이동객체의 궤적에 대한 연속 최근접 질의 처리

(Continuous Nearest Neighbor Query Processing on Trajectory of Moving Objects)

지 정 희 † 최 보 윤 † 김 상 호 †† 류 근 호 †††

(Jeong Hee Chi) (Bo Yoon Choi) (Sang Ho Kim) (Keun Ho Ryu)

요 약 최근 위치 기반 서비스 기술에 관한 관심이 증가하면서, 시간에 따라 연속적으로 변하는 이동 객체에 관한 많은 연구들이 활발하게 수행되고 있다. 또한 이 시스템들이 자주 사용되는 질의 처리 기법 중 하나인 최근접(nearest neighbor, NN) 질의에 대한 연구도 다양하게 수행되고 있다. 그러나, 기존의 최근접 질의 처리 기법들은 질의와 객체가 이동하면 그들이 결과가 유효하지 않게 되므로, LBS를 위한 이동객체 관리 시스템에는 적합하지 않을 수 있다.

이러한 문제들을 해결하기 위해서 이동객체에 대한 정확하고 연속적인 질의 처리가 가능한 새로운 최근접 질의 처리 기법을 제안하였으며, 이를 연속 궤적 최근접(continuous trajectory NN, CTNN) 질의라 부른다. 이 논문에서는 빠른 응답 시간을 얻기 위한 근사 연속 궤적 최근접(approximate CTNN, ACTNN) 질의 처리 기법과 정확한 최근접 탐색을 가능하게 하는 정확 연속 궤적 최근접(exact CTNN, ECTNN) 질의 처리 기법을 제안하였다. 우리는 여러 데이타 셋을 기반으로 실험을 하였으며, 실험결과는 제안된 ECTNN 기법의 경우 정확도는 상당히 높은 반면, 응답시간은 약간 낮은 성능을 보였다. 또한 ACTNN 기법의 경우 정확도는 ECTNN 기법에 비해 낮은 반면, 응답시간은 높은 성능을 보였다.

제안된 기법들은 항해 시스템, 교통 통제 시스템, 물류정보 시스템 등 각종 위치 기반 서비스에 다양하게 사용될 수 있고, 특히 질의 객체와 데이타 객체가 모두 이동 점 객체이면서 이들의 궤적 정보를 미리 파악할 수 있는 경우에 가장 적합하다.

키워드 : 최근접 질의, 이동객체, 연속 최근접 질의

Abstract Recently, as growing of interest for LBS(location-based services) techniques, lots of works on moving objects that continuously change their information over time, have been performed briskly. Also, researches for NN(nearest neighbor) query which has often been used in LBS, are progressed variously. However, the results of conventional NN query processing techniques may be invalidated as the query and data objects move. Therefore, they are usually meaningless in moving object management system such as LBS.

To solve these problems, in this paper we propose a new nearest neighbor query processing technique, called CTNN, which is possible to meet accurate and continuous query processing for moving objects. Our techniques include an Approximate CTNN(ACTNN) technique, which has quick response time, and an Exact CTNN(ECTNN) technique, which makes it possible to search nearest neighbor objects accurately. In order to evaluate the proposed techniques, we experimented with various datasets. Experimental results showed that the ECTNN technique has high accuracy, but has a little low performnce for response time. Also the ACTNN techinque has low accuracy comparing with the ECTNN, but has quick response time.

The proposed techniques can be applied to navigation system, traffic control system, distribution information system, etc., and specially are most suitable when both data and query are moving objects and when we already know their trajectory.

Key words : nearest neighbor query, moving objects, continuous nearest neighbor query

---

· 이 연구는 대학 IT 연구센터 육성 사업, BK 사업단, 과학기술부 RRC(청주대학교 정보통신연구센터)의 연구비 지원으로 수행되었음

† 비 회 원 : 충북대학교 전자계산학과
jhchi@dblab.chungbuk.ac.kr
bychoi@dblab.chungbuk.ac.kr

†† 비 회 원 : 충북대학교 연구원
shkim@dblab.chungbuk.ac.kr

††† 종신회원 : 충북대학교 전기전자 및 컴퓨터공학부 교수
khryu@dblab.chungbuk.ac.kr

논문접수 : 2003년 11월 27일
심사완료 : 2004년 7월 19일

1. 서 론

최근 무선 인터넷 및 모바일 컴퓨팅 기술이 급속히 발전함에 따라 위치 기반 서비스 관련 기술 개발이 활발히 진행되고 있다. 이 서비스들은 실 세계 시공간 데이타에 다양한 질의 처리 기법이나 효율적인 데이타 저장 및 관리 기법들을 적용하여 사용자에게 원하는 정보를 제공하고자 한다. 우리가 자주 사용하는 질의 처리 기법 중 하나인 최근접 질의는 사용자의 선택 위치와 가장 가까운 곳에 존재하는 객체를 결과로 반환하며, 위치 기반 서비스 내에서 다음의 예제와 같이 적용될 수 있다: A씨는 집에서 핸드폰으로 가장 가까운 곳에 위치한 피자집을 검색하여 피자를 주문한다(정적-정적). 운전 중인 B씨는 자신의 차에 장착된 네비게이터(navigator)를 통해서 가장 가까운 곳에 위치한 주유소를 검색한다(동적-정적). 항해 중인 선박 C는 근처에서 항해하고 있는 주유선을 검색하여 급유를 요청하고자 한다(동적-동적). 고속버스 운전기사 D씨는 고속도로 위에서 사고를 겪었다. D씨는 가장 먼저 자신의 위치에 도착 가능한 버스를 검색하여 도움을 요청하였고, 승객들은 모두 다른 버스를 이용하여 목적지에 무사히 도달할 수 있었다(정적-동적). 이 때, 괄호 안의 요소는 질의-데이타 객체 타입을 나타낸다.

이와 같이 위치 기반 서비스들은 여러 타입의 데이타 객체들에 대한 질의를 처리하며, 효율적인 성능을 위해 이에 따른 여러 기법들이 제안되었다. 공간 객체와는 달리, 이동객체들은 시간의 흐름에 따라 연속적으로 자신의 위치를 변경하며 움직이기 때문에 어떤 한 시점에서 계산된 최근접 질의 결과는 다른 시간에서 유효하지 않을 수 있다. 하지만 대부분의 시스템이 사용하는 최근접 질의 처리들은 이러한 속성을 무시한 채, 이동객체 궤적의 시작 시점에서만 객체간 거리를 비교하여 최근접 객체를 선택하거나, 샘플링 된 일정 간격 시간점 위에서 거리를 비교하여 결과를 반환한다. 따라서 전체 질의 시간에 대해 항상 유효한 정보를 사용자에게 제공할 수 없다. 이를 해결하기 위해서 연속 또는 영속 질의 처리 개념[1]을 도입한 기법들이 제안되기도 하였지만, 이들 역시 객체 궤적 전체에 대한 최근접 질의를 행하지 않기 때문에 정확한 결과를 제공하기에는 미흡하다. 즉 이들 기법은 서로 가까워지는지 멀어지는지 등, 궤적 정보를 전혀 고려하지 않기 때문에 부정확한 결과를 보이며, 또한 하나의 최근접 객체를 선택해야 하는 질의에서 여러 후보 객체를 가지는 경우, 결과 선택 기준이 명확하지 않기 때문에 임의로 결과를 선택하게 되며 이는 부정확한 결과를 낳을 수 있다.

따라서 이 논문에서는 기존 연구의 문제점을 해결하기 위하여, 객체간 궤적 정보를 비교하면서 연속적으로 질의와 가장 가까운 거리를 유지하며 움직이는 객체를 찾는 것이 가능한 새로운 최근접 질의 처리 기법인 연속 궤적 최근접(CTNN) 질의 처리 기법을 제안하였다. 이 기법은 궤적 정보 비교를 통해 질의와 가장 가까운 위치를 유지하며 움직이는 객체를 정확히 탐색할 뿐만 아니라 여러 후보 객체가 있을 때 k값(1또는 n)에 따라 결과를 선택할 수 있는 기준을 제공하며, 질의 객체의 궤적 상에서 최근접 객체 정보가 변경되는 시점을 연속적으로 지정한 후 그 시간점 위에서 결과가 변경되는지 여부를 비교함으로써 질의 전체에 대해 항상 유효한 결과를 얻을 수 있도록 하였다. 또한, 이 논문에서는 응답 시간을 빠르게 하기 위한 근사 연속 궤적 최근접(Approximate CTNN, ACTNN) 기법과 정확한 최근접 객체를 탐색하기 위한 정확 연속 궤적 최근접(Exact CTNN, ECTNN) 기법을 제안하였다.

논문의 나머지는 다음과 같이 구성되어 있다. 2절에서는 기존 연구들을 살펴보고, 3절에서는 이 논문에 적용된 기본 모델과 가정들을 살펴보며, 4절에서는 이 논문에서 제안하는 CTNN 기법들에 대한 질의 처리 과정과 알고리즘을 소개하고, 5절에서는 제안 기법의 성능평가를 보인다. 그리고 마지막으로 6절에서 결론 및 향후 연구를 제시한다.

