연속 최근접 이웃(CNN) 탐색의 성능향상을 위한 탐색구간 제한기법

A Search Interval Limitation Technique for Improved Search Performance of CNN

한 석* 오 덕 신** 김 종 완***
Seok Han Dukshin Oh Jongwan Kim

요 약

위치기반 서비스(LBS, Location Based Services)에 대한 관심이 증가함에 따라 사용자가 이동 중에도 질의를 통한 최근접 이웃(NN, Nearest Neighbor) 탐색에 대한 필요성이 증가하였다. 이와 같은 동적환경에서의 최근접 이웃 탐색은 탐색 구간에 대해 NN탐색기법을 반복적용하여 수행해 왔으나 이는 불필요한 중복연산이 발생하여 탐색 비용이 증가하는 단점이 있다. 본 논문에서는 이동 중에도 탐색 구간에 대해 연속적인 최근접 이웃(CNN, Continuous Nearest Neighbor)을 탐색하는 새로운 기법인 *Slabbed_CNN*을 제안한다. *Slabbed_CNN*은 슬랩을 이용하여 탐색 구간을 줄임으로써 기존 CNN기법의 탐색영역과 계산비용을 감소시킴으로써 기존 CNN보다 연산비용을 감소시키고 빠른 서비스를 제공한다.

Abstract

With growing interest in location-based service (LBS), there is increasing necessity for nearest neighbor (NN) search through query while the user is moving. NN search in such a dynamic environment has been performed through the repeated application of the NN method to the search segment, but this increases search cost because of unnecessary redundant calculation. We propose slabbed continuous nearest neighbor (Slabbed_CNN) search, which is a new method that searches CNN in the search segment while moving. Slabbed_CNN reduces calculation costs and provides faster services than existing CNN by reducing the search area and calculation cost of the existing CNN method through reducing the search segment using slabs.

☞ keyword : 최근접이웃 탐색, 연속적 최근접이웃 탐색, 슬랩, 위치기반서비스, Nearest Neighbor, CNN(Continuous NN), Slab, Location-Based Services

1. 서 론

위치기반서비스 (LBS : Location Based Services)는 사용자에게 무선 통신을 통해 호텔, 병원, 택시 등의 위치와 같은 다양한 정보를 제공한다 [1]. 사용자는 서비스를 탐색하기 위해 응용 프로그램 상에서 객체나 위치를 선택하고 그 위치에 가장 가까운 객체를 찾는 최근접이웃(NN) 질의를 한다 [2]. 이는 사용자나 객체가 모두 정적(static)인 상태일 때 수행되는 것이다. 그러나 모바일 장치를 휴대한 사용자는 질의를 한 후 이동이 가능하기 때문에 LBS 서버는 이 경우에도 최근접 이웃을 탐색하여 사용자에게 제공해야 한다. 따라서 그림 1과 같이 사용자가 이동할 때 질의 시작점과 끝점 사이에서 가장 가까운 호텔을 최근접이웃으로 탐색하고자 할 때, 연속적인 최근접 이웃(CNN) 탐색이 필요하다. 이때, 사용자가 이동함에 따라 최근접 이웃은 다르게 나타난다.

CNN 탐색은 질의 경로를 따라 탐색해 나가면서 질의에 대한 최근접 이웃의 정보가 변하는 시

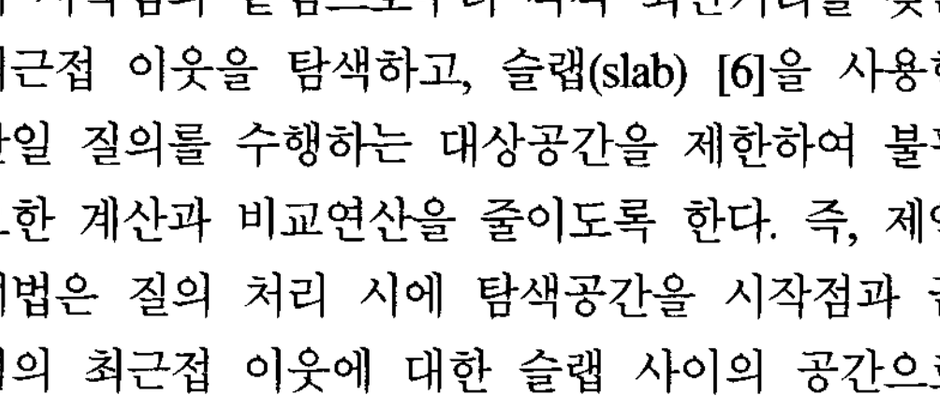
* 정 회 원 : 한미 연합군 사령부 군수참모부 군수준비 태세처 seok0446@hotmail.com
** 정 회 원 : 삼육대학 경영정보학과 부교수 ohds@syu.ac.kr
*** 정 회 원 : 삼육대학교 경영정보학과 외래 교수 wany@korea.ac.kr(교신저자)
[2007/09/06 투고 - 2007/09/14 심사 - 2007/12/21 심사완료]

점, 즉 분할점(split point) s_i를 찾는 것이 목적이다. CNN에서는 탐색 구간의 모든 점에 대한 최근접 이웃을 분할점과 대상 객체 사이의 직선거리를 계산하여 탐색한다 [3][4][5].

