하기 위해 반복적인 최근접 이웃 질의를 수행하는 문제가 발생한다. 결국 반복적인 질의 수행은 계산비용의 증가를 초래한다.

2.2 연속적인 최근접 이웃 탐색기법 (CNN Search)

시간 매개화 질의를 이용할 때 발생하는 반복적인 최근접 질의 수행으로 인한 계산비용 증가의 문제를 해결하기 위해 연속적인 최근접 이웃 탐색 기법이 제안되었다[8].

공간상의 한 점집합을 P라 하자. 연속적인 최근접 이웃 질의는 직선 세그먼트 $q=[s_0, s_e]$에 있는 모든 점의 최근접 이웃과 분할점의 리스트 SL(Split List)을 반환한다. SL은 분할점과 최근접 이웃의 쌍으로 나타낸다. 최근접 이웃이 변하는 질의 세그먼트 상의 점을 분할점이라 한다. 분할점을 중심으로 하고 분할점의 최근접 이웃과의 거리를 반경으로 하는 원을 경계원이라 한다. 분할점의 경계원 내부에 있는 점들만이 SL을 갱신시킨다. 결과 값은 <R, T> 튜플 집합으로 구성된다. R은 질의를 만족하는 P의 한 점이고, T는 R이 q의 최근접 이웃인 구간이다. 그림 2는 연속적인 최근접 이웃과 분할점 탐색과정을 보여준다.

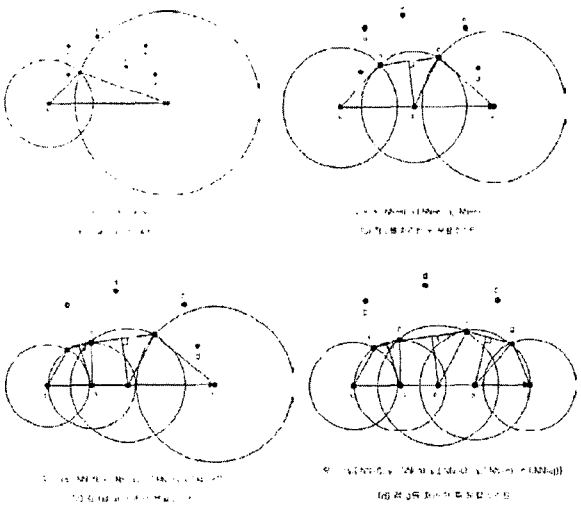

그림 2. 연속적인 최근접 이웃과 분할점 탐색

그림 2에서 P={a, b, c, d, e, f, g}이고, 임의로 주어진 알파벳 순서에 따라 P의 원소들을 처리한다. 그림 2 (a)에서 처음 처리되는 점 a가 q의 현재의 최근접 이웃이 되고 SL에 삽입된다. 그림 2 (b)에서 다음 점 b는 s_0 와 s_e의 경계원 외부에 있으므로 최근접 이웃이 될 수 없다. 그러나 c는 s_e의 경계원 내부에 있으므로 최근접 이웃이 되고, a와 c를 잇는 수직이등분선, $\perp(a,c)$과 q와의 교점 s_1이 분할점으로 삽입되고 SL이 갱신된다. 그림 2 (c)에서 점 d와 e는 어떠한 경계원 내부에도 있지 않으므로 무시한다. 다음 점 f는 s_0의 경계원 내에 있으므로 처리한다. SL은 f와 a의 수직이등분선과 q와의 교점 s_1이 추가로 삽입되어 갱신된다. 마지막 점 g는 s_e의 경계원 내부에 있으므로 그림 2 (d)와 같이 $\perp(c,g)$와 q의 교점 s_3이 삽입되어 SL이 갱신된다.

