■ 2007년도 학생논문 경진대회 수상작

도로 네트워크 데이타베이스에서 근사 색인을 이용한 k-최근접 질의 처리

(k-Nearest Neighbor Query Processing using Approximate Indexing in Road Network Databases)

이 상 철 † 김 상 욱 ††

(Sang-Chul Lee) (Sang-Wook Kim)

요 약 본 논문에서는 도로 네트워크 데이타베이스에서 정적 객체의 k-최근접 이웃 질의를 효율적으로 처리하기 위한 방안을 논의한다. 기존의 여러 기법들은 인덱스를 사용하지 못했는데, 이는 네트워크 거리가 순서화된 거리함수가 아니며 삼각 부등식(triangular inequality) 성질 또한 만족하지 못하기 때문이다. 이러한 기존 기법들은 질의 처리 시 심각한 성능 저하의 문제를 가진다. 선계산된 네트워크 거리를 이용하는 또 다른 기법은 저장 공간의 오버헤드가 크다는 문제를 갖는다. 본 논문에서는 이러한 두 가지 문제점들을 동시에 해결하기 위하여 객체들 간의 네트워크 거리를 근사하여 객체들에 대한 인덱스를 구축하고, 이를 이용하여 k-최근접 이웃 질의를 처리하는 새로운 기법을 제안한다. 이를 위하여 본 논문에서는 먼저 네트워크 공간 상의 객체를 유클리드 공간 상으로 사상하기 위한 체계적인 방법을 제시한다. 특히, 삼각 부등식 성질을 만족시키기 위하여 **평균 네트워크 거리**라는 새로운 거리 개념을 제시하고, 유클리드 공간으로의 사상을 위하여 FastMap 기법을 사용한다. 다음으로, 평균 네트워크 거리와 FastMap을 사용하여 네트워크 공간 상의 객체들로 인덱스를 구축하는 **근사 색인** 알고리즘을 제시한다. 또한, 구축한 인덱스를 사용하여 k-최근접 이웃 질의를 효과적으로 수행하는 알고리즘을 제안한다. 마지막으로, 실제 도로 네트워크를 이용한 다양한 실험을 통하여 제안된 기법의 우수성을 규명한다.

키워드 : k-최근접 이웃 질의, 도로 네트워크 데이타베이스, 근사 색인

Abstract In this paper, we address an efficient processing scheme for k-nearest neighbor queries to retrieve k static objects in road network databases. Existing methods cannot expect a query processing speed-up by index structures in road network databases, since it is impossible to build an index by the network distance, which cannot meet the triangular inequality requirement, essential for index creation, but only possible in a totally ordered set. Thus, these previous methods suffer from a serious performance degradation in query processing. Another method using pre-computed network distances also suffers from a serious storage overhead to maintain a huge amount of pre-computed network distances. To solve these performance and storage problems at the same time, this paper proposes a novel approach that creates an index for moving objects by approximating their network distances and efficiently processes k-nearest neighbor queries by means of the approximate index. For this approach, we proposed a systematic way of mapping each moving object on a road network into the corresponding absolute position in the m-dimensional space. To meet the triangular inequality this

· 본 연구는 한국학술진흥재단을 통한 2007년도 교육인적자원부 학술연구조성사업비의 지원 (KRF-2007-314-D00221)과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원(IITA-2008-C1090-0801-0040)으로 수행되었습니다.

† 학생회원 : 한양대학교 전자컴퓨터통신공학과
korly@hanyang.ac.kr
†† 종신회원 : 한양대학교 정보통신학부 교수
wook@hanyang.ac.kr

논문접수 : 2007년 10월 5일
심사완료 : 2008년 5월 8일

정보과학회논문지: 데이타베이스 제35권 제5호(2008.10)

paper proposes a new notion of *average network distance*, and uses FastMap to map moving objects to their corresponding points in the m-dimensional space. After then, we present an *approximate indexing* algorithm to build an R*-tree, a multidimensional index, on the m-dimensional points of moving objects. The proposed scheme presents a query processing algorithm capable of efficiently evaluating k-nearest neighbor queries by finding k-nearest points (i.e., k-nearest moving objects) from the m-dimensional index. Finally, a variety of extensive experiments verifies the performance enhancement of the proposed approach by performing especially for the real-life road network databases.

Key words : k-nearest neighbor queries, road network databases, approximate indexing

1. 서 론

최근, 위성 항법 장치(global positioning system: GPS) 기술의 급속한 발달로 인하여 이를 이용한 위치 기반 서비스(location-based service)가 널리 사용되고 있다. 위치 기반 서비스는 사용자의 위치 정보를 활용하여 제공되는 서비스를 의미한다[1]. 텔레매틱스(telematics)는 자동차와 여객선 등의 위치 정보를 이용하여 위치 기반 서비스를 제공하는 새로운 응용 분야이다[2]. 여기서, 자동차와 여객선과 같이 시간의 흐름에 따라 공간상의 위치 정보가 변화하는 객체를 이동 객체(moving object)라 한다[3].

이동 객체가 위치하고 있는 공간은 이동 객체 움직임의 제약 유무에 따라 이동의 제약이 전혀 없는 공간과 이동이 제약된 공간으로 나눌 수 있다[4,5]. 제약이 전혀 없는 공간의 대표적인 예는 유클리드 공간(Euclidean space)이다[6]. 유클리드 공간상의 이동 객체로는 바다 위를 항해하는 배나 하늘을 비행하는 항공기 등을 들 수 있다. 제약이 있는 공간의 대표적인 예는 도로 네트워크 공간(road network space)이다[7]. 도로 네트워크 공간상의 이동 객체로는 도로 위를 움직이는 자동차와 선로를 따라 움직이는 기차 등을 들 수 있다. 텔레매틱스와 같은 실제 응용에서는 주로 도로 네트워크 위를 움직이는 이동 객체를 주 대상으로 한다.