2. 관련 연구

최근접 질의 처리 기법들은 멀티미디어 데이타베이스[2], 데이타 마이닝[3], 공간·시공간·이동체 데이타베이스[4-8] 등 광범위한 분야에서 연구되고 있으며, 최근접 질의 처리를 위한 인덱스나[9,10], 응답 시간을 줄이기 위한 근사(approximate) 질의 처리 기법[11-14]이 제안되기도 하였다. 이 절에서는 질의 객체와 데이타 객체가 모두 이동객체인 경우에 적용 가능한 최근접 질의 처리 기법들을 살펴본다.

[15]는 쌍대성 변환(Duality Transform) 기법을 사용하여 (x, y) 평면 상의 이동객체 궤적 선분들을 (v, a) 평면 (속도, y절편) 상의 점으로 변환하고, 점 간의 유클리안 거리를 계산하여 최근접 객체를 선택한다. 이는 모든 질의-데이타 객체 타입에 적용 가능하지만 연속적인 최근접 탐색이 불가능하며 3차원 (x, y, t) 공간 상의 선분으로 표현되는 이동객체 궤적에는 부적합하다.

[16]은 TPR-tree[17]을 사용하여 데이타를 인덱스하고, DF 탐색 방법[4]에 따라 미분 함수를 사용하여 노드 사각형과 질의점 사이의 거리를 계산한 후 [now, now+△] 간격마다 가장 가까운 거리를 가지는 객체를 찾아 결과로 반환하는, 영속 최근접 질의 처리 기법을 제안하였다. 이는 단순 탐색 기법을 따르기 때문에 한 데이타에 대해 여러 번 중복된 거리 계산을 행하기도

하고, 이전에 탐색된 최근접 객체가 현재에도 결과로서 유지되는지 아닌지를 비교하는 작업들을 자주 요구한다. 또한 거리 계산 함수는 객체의 위치나 속도에 의존적이기 때문에 이 값들이 변경될 때마다 함수가 재계산되므로 큰 오버헤드를 초래할 수 있다.

[18]은 TPNN(time-parameterized)질의와 CNN(continuous)질의를 제안하였다. 이들은 최근접 객체를 선택하면서, 전체 질의 시간 선분 위에서 최근접 객체 정보가 변경될지도 모르는 (TPNN) 또는 변경되는 (CNN) 시간점을 계산하고, 각각의 최근접 객체 정보와 유효시간 인터벌을 시간 리스트 TL에 저장하였다가 질의 마지막 시간에 결과로 반환하는 연속 최근접 질의이다. 즉, 결과는 TL={<a, [t_1, t_2]>, <b, [t_2, t_3]>}과 같이 여러 개의 연속적이고 분리된 하위 시간 인터벌과 각 인터벌 내에서 유효한 최근접 객체 정보로 구성된다. TPNN질의는 모든 객체에 대해 질의와 가장 가까워지는 가까운 미래 시간을 미리 계산하고, 계산된 시간 값에 따라 순서대로 계산된 시간 점 위에서 실제 최근접 객체가 되는지를 검사한 후, 결과를 선택한다. TPNN 질의의 계산적 오버헤드를 줄이기 위하여 제안된 것이 CNN 질의이며, 이는 질의 시작 시간에서 객체들과 질의 q 사이의 거리를 계산한 후 거리를 오름차순으로 정렬하고 첫 번째 최근접 객체 p를 선택한다. 다음으로 다른 객체가 p와 q 사이에 생성된 근접원(vicinity) 안에 포함되는지를 거리 순서대로 비교하고, 포함된다면 그 객체를 최근접 객체 p′로 선택하면서 유효시간 인터벌을 계산한다. 그리고 p′와 q 사이에 또 근접원을 그리고, 동일 과정을 질의 마지막 시간까지 반복한 후 TL내의 결과를 반환한다. 질의와 데이타가 모두 이동객체인 경우를 위한 CkNN 질의는 최근접 객체가 갱신되는 시점의 타임스탬프 집합으로 구성된 분할 리스트를 유지하여, 분할 리스트에 유지되는 타임스탬프 상에서의 kNN 객체를 탐색하게 되며, 따라서 다른 타임스탬프는 고려할 필요가 없게 된다. 그러나, CkNN 기법의 분할 리스트를 결정하기 위해서는 최근접 객체가 갱신되는 순간을 다루기 위해 모든 시간에서의 최근접 객체에 대한 갱신여부를 체크하게 된다. 또한, CkNN 기법을 이동객체의 궤적에 적용할 경우 타임스탬프 단위에 따라 이동객체의 위치를 추정하여 최근접 객체 여부를 비교하게 되므로, 타임스탬프 단위에 따라, 예를 들면 1분, 10분, 1시간, 서로 다른 객체가 최근접 객체로 선택되게 되며, 따라서, 부정확한 결과값을 반환할 수 있다.

3. 질의 모델

이 절에서는 CTNN 질의의 기본 데이타 구조와 질의 타입을 설명하며, 이 논문이 포함하고 있는 몇 가지 가정 및 표기들을 기술한다.

3.1 데이타 모델

이 논문은 시간의 흐름에 따라 자신의 위치를 변경하며 연속적으로 움직이는 이동 점객체 예를 들면, 자동차, 사람, 비행기, 선박 등에 대한 최근접 질의를 대상으로 한다. 객체의 위치 정보는 GPS 등의 장치를 통하여 주기적으로 추출되고, 객체의 삽입, 삭제, 속도나 이동 방향의 변경 등과 같은 갱신이 발생할 때 데이타베이스에 저장된다. 즉, 이동객체는 갱신이 발생되는 각 시간 t에서 <id_i, (x, y), t> 형태로 저장되며, 이 때 id_i는 객체 id의 i번째 저장 정보를, (x, y)는 시간 t에서의 공간 좌표를 나타내고, 속도나 방향 정보는 저장하지 않는다.

정의 1. 궤적 Tr_i는 이동객체 i가 움직인 경로이며, 3차원 공간 (x, y, t) 상의 폴리라인(polyline) 즉, 선분 세그먼트 S들의 집합으로 표현된다. 즉 n개의 선분 세그먼트로 구성된 이동객체 i의 궤적 Tr_i는 다음과 같다. 여기서, S_{in}은 i번째 객체의 n번째 세그먼트를 의미한다.

$$Tr_i = \{ S_{i1}, S_{i2}, S_{i3}, \cdots, S_{in}\}$$

정의 2. 이동객체의 궤적 Tr_i을 구성하는 선분 세그먼트 S_{in}는 데이타베이스 내에서 연속되어 나타나는 두 개의 데이타 점으로 구성된다. 여기서 id_n은 객체 i의 식별자는 id이고, n번째 갱신되었음을 나타내며, x_n, y_n, t_n은 각각 n번째 공간 좌표와 시간을 나타낸다.

$$S_{in} = \{<id_n, (x_n, y_n), t_n>, <id_{n+1}, (x_{n+1}, y_{n+1}), t_{n+1}>\}$$

이동객체 궤적은 크게 두 가지, 자유궤적과 제약궤적으로 구분된다[19]. 제약궤적은 자유궤적의 특별한 경우로 철로 위의 기차, 고속도로 위의 자동차 등과 같이 객체의 이동 경로가 제한된 것을 말한다. 우리는 미리 이동 경로를 예측할 수 있는 객체를 '제약궤적을 가진 객체'로 간주하며, 이 논문에서 제안되는 CTNN 기법들은 버스, 비행기, 기차 등 제약궤적 이동 점객체에 적용 가능하다.

데이타베이스에 저장되지 않은 시점에서의 공간좌표는 다음의 시간 종속적 선형 위치 추정 함수를 사용하여 추정될 수 있다.

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

이 함수는 데이타베이스에 저장된 시간 t_i와 t_{i+1}에서의 좌표값을 이용하여, 그 사이에 존재하는 시간 t에서의 객체 위치를 계산하기 때문에, 한 선분 세그먼트의 양끝 점 정보만으로 한 선분 세그먼트 위의 객체 공간 점 위치를 추정할 수 있다.

3.2 질의 타입

범위(range) 질의는 다른 타입의 질의를 처리하는 과

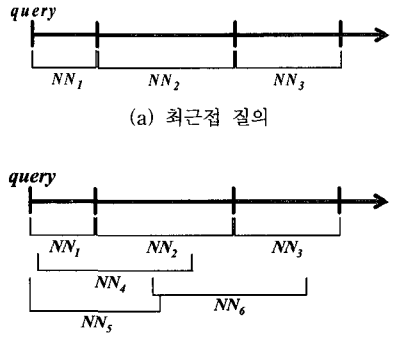
정에서 데이타의 양을 줄이는 전처리 작업으로 많이 사용되며, CTNN 질의 역시 시간 범위질의를 사용하여 검색대상이 되는 데이타 객체들을 필터링 한다. [20]에 따르면 시공간 최근접 질의는 크게 두 가지, 시간 범위 질의와 공간 최근접 질의의 합, 공간 범위 질의와 시간 최근접 질의의 합으로 구분된다. CTNN은 기본적으로 전자의 경우를 따르며, 좀 더 엄밀히 말해서 최근접 객체가 선택된 한 시점으로부터 결과가 변경되는 가장 가까운 미래 시간을 빠르게 찾아내고 그 시간점에서 공간적으로 가장 가까운 위치를 가지는 객체를 선택하기 때문에 공간 최근접 탐색과 시간 최근접 탐색이 동시에 가능한 시공간 최근접 질의 타입이다.