이 기법은 질의 세그먼트의 길이가 길어지면, 최근접 이웃을 검색하기 위한 탐색공간이 늘어나기 때문에 처리해야 할 객체의 수가 증가하게 된다. 또한, 데이터 집합의 처리되는 순서에 따라 성능에 영향을 크게 받는다. 예로, 질의 세그먼트로부터 멀리 떨어진 객체를 먼저 처리 시 탐색공간이 증가되어 처리해야 할 객체의 수가 증가한다. 이는 결국 거리 계산비용을 증가시키게 되어 질의 성능이 저하되는 문제점을 발생시킨다.

3. 슬랩을 이용한 연속적 최근접 이웃 탐색기법

본 논문에서 제안하는 기법은 기존의 연속적인 최근접 이웃 탐색기법과 달리 질의 시에 질의 세그먼트의 시작점 s_0와 끝점 s_e로부터 각각의 최근접 이웃을 탐색한다. 이때 탐색한 두 최근접 이웃 각각의 슬랩 사이의 공간으로부터 탐색공간을 점진적으로 줄여 기존 기법보다 효율적인 연속적 최근접 이웃 탐색이 가능하다.

점집합 P = {a, b, c, d, e, f, g}이고, P의 원소들은 임의의 순서가 아닌 질의 세그먼트와의 최단거리에 따라 처리되며, 탐색공간과 계산비용을 줄이기 위해 슬랩을 사용한다. 슬랩은 P의 임의의 원소의 수직 연장선이다. p_i와 p_j를 P의 임의의 두 원소라 하고, 이 두 점의 슬랩 사이의 공간은 $slab(p_i, p_j)$으로 나타낸다. 두 점사이의 최단거리는 유클리드(Euclid) 거리 측정 기법을 사용하여 측정한다. 질의 세그먼트 위의 점에 대한 최근접 이웃은 P의 원소 중 q와 최단거리를 갖는 점을 의미한다. 처음 시작 시, s_0 와 s_e 각각의 최근접 이웃은 P의 모든 원소와의 거리를 측정하여 결정한다. 그림 3은 슬랩을 이용한 연속적인 최근접 이웃 탐색과 SL 갱신 과정을 보여준다.

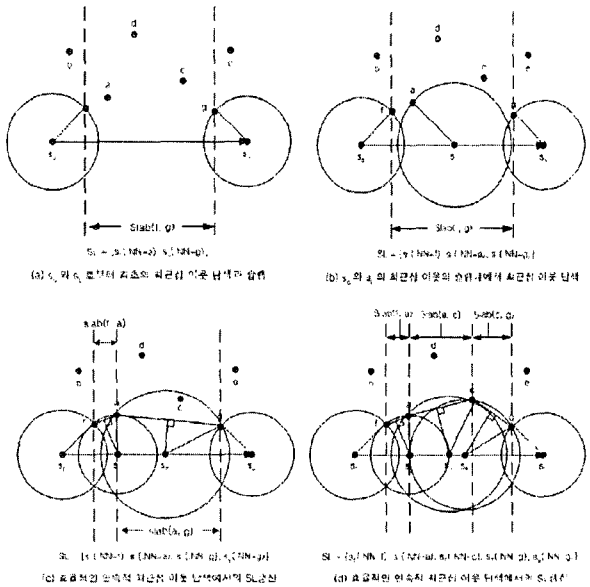

그림 3. 슬랩을 이용한 연속적인 최근접 이웃 탐색과 SL 갱신

q 상의 점들중에 최근접 이웃이 변경되는 점을 분할점 s_i라 하면, 이러한 분할점들과 각각의 최근접 이웃 $s_i.NN$의 쌍의 집합인 SL이 결과값이 된다. 초기에 $q = \{s_0, s_e\}$이고, s_0와 s_e의 최근접 이웃은 알려져 있지 않다. 그림 3 (a)에서 먼저, P의 모든 원소중에 s_0와 최단거리를 갖는 점 f와 s_e와 최단거리