도로 네트워크 정보, 이동 객체의 정보, 정적 객체 정보 등이 저장된 데이타베이스를 도로 네트워크 데이타베이스(road network databases)라 한다. 도로 네트워크 데이타베이스에서 도로 네트워크는 방향성을 가진 그래프로 모델링 된다[5]. 하나의 도로 세그먼트(road segment)는 그래프의 간선(edge)에 해당되며, 서로 다른 두 도로 세그먼트가 만나는 지점, 즉 교차로가 그래프의 노드(node)에 해당된다. 또한, 도로 네트워크 위에 정류소, 학교, 호텔 등과 같은 시설물들은 정적 객체(static object)로 모델링되며, 자동차, 사람과 같은 이동성을 가지고 있는 객체는 이동 객체로 모델링된다.

도로 네트워크 데이타베이스에서 사용되는 질의들은 k-최근접 이웃 질의(k-nearest neighbor queries), 영역 질의(range queries), 공간 조인 질의(spatial join queries) 등이 있다[8]. 본 논문에서는 네트워크 위를 움직이는 이동 객체의 위치를 질의 점으로 거리가 가장 가까운 k개의 정적 객체를 검색하는 k-최근접 이웃 질의에 연구의 초점을 맞추고자 한다. 기존 유클리드 공간상에서는 각 객체의 절대 좌표를 사용하여 임의의 두 객체들 간의 유클리드 거리를 계산할 수 있다. 그러나 도로 네트워크 공간상에서 이동 객체는 미리 정의된 도로 네트워크 위로만 움직일 수 있기 때문에, 각 객체의 절대 좌표만으로는 객체들 간의 네트워크 거리(network distance)를 계산할 수 없다. 여기서, 두 객체간의 네트워크 거리란 도로 네트워크에서 해당 두 객체간의 최단 경로상에 존재하는 도로 세그먼트 길이들의 총 합이다[8].

그림 1은 도로 네트워크에서 k-최근접 이웃 질의의 예를 나타낸 것이다. 검은색의 점은 질의 점인 자동차의 위치를 나타내고 있으며, 빗금 친 일곱 개의 사각형은 정적 객체의 위치를 나타낸다. 1-최근접 이웃 질의가 들어왔을 때, 유클리드 공간에서는 유클리드 거리(그림에서 점선으로 나타낸 거리)를 기준으로 가장 가까운 정적 객체 A를 검색해 반환한다[9]. 그러나 도로 네트워크 공간에서 이동 객체는 도로 네트워크 위만을 움직이기 때문에 네트워크 거리를 기준으로 검색해야 한다. 따라서 이 경우에는 네트워크 거리(그림에서 실선으로 나타낸 거리)가 가장 가까운 정적 객체 B가 검색되어야 한다.

그림 1의 예에 나타난 바와 같이, 임의의 두 점간 네

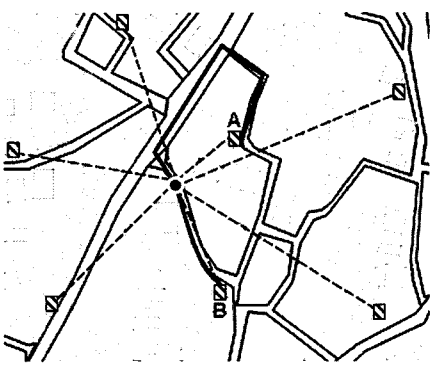


그림 1 도로 네트워크에서 k-최근접 이웃 질의의 예

트워크 거리는 두 점의 절대 좌표만을 가지고 계산할 수 없다. 그 이유는 네트워크 거리가 절대 좌표에 의한 유클리드 거리가 아닌 두 점을 연결하는 도로 네트워크 구조에 따라 결정되기 때문이다. 이러한 이유로 현재까지 정확한 네트워크 거리를 효율적으로 구하기 위한 다양한 방법들이 연구되어 왔으며, 대표적인 예로는 IER 기법, INE 기법, VN^3 기법 등이 있다[8,10]. IER 기법은 유클리드 거리가 네트워크 거리보다 항상 작거나 같은 성질을 이용한다. 먼저, 유클리드 거리를 기준으로 후보들을 찾고, 이들만을 대상으로 실제 네트워크 거리를 구한다. 이러한 IER 기법은 저장 공간 오버헤드가 작다는 장점을 가지고 있지만, 여러 번의 시행 착오를 겪기 때문에 질의 처리 성능이 크게 떨어진다. INE 기법은 질의 점으로부터 도로 세그먼트를 차례로 확장해 가면서 정적 객체의 유무를 검색한다. IER 기법과 마찬가지로, 저장 공간 오버헤드가 작다는 장점을 가지고 있지만, 다수의 디스크 접근이 필요하기 때문에 질의 처리 성능이 좋지 않다. VN^3 기법은 각 정적 객체와 인접한 다른 정적 객체간의 거리가 같은 지점들을 기준으로 전체 네트워크 공간을 셀들의 집합으로 나눈다. 효과적인 질의 처리를 위하여 각 셀 안의 모든 정적 객체, 노드, 셀 경계선간 거리를 선계산(pre-computation)해 둔다. VN^3 기법은 IER 기법 및 INE 기법과 비교하여 질의 처리 성능이 뛰어 나지만, 저장 공간의 오버헤드가 지나치게 크다는 단점이 있다.

기존 기법들은 질의 처리를 위하여 네트워크 거리를 기반으로 하는 정적 객체 색인을 시도하지 않았다. 그 이유는 R^*-트리와 같은 기존의 많은 다차원 인덱스들이 유톨리드 거리 등의 순서화된 거리를 지원할 뿐 네트워크 거리는 지원하지 않기 때문이다. 또한, 네트워크 거리가 삼각 부등식(triangular inequality)을 만족하지 못하기 때문에 FastMap[11] 혹은 M-트리[12] 등의 최근 인덱스 기법도 사용하지 못한다. 그러나 만약 네트워크 거리에 대해서 인덱스를 사용할 수 있다면, 비교적 작은 공간 오버헤드만으로도 빠르게 질의 처리하는 것이 가능하게 된다. 본 논문에서는 네트워크 거리를 기반으로 정적 객체를 **근사 색인**(*approximate indexing*)하고, 이를 이용하여 효과적으로 질의를 처리하는 새로운 기법을 제안한다.