또한 전체 질의 시간 상에서 최근접 객체가 변경될 시점을 선택한 후 그 시간점에서 실제 변경 여부를 판단하여 저장하기 때문에 항상 유효하고 연속적인 결과를 얻을 수 있다. 즉, 연속 최근접 질의 타입을 가진다.

그리고 CTNN 질의는 사용자가 몇 개의 최근접 객체를 결과로 원하는지에 상관없이 조건을 만족시킬 수 있다는 장점을 가진다. 즉, 최근접 질의, k-최근접 질의가 모두 가능하며, k값은 사용자의 지정에 따르거나 결과로 나타날 수 있는 모든 후보 객체의 갯수가 될 수 있다. 그림 1은 최근접 질의와 k-최근접 질의의 결과 객체 정보와 유효시간 인터벌간 차이를 보인다. 예를 들면, 질의 시간이 $[t_s , t_e]$로 주어졌을 때, 그림 1(a)의 최근접 질의 결과는 $TL = \{<NN_1 , [t_s , t_1]>, <NN_2 , [t_1 , t_2]>, <NN_3 , [t_2 , t_e]>\}$ 형태가 된다. 즉, 전체 질의 시간 상에서 각각의 개별적이고 분리된, 연속적인 유효시간 인터벌과 각 인터벌 내에서 유효한 하나의 최근접 객체 정보로 구성된다. 또한 그림 1(b)의 k-최근접 질의 결과는 $TL = \{<NN_1 , [t_s , t_3]>, <NN_4 , [t_1 , t_5]>, <NN_5 , [t_s , t_4]>, <NN_2 , [t_3 , t_6]>, <NN_3 , [t_6 , t_e]>\}$ 형태가 되며, 이는 시간 인터벌이 서로 겹치는 여러 하위 유효시간 인터벌과 해당 인터벌 내에서 유효한 최근접 객체들을 포함하게 된다.

3.3 가정 및 표기

CTNN 질의는 다음의 가정을 바탕으로 처리된다.



(a) 최근접 질의

(b) k-최근접 질의

그림 1 CTNN 질의에서의 결과 유효시간 인터벌

1. 어떤 객체 p가 최근접 객체로 선택된 후 1분 이내에 변경된다면, p는 무시되고 유효시간 인터벌은 이전 또는 이후 인터벌에 포함된다. 이 시간 인터벌은 하나의 결과를 표현하기에는 짧은 시간이라고 판단되며, 실 세계에서는 무의미하게 사용될 수 있다. 그렇지만, 응용에 따라 시간 단위를 변경할 수 있다.
2. 두 객체 사이의 거리가 1m 이내인 경우, 이 객체들은 서로 만나거나 교차한다고 간주한다. 실 세계에서 객체들이 같은 시간에 같은 좌표를 가지는 경우는 충돌을 의미하며, 이러한 상황은 거의 발생하지 않는다. 따라서 어떤 시간 점 위에서 1m 이내에 존재하는 두 객체는 같은 시간에 동일 위치에 존재하는 객체로 간주한다.

그리고 표기들은 표 1과 같이 사용한다.

4. 연속 궤적 최근접 질의

이 절에서는 CTNN 질의에 사용되는 기본 알고리즘과 계산 과정들을 소개한 후, 이 논문이 제안하는 두 가지 질의 처리 기법, ACTNN 과 ECTNN의 알고리즘을 기술한다. 그리고 k-CTNN 질의로의 확장에 관해 기술한다.

4.1 CTNN 질의 기본 알고리즘

CTNN 질의는 먼저, 주어진 질의 시간 범위 내에 존

표 1 표기

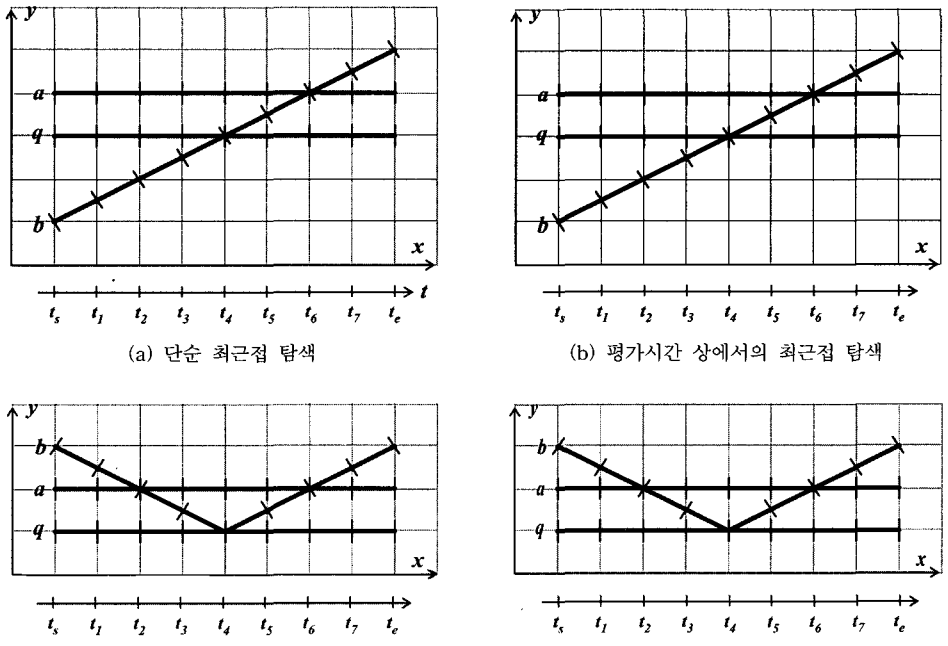
$\square a_i$	객체 a의 i번째 세그먼트 ($i \geq 1$ 정수)
(x_a^t, y_a^t)	시간 t에서 객체 a의 공간 점 좌표
(x_{ai} , y_{ai})	객체 a의 i번째 저장 좌표
Δx (Δy)	$x(y)$ 축에서 객체가 단위 시간당 움직인 거리
x_{ab} (y_{ab})	객체 a, b의 $x(y)$ 좌표 차이 값
$\|a - b\|^t$	시간 t에서 a, b사이의 공간 거리 = $\sqrt{(x_{ab}^t)^2 + (y_{ab}^t)^2}$
$/p$	세그먼트 p의 기울기 = $\Delta x / \Delta y$
∂p	세그먼트 p의 변위 값. 객체가 단위 시간당 움직인 거리

재하는 모든 살아 있는 데이타 객체들을 검색하고 질의 시간 인터벌로 이 세그먼트들의 시간 인터벌을 클리핑(Clipping)한 후, 최근접 질의 처리를 통해 질의 궤적과 가장 가까운 거리를 유지하는 세그먼트를 결과로 선택한다. 최근접 객체를 선택하는 과정은 크게, 연속 질의를 위한 평가시간 설정 단계와 궤적 질의를 위한 객체 궤적 정보 비교 단계로 구성된다. 평가시간이란, 질의 객체의 전체 궤적 선분 상에서 최근접 객체가 획득되는 시간, 또는 최근접 객체 정보가 변경되는 시간을 말한다. 즉, 질의가 평가되는 시간이며, 탐색된 결과의 유효성을 판단하기 위해서 질의가 재평가되는 시간점이다. 두 개의 연속된 평가시간 점은 하나의 평가시간 인터벌을 구성하게 되고, 각각의 최근접 객체들은 탐색될 때마다 자신이 유효하게 존재하는 평가시간 인터벌과 함께 시간 리스트 TL에 {<a, [t_1, t_2]>, <b, [t_2, t_3]>} 형태로 저장되었다가, 질의 마지막 시간에 결과로 반환된다.

4.1.1 평가시간 설정

평가시간은 기본적으로 데이타 객체와 질의 객체를 포함한 모든 세그먼트의 양 끝점 시간 즉, 객체 갱신이 발생하는 각각의 변곡점들과 세그먼트 간의 교차점으로 구성된다. 그림 2에서 보이는 것처럼, 탐색된 최근접 객체 결과는 다른 세그먼트와 교차하는 시점에서 변경된다. 즉, 하나의 최근접 객체 p를 선택한 후 p가 어떤 시간 t 에서 p' 객체와 교차한다면, t시간 이전의 최근접 객체는 p, 이후 시간의 최근접 객체는 p'가 된다: TL = {<p , [t_s , t]>, <p' , [t , t_e]>}. 그림 2(a)와 같은 예제가 주어졌을 때, 단순 방법으로 세그먼트 시작 시점에서만 최근접 객체를 탐색한다면 결과는 TL = {<a , [t_s , t_e]>}가 될 것이다. 정확한 결과를 얻고자 한다면, 모든 시점에서 최근접 객체 정보 변경 여부를 파악해야만 하며, 총 7번의 거리 계산이 추가적으로 이루어진다 (t_1~t_7). 만일 그림 2(b)와 같이 교차점과 변곡점 위에서 최근접 객체를 계산한다면 TL = {<a , [t_s , t_4]>, <b , [t_4 , t_6]>, <a , [t_6 , t_e]>}의 결과를 얻게 될 것이다. 즉, 맨 처음 탐색된 객체 a는 객체 b가 질의 세그먼트와 교차하는 시간 t_4까지의 최근접 객체이며, 시간 t_4 이후의 최근접 객체인 b는 시간 t_6에서 객체 a와 교차하므로, 결과 정보는 또 한번 변경된다. 이는 단순 방법보다는 좀 더 정확한 결과를 제공하기는 하지만 충분히 정확한 결과를 제공하지는 못한다.