CNN 탐색기법에서는 탐색 구간에 대해 단일 질의를 수행하여 반복적인 단순 최근접 이웃질의를 수행하는 기존의 문제점을 해결하고자 하였다 [4]. 그러나 객체의 처리 순서가 탐색성능에 큰 영향을 미쳐 심각한 계산 오버헤드(overhead)가 발생하는 문제점이 있다. 즉, 최초로 처리되는 객체가 질의 구간에서 원거리에 위치한 경우, 탐색해야 할 공간이 증가하여 객체의 수가 증가하게 된다. 이는 계산비용의 증가를 초래하여 탐색성능을 저하시킨다.

본 논문에서는 기존의 CNN이 대상 객체의 처리 순서에 따라 탐색성능에 영향을 받는 문제점을 해결하기 위한 새로운 *Slabbed_CNN* 탐색 구간 제한기법을 제안한다. 제안 기법에서는 탐색 구간의 시작점과 끝점으로부터 각각 최단거리를 갖는 최근접 이웃을 탐색하고, 슬랩(slab) [6]을 사용한 단일 질의를 수행하는 대상공간을 제한하여 불필요한 계산과 비교연산을 줄이도록 한다. 즉, 제안 기법은 질의 처리 시에 탐색공간을 시작점과 끝점의 최근접 이웃에 대한 슬랩 사이의 공간으로 점진적으로 줄여나감으로써 계산 비용을 감소시키는 탐색기법이다.

논문에서 이동 객체는 위치만 변하는 객체로 가정하고, 동적 질의와 정적 대상에 대한 연속적인 최근접 이웃탐색으로 제한한다. 슬랩은 질의 구간의 최근접 이웃의 x축에 대한 수직연장선이다.



(그림 1) 이동환경에서의 최근접 이웃 탐색

논문의 구성은 다음과 같다. 2절에서는 최근접 이웃을 탐색하기 위한 기존 기법과 탐색영역을 분할하는 슬랩의 개념을 살펴본다. 3절에서는 기존 연속적 최근접 탐색 기법의 단점을 극복하기 위해 슬랩을 적용한 연속적 최근접 탐색 기법을 소개한다. 4절은 슬랩을 적용한 R-tree기반 인덱스들의 성능을 비교하여 제안 기법의 향상된 성능을 보인다. 5절은 논문의 요약과 함께 향후 연구에 대해 기술한다.

2. 관련연구

2.1 최근접 이웃 탐색 기법

사용자에게 위치기반 서비스를 제공하기 위한 최근접 이웃 탐색기법은 TP(Time-Parameterized) NN [3], CNN(Continuous Nearest Neighbor) [4], Voronoi 다이어그램을 이용한 기법 [5] 등이 연구되었다.

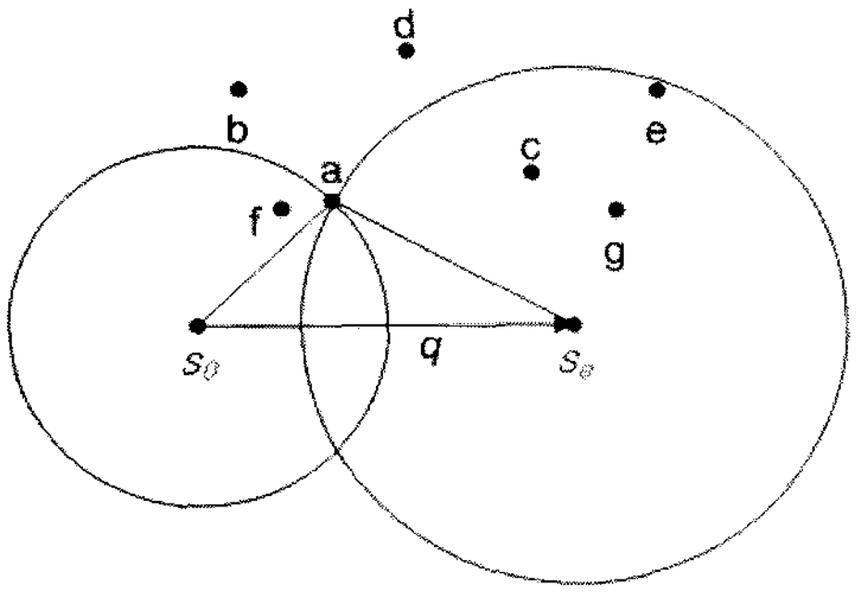
보로노이(voronoi) 다이어그램을 사용하는 기법은 질의점이 계속해서 이동하는 경우, 최근접 이웃의 계속적인 갱신을 처리하기 위해 단순한 최근접 이웃탐색 알고리즘의 반복 응용에 기반한다. 여기에는 불필요한 중복연산이 존재한다.