제안하는 근사 색인 기법은 다음과 같다. 먼저, 도로 네트워크 위의 모든 정적 객체들을 m-차원 유클리드 공간상의 절대 좌표로 사상(mapping)한다. 이 때, 유클리드 공간상의 절대 좌표 설정 기준은 임의의 두 정적 객체간의 유클리드 거리가 실제 네트워크 거리를 최대한 반영하도록 하는 것이다. 이와 같이, 사상된 정적 객체들의 유클리드 좌표를 다차원 공간 인덱스인 R*-트리[13,14]를 이용하여 색인한다. k-최근접 이웃 질의의 처리 시에는 주어진 점 역시 다른 모든 정적 객체와의 네트워크 거리를 최대한 반영하도록 m-차원 유클리드 공간상의 절대 좌표로 사상한다. m-차원 유클리드 공간상의 질의 점을 이용하여 색인 단계에서 구축한 R*-트리를 탐색함으로써 k-최근접 이웃 질의를 처리한다.

제안하는 인덱스 기반 기법은 네트워크 거리 기반의 인덱스를 사용하지 않는 INE 기법 및 IER 기법에 비해 질의 처리에 있어서 우수한 성능을 제공할 수 있다. 또한 인덱스만을 이용하여 질의 처리가 가능하므로 VN^3 기법에 비해 매우 작은 저장 공간만으로 k-최근접 이웃 질의를 처리할 수 있게 된다. 그러나 제안한 기법은 네트워크 공간의 점을 유클리드 공간의 점으로 사상하는 데 있어서 근사적 기법을 사용하므로, 질의 처리에 있어서 착오 기각(false dismissal)이 발생할 수 있다. 본 논문에서는 이러한 착오 기각이 그다지 심각한 문제가 아님을, 즉 착오 기각의 비율이 매우 작음을 실험을 통하여 자세히 설명한다. 또한, 착오 기각의 문제를 해결하기 위한 방법으로, 주어진 k 보다 약간 많은 수의 정적 객체를 검색하는 기법을 도입하고 이의 도입 효과에 대해서 논의한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 기존 연구의 접근 기법을 소개하고, 그 장단점을 지적한다. 제 3장에서는 제안하는 기법을 상세히 설명한다. 제 4장에서는 실험을 통해 제안하는 기법의 정확도를 검증한다. 마지막으로, 제 5장에서는 본 논문을 요약하고, 결론을 내린다.

2. 관련 연구

본 장에서는 도로 네트워크 데이타베이스에서의 공간 질의 처리를 위한 기존의 대표적 접근법을 소개하고, 그 장단점을 논의한다.

k-최근접 이웃 질의를 처리하기 위한 대표적인 방식들은 INE 기법[8], IER 기법[8], VN^3 기법[10] 등이 있다. INE(incremental network expansion) 기법[8]은 우선 질의 점이 포함된 도로 세그먼트를 찾는다. 그런 다음, 이 도로 세그먼트로부터 시작하여 인접 도로 세그먼트를 하나씩 차례로 확장해 가며, 확장된 도로 세그먼트에 정적 객체가 있는지의 여부를 확인한다. 정적 객체가 해당 세그먼트상에 존재하는지의 여부를 빠르게 파악하기 위하여 세그먼트들에 대한 R*-트리[13,14]를 이용한다. 위의 과정을 사용자가 요구한 k개의 정적 객체를 모두 찾을 때까지 반복한다. 이 방식은 저장 공간이 작다는 장점을 가지고 있지만, 각 도로 세그먼트를 확장할 때마다 도로 세그먼트들에 대한 정보를 디스크로부터 매번 접근해야 하는 단점이 있다. 또한, 해당 도로 세그먼트 상의 정적 객체 유무에 대한 결과를 R*-트리를

통하여 매번 검색해야 하므로 질의 성능이 떨어진다.

IER(incremental Euclidean restriction) 기법[8]은 두 점간의 네트워크 거리는 유클리드 거리보다 항상 크거나 같다는 특성을 이용한다. 우선, 정적 객체들을 대상으로 구축된 R-트리를 검색하여 유클리드 거리를 기준으로 질의 점에서 가장 가까운 k개의 후보 정적 객체들을 찾아낸다. 그런 다음, 후보 정적 객체들의 질의 점까지 실제 네트워크 거리를 계산하고, 그들 중 가장 큰 값을 반지름으로 하는 영역 질의를 구성하여 R-트리를 다시 검색한다. 이 때 검색된 정적 객체 집합에서 이전 단계에서 구한 후보 정적 객체들을 제외하고, 유클리드 거리를 기준으로 가장 가까운 정적 객체를 찾아 질의 점과의 네트워크 거리를 계산한다. 이 거리가 후보 정적 객체들 중 질의 점과의 네트워크 거리가 가장 큰 정적 객체보다 작으면, k번째 정적 객체는 후보에서 탈락시키고 새로 검색된 정적 객체를 k개의 후보 정적 객체에 넣는다. 위의 과정을 계속 반복하여 질의 점에서 가장 가까운 k개의 정적 객체를 검색한다. 이 방식은 INE 기법과 비슷한 저장 공간을 필요로 하지만 여러 번의 시행 착오를 겪기 때문에 질의 처리 성능이 크게 떨어진다[8].

VN^3(Voronoi network nearest neighbor)[10]는 각 정적 객체와 인접한 다른 정적 객체간의 거리가 같은 지점들을 기준으로 전체 네트워크 공간을 셀들의 집합으로 나눈다. 셀의 중심에는 한 개의 정적 객체만이 존재하고, 이것이 셀 내부의 어느 지점에서도 가장 가까운 정적 객체가 된다. 셀 내부적으로는 노드와 정적 객체, 경계점과 정적 객체, 그리고 셀 안의 모든 경계점간 거리를 미리 계산하여 저장되어 있다. 또한, 이 외에도 인접 셀에 대한 정보가 추가적으로 저장되어 있다. k-최근접 질의 처리 시, k가 1일 때는 질의 점이 떨어진 셀 안의 정적 객체가 가장 가까운 정적 객체가 된다. k가 2 이상일 때에는 해당 셀과 그 인접 셀 내부의 정적 객체들을 후보로 둔다. 질의 점과 질의 점이 포함된 셀의 경계까지의 거리를 계산하면, 후보의 셀들 중 어떤 셀 안의 정적 객체가 두 번째로 가까운 정적 객체인지 알게 된다. 이렇게 두 번째 셀을 발견하게 되면, 두 번째 셀의 인접 셀 내부의 정적 객체들 역시 후보가 된다. k개의 정적 객체가 다 찾아질 때까지 위의 질의 처리 과정을 반복한다. 이 방식은 미리 계산된 정보를 이용하기 때문에 질의 처리 성능이 매우 뛰어나다. 그러나 미리 계산하여 저장해야 할 정보가 많기 때문에 저장 공간의 오버헤드가 매우 심하다는 단점이 있다[15].