따라서 더욱 세밀한 세그먼트간 교차점 정보를 얻기 위해 CTNN 질의는 그림 2(c)와 같이 선분 대칭이동 단계를 추가한다. 즉, 질의 세그먼트를 기준으로, 모든 객체 세그먼트들을 한쪽 공간으로 모은 후 교차점 정보를 얻는다면, 그림 2(d)와 같이 좀 더 정확한 최근접 객체 정보를 얻을 수 있다.



(a) 단순 최근접 탐색 (b) 평가시간 상에서의 최근접 탐색

(c) 질의 궤적에 대한 선분 대칭이동 (d) 대칭이동 이후, 평가시간 상에서의 최근접 탐색

그림 2 CTNN 질의 기본 알고리즘 예

위에서 설명한 CTNN 질의에 대한 전체 평가시간은 데이타 객체와 질의 객체를 포함한 모든 세그먼트의 양 끝점 시간과 세그먼트 간의 교차가 발생한 시간으로 구성되며, 알고리즘 1를 통해서 설정된다. 먼저, 데이타 객체와 질의 객체를 포함한 모든 세그먼트의 양 끝점 시간은 EL에 저장된다. 예를 들면, 질의 세그먼트의 시작시간이 t_s 이고 끝시간이 t_e일 경우 EL={<t_s, t_e>} 형태로 저장된다.

알고리즘 1. Set_EvaluationTime

Input: 데이타 객체 세그먼트 집합 *seg*, 질의 세그먼트 *q*
Output: 교차점 리스트 *IL*, 변곡점 리스트 *EL*

1 *seg*, *q*의 세그먼트 양 끝점 → *EL*
2 **Find_Intersection( )** → *IL*
3 *EL*, *IL* 내의 시간 점 반환

그리고, 세그먼트 간의 교차가 발생한 시간은 알고리즘 2에 의해 IL에 저장된다. 먼저, 세그먼트간의 교차점을 탐색하기 위해 2차원 공간상의 선분 간의 교차점 추출을 위한 [21]의 스윕 라인 알고리즘(Sweep Line Algorithm)을 이용하여 교차점을 탐색하며, 탐색된 교차점의 교차 시간을 설정한다. 예를 들면 a 와 b가 교차점 ip에서 교차한 시간이 t_i 이라고 하면, 교차점 리스트 IL={<t_i, ip, (a,b)>}이 저장된다.

알고리즘 2. Find_Intersection

Input: 질의 객체와 데이타 객체 세그먼트 집합 *seg*
Output: 교차점 리스트 *IL*

1 교차점 탐색
  1.1 *seg*를 2차원 공간 상의 선분 세그먼트로 간주
  1.2 **Sweep Line Algorithm** → 교차점 *ip* 추출
2 *ip*의 교차 시간 *t* 설정 // 객체 A와 B가 각각 t_A, t_B 시간에 *ip*에 도달한다고 가정하였을 때,
  2.1 $t_A > t_B$ 라면 $t = t_B$
  2.2 이외의 경우 $t = t_A$
3 *ip* , *t* → *IL*
4 *IL* 반환

4.1.2. 변위 계산

궤적 최근접 질의의 기본 목적은 질의와 가능한 한 가까운 거리를 유지하면서 움직이는 객체를 최근접 객체로 선택하는 것이다. 따라서 각 평가시간 점에서 최근접 객체를 선택할 때, k값에 따라 궤적 질의 조건을 만족하는지 여부를 판단할 필요가 있다. 예를 들어, 한 평가시간 점 *t*에서 후보 객체가 1개라면 CTNN 질의는 더 이상의 비교 없이 그 객체를 결과로 선택한다. 하지만 정확한 결과를 얻기 위해서는 질의와 가장 가까운 거리를 가지는 객체가 여럿 존재하는 경우, 이 중 1개의 객체만을 최근접 객체로서 선택할 수 있는 기준이 필요하다. 이 경우, CTNN 질의는 후보 데이타 객체 세그먼트들과 질의 세그먼트 사이의 기울기나 변위 값 등을 비교하여 하나의 최근접 객체를 선택한다. 하지만 각각의 기울기, 방향, 변위 값을 계산한다면 시간적 오버헤드를 초래할 수도 있기 때문에 CTNN 질의는 알고리즘 3을 통해 1개의 최근접 객체를 선택하며, 이를 변위 계산이라 부른다.

알고리즘 3. Calculate_Displacement

Input: □*A* , □*B* // x_{ab}, y_{ab}는 객체 A, B의 *x* 와 *y* 좌표 차이 값
Output: 정수 *T*
// 두 세그먼트가 서로 일치하거나, 평행하는 경우
1 */A* == */B*이고 $\partial A == \partial B$인 경우,
  x_{ab} == y_{ab} == 0 이면 *T* = 1 // 두 세그먼트가 일치
  1.1 이외의 경우 *T* = 0 // 두 세그먼트가 평행
  // 두 세그먼트의 기울기가 서로 역관계인 경우
2 |*/A*| == 1/|*/B*|인 경우,
  // 서로 같은 방향으로 진행하는 세그먼트의 경우
  2.1 */A*의 부호 == */B*의 부호 일 때,
    2.1.1 x_{ab} == y_{ab} 이면 $T = x_{ab} / ||\Delta y|-|\Delta x||$
    2.1.2 이외의 경우 *T* = 0 // 두 세그먼트가 교차하지 않음
  // 서로 반대 방향으로 진행되는 세그먼트의 경우
  2.2 */A*의 부호 != */B*의 부호 일 때,
    2.2.1 $(|\Delta y|+|\Delta x|) / ||\Delta y|-|\Delta x|| == x_{ab} / y_{ab}$ 이면 $T = x_{ab} / (|\Delta y|+|\Delta x|)$
    2.2.2 이외의 경우 *T* = 0 // 두 세그먼트가 교차하지 않음
  // 두 세그먼트가 사로 대칭 관계에 있는 경우
3 */A* == -*/B*인 경우,
  // x축을 기준으로 대칭일 경우
  3.1 x_{ab} == 0 이고 y_{ab} != 0 이면 $T = y_{ab} / (|\Delta y|+|\Delta x|)$
  // y축을 기준으로 대칭일 경우
  3.2 y_{ab} == 0 이고 x_{ab} != 0 이면 $T = x_{ab} / (|\Delta y|+|\Delta x|)$
  3.3 이외의 경우 *T* = 0 // 두 세그먼트가 교차하지 않음
4 이외의 모든 경우, $X = x_{ab} / |\partial A - \partial B|$ 이고 $Y = y_{ab} / |\partial A - \partial B|$
  4.1 */A* > 1이고 */B* > 1 이라면,
    4.1.1 *X* > *Y* 이면 *T* = *X*
    4.1.2 이외의 경우 *T* = *Y*
  4.2 (*/A*와 */B*) > 0 이고 (*/A* 또는 */B*) < 1 이라면, $T = |x_{ab} - y_{ab}| / |\partial A - \partial B|$ (반올림)
  4.3 이외의 경우 *T* = *X* + *Y*

변위 계산은 두 세그먼트 사이에서 발생되는 교차 시점을 계산해낸다. 즉, 시간 t에서 객체 a와 q사이의 변위 계산 결과가 1이라면, 이는 a와 q가 t+1시간에 교차한다는 것을 의미한다. 후보 객체들과 질의 사이에 변위 계산을 수행하였을 때, 가장 작은 값을 가지는 객체는 질의와 가장 먼저 만나는 객체를 뜻하며, 이는 동시에 질의와 가장 가까운 거리를 유지하면서 움직이는 객체를 뜻한다. 따라서 여러 후보 객체가 있을 때, 각 객체와 질의 사이에 변위 계산을 행하고, 가장 작은 결과 값을 가지는 객체 순서대로 k값에 따라 선택함으로써 정확한 최근접 객체 결과를 얻을 수 있다. 이 때 음수는 두 세그먼트가 t보다 과거 시간에 교차하였다는 것을 의미하며, 0은 t시간 이후로 q와 만나지 않는 세그먼트, 즉 평행하게 움직이거나 멀어지는 방향으로 움직이는 세그먼트를 의미한다.

4.1.3 최근접 객체 탐색

CTNN 질의의 최근접 객체 탐색 단계는 기본적으로 다음의 알고리즘 4와 같이 처리된다. 여기서 우리는 1개의 최근접 객체를 선택하는 최근접 질의 처리를 가정한다.