TP 최근접 질의는 탐색 대상 객체들의 작용시간(influence time)을 거리 측정에 사용한다. 이때, 작용시간이란 데이터 객체가 질의 객체에 가까워지기 시작하는 시간이다. 이 기법은 각 작용시간마다 질의의 대상이 되는 점집합 P의 점이 현재의 최근접 이웃보다 탐색 구간에 더 가깝게 되는 영향점(influence point)에서 객체들 간의 거리를 계산하기 때문에 연산비용이 높다. 즉, 최근접 이웃의 계산비용이 영향점의 개수에 비례하여 증가하기 때문에 대규모의 질의와 데이터 집합에서 CNN 탐색 시 매우 심각한 CPU 부하를 유발하는 문제점이 있다. CNN은 기존의 문제점을 보완하고자 사용자가 이동 중에 탐색할 구간인 직선 탐색 질의 구간 $q=[s_0, s_e]$ 의 최근접 이웃집합과 분할점 리스트 SL(Split List)을 구한다. 객체 사이의

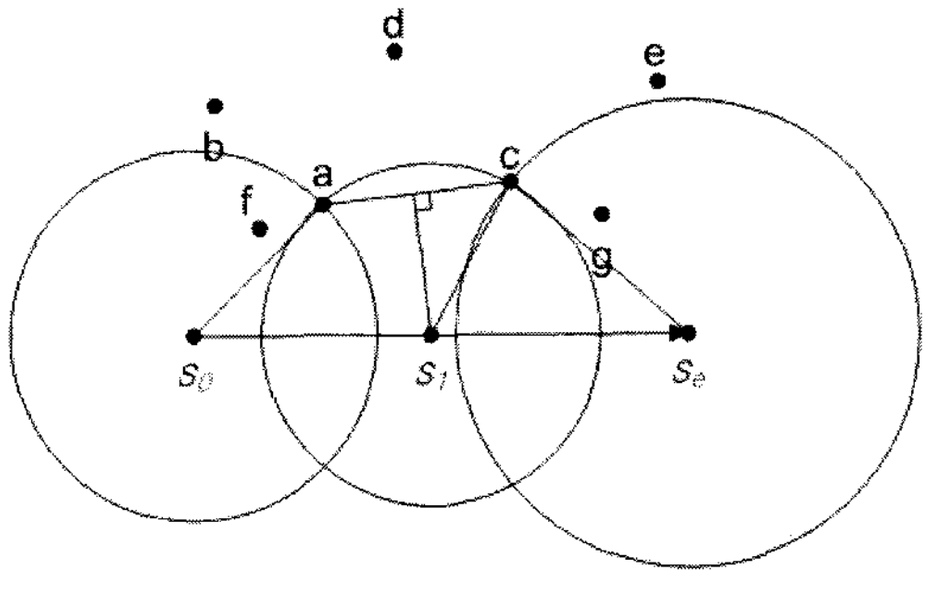
직선 구간에 대한 CNN은 NN을 반복 적용하는 것보다 효과적이다.

그림 2, 그림 3은 연속적인 최근접 이웃과 분할점 탐색과정을 보여준다. 초기에 SL은 오직 두 개의 분할점 s_0, s_e를 가지며, 질의 구간 q=[s_0, s_e] 내의 모든 점에 대한 최근접 이웃은 주어지지 않는다. 각 단계에서 SL은 그때까지 처리된 모든 점들의 결과 값을 가진다. 마지막 결과 리스트는 탐색이 종료된 후에 각 분할점 s_i와 최근접 이웃 s_i.NN을 포함한다. 그림 2에서 공간상의 한 점집합은 P={a, b, c, d, e, f, g} 이고, 임의로 주어진 알파벳 순서에 따라 P의 원소들을 처리한다. 이때, 가장 먼저 처리되는 P의 점 a가 질의 구간 q에 대한 현재의 최근접 이웃이 되고 SL에 삽입된다. |s_0, a| (s_0 에서 a 까지의 거리)와 |s_e, a|를 반지름으로 하고 s_0 와 s_e 를 각각 중심으로 가지는 경계원을 각각 그린다. 이때의 결과 리스트는 {a, [s_0, s_e]}가 된다. 다음에 처리될 객체가 경계원에 포함되지 않을 경우엔 처리하지 않는다.

그림 3에서 다음 점 b는 s_0와 s_e의 경계원 외부에 있으므로 최근접 이웃이 될 수 없다. 그러나 다음 점 c는 s_e의 경계원 내부에 있으므로, a와 c를 잇는 수직이등분선 ⊥(a, c) 와 q와의 교점이 분할점 s1이 되어 SL에 삽입되어 SL이 갱신되고 결과 리스트는 {<a, [s_0, s1]>, <c, [s1, s_e]>}가 된다. 이와 같은 과정을 반복 수행하면 최종 결과 리스트로써 {<f, [s_0, s1]>, <a, [s1, s2]>, <c, [s2, s3]>, <g, [s3, s_e]>}가 반환된다.



(그림 2) CNN과 분할점 탐색: 점 a를 처리한 후 분할리스트 (SL={s_0(.NN=a), s_e(.NN=a)})



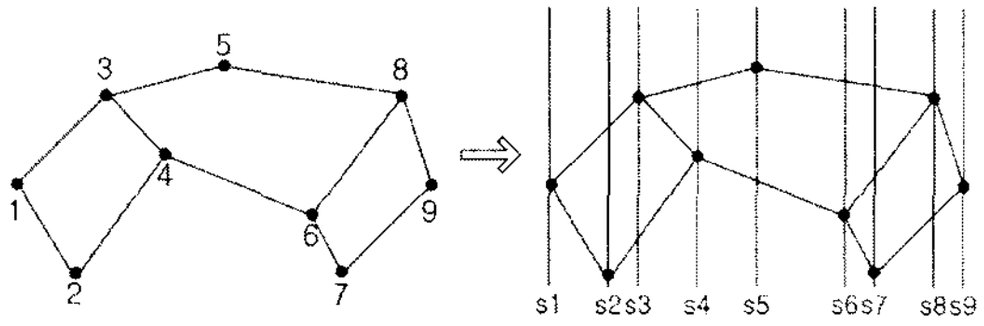
(그림 3) CNN과 분할점 탐색: 점 c를 처리한 후 분할 리스트 (SL={s_0(.NN=a), s1(.NN=c), s_e(.NN=c)})