3. 제안하는 기법

본 장에서는 도로 네트워크 데이타베이스에서 질의를 효과적으로 처리하기 위한 새로운 기법을 제안한다. 먼저, 제3.1절에서는 제안하는 기법의 기본 전략을 설명하고, 제3.2절에서는 제안하는 근사 색인 기법을 설명한다. 제3.3절에서는 이를 이용한 k-최근접 이웃 질의 처리 기법을 설명한다.

3.1 기본 전략

제2장에서 언급한 바와 같이, 기존 기법들은 네트워크 거리를 기반으로 하는 정적 객체의 색인을 시도하지 않았다. 이와 같이, 기존 기법들이 네트워크 거리를 기반으로 한 색인을 시도하지 않은 이유는 1) R*-트리 부류의 인덱스들이 유클리드 거리와 같이 순서화되는 거리만을 지원할 뿐 네트워크 거리를 지원하지 못하며, 2) 네트워크 거리가 삼각 부등식을 만족시키지 못하는 거리 함수이기 때문이다[16]. 이와 같은 제약 조건을 해결하지 않고 네트워크 거리를 그대로 색인하게 되면 질의 처리 시 예상치 못한 검색 결과를 얻을 수 있다. 그런데, 네트워크 거리에 대한 정확한 혹은 근사적인 색인이 가능하다면 작은 공간 오버헤드만으로 빠르게 질의 처리하는 것이 가능하게 된다. 본 연구에서는 네트워크 거리를 기반으로 정적 객체를 근사 색인하고, 이를 이용하여 효과적으로 질의를 처리하는 기법을 제안한다.

네트워크 거리를 색인하기 위해서는 1) 네트워크 거리를 유클리드 거리로 접근하여 R*-트리 부류의 기존 인덱스를 사용하는 방법과 2) 임의의 두 객체 간의 거리를 직접 색인하는 새로운 타입의 인덱스를 사용하는 방법을 사용할 수 있다. 후자의 대표적인 예로는 M-트리[12]를 들 수 있다. M-트리는 객체들 간의 거리를 기준으로 페이지를 분할하여 저장하는 균형 트리이다. M-트리를 사용한 k-최근접 질의 처리 시에는 질의 점부터 상위 노드의 각 객체와의 거리를 구한 후, 이를 바탕으로 계속하여 탐색할 자식 노드를 결정한다. 위와 동일한 방법으로 단말노드까지 탐색하여 k개의 최근접 객체를 반환한다. 그러나 도로 네트워크 환경에서의 거리 계산은 많은 부하를 유발하므로, 질의 시점에 탐색 대상 노드의 모든 객체에 대해서 이러한 네트워크 거리 계산을 수행하는 것은 실용적이지 못하다. 따라서 본 논문에서는 이중 전자의 방법을 사용한다.

제안하는 근사 색인 방법이 동작하는 절차를 요약하면 다음과 같다.

- 인덱스 구축 단계
  - ① 모든 정적 객체를 m-차원 유클리드 공간상의 절대 좌표로 사상한다. (이때, 절대 좌표의 설정 기준은 m-차원 유클리드 공간상의 임의의 두 정적 객체간의 유클리드 거리가 원래의 네트워크 공간상의 실제 네트워크 거리를 최대한 반영하도록 하는 것이다.)

② 사상한 정적 객체의 위치를 다차원 인덱스인 R*-트리를 사용하여 색인한다.

• 질의 처리 단계

① 질의 점 역시 다른 모든 정적 객체와의 네트워크 거리를 가급적 그대로 유지하도록 m-차원 유클리드 공간상의 한 점으로 사상한다.

② 유클리드 공간으로 사상된 질의 점을 이용하여 정적 객체를 대상으로 구성된 R*-트리에 검색하여 질의 처리를 수행한다.

결국, 제안하는 방법은 네트워크 공간상의 객체(혹은 점)들을 유클리드 공간상으로 사상하고, 성능이 우수한 기존 다차원 인덱스를 사용하여 질의 처리를 수행하는 방법을 취한다고 볼 수 있다. 그런데, 이러한 본 연구의 접근법이 가능하기 위해서는 네트워크 공간상의 객체를 유클리드 공간상의 객체로 사상하는 체계적인 방법이 필요하다.

그림 2는 네트워크 공간내의 정적 객체 및 이동 객체를 m-차원 유클리드 공간상으로 사상하는 과정을 나타낸다. 도로 네트워크상을 이동하는 자동차가 현재 자신의 위치에서 가장 가까운 주유소를 찾는다고 가정하자. 제안하는 기법은 우선 모든 주유소 쌍 간의 네트워크 거리를 이용하여 주유소들의 위치를 m-차원 유클리드 공간상의 절대 좌표로 사상한다. 사상 후, m-차원 유클리드 공간상의 임의의 두 주유소 간 유클리드 거리는 원래의 네트워크 거리를 근사하게 된다. 질의 점인 자동차의 위치도 이와 동일한 방법으로 m-차원 유클리드 공간상의 절대 좌표로 사상시킨다. 정적 객체들과 질의 점을 m-차원 유클리드 공간상의 절대 좌표로 사상하기 위한 다양한 방법들이 존재할 수 있다[17]. 본 연구에서는 사상 시 발생하게 되는 오차가 적으면서, 빠르게 사상할 수 있는 FastMap 알고리즘[11]을 사용한다.

FastMap 알고리즘[11]

FastMap은 N개의 객체와 거리 함수 D가 주어졌을 때, 이 객체들을 m-차원 유클리드 공간상의 절대 좌표로 사상하는 기법이다. FastMap을 사용하기 위해서는 거리 함수가 다음 세 가지 조건을 만족해야 한다.

(조건 1) 그 결과가 항상 양수(non-negative)여야 한다.

(조건 2) 대칭성(symmetric)을 만족해야 한다.