알고리즘 4. Find_NNObject

Input: 데이타 객체 세그먼트 집합 *seg* , n개의 세그먼트 $(q_1,\cdots,q_n)$로 구성된 질의 객체 궤적 q
Output: 시간 리스트 *TL*

// 단계 수행 중 후보 객체가 여럿 존재하는 경우에는 항상 **Calculate_Displacement( )** 적용
1 q를 기준으로, 한 쪽 공간의 *seg*들을 다른 쪽 공간으로 대칭 이동
2 **Set_EvaluationTime( )**
3 $|seg_i - q_i|^{tsi}$ // q_i의 시작 시간 tsi
  3.1 *mindist* 가지는 최근접 객체 p 선택
  3.2 $TL = <p, [t_{si} , t_{ei}]>$ //□q_1의 끝 시간 t_{ei}
4 **Until** t_{el}
  4.1 t_i시간에 *IL*내의 교차점 ip_1이 존재할 때, // 단계 순서대로 조건이 만족하면 TL 갱신
    4.1.1 □q와 □p'가 교차한다면 $TL = \{<p, [t_{si}, t_i]>, <p', [t_i, t_{ei}]>\}$
    4.1.2 □p와 □p''가 교차한다면 $TL = \{<p, [t_{si}, t_i]>, <p'', [t_i, t_{ei}]>\}$
    4.1.3 이외의 경우 $TL = <p , [t_{si} , t_{ei}]>$
  4.2 *EL*내의 t_c시간에 객체 p'의 시작점이 존재할 때,
    4.2.1 $|p' - q|^{tc} < |p - q|^{tc}$ 이면 $TL = \{<p , [t_{si} , t_c]>, <p' , [t_c , t_{ei}]>\}$
    4.2.2 이외의 경우 $TL = <p , [t_{si} , t_{ei}]>$
  4.3 *EL*내의 t_d시간에 객체 p''의 끝점이 존재할 때,
    4.3.1 p'' == p이면 2단계 반복하고 최근접 객체 p'' 선택, $TL = \{<p , [t_{si} , t_d]>, <p'' , [t_d , t_{ei}]>\}$
    4.3.2 이외의 경우 $TL = <p, [t_{si} , t_{ei}]>$
5 **Until** t_e // q의 끝 시간, 즉 세그먼트 q_n의 끝시간과 동일
  5.1 t_{si}에서의 최근접 객체 p를 기준으로 t_{ei}까지, 각 질의 세그먼트 단위 별로 4단계 반복
6 *TL* 반환

CTNN 질의는 시작 시간에서 전체 데이타 객체와 질의점 사이의 거리 계산을 행한 후, 최소한의 추가 계산 작업만으로 다음 하위 시간 인터벌의 최근접 객체들을 선택한다. 단지 현재 p의 유효시간 인터벌 내에서 어떤 객체가 삽입되었거나 삭제되었을 때만 이 갱신으로 인해 유효시간 인터벌이 변경되는지 다른 최근접 객체가 선택되는 지를 체크하고 최소한의 거리 계산을 행하며, 나머지 평가시간 위에서는 질의와 교차하는 세그먼트 p′, 또는 현재의 p와 교차하는 p′가 있을 때 p′로 최근접 객체 정보를 변경시켜주기만 하면 된다. CNN 질의는 어떤 객체의 속도나 방향이 변경될 때마다 거리 계산 함수를 재계산하고, 함수가 변경될 때마다 객체들과 질의 사이의 거리를 재계산하며 이들을 정렬한다. 또한 최근접 객체 선택과정에서 유효시간 인터벌을 결정하기 위한 분할점 계산 작업도 필요하다. CTNN은 이러한 불필요하고 반복적인 작업들을 줄이고, 간단한 계산을 통해 정확하고 연속적인 최근접 탐색을 가능하게 하였다.

4.2 근사 연속 궤적 최근접 질의

실 세계 응용에서 어떤 사용자는 자신이 원하는 응답을 얻을 때, 결과의 정확도보다는 빠른 응답 시간을 더욱 중요하게 생각할 수도 있다. 즉, 약간의 오차가 허용되는 범위 내에서 빠르게 결과를 얻기를 원할 지도 모른다. 이런 경우 사용되는 것이 근사 질의 처리 기법이다. CTNN 질의는 선분 대칭 이동과 교차점, 변곡점에서의 결과 변경 여부 비교를 통해 사용자에게 정확한 결과를 제공하지만 세그먼트 수가 많아질 수록, 세그먼트 교차 수가 증가할 수록 대칭 이동 횟수나 평가시간점 개수가 증가하기 때문에 빈번한 비교 작업을 행하게 된다. 이로 인해 응답 시간이 느려지는 것을 방지하기 위하여, 우리는 다음의 알고리즘 5와 같이, 좀 더 빠른 시간 안에 근사 정보를 얻을 수 있는 ACTNN 기법을 제안한다.

알고리즘 5. ACTNN

Input: 데이타 객체 세그먼트 집합 *seg*, n개의 세그먼트 $(q_1,\cdots,q_n)$로 구성된 질의 객체 궤적 q
Output: 시간 리스트 *TL*

// 단계 수행 중 후보 객체가 여럿 존재하는 경우에는 항상 **Calculate_Displacement( )** 적용

1 **set_EvaluationTime()** // 두 세그먼트의 실제 교차점만을 평가시간으로 선택
2 $|q_i - seg_i|^{tsi}$ // q_i의 시작 시간 t_{si}
2.1 *mindist* 가지는 최근접 객체 p 선택
2.2 $TL = <p , [t_s , t_{ei}]>$
3 탐색원 h 그리기 (지름: $|q_i - p|^{tsi}$, 중심점: 선분q_ip의 중점)
4 h와 교차하는 $\square p'$가 존재하는지 검색
4.1 $\square p'$가 존재한다면,
4.1.1 교차시간 t_h 설정
4.1.1.1 h와 $\square p'$가 교차하는 두 점 중, 이후 시간을 가지는 교차점 선택
4.1.1.2 한 점만 교차한다면, 그 점을 교차점으로 선택
4.1.2 $|q_i - p'|^{th} < |q_i - p|^{th}$ 라면 $TL = \{<p , [t_{si} , t_h]>, <p' , [t_h , t_{ei}]>\}$
4.1.3 아니면 $TL = <p , [t_{si} , t_{ei}]>$
4.2 $\square p'$가 존재하지 않는다면, $TL = <p , [t_{si} , t_{ei}]>$
5 **Until** t_{ei} // $\square q_i$의 끝 시간
5.1 **Find_NNObject()**의 4단계 수행
5.2 $\square q$에 의해 발생되지 않은 교차점 t_i에 대해, 3~4단계 반복
6 **Until** t_e // q의 끝 시간, 즉 세그먼트 q_n의 끝시간과 동일
6.1 각 질의 세그먼트 q_i에 대해, t_{si}에서 3~4단계, t_{ei}까지 5단계 반복 수행
7 TL 반환

ACTNN 질의는 선분 대칭이동 작업을 행하지 않으며, 두 세그먼트가 실제로 생성하는 교차점만을 평가시간으로 가진다. 따라서 평가시간 개수를 줄일 수 있으며, 이로 인해 최근접 객체 정보 변경 비교 횟수를 줄일 수 있다. 대신에 정확도를 보충하기 위하여 탐색원과 세그먼트의 교차점을 평가시간에 포함한다.

4.3 정확 연속 궤적 최근접 질의

ECTNN 기법의 기본 아이디어는 선분 세그먼트 대칭 이동과 탐색원생성 과정을 혼합한 것이다. 먼저 기본 CTNN 질의와 동일하게, 첫 번째 교차점이 존재하는 공간이 기준공간으로 선택되며, 질의 시작시간에서 ACTNN 질의와 동일하게 모든 객체들과 질의점 사이의 거리를 계산한 후, 가장 작은 거리를 가지는 객체 p를 선택한다. 그리고 p가 이동공간 내의 객체라면 이를 기준공간으로 이동시키고, p와 기준공간 내 세그먼트간의 교차점을 탐색한다. p와 q 사이에 탐색원을 생성하고 이에 교차하는 다른 세그먼트가 존재하는지 체크한다. 만일 질의 한 세그먼트가 끝나기 전에 p가 다른 교차점과 만나거나 새로운 시작점 또는 끝점이 생긴다면 *Find_NNObject()*를 따라 처리한다. 그리고 p가 질의 세그먼트와 교차점을 생성하는 경우에만 교차점 이후 시간에 존재하는 p의 일부 세그먼트를 반대쪽 공간으로 대칭이동 시킨다. ECTNN 질의는 알고리즘 6과 같이 수행된다.