CNN 기법은 질의 구간의 길이가 길어지면, 최근접 이웃을 탐색하기 위한 탐색공간이 늘어나기 때문에 처리해야 할 객체의 수가 증가하게 된다. 또한, 데이터 집합이 처리되는 순서에 따라 성능에 영향을 준다. 즉, 초기에 선택된 임의의 객체가 질의 구간으로 부터 멀리 떨어진 경우 탐색공간이 증가되어 처리해야 할 객체의 수가 증가한다. 결과적으로 거리 계산비용을 증가시키고 객체간의 거리 비교연산도 증가시켜 질의 성능이 저하되는 문제가 있다.

2.2 슬랩(Slab)과 질의영역의 재분할

슬랩은 특정 점들을 기준으로 수직 또는 수평선을 그어 탐색영역을 축소하는 방법으로 평면에서의 점 위치 문제를 처리할 때 매우 유용한 방법이다 [6]. 먼저, s를 n개의 모서리를 가진 평면 재분할(planar subdivision)이라 할 때, 평면에서의 점 위치 문제는 주어진 질의점 q를 포함하는 s의 면(face)을 알려주도록 s를 저장하는 것이다. 특정 면을 알아내기 위해 각 점을 분할하는 슬랩을 사용한다. 즉, 그림 4와 같이 평면 재분할의 모든 꼭지점을 통과하는 수직선을 그린 다음 수평, 수직 슬랩으로 분할하고 저장한다.

이때, 슬랩이 생성되는 꼭지점의 x축 좌표는 배열 안에 정렬된 순서로 저장된다. 이것은 O(log n)



(그림 4) 수직 슬랩을 이용한 평면 재분할

시간동안에 질의점 q를 포함하는 슬랩을 결정할 수 있도록 한다.

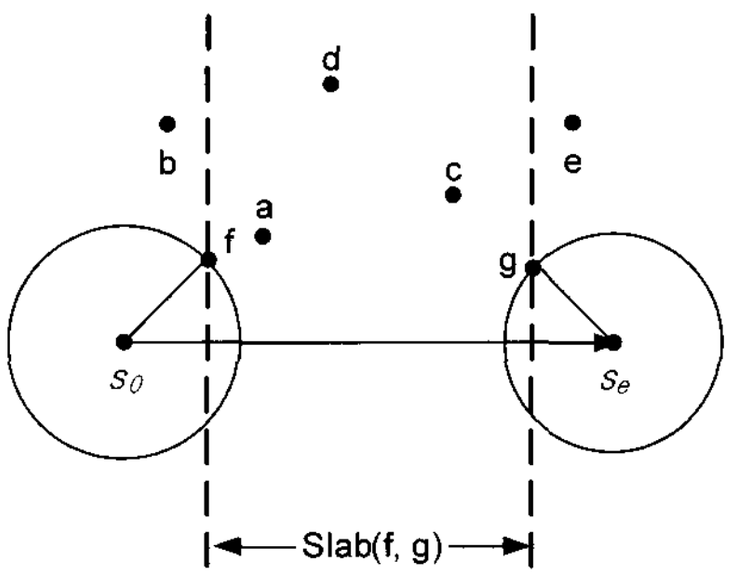
제안 기법에서는 연속적인 최근접 이웃을 탐색하기 위한 탐색영역을 줄이고 계산비용을 감소시키기 위해 최근접 이웃의 수직 슬랩을 이용하여 탐색영역을 재분할한다. 질의 구간의 시작점과 끝점으로부터 각각 최단거리를 갖는 최근접 이웃을 탐색하고, 각각의 최근접 이웃의 수직 슬랩을 사용하여 탐색영역을 분할한다. 질의를 처리하는 동안 질의 구간에 대한 단일 질의를 수행하는 탐색영역을 점진적으로 줄여나간다. 이와 같이, 슬랩을 이용하면 최근접 이웃의 수직 슬랩으로 재분할된 탐색영역에서 질의 구간의 다음 최근접 이웃을 탐색하여 불필요한 계산과 비교연산이 감소한다.