(조건 3) 삼각 부등식(triangular inequality) 성질을 만족해야 한다.

사상의 자세한 과정은 다음과 같다. 우선, 주어진 거리 함수를 기준으로 하여 멀리 떨어져 있는 두 객체 O_a와 O_b를 찾는다. 이 두 객체를 피봇 쌍(pivot pair)이라고 부르고, 각 객체를 피봇 객체(pivot object)라 부른다. 이 피봇 쌍을 잇는 선분에 아래와 다음 식 (1)의 코사인 법칙을 이용하여 나머지 객체 O_i들을 프로젝션(projection) 시킨다.

$$D(O_b,O_i)^2 = D(O_a,O_i)^2 + D(O_a,O_b)^2 - 2x_iD(O_a,O_b) \quad (1)$$

$D(X,Y)$는 주어진 거리 함수에 의한 객체 X와 객체 Y간의 실제 거리를 의미하고, x_i는 객체 O_i를 피봇 쌍을 잇는 선분 상에 프로젝션 시킨 값을 의미한다. 식 (1)을 이용하여 식 (2)를 유도할 수 있다.

$$x_i = \frac{D(O_a,O_i)^2 + D(O_a,O_b)^2 - D(O_b,O_i)^2}{2D(O_a,O_b)} \quad (2)$$

식 (2)를 통하여 피봇 쌍을 잇는 선분 상에 모든 객체들을 프로젝션 시킬 수 있다. 이 때, 피봇 쌍을 잇는 선분에 프로젝션 된 x_i가 객체 O_i의 한 차원의 값이 된다. 그리고 나머지 차원의 값을 계산하기 위한 목적으로 피봇 쌍을 잇는 선분에 수직인(orthogonal) 초평면(hyperplane)상에서 객체들 간 거리를 재계산한다. 식 (3)은 피봇 쌍에 수직인 초평면에서의 객체 간 거리를 재계산하는 식을 나타낸다. O_i'와 O_j'는 초평면에 프로젝션 된 객체 O_i와 O_j를 말하며, $D(O_i',O_j')$는 주어진 거리 함수에 의한 초평면상에서의 객체 O_i'와 O_j'간의 거리를 나타낸다. 식 (3)에서 x_i와 x_j는 식 (2)를 통하여 O_i와 O_j가 피봇 쌍을 잇는 선분에 프로젝션 된 값을 나타낸다.

$$D(O_i',O_j')^2 = D(O_i,O_j)^2 - (x_i - x_j)^2 \quad (3)$$

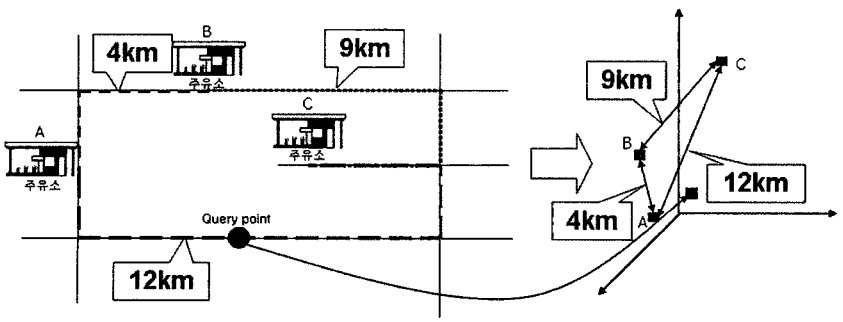


그림 2 m-차원 유클리드 공간으로 사상의 예

재계산된 객체간의 거리에 준하여 가장 먼 두 객체를 두 번째 피봇 쌍으로 선택하고, 상기 과정을 반복하여 각 객체의 두 번째 차원 값을 구하게 된다. 이와 같은 과정을 m번 반복함으로써 N개의 객체들을 m-차원상의 점으로 사상하게 된다. 이 m은 사용자에 의해 임의로 주어지며, 반복하는 횟수가 m에 도달하기 전에라도 객체들 간 거리가 모두 0이 되는 경우에는 그대로 종료한다[11].

그림 3은 FastMap을 이용하여 객체를 유클리드 공간상의 점으로 사상하는 과정을 나타낸 다. O_x는 사상되는 객체를 나타내며, 이 중 O_a와 O_b는 피봇 쌍을 의미한다. 그림 3(a)는 O_i가 피봇 쌍인 O_a와 O_b를 잇는 선분 $\overline{O_aO_b}$상의 x_i로 프로젝션되는 과정을 나타낸다. 그림 3(b)는 O_i와 O_j가 선분 $\overline{O_aO_b}$상의 x_i와 x_j로 프로젝션되었을 때, O_i와 O_j를 선분 $\overline{O_aO_b}$와 수직인 초평면상으로 프로젝션시키고, 프로젝션된 O_i'와 O_j'간 거리를 구하는 과정을 나타낸 것이다. 이러한 FastMap은 유클리드 거리가 아닌 다른 거리들을 유클리드 공간상으로 비교적 정확하게 사상시키는 것으로 알려져 있다[17].

3.2 색인

FastMap을 사용하여 색인을 하기 위해서는 우선 모든 정적 객체 쌍 간의 네트워크 거리 정보가 필요하다. 이러한 정적 객체 쌍 간의 네트워크 거리는 다익스트라 알고리즘(Dijkstra algorithm)을 이용하여 쉽게 구할 수 있다. FastMap은 이 네트워크 거리 정보를 이용하여 각 정적 객체를 m-차원 유클리드 공간상의 절대 좌표로 사상한다. 그러나 네트워크 거리는 FastMap이 요구하는 거리 함수의 세 가지 조건들 중 (조건 2) 대칭성 조건과 (조건 3) 삼각 부등식 조건을 만족하지 못한다. 그 이유는 도로 네트워크의 경우 일방 통행, 비대칭 도로 등이 존재하기 때문이다. 따라서 네트워크 거리를 FastMap에 바로 적용하는 것은 불가능하다.

이 문제를 해결하기 위하여 본 논문에서는 두 객체의 네트워크 거리 대신에 두 객체의 평균 네트워크 거리를 사용한다. 여기서 평균 네트워크 거리란 두 정적 객체 간 왕복 네트워크 거리의 평균을 의미하며, 다음과 같이 정형적으로 정의할 수 있다.