알고리즘 6. ECTNN

Input: 데이타 객체 세그먼트 집합 seg , n개의 세그먼트 $(q_1,\cdots,q_n)$로 구성된 질의 객체 궤적 q
Output: 시간 리스트 TL

// 단계 수행 중 후보 객체가 여럿 존재하는 경우에는 항상 **Calculate_Displacement( )** 적용
1 **Set_EvaluationTime()**
2 $|q_i - seg_i|^{tsi}$ // q_i의 시작 시간
2.1 *mindist* 가지는 최근접 객체 p 선택
2.2 $TL = <p , [t_{si} , t_{ei}]>$
3 탐색원 h 그리기 (지름: $|q_i - p|^{tsi}$, 중심점: 선분q_ip의 중점)
4 h와 교차하는 $\square p'$가 존재하는지 검색
4.1 $\square p'$가 존재한다면,
4.1.1 교차시간 t_h 설정
4.1.1.1 h와 $\square p'$가 교차하는 두 점 중, 이후 시간을 가지는 교차점 선택
4.1.1.2 한 점만 교차한다면, 그 점을 교차점으로 선택
4.1.2 $|q_i - p'|^{th} < |q_i - p|^{th}$ 라면 $TL = \{<p , [t_{si} , t_h]>, <p' , [t_h , t_{ei}]>\}$
4.1.3 아니면 $TL = <p , [t_s , t_{ei}]>$
4.2 $\square p'$가 존재하지 않는다면, $TL = <p , [t_s , t_{ei}]>$
5 **Until** t_{ei} // $\square q_i$의 끝 시간
5.1 **Find_NNObject()**의 4.2 ~ 4.3단계 수행
5.2 t_i시간에 IL내의 교차점 ip_I가 존재할 때,
5.2.1 $\square q$와 p'에 의해 발생된 교차점이라면,
5.2.1.1 t_i시간 이후의 $\square p'$를 반대쪽 공간(기준 공간)으로 대칭이동
5.2.1.2 $\square p'$와 기준공간 내 seg_i에 대해 **Find_Intersection()** 수행 → IL
5.2.2 $\square q$에 의해 발생되지 않은 교차점이라면, ip_I과 q_i에 대해 3~4단계 반복
6 **Until** t_e // q의 끝 시간, 즉 세그먼트 q_n의 끝시간과 동일
6.1 각 질의 세그먼트 q_i에 대해, t_{si}에서 3~4단계, t_{ei}까지 5단계 반복 수행
7 TL 반환

4.4 k-CTNN 질의

위에서 살펴본 CTNN 기법들은 모두 k-CTNN 질의로 쉽게 확장될 수 있다. 그림 1에서도 살펴보았듯이, k-CTNN 질의에서는 질의 전체 시간 내에서 각각의 하위 인터벌들이 서로 어떤 시간 인터벌을 공유하거나 개별적으로 존재한다. 즉, 어떤 시점에서 여러 개의 최근접 객체 후보들이 존재하는 경우, 이들은 Calculate_Displacement()를 통하여 반환된 정수의 오름차순으로 정렬되고, k값에 따라 차례로 최근접 객체로 선택된다. 따라서 1개의 최근접 객체부터 사용자가 지정한 k개의

값을 만족할 수 있는 만큼, 또는 선택되는 모든 후보 객체 개수만큼 결과로 출력될 수 있다. 그리고 이 후보들 각각은, 자신이 발생시키는 어떤 교차점까지 유효시간 인터벌을 할당 받게 된다. 예를 들어, 시간 t_s에서 *mindist*를 가지는 객체 a, b가 존재하는 경우, a는 시간 t_1에서 c와, b는 시간 t_2에서 d와 교차한다면, TL = {<a, [t_s, t_1]>, <b, [t_s, t_2]>, <c, [t_1, t_3]>, <d, [t_2, t_e]>} 형태로 갱신된다.

사용자가 k 값을 지정하는 경우, CTNN 기법은 질의 시작 점에서 가장 작은 거리 순서대로 후보 객체들의 개수를 비교하면서 k개를 선택한다. 그리고 두 번째 선택 조건은 변위 계산 결과 값이 된다. 예를 들어, 5개의 최근접 객체를 선택하고자 할 때, 질의 점과의 거리 순서가 a < b = c < d = e = f = g라면, 먼저 거리가 작은 순서대로 a, b, c를 최근접 객체로 선택한 후 d, e, f, g 중 가장 작은 변위 계산 값을 가지는 객체 두 개를 최근접 객체로 추가 선택한다. 이렇게 k값 만큼 최근접 객체를 선택하면 이들은 모두 질의 시작 시간 t_s부터 각 객체가 어떤 다른 객체와 교차하는 시점까지 하위 시간 인터벌을 할당 받게 된다. 중간에 질의와 어떤 세그먼트 p′가 교차한다면 가장 먼 조건을 가지는 객체를 p′로 교체한다. 객체가 선택될 때마다 이들은 TL에 저장되었다가 다른 변경사항에 따른 갱신들을 거친 후, 질의 마지막 시간 t_e에서 반환된다. k값이 지정되지 않은 경우는 *mindist*를 가지는 객체들만을 최근접 객체로 선택하고(1 또는 k개) 그들의 유효시간 인터벌과 함께 TL에 저장시킨다.

5. 실험 및 성능 평가

이 절에서는 ACTNN과 ECTNN 기법의 응답시간, 질의 결과 정확도 성능을 비교하였다. CTNN 질의는 연속 질의 개념과 궤적 질의 개념을 가지고 있고 연속 질의 개념 측면에서 비교될 수 있는 CNN 질의와 비교 평가 하였다. 이동객체를 위한 CNN 질의는 k-최근접 질의를 위해 제안된 기법이지만, 이 실험에서는 CNN의 k 최근접 질의 처리 단계 중 가장 먼저 수행되는 분할 리스트를 결정하는 단계, 즉 최근접 객체가 갱신되는 시점을 결정하는 단계에 초점을 맞추어 비교평가 하였다.

5.1 실험 환경

제안된 CTNN 기법들은 256MB의 주 메모리, 40GB의 하드 디스크를 가진 Window 2000 Professional 운영체제 하의 Pentium IV 1.4GHz PC에서, MS SQL Server 2000 DBMS 와 Visual C++을 통해 구현되었다. 실험 데이타는 GSTD(Generator of SpatioTemporal Datasets) [22]를 통해 <ID, 시간, 분, 공간좌표 (x, y)> 형태로 생성되었다. 그리고 실험 파라미터로 이동객체의 궤적 세그먼트 수, 객체의 이동 공간 범위, 세그먼트의 양 끝점 시간 차(끝 시간 - 시작 시간)를 사용하였다.

우리는 GSTD를 사용하여 각 시간 범위 내의 각 이동 공간 범위 내에서 데이타를 각 개수만큼 임의로 10번씩 생성하였고, 모든 실험은 생성된 모든 데이타 집합에 대해 이루어졌으며, 결과는 동일 조건 10개 집합의 실험에 대한 평균값이다.

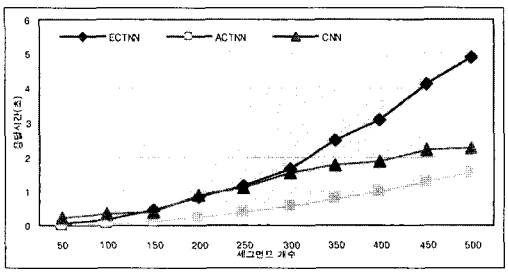
우리는 세그먼트가 분포하고 있는 공간범위에 따라, 세그먼트의 양끝점 시간차에 따라 이동객체 세그먼트 수 당, 각각의 질의 처리 기법이 각각 얼마만큼의 질의 응답시간을 보이는지 실험하였다. 또한 세그먼트의 끝점 시간차에 따라 또는 공간범위에 따라 세그먼트 수 당 각각의 기법들이 얼마만큼의 결과 정확도를 가지는지 보이기 위해, 임의로 생성한 5, 10, 15, 20, 25, 30, 50, 100개의 세그먼트에 대해 단순 방법으로 최근접 객체를 계산해보고 이 결과와 ACTNN, ECTNN 기법이 탐색해 낸 결과의 하위 인터벌 개수를 비교해보았다. 이 때 각 하위 인터벌의 시작, 끝 시간이 단순 계산 결과의 시작, 끝 시간과 1초 이내의 범위를 가지는 경우만을 정확한 결과라고 간주하였다.

5.2 공간 분포 범위에 따른 응답시간

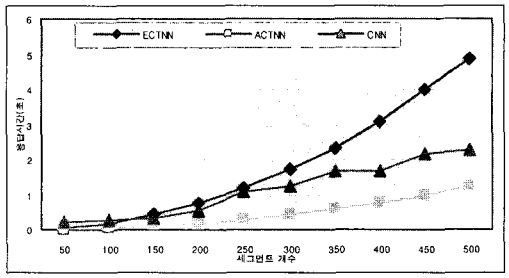
우리는 먼저 세그먼트의 공간 분포 범위와 ECTNN, ACTNN, CNN 기법들의 질의 응답시간에 따른 변화를 살펴보았다. 그림 3은 각 공간 분포에 따른 질의 응답시간을 나타내고 있다. 그림 3에서 알 수 있듯이 ECTNN, ACTNN, CNN 기법은 모두 공간 범위에 따른 질의 응답 시간에 큰 영향을 받지 않는다. 특히, ECTNN과 ACTNN 기법의 경우 그림 4에서 보여주듯이 추출한 교차점 개수가 공간 범위에 따라 크게 달라지지 않기 때문이다.