3. 슬랩을 이용한 연속적 최근접 이웃탐색

3.1 탐색영역 분할과 최근접 이웃

본 논문에서 제안하는 *Slabbed_CNN*은 다음 사항을 가정한다. 점집합 P={a, b, c, d, e, f, g} 이고, P의 원소들은 임의의 순서가 아닌 질의 구간 q=[s_0, s_e] 와의 최단거리에 따라 처리되며, 탐색공간과 계산비용을 줄이기 위해 슬랩을 사용한다. 이때, 수직 슬랩이 사용되며, 그림 5와 같이 q의 최근접 이웃이 되는 P의 임의의 원소의 수직 연장선이다. p_i와 p_j를 P의 임의의 두 원소라 하면, 두 점의 수직 슬랩 사이의 공간은 slab(p_i, p_j) 으로 나타낸다. 질의 구간 q상의 모든 점에 대한 최근접 이웃은 P의 원소 중에 q와 최단거리를 갖는 점을 의미한다. 두 점 사이의 최단거리는 유클리드(euclidian) 거리로 측정한다. 초기에 SL은 오직 두 개의 분할점 s_0, s_e를 가지며, 질의 구간 q=[s_0, s_e] 내의 모든 점에 대한 최근접 이웃은 주어지지 않는다.

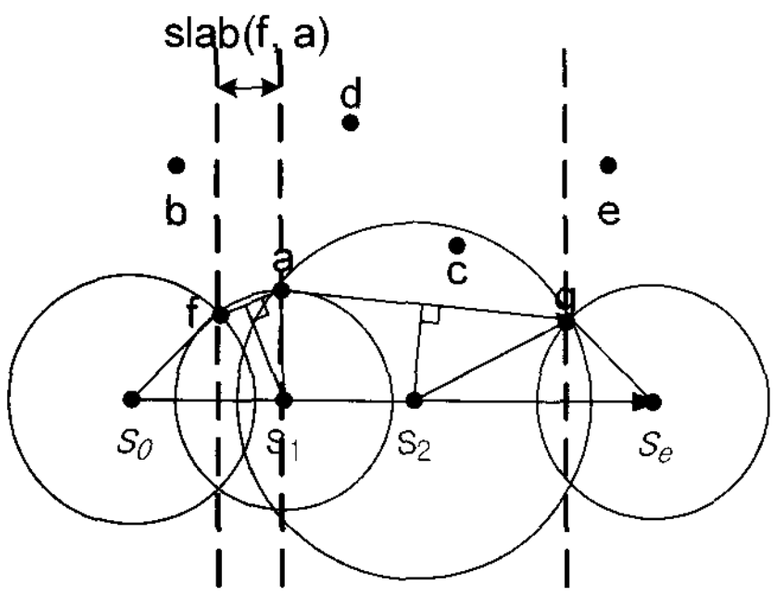
슬랩을 이용한 CNN 계산과 분할점 리스트 SL의 갱신과정은 그림 5, 그림 6과 같다. 그림 5는 최초의 최근접 이웃을 찾은 후 슬랩에 대한 구성을 나타낸다. 먼저, P의 모든 원소 중에 질의 구간의 시작점 s_0 와 최단거리를 갖는 점 f를 최근접 이웃으로 탐색하게 된다. g의 경우도 마찬가지 방법으로 탐색한다. NN을 찾는 과정은 다음과 같다. q 상의 점들중에 최근접 이웃이 변경되는 점을 분할점 s_i라 하면, 이러한 분할점들과 각각의 최근접 이웃 s_i.NN 쌍의 집합이 SL이다. 초기에 q = {s_0, s_e} 이고, s_0 와 s_e 의 최근접 이웃은 알지 못한다. 그림 5에서 먼저, P의 원소 중 질의 세그먼트의 시작점 s_0 와 최단거리를 갖는 점 f를 최근접 이웃으로 검색하게 된다. 이때 f의 x축 좌표를 따라 수직 슬랩을 그려서 탐색영역을 분할한다. f의 수직 슬랩의 오른쪽 영역이 다음 탐색영역이 된다. 이 영역에서 s_e 와 최단거리를 갖는 점 g를 s_e 의 최근접 이웃으로 탐색하고 g의 수직 슬랩을 그린다. 다음으로, s_0 와 s_e를 각각 중심으로 하고 각각의 최근접 이웃 f와 g사이의 거리 dist(s_0, f)=|s_0, f| 와 dist(s_e, g)=|s_e, g| 를 반지름으로 하는 경계원을 얻는다. 이때 얻어지는 SL = {s_0(.NN=f), s_e(.NN=g)} 이다.



(그림 5) s_0 와 s_e 로부터 최초의 최근접 이웃 탐색과 슬랩 구성, SL={f, g}

그림 6은 그림 5의 최근접 이웃 f와 a의 수직 이등분선 ⊥(f, a)와 질의 구간 q와의 교점 s1을 찾고 최근접 이웃을 slab(f, a)에서 탐색한다. 그리고 a와 g의 수직이등분선 ⊥(a, g)와 질의 구간 q와의 교점 s2를 찾고 최근접 이웃을 slab(a, g)에서 탐색한다. 탐색결과는 SL={s_0(.NN=f), s1(.NN=a), s2(.NN=g), s_e(.NN=g)}가 된다.

최종 결과를 얻기 위해 각 경계원 내에 더 가까운 점이 있는지 검사한다. s1의 경계원 내에 다른 점이 존재하는지 slab(f, a)에서 검사하고, s2의 경계원 내에 다른 점이 존재하는지 slab(a, g)에서 검사한다. 이때, |s2, g|>|s2, c|이므로 SL은 갱신되어야 한다. 이 과정을 반복하여 수행하면 구간의 모든 CNN이 계산되며 최종 SL={s_0(.NN=f), s1(.NN=a), s2(.NN=c), s3(.NN=g), s_e(.NN=g)}이 된다. s3는 s2와 s_e사이에 있는 점이며, 최근접이웃 g를 s_e와 공유한다. 최종결과는 SL에 남아있는 분할점과 분할점의 최근접 이웃이 반환된다. 슬랩을 이용한 탐색 구간의 제한은 질의 구간으로부터 멀리 떨어져 있어 조사대상이 되지 않는 대상 객체들을 탬색에서 제외시킴으로써 탐색 비용의 절감을 가져온다. 이는 사용자가 위치기반서비스의 결과를 보다 빨리 받아볼 수 있게 한다.