정의 1. 도로 네트워크 상의 두 객체 O_x와 O_y의 **평균 네트워크 거리**(*average network distance*)는 객체 O_x에서 O_y로의 네트워크 거리 $D(O_x,O_y)$와 객체 O_y에서 O_x로의 네트워크 거리 $D(O_y,O_x)$의 평균값으로 정의하며, $\overline{D}(O_x,O_y)$로 표기한다. 다음 식 (4)는 두 객체 O_x와 O_y의 평균 네트워크 거리를 수식으로 표현한 것이다.

$$\overline{D}(Ox,Oy)=\frac{D(Ox,Oy)+D(Ox,Oy)}{2} \quad (4)$$

본 논문에서 평균 네트워크 거리를 사용하는 이유는 많은 경우에 있어서 평균 네트워크 거리 $\overline{D}(O_x,O_y)$는 원래의 네트워크 거리인 $D(O_x,O_y)$ 및 $D(O_y,O_x)$와 그 값이 유사하며, 덧붙여 앞서의 두 가지 조건인 (조건 2)와 (조건 3)을 충족 시켜주기 때문이다.

새롭게 정의한 평균 네트워크 거리는 대칭성 조건을 자연스럽게 만족한다. 즉, 정의 1에 의해 $\overline{D}(O_x,O_y)$ = $\overline{D}(O_y,O_x)$이 성립하고, 따라서 (조건 2)를 만족한다. 다음으로, 평균 네트워크 거리가 삼각 부등식의 (조건 3)을 만족함을 다음 두 가지 보조정리들을 통하여 증명한다.

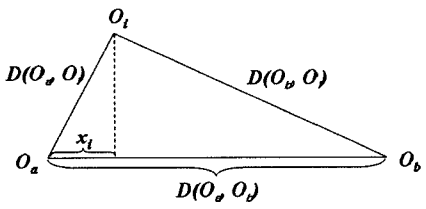
보조정리 1. 도로 네트워크 상의 두 객체 O_a와 O_b가 주어였을 때, 네트워크 거리 D에 대하여 다음 두 식 (5)와 (6)은 항상 만족한다.

$$D(O_a,O_b) \le D(O_a,O_i) + D(O_i,O_b) \quad (5)$$

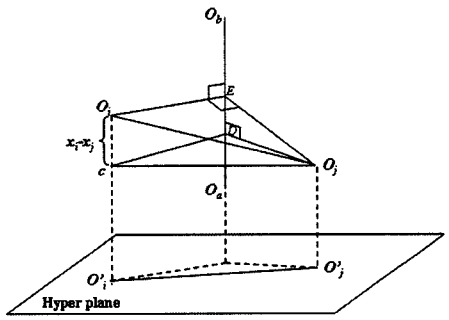
$$D(O_b,O_a) \le D(O_b,O_i) + D(O_i,O_a) \quad (6)$$

여기서, O_i는 네트워크 상의 임의의 노드이다.

증명: 먼저, 식 (5)의 증명을 위해 $D(O_a,O_b) > D(O_a,O_i) + D(O_i,O_b)$인 경우가 있다고 가정하자. 그렇다면, O_a에서 O_b까지의 최단 거리는 $D(O_a,O_i) + D(O_i,O_b)$가 되므로 $D(O_a,O_b)$는 O_a에서 O_b까지의 네트워크 거리가 아님을 의미한다. 따라서 $D(O_a,O_b) >$



(a) 선분 OaOb로의 객체 프로젝션



(b) 선분 OaOb에 수직인 초평면에 객체 프로젝션

그림 3 FastMap의 기본 원리

$D(O_a,O_i) + D(O_i,O_b)$ 인 경우는 발생할 수 없게 되고, 식 (5)는 항상 만족됨을 알 수 있다. 식 (6) 역시 이와 동일한 방법으로 만족됨을 증명할 수 있다. ■

보조정리 2. 도로 네트워크 상의 두 정적 객체 O_a와 O_b 간 평균 네트워크 거리 $\overline{D}$는 삼각 부등식을 만족한다.

$$\overline{D}(O_a,O_b) \le \overline{D}(O_a,O_i) + \overline{D}(O_i,O_b) \quad (7)$$

여기서, O_i는 네트워크 상의 임의의 노드이다.

증명: 보조정리 1에 의해서 식 (8)과 식 (9)가 만족됨을 알 수 있다.

$$D(O_a,O_b) + D(O_b,O_a) \le (D(O_a,O_i) + D(O_i,O_b)) + (D(O_b,O_i) + D(O_i,O_a)) \quad (8)$$

$$= (D(O_a,O_i) + D(O_i,O_a)) + (D(O_b,O_i) + D(O_i,O_b)) \quad (9)$$

식 (9)에 식 (4)를 대입하면, 식 (10)이 됨을 알 수 있다.

$$2\overline{D}(O_a, O_b) \le 2\overline{D}(O_a,O_i) + 2\overline{D}(O_b,O_i) \quad (10)$$

따라서 식 (7)은 만족하며, 보조정리 2는 증명된다. ■

평균 네트워크 거리 $\overline{D}$는 결과가 항상 양수여야 하는 조건, 대칭성 조건, 삼각 부등식 조건을 모두 만족한다. 이제 $\overline{D}$를 FastMap의 거리 함수로 사용하여 각 정적 객체를 m-차원 유클리드 공간상의 절대 좌표로 사상할 수 있게 된다. 그리고 사상된 정적 객체들을 다차원 공간 인덱스인 R*-트리를 이용하여 색인할 수 있다.