를 갖는 점 g를 s_0, s_e 각각의 최근접 이웃으로 탐색하고 SL에 삽입한다. 이때, s_0와 s_e를 각각 중심으로 하고 각각의 최근접 이웃 f와 g 사이의 거리 dist(s_0, f)=|s_0, f|와 dist(s_e, g)=|s_e, g|를 반지름으로 하는 경계원을 얻는다. 그림 3 (b)에서는 다음 최근접 이웃을 찾기 위해 q에서 s_0와 s_e의 경계원 내에 포함되는 구간을 뺀 나머지 구간의 중점 s_1을 분할점으로 하여 slab(f, g)의 영역 내에서 최근접 이웃 a를 탐색하고 SL에 삽입한다. q가 경계원으로 모두 덮일 때까지 최근접 이웃을 탐색한다. 그림 3 (c)에서는 (b)에서 얻은 최근접 이웃 f와 a의 수직이등분선 $\perp(f,a)$와 q의 교점 s_1와 a와 g의 수직이등분선 $\perp(a,g)$와 q의 교점 s_2 각각의 최근접 이웃을 slab(f, a)와 slab(a, g)에서 각각 탐색하고 SL에 삽입한다. 그리고 s_1의 경계원 내에 다른 점이 존재하는지 slab(f, a)에서 검사하고, s_2의 경계원 내에 다른 점이 존재하는지 slab(a, g)에서 검사한다. |s_2, g| > |s_2, c| 이므로 그림 3 (d)에서는 $\perp$(c, g) 와 q의 교점을 새로운 분할점 s_3로 하는 경계원을 생성한다. s_3와 s_3의 최근접 이웃 c를 삽입하여 SL을 갱신한다. 마지막까지 SL에 남아있는 분할점과 최근접 이웃이 최종 결과가 된다.

```
[SCNN Algorithm Search]
1 // P : 무작위 함수에 의해 생성된 N개의 점집합, P={p1,p2,…,pn}
2 // pi : P의 임의의 원소, pi = (pix , piy) ∈ P
3 // SL : 분할점 si = (six, siy)와 si.NN = pj 의 순서쌍 (si, pj) 집합
4 // si.NN : si의 최근접 이웃
5 // 초기 SL = { s0, se }
6 // 중심 si의 최근접이웃 pj에 대한 최소경계원(MBC)의 반지름(rsi):
7 // rsi = |si, pj| = sqrt((six - pjx)^2 + (siy - pjy)^2)
8 // q : 직선 질의 세그먼트
9 // qL : q의 길이, qL = |s0, se|
10
11 Begin FindNN
12
13 compute dist (si, pj);  // s0, se 와 pj 사이의 거리계산
14  choose si.NN with min(dist(si, pj); // s0, se 의 NN 결정
15 if ( pjx < s0.NN or pjx > se.NN )
16    delete pj from P; // s0의 슬랩과 se의 슬랩 사이의 공간에
17                      // 있지 않은 점들을 제거
18 while ((qL - Σrsi) > 0) {
19  search sj and sj.NN; // si의 MBC에 포함되지 않고 남은 q의
20                       // 구간의 중점 sj 와 최근접 이웃 sj.NN 검색
21  create MBC of sj; // sj의 MBC 생성
22 }
23 compute sk and sk.NN; // ⊥( Sj.NN , Sj+1.NN ) 과 q의 교점
24                        // Sk 와 최근접 이웃 sk.NN 계산
25 update SL with (sk, sk.NN); // SL에 (sk, sk.NN) 삽입/갱신
26 create MBC of sk; // sk의 MBC 생성
27    // sk의 MBC 내에 더 가까운 점이 있는 지 검사, ps는
28    // sk-1.NN의 슬랩과 sk.NN의 슬랩 사이의 점
29 if (rsk ≥ dist (sk ,ps))
30   insert (sk ,ps) into SL; // 더 가까운 점 ps가 있으면 SL에
31                            // (sk ,ps)을 삽입하고 갱신
32 return SL; // 최종 결과값으로 SL을 반환
33
34 End FindNN
```

그림 4. 슬랩을 이용한 연속적 최근접 이웃 탐색 알고리즘

4. 결론 및 향후 연구

본 논문에서 제안한 알고리즘을 사용하면 질의점이 직선 세그먼트 위를 계속적으로 움직여서 시간에 따라 질의의 위치가 달라지는 연속적인 최근접 이웃을 기존 기법보다 효율적으로 탐색할 수 있다.