(그림 6) 슬랩에 의한 탐색영역 분할과 SL갱신 (SL={s_0(.NN=f), s1(.NN=a), s2(.NN=g), s_e(.NN=g)})

3.2 탐색 알고리즘

슬랩을 이용한 연속적 최근접 이웃 탐색 알고리즘은 그림 7과 같다. 데이터 집합 P={p1, p2, …, pn}은 무작위 함수에 의해 생성된 N개의 점집합이고, p_j 는 (p_j.x , p_j.y)∈P인 P의 임의의 원소이다. NN이 변경되는 질의 구간 q의 점인 분할점의 리스트 SL은 분할점 s_i=(s_i.x, s_i.y)와 s_i.NN=p_j의 순서쌍 (s_i, p_j)의 리스트이다. s_i.NN은 s_i의 최근접 이웃을 나타낸다. 초기 SL은 SL={s_0, s_e}로 설정된다. 또한, 분할점 s_i를 중심으로 하고 최근접 이웃 p_j에 대한 최소경계원 (Minimum Bounding Circle, MBC)의 반지름 rsi는 다음과 같다.

$$rs_i = |s_i, p_j| = \sqrt{(s_{ix} - p_{jx})^2 + (s_{iy} - p_{jy})^2}$$

연속적 최근접 이웃인 SL을 구하는 과정은 그림 7의 알고리즘과 같이 먼저, 탐색구간인 s_i(s_0, s_e) 그리고 구간의 한점인 p_j사이의 거리를 구한다. 그 결과 s_i와 p_j사이에 가장 가까운 최근접이웃(NN)을 선택한다 (2-3줄). 다음으로 s_0의 슬랩과 s_e의 슬랩 사이의 공간에 있지 않은 점들을 제거함으로써 탐색범위를 줄인다 (4-5줄). 이제 반복문을 실행하여 s_i의 최소경계원(MBC)에 포함되지 않고 남은 q의 구간의 중점 s_j 와 최근접 이웃 s_j.NN을 검색하고 s_j를 중심으로 하는 최소경계원을 생성한다 (6-8줄). 탐색 구간에 있는 각점의 최근접이웃 사이의 수직 이등분선을 긋고 q의 교점 S_k 와 최근접 이웃 S_k.NN을 계산한다. 계산된 S_k와 S_k.NN은 SL에 삽입하여 목록을 갱신한다 (10-11줄). 다음으로 할 일은 S_k의 최소경계원을 생성하고 그 안에 더 가까운 점이 있는지를 검사한다. 이때, p_s는 s_{k-1}.NN의 슬랩과 s_k.NN의 슬랩 사이의 점일 경우 S_k와의 거리를 측정하여 더 가까운 점 p_s가 있으면 SL에 (s_k ,p_s)을 삽입하고 갱신한다 (12-14줄). 연속적최근접 이웃에 대한 결과는 리스트 SL에 저장되며, 알고리즘은 SL을 반환하게 된다 (15줄).

```
Procedure: SCNN_Search(q, P)
Input: query line segment q, dataset P
Output: final answer set
1. Begin FindNN
2.    compute dist(si, pj); //s0, sc와 pj사이 거리
3.    choose si.NN with min(dist(si, pj)); //NN 결정
4.    if(pj.x < s0.NN or pj.x > se.NN)
5.        delete pj from P;
6.    while((qL - Σrsi) > 0){
7.    search sj and sj.NN;
8.    create MBC of sj; //sj의 MBC 생성
9.    }
10.   compute sk and sk.NN;
11.   update SL with (sk, sk.NN);
12.   create MBC of sk; //sk의 MBC 생성
13.   if(rsk ≥ dist(sk ,ps))
14.       insert (sk ,ps) into SL;
15.   return SL; //최종 결과값으로 SL을 반환
16. End FindNN
```

(그림 7) 슬랩을 이용한 연속적 최근접 이웃 탐색 알고리즘

4. 실험 및 성능 평가

4.1 실험 환경

성능평가에 사용된 시스템은 펜티엄-Ⅳ 2.0GHz 프로세서에 256MB의 메모리를 가지며, 윈도우 XP 환경에서 실험하였다. 질의 구간은 무작위로 생성하였으며 구간의 길이는 파라미터(parameter)로써 설정하였다. 전체 성능은 100개의 질의의 수행결과의 평균값으로 평가하였으며, 성능평가에 사용된 데이터 집합은 R-tree Portal [7]의 실제 데이터 집합을 사용하였다. 또한, 질의는 직선 구간 q=[s_0, s_e]로 주어지고, 대상 데이터 집합은 정적 데이터 집합으로 주어진다. 색인 구성 시 한 노드의 크기는 4KB로 설정하였고, 정적 데이터 집합이므로 R-tree를 사용하여 색인하였다 [8][9].