알고리즘 1은 정적 객체들을 색인하는 과정을 의사 코드로 나타낸 것이다. 단계 (1)은 모든 정적 객체 쌍 (O_i,C_j)간의 평균 네트워크 거리 $\overline{D}(O_i,C_j)$를 구하는 과정을 나타낸다. $\overline{D}(O_i,C_j)$는 다익스트라 알고리즘을 구한 이용하여 두 정적 객체의 실제 네트워크 거리 $D(O_i,C_j)$와 $D(C_j,O_i)$의 평균이 된다. 단계 (2)는 구해진 평균 네트워크 거리를 FastMap을 위한 거리 함수로 이용하여 각 정적 객체를 m-차원 유클리드 공간상의 절대 좌표로 사상한다. 단계 (3)은 m-차원 유클리드 공간 내의 정적 객체를 색인하는 과정이다. 마지막으로, 단계 (4)에서는 도로 네트워크의 각 노드에 대하여 모든 피봇까지의 네트워크 거리를 계산하여 저장한다. 단계 (4)가 필요한 이유는 제 3.3절에서 자세히 설명한다.

```
Algorithm Approximate-Indexing
float facility_distance_matrix[n][n];
int Pivot_Pair[k][2];
float Pivot_Pair_distance[k];
float x[n][k];
float node[m][2k];

void Indexing(facility_coordinate[n]){
  int i, j, c;
  for(i=0;i<n;i++)
    for(j=i+1;j<n;j++)
      facility_distance_matrix[i][j]
        ={Dijstra(facility_coordinate[i],facility_coordinate[j])
        +Dijstra(facility_coordinate[j],facility_coordinate[i])}/2;   (1)
        facility_distance_matrix[i][j]=facility_distance_matrix[j][i];
  }
  Fastmap(facility_distance_matrix, x);                               (2)
  Indexing(x, facility_index);                                        (3)
  while(!node){
    while(!pivot_pair){
      node[i][c]
        ={Dijstra(node_coordinate[i],Pivot_pair[j][1])
        +Dijstra(Pivot_pair[1],node_coordinate[i])}/2;
      node[i][c+1]
        ={Dijstra(node_coordinate[i],Pivot_pair[j][2])
        +Dijstra(Pivot_pair[2],node_coordinate[i])}/2;                (4)
      j++; c++;
    }
    i++;
  }
}
```

알고리즘 1 색인 알고리즘

3.3 질의 처리

도로 네트워크 상의 임의의 위치에 질의 점이 주어졌을 때, 제 3.2절에서 제안한 인덱스를 이용하기 위해서는 정적 객체들이 사상된 m-차원 유클리드 공간상으로 사상하여야 한다. 이를 위해서는 질의 점에서 각 피봇 객체까지의 네트워크 거리 정보가 필요하다. 그러나 질의 처리 시점에 질의 점에서 $2 \cdot m$개의 피봇 객체까지의 평균 네트워크 거리를 계산하는 것은 매우 큰 처리 비용을 요구한다. 본 논문에서는 이 문제를 해결하기 위하여 도로 네트워크 상의 각 노드에서 각 피봇 객체까지의 거리를 미리 계산하여 유지하는 기법을 제안한다. 보조정리 3을 사용하면 제안한 기법이 질의 점에서 피봇 객체까지의 평균 네트워크 거리를 빠르고 정확하게 계산할 수 있음을 알 수 있다.

보조정리 3. 도로 네트워크 상에서 질의 점이 Q, 거리 계산이 필요한 피봇 객체가 P, Q가 위치하는 도로 세그먼트 양 끝 노드가 N_a, N_b라 할 때, Q에서 P로의 평균 네트워크 거리 $\overline{D}(Q,P)$는 다음 식 (11)로 구할 수 있다.

$$\overline{D}(Q,P) = \begin{cases} Case\ 1: & MIN(\overline{D}(Q,N_a)+\overline{D}(N_a,P), \overline{D}(Q,N_b)+\overline{D}(N_b,P)) \\ Case\ 2: & \overline{D}(Q,P) \end{cases} \quad (11)$$

식 (11)에서 *Case* 1은 Q와 P가 다른 도로 세그먼트에 존재할 경우이며, *Case* 2는 Q와 P가 같은 도로 세그먼트에 존재할 경우이다. $MIN(x,y)$는 x, y값 중 작은 값을 반환하는 함수이다.

증명: 질의 점 Q가 주어졌을 때, P와 Q가 속한 도로 세그먼트가 같은 경우를 제외하고 Q에서 P까지의 최단 경로는 반드시 N_a 또는 N_b 노드를 포함하게 된다. 그 이유는 도로 네트워크 상의 이동 객체는 Q가 속한 도로 세그먼트의 양 끝 노드 중 어느 한 노드는 반드시 지나

야만 P에 도달 할 수 있기 때문이다. 따라서 Q에서 P까지의 네트워크 거리는 Q에서 N_a까지의 거리와 N_a에서 P까지의 네트워크 거리의 합 또는 Q에서 N_b까지의 거리와 N_b에서 P까지의 네트워크 거리의 합 중 작은 값이 된다. P와 Q가 같은 도로 세그먼트상에 존재할 때에는 이들 간의 평균 네트워크 거리를 계산한다. ■

질의 점에서 각 피봇 객체 간의 거리를 효과적으로 파악하기 위해서 알고리즘 1의 단계 (4)에서는 각 노드에서 모든 피봇 객체까지의 평균 네트워크 거리를 미리 계산해 둔 바 있다. 이를 이용하여 질의 점이 주어지면, 먼저 질의 점이 속해 있는 해당 도로 세그먼트를 찾는다. 이 세그먼트의 양 끝 노드에서 피봇 객체들까지의 평균 네트워크 거리는 미리 계산 되어 있으므로, 식 (11)을 이용하여 질의 점에서 각 피봇 객체까지의 평균 네트워크 거리를 쉽게 구할 수 있다. 이를 이용하여 질의 점을 m-차원 유클리드 공간상의 점으로 사상하여 k-최근접 이웃 질의를 다음 알고리즘 3.2와 같이 수행한다.