표 2 실험 파라미터

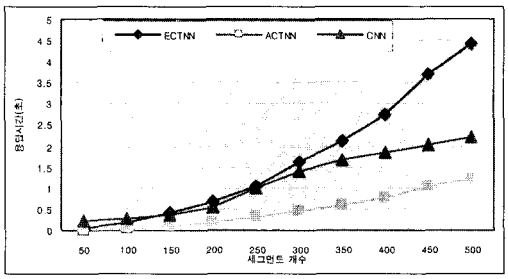
실험 파라미터	값
질의 시간 측정을 위한 객체 세그먼트 개수	50 , 100 , 150 , 200 , 250 , 300 , 350 , 400 , 450 , 500
결과 정확도 측정을 위한객체 세그먼트 개수	5 , 10 , 15 , 20 , 25 , 30, 50, 100
객체 이동 공간 범위	5만m^2, 1만m^2, 5천m^2, 1천m^2
세그먼트의 양 끝점 시간 차	[09 - 19] , [10 - 15] , [11 - 14]



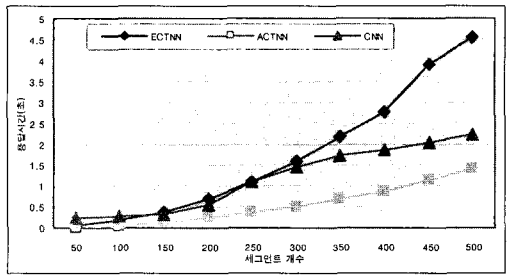
(a) 5만m^2



(b) 1만m^2



(c) 5천m^2



(d) 1천m^2

그림 3 공간 범위 분포에 따른, 세그먼트 개수 vs. 평균 응답 시간

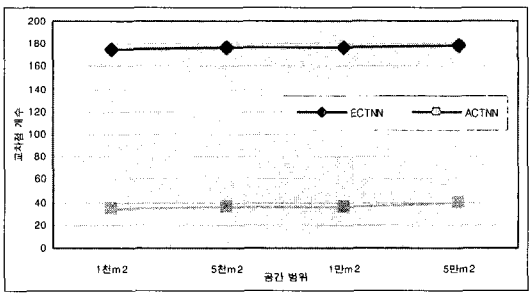
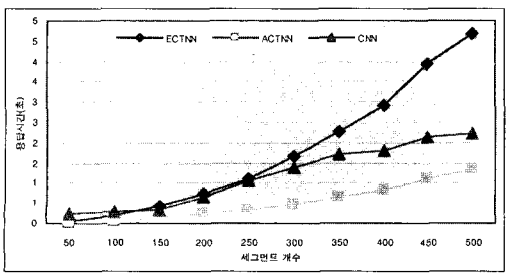


그림 4 공간 범위 분포에 따른 교차점 개수

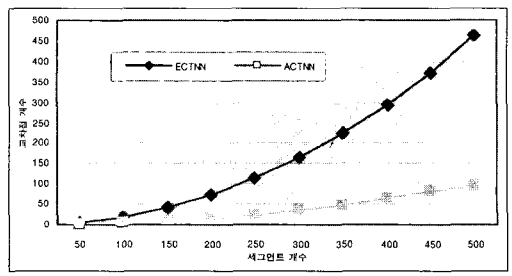
5.3 세그먼트 개수에 따른 응답시간

우리는 앞의 실험에서 ECTNN과 ACTNN 기법의 경우 공간범위에 따른 세그먼트의 교차점의 개수가 비슷하기 때문에 공간 범위분포에는 영향을 받지 않음을 알 수 있었다. 이번 실험에서는 세그먼트의 개수에 따른 각 기법들의 응답시간을 살펴 보았다. 그림 5(a)는 세그먼트의 개수에 따른 각 기법들의 질의 응답시간들을 보여주고 있다.

그림 5(a)에서 보여 주듯이 각 기법들 모두 세그먼트의 개수에 비례하여 응답하는 것을 볼 수 있다. ACTNN기법의 경우 가장 빠른 질의 응답을 보였으며, ECTNN 과 CNN의 경우 세그먼트의 수가 50~300까지는 비슷한 응답 시간을 보였지만, 세그먼트수가 증가할수록 ECTNN 기법이 보다 늦은 응답 시간을 보였다. 이러한 이유는 세그먼트에 따른 ACTNN 과 ECTNN



(a) 질의 응답 시간



(b) 교차점 개수

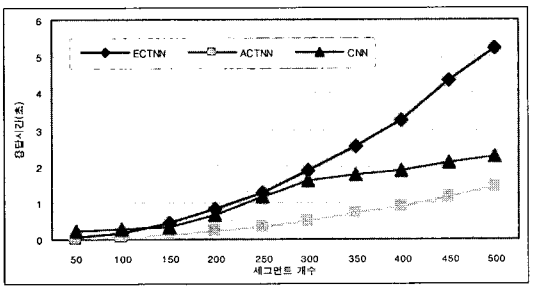
그림 5 세그먼트 개수에 따른 질의 응답 시간

기법이 발생하는 교차점 개수를 나타내고 있는 그림 5(b)에서 보여주듯이 ECTNN 기법의 경우 세그먼트의 개수가 증가할수록 평가해야 할 교차점의 수가 상당히 증가하기 때문이다.

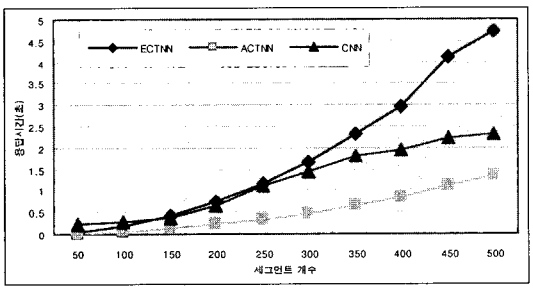
5.4 세그먼트의 양 끝점 시간 차에 따른 응답시간

우리는 모든 데이타의 시작시간과 종료시간이 동일하다는 가정을 기반으로, 일정 시간간격으로 임의의 세그먼트를 생성한 경우, 세그먼트의 개수에 따라 각 CTNN 기법의 질의응답 시간이 어떻게 나타나는지 살펴보았다. 이 시간은 세그먼트의 분포 공간 범위에 따른 10개 데이타 집합의 실험 응답 시간을 평균한 값이다.

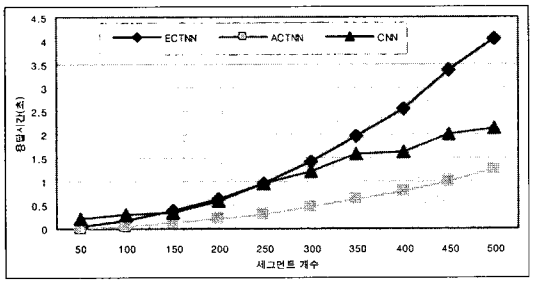
그림 6에서도 알 수 있듯이 ACTNN과 ECTNN 기법은 모두 세그먼트의 양 끝점시간 차에 따라 질의 응답 시간에 영향을 받는 반면, CNN 기법은 세그먼트의 양 끝점시간에 큰 영향을 받지 않음을 보여주고 있다. 이는 ACTNN과 ECTNN 기법의 경우, 각 세그먼트의 양 끝 시간이 정해진 경우, 일정 범위 내에서 세그먼트를 임의로 생성하면 각 세그먼트의 속도에 차이가 생기며, 따라서 세그먼트의 길이나 세그먼트들의 분포 밀도에 영향을 끼치면서 동시에 교차점도 영향을 받기 때문이다.



(a) 10시간



(b) 5시간



(c) 3시간

그림 6 세그먼트 양 끝점 시간 차에 따른 평균 응답 시간

5.5 정확도 비교

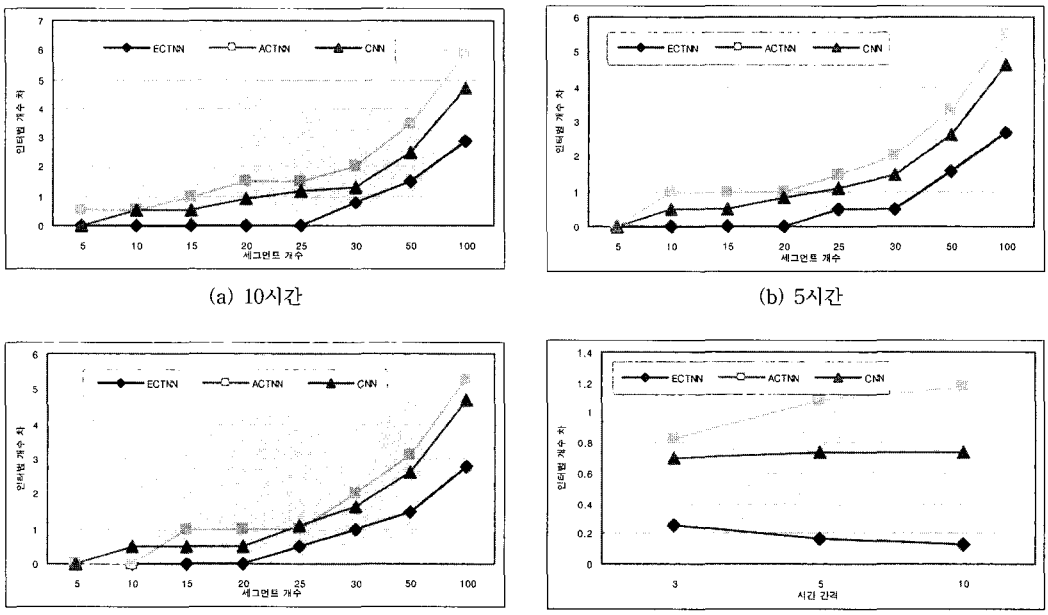
우리는 세그먼트의 수가 5, 10, 15, 20, 25, 30, 50, 100개 인 경우, 일정 공간 비율과 일정 시간 범위를 가지는 세그먼트들을 각각 임의로 생성한 후 ECTNN, ACTNN, CNN 기법을 적용하였을 때 각 기법들이 어느 정도의 정확도를 보이는지 살펴보았다.