성능평가에 사용된 옵션은 표 1과 같다. 먼저, 데이터 집합에서 전체 질의를 수행하는데 소요되는 CPU 시간을 비교하기 위해 CPU의 클럭을 측정하였다. 또한 질의 구간의 크기를 변경시키면서 질의를 처리하는데 소요되는 시간을 측정하였다.

(표 1) 성능평가 옵션

항목	값
질의 구간 길이(%)	전체영역의 5~30
데이터 집합의 객체 수 [개]	35,000~600,000
CPU 비용 [sec]	CPU 클럭 측정값
노드 방문 [회]	R-tree의 노드 방문회수

질의는 다음과 같이 생성된다. 첫째, 질의 구간의 시작점은 데이터 영역에 균등하게 분포한다. 둘째, 질의 구간이 x축과 이루는 각은 [0, 2π) 사이의 임의의 값을 가진다. 셋째, 동일한 작업부하를 갖는 모든 질의에 대해 질의 구간의 길이는 고정된다.

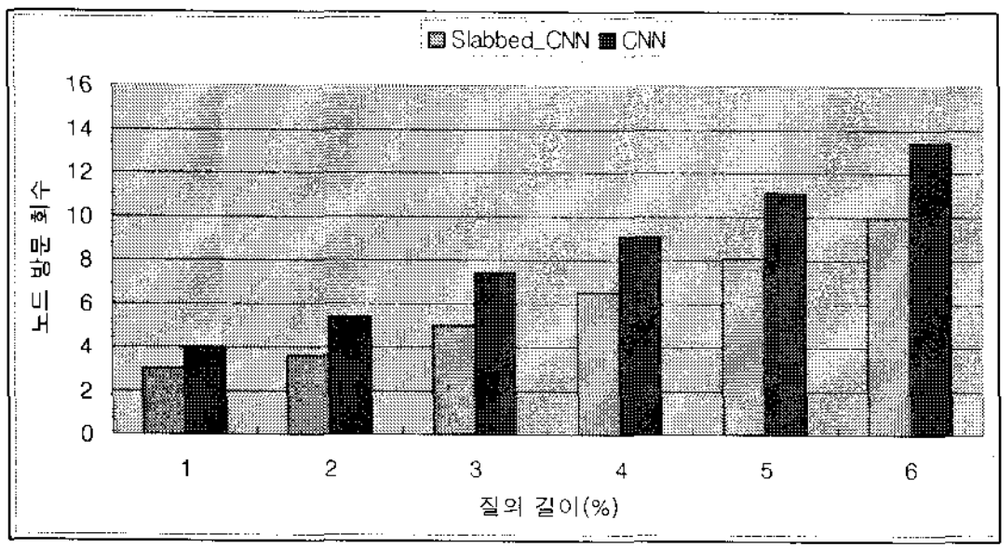
4.2 기존 기법과의 비교 평가

데이터 집합에 대해 질의 구간의 길이를 전체 데이터 영역의 5%~30%까지 변화시키면서 *Slabbed_CNN* 탐색기법과 CNN 탐색기법의 노드 방문회수를 측정하여 각 기법의 성능을 비교한 결과는 그림 8과 같다. 이때, 제안 기법의 노드 방문회수가 CNN의 노드 방문회수보다 더 적은 것을 알 수 있다. 이것은 슬랩을 이용하여 탐색 영역을 줄임으로써 처리해야 할 데이터 수가 감소하였기 때문이다.

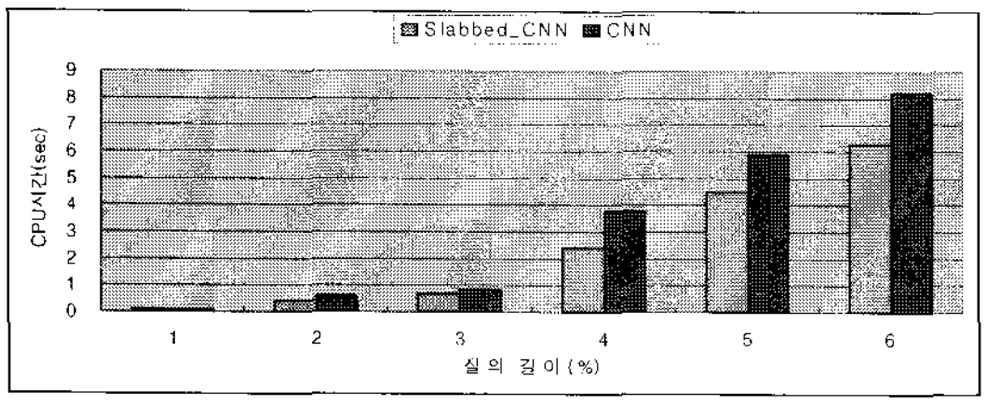
그림 9는 두 기법의 CPU 오버헤드를 비교한 것으로써 질의 구간의 길이 증가에 따른 CPU 클럭을 측정한 결과를 보여주고 있다. 질의 구간의 길이가 증가할수록 두 기법간의 시간 차이가 커지고 있다. 이것은 CNN 기법이 *Slabbed_CNN* 기법에 비해 질의 구간의 길이가 길어질수록 더 많은 거리계산을 유발하기 때문이다.