본 논문에서는 탐색영역과 계산비용을 줄이기 위해 슬랩을 이용한 연속적 최근접 이웃 탐색기법을 제안하였다. 본 기법을 통해 기존 기법의 높은 처리 비용과 질의 결과의 부정확성을 방지한다. 또한, 연속되는 최근접 이웃의 슬랩을 이용하여 탐색영역과 계산비용을 줄일 수 있다. 먼저 질의 세그먼트의 시작점과 끝점의 최근접 이웃을 각각 탐색한다. 이 두 최근접 이웃의 슬랩 사이의 공간이 다음 최근접 이웃을 찾기 위한 탐색 공간이 된다. 이와 같은 방법으로 남은 질의 세그먼트의 구간에서 질의를 계속 수행하면 탐색할 공간이 점진적으로 줄어들기 때문에 필요 없는 객체가 제거되며, 계산비용도 감소한다. 본 기법은 K-최근접 이웃 질의[6]나 도로 네트워크에서의 경로를 입력으로 하는 응용[7]에서의 적용을 위해 확장될 수 있으며 지리정보시스템과 이동 컴퓨팅과 같은 응용들에도 유용하다.

향후 질의대상이 되는 데이터의 분포에 따른 제안기법의 성능평가와 시간 매개화 질의[5]나 연속적 최근접 이웃 탐색기법[8]과 같은 기존 기법과의 비교평가를 위한 시뮬레이션을 진행할 것이다.

한편, 다차원 공간으로 확장 시 사용가능하도록 기존의 거리측정 기법과 불필요한 데이터 제거 기법의 수정을 위한 연구도 진행되어야 한다. 또한, 동적 데이터 집합에 대한 응용과 시공간 데이터베이스에서 제안된 R*-트리[9]와 같은 동적 색인기법과 본 논문의 제안기법과의 결합도 고려중이다.

[참고문헌]

[1] Martin Erwig, Ralf Hartmut Guting, Markus Schneider and Michalis Vazirgiannis. "Spatio-Temporal Data Types : an Moving Object in Databases", CHOROCHRONOS Approach to Modeling and Querying Technical Report, 1997.

[2] D.E.Knuth. "Art of Computer Programming, Volumn 3 : Sorting and Searching. Addison-Wesley Pub Co,1998.

[3] L. Forlizzi, R. H. Guting, E. Nardeli and M. Schneider. "A Data Model and Data Structures for Moving Objects Databases", ACM SIGMOD, 2000.

[4] Tao, Y. Papadias, D. "Time Parameterized Queries in Spatio-Temporal Databases. ACM SIGMOD, 2002.

[5] N. Roussopoulos, S. Kelly, and F. Vincent. "Nearest Neighbor Queries", ACM SIGMOD, 1995.

[6] Song, Z., Roussopoulos, N. "K-Nearest Neighbor Search for Moving Query Point", SSTD, 2001.

[7] Christian S.Jensen, Jan Kolar, Torben Bach Pedersen, Igor Timko. "Nearest Neighbor Queries in Road Networks", GIS '03.

[8] Y.Tao, D.Papadias, and Q.Shen. "Continuous Nearest Neighbor Search", 28th VLDB, 2002.

[9] Beckmann, N., Kriegel, H. P., Schneider, R., Seeger, B. "The R*-tree : An Efficient and Robust Access Method for Points and Rectangles", ACM SIGMOD, 1999.