우리는 정확한 결과값과 각 기법이 탐색해 낸 하위 시간 인터벌들 간의 개수를 비교하여 평가하였다. 그 이유는 인터벌 개수가 동일한 경우 시점 간의 차를 정확히 판별할 수 있기 때문이다. 따라서, 각 데이타 집합에 대해 단순 방법으로 최근접 객체를 탐색한 경우와 각 CTNN 기법이 탐색한 결과를 비교하고, 인터벌 개수가 실제결과와 동일한 경우 시작 또는 끝 시간 차가 1초 이내라면 정확히 일치하는 것으로 간주하였으며, 아니라면 인터벌 차이 수를 1로 설정하였다. 그리고 각 기법이 탐색해 낸 결과와 정확한 결과 사이의 인터벌 개수 차를 정확도 값으로 산출하였다.

그림 7의 실험 결과에서 보여주듯이, 모든 시간 간격에 대해 대부분 같은 경향을 보였다. 즉, 각 기법들 모두 세그먼트의 개수가 증가할수록 인터벌 개수 차를 많이 나타내였다. ECTNN 기법의 경우 모든 경우 정확도가 높았으며, ACTNN과 CNN의 경우 CNN이 약간 좋은 성능을 보였다. 또한 시간 간격에 따른 전체적인 정확도를 비교해 보았을 때 그림 4(d)에서 보여주듯이 ECTNN과 ACTNN 기법의 경우 시간간격에 따른 영향을 받는 반면, CNN은 시간간격에 따른 영향을 받지 않는 것으로 나타났다. 이러한 이유는 세그먼트의 양끝점 시간간격에 따른 실험에서도 살펴보았듯이 각 세그먼트의 양끝 시간이 정해진 경우, 일정 범위 내에서 세그먼트를 임의로 생성하면 각 세그먼트의 속도에 차이가 생기며, 따라서 세그먼트의 길이나 세그먼트들의 분포 밀도에 영향을 끼치기 때문이다.

6. 결 론

이 논문에서 우리는 질의와 데이타 객체가 모두 이동 객체인 경우에 가장 유용하게 사용될 수 있는 새로운 최근접 질의 처리 기법들을 제안하였다. 기존의 연구들은 데이타와 질의의 궤적 정보를 전혀 고려하지 상태에서 최근접 객체를 선택하기 때문에 부정확한 결과를 보



(a) 10시간 (b) 5시간

(c) 3시간 (d) 시간 간격에 따른 인터벌 개수

그림 7 시간 간격에 따른 정확도

일 수 있다. 또한 선택된 최근접 객체 정보가 빠른시간 안에 변경될 수도 있기 때문에 잦은 갱신을 발생시킬 수 있다. CTNN 기법은 이런 문제점들을 해결하기 위하여 궤적 최근접 질의 개념을 사용하였으며, 변위계산을 통해 질의 궤적과 가장 가까운 위치를 유지하면서 움직이는 객체를 최근접 객체로 선택하도록 하였다. 또한 최근접 객체가 변경되는 연속적인 시간을 탐색해내기 위하여 객체 세그먼트들의 변곡점과 교차점 위에서 결과 변경 여부를 비교하였다. 우리는 이와 같은 궤적 최근접 질의를 수행하기 위한 기본적인 CTNN 알고리즘과 함께, 응답시간을 줄이기 위한 ACTNN 기법과 정확도 향상을 위한 ECTNN 기법의 알고리즘을 제안하였다. 그리고 각 기법들이 세그먼트의 공간범위나 시간범위에 따라 얼마만큼의 응답시간 차이를 보이는지, 정확도 차이를 보이는지 실험을 통해 평가하였다. 실험 결과에서는 ECTNN 기법의 경우 높은 정확도를 갖는 반면, 응답시간에 있어서는 조금 낮은 성능을 보였다. 또한 ACTNN 기법의 경우 빠른 응답시간을 보인반면, 정확도 측면에서는 ECTNN 보다 낮은 성능을 보였다. 따라서, 우리는 실험을 통해 정확한 답을 원하는 경우에는 ECTNN 기법을 사용하고, 빠른 응답을 원할 경우는 ACTNN 기법을 사용할 수 있음을 보였다.

향후연구에서는 인덱스 구조를 이용한 효율적인 CTNN 기법 처리에 관한 연구를 수행할 것이며, 또한 이동 영역객체에 대한 CTNN 기법에 관한 연구도 수행할 것이다.

참 고 문 헌

[1] A. Prasad Sistla, Ouri Wolfson, Sam Chamberlain, Son Dao, "Modeling and Querying Moving Objects," ICDE 1997, pp.422~432.

[2] Flip Korn, Nikolas Sidiropoulos, Christos Faloutsos, Eliot Siegel, Zenon Protopapas, "Fast Nearest Neighbor Search in Medical Image Databases," VLDB 1996, pp.215~226.

[3] Like Gao, Zhengrong Yao, Xiaoyang Sean Wang, "Evaluating Continuous Nearest Neighbor Queries for Streaming Time Series via Pre-Fetching," CIKM 2002, pp.485~492.

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

[5] Apostolos Papadopoulos, Yannis Manolopoulos, "Performance of Nearest Neighbor Queries in R-tree," ICDT 1997, pp.394~408.

[6] Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, Thomas Seidl, "Fast Nearest Neighbor Search in High-Dimensional Space," ICDE 1998, pp.209~218.

[7] 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.

[8] Danxhou Liu, Ee-Peng Lim, We Keong Ng,

"Efficient k Nearest Neighbor Queries on Remote Spatial Database Using Range Estimation," SSDBM 2002, pp.121~130.

[9] David A. White, Ramesh Jain, "Similarity Indexing with the SS-tree," ICDE 1996, pp.516~523.

[10] Norio Katayama, Shin'ichi Satoh, "The SR-tree : An Index Structure for High-Dimensional Nearest Neighbor Queries," GISMOD Conference 1997, pp.369~380.

[11] Suni Araym, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, Angela Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," JACM 1998, pp.891~923.

[12] Piotr Indyk, Rajeev Motwani, "Approximate Nearest Neighbors : Towards Removing the Curse of Dimensionality," STOC 1998, pp.604~613.

[13] King-Ip, Congjun Yang, "The ANN-tree : An Index for Efficient Approximate Nearest Neighbor Search," DASFAA 2001, pp.174~181.

[14] Stephan Volmer, "Fast Approximate Nearest Neighbor Queries in Metric Feature Spaces by Buoy-Indexing," VISUAL 2002, pp.36~49.

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

[16] Rimantas Benetis, Christian S. Jensen, Gytis Karciauskas, Simonas Saltenis, "Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects," IDEAS 2002, pp.44~53.

[17] Simonas Saltenis, Christian S. Jensen, "Indexing the Position of Continuously Moving Object," SIGMOD Conference 2000, pp.331~342.

[18] Yufei Tao, Dimitris Papadias, "Spatial Queries in Dynamic Environments," TODS 2003.

[19] Jose Moreira, Cristina Ribeiro, Jean-Marc Saglio, "Representation and Manipulation of Moving Points: An Extended Data Model for Location Estimation," Cartography and Geographic Information Systems(CaGIS), ACSM, Vol.26, No.2, April 1999.

[20] Kriengkrai Porkaew, Iosif Lazaridis, Sharad Mehrotra, "Querying Mobile Objects in Spatio-Temporal Databases," SSTD 2001, pp.59~78.

[21] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, Computational Geometry Algorithms and Applications. Springer-Verlag, March 1997.

[22] Yannis Theodoridis, Mario A. Nascimento, "Generating Spatiotemporal Datasets on the WWW," SIGMOD 2000, pp.39~43.

지 정 희

1999년 충주대학교 전자계산학과 졸업 2001년 충주대학교 대학원 전자계산학 석사. 2003년 충북대학교 대학원 전자계산학 박사수료. 관심분야는 시공간 데이타베이스, Temporal GIS, 시공간 질의 최적화, 시공간 색인기법, 이동객체 관리 기법

최 보 윤

2002년 한성대학교 정보시스템공학과 졸업. 2004년 충북대학교 대학원 전자계산학 석사. 관심분야는 시공간 데이타베이스, 이동객체 질의 처리기법, 위치 기반 서비스

김 상 호

1997년 충북대학교 컴퓨터과학과 졸업 1999년 충북대학교 대학원 전자계산학 석사. 2004년 충북대학교 대학원 전자계산학 박사. 2004년~현재 충북대학교 연구원. 관심분야는 시공간 데이타베이스, Web Visualization, Component GIS, 유비쿼터스, 이동객체 관리 기법

류 근 호

정보과학회논문지 : 데이타베이스 제 31 권 제 1 호 참조