실험에 나타난 것처럼 질의 구간의 길이가 길어질수록 탐색해야 할 영역과 그 영역 내의 처리해야 할 데이터가 증가하기 때문에 두 기법간의

성능 차이는 커진다. 즉, 질의 구간의 길이가 증가할수록 거리 계산을 발생시키는 분할점의 숫자가 증가하기 때문에 CPU 시간이 증가하게 된다. *Slabbed_CNN* 탐색기법에서는 최근접 이웃의 수직 슬랩을 이용하여 탐색영역을 분할하고 분할된 영역 내에 있는 데이터에 대해서만 처리하기 때문에 기존의 CNN 탐색기법보다 거리 계산이 적다.



(그림 8) 질의 길이에 따른 노드 방문회수



(그림 9) 질의 길이에 따른 CPU 시간

5. 결론 및 향후 연구

이동객체에 대한 최근접 이웃 질의를 처리하기 위한 기존의 기법들은 특정 타임스탬프(timestamp)에서 공간 좌표를 비교하여 가장 가까운 객체를 찾는 것이 대부분이었다. 하지만 질의 객체가 동적인 경우, 즉 직선 구간으로 주어지는 경우, 특정 시간에서의 최근접 이웃의 정보는 그 시간이 지나면 잘못된 정보가 될 수 있기 때문에 적합하지 않다. 질의 구간에 대한 최근접 이웃의 정보는 CNN 질의 처리 기법으로 처리하는 것이 더 정확한 결과를 얻을 수 있다. 그러나 기존의 CNN 탐색에 관한 연구들은 질의의 동적 속성은 고려했지만 대상 데이터 객체들의 처리 순서와 분포에 따른 탐색영역과 계산비용은 고려하지 않았다. 따라서 데이터의 처리 순서와 분포에 따라 탐색영역과 계산비용이 일정하지 않아 좋은 질의 성능을 보장할 수 없었다.

본 논문에서는 대상 데이터 객체들의 처리 순서에 따른 탐색영역과 계산비용을 줄이기 위해 슬랩을 이용한 연속적인 최근접 이웃 탐색기법인 *Slabbed_CNN*을 제안하였다. 제안 기법을 통해 기존 기법의 높은 처리 비용과 불필요한 연산을 방지할 수 있었으며, 질의점이 직선 구간 위에서 계속적으로 이동하여 시간에 따라 질의의 위치가 달라지는 연속적인 최근접 이웃을 기존 기법보다 효율적으로 탐색한다. 향후에는 슬랩을 수직, 수평에 모두 적용하여 탐색영역을 감소시킴으로써 탐색 성능의 향상을 꾀하고자 한다.

참 고 문 헌

[1] J. Kim, S. J. Im, S. W. Kang, C. S. Hwang, S. K. Lee, "SQR-tree: A Spatial Index Using Semi-quantized MBR Compression Scheme in R-tree", Journal of Information Science and Engineering (JISE), Vol. 23, No. 5, pp.154-1563, 2007.

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

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

[4] Y. Tao, D. Papadias, and Q. M. Shen. "Continuous Nearest Neighbor Search", Proceeding of VLDB '02, 2002.

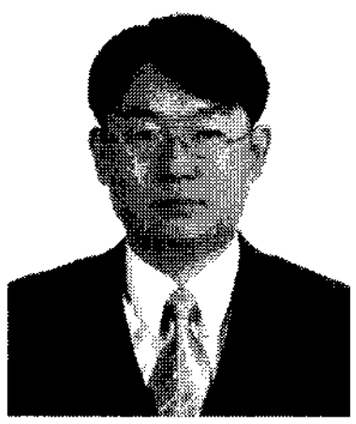
[6] Mark de Berg, Marc van Kreveld, Mark Overmars, Otfried Schwarzkopf. "Computational Geometry : Algorithms and Applications - Ch 6. Point Location", Springer - Verlarg 2nd rev. ed. 2000.

[7] http://www.rtreeportal.org/spatial.html
[8] A. Guttman, "R-trees : A Dynamic Index Structure for Spatial Searching", ACM SIGMOD, 1984.
[9] T. Sellis, N. Roussopoulos, C. Faloutsos, "The R^+-tree : a Dynamic Index for Multi-Dimensional Objects", VLDB, 1987.

◐ 저 자 소 개 ◑

한 석
2005년 고려대학교 정보통신대학원 컴퓨터학과 졸업(이학석사)
1998년 ~ 현재 한미 연합군 사령부 군수참모부 군수준비태세처
관심분야 : 이동 객체 데이터베이스, 이동 컴퓨팅, 유비쿼터스시스템
E-mail : seok0446@hotmail.com

오 덕 신
2000년 상명대학교 대학원 경영학과 박사수료
1997~현재 삼육대학 경영정보학과 부교수
관심분야 : 경영정보시스템, 전자상거래, e-Learning
E-mail : ohds@syu.ac.kr



김 종 완
2007년 고려대학교 대학원 컴퓨터학과 졸업(이학박사)
현재 삼육대학교 경영정보학과 외래 교수
관심분야 : 위치기반서비스(LBS), 모바일데이터/센서데이터 관리, 소프트웨어공학
E-mail : wany@korea.ac.kr