알고리즘 2는 질의 처리 과정을 의사 코드로 나타낸 것이다. 단계 (1)은 질의 점이 위치한 도로 세그먼트를 찾는 과정이다. 유클리드 공간 상의 도로 세그먼트의 위치 정보에 대한 인덱스가 구성되어 있다면, 이 세그먼트를 쉽게 찾을 수 있다. 단계 (2)는 질의 점으로부터 각 피봇 객체까지의 평균 네트워크 거리를 구하는 과정이다. 색인 과정에서 계산된 노드에서 피봇 객체까지의 평균 네트워크 거리를 이용하여 질의 점으로부터 각 피봇 객체까지의 평균 네트워크 거리를 쉽게 구할 수 있다. 단계 (3)은 FastMap을 이용하여 질의 점을 m-차원 유클리드 공간상의 절대 좌표로 사상시키는 과정이다. 색인 과정에서 구해진 피봇 객체들 간의 평균 네트워크 거리 정보를 활용하여 구한다. 단계 (4)는 색인 과정에서 생성한 m-차원의 R*-트리를 이용하여 질의 점으로부터 가장 가까운 정적 객체 k개를 찾는다.

```
Algorithm Query-Processing
void NN_query(Q_coordinate){
  float q[2k];
  int segment_id = find_segment(Q_coordinate);                    (1)
  for(i=0;i<2k;i++){
    q[i] = min(distance_query_to_node+distance_node_to_pivot[i]);  (2)
  }
  Fastmap(q, x);                                                  (3)
  result = NN_search(x, # of facility, facility_index);           (4)
}
```

알고리즘 2 k-최근접 이웃 질의 처리 알고리즘

알고리즘 1과 2의 근사 색인 기법은 객체들 간의 실제 네트워크 거리를 최대한 반영하여 유클리드 공간 상으로 사상하므로 착오기각을 최소로 유지할 수 있으며, 기존 기법들에서는 사용하지 못 하던 인덱스를 이용하므로 질의 처리 성능을 크게 개선시킬 수 있다. 또한, 인덱스만을 이용하여 질의를 처리하기 때문에 저장 공간 오버헤드를 작게 유지할 수 있다.

제안하는 근사 색인 기법은 성능 및 정확도에 있어서 우수한 결과를 보이게 된다. 먼저, 기존 기법들에서는 사용하지 못한 인덱스를 사용하므로 질의 처리 성능을 크게 개선시킬 수 있다. 다음으로, 객체들은 상호간의 실제 네트워크 거리를 최대한 반영하도록 유클리드 공간 상으로 사상되므로, 인덱스만으로 검색한 경우에도 비교적 정확한 질의 처리가 가능하다(질의 처리의 정확도에 대해서는 다음 제 4장의 성능 평가에서 자세히 논의한다.). 또한, 질의 처리 시에 인덱스만을 이용하기 때문에 VN^3기법 등에 비해 저장 공간 오버헤드를 작게 유지할 수 있다.

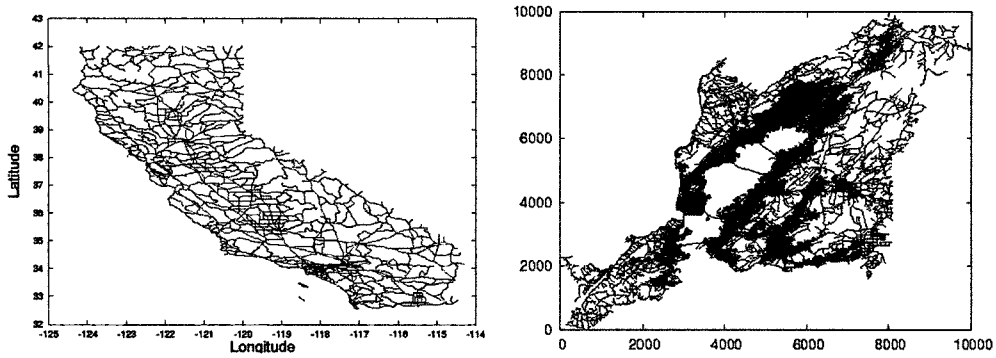
FastMap을 사용한 근사 색인 기법의 문제점은 착오기각이 발생할 수 있다는 것이다. 이는 FastMap 자체가 객체 간의 네트워크 거리를 100% 정확하게 유클리드 공간으로 사상하는 것이 아니라 근사적으로 사상하기 때문이다[17]. 즉, 근사적인 사상으로 인하여 원래는 k개 해답에 속 하나 인덱스 검색 결과에는 속하지 않는 착오기각이 발생할 수 있다. 이 문제를 해결하기 위한 방안으로 k개 이상의 k'개의 정적 객체를 검색함으로써 착오기각을 완화시킬 수 있다. 그 이유는 검색할 정적 객체의 수를 k'개로 늘려 줌에 따라 k개를 검색하는 경우에는 착오기각 되었던 정답이 검색 결과에 포함될 확률이 높아지기 때문이다. 따라서 정적 객체의 검색 개수를 k개 이상으로 설정하여 인덱스를 검색한다면, 착오기각의 문제점을 크게 완화시킬 수 있다. 이 때, 정답이 아닌 정적 객체들이 k'개에 포함될 수 있기 때문에 k'의 수를 적절히 조절해야 한다. 본 논문에서는 실험을 통하여 k'개의 정적 객체를 검색함으로써 착오기각을 크게 완화시킬 수 있음을 보인다. 또한 실험을 통하여 최적의 성능을 보이는 k에 대한 k'의 비율을 보인다.

4. 성능 평가

본 장에서는 다양한 실험을 통하여 제안된 근사 색인 기법의 우수성을 보인다. 제 4.1절에서는 실험 환경을 설명하고, 제 4.2절에서는 실험 결과를 제시하고 분석한다.

4.1 실험 환경

실험을 위한 실제 도로 네트워크 데이타로서 Oldenburg Road Network[18]와 California Road Network [18]를 사용하였다. 그림 4는 실험에서 사용한 도로 네트워크 데이타를 나타낸 것이며, 이후부터 이들을 각각 간략히 ORN과 CRN이라 부른다. ORN은 노드의 수가 6,105개이고, 도로 세그먼트의 수가 7,034개인 도로 네트



(a) Oldenburg 도로 네트워크 (b) California 도로 네트워크

그림 4 실험에 사용한 도로 네트워크 데이타

워크이다. CRN은 노드의 수가 21,047개이고, 도로 세그먼트의 수가 21,691개인 도로 네트워크이다. 도로 네트워크 내의 정적 객체의 수는 ORN과 CRN에서 각각 도로 세그먼트 수의 0.5%를 취하였으며, 이들의 위치는 전체 도로 네트워크 내에서 균등하게 분포하도록 하였다.

본 실험에서는 제안된 기법에서 절대 좌표 사상 시 발생하는 오차를 측정하기 위하여 질의 점으로부터 임의의 두 객체까지 거리의 순서가 바뀌는 정도를 측정하였다. 본 연구에서는 이를 오차율이라 부른다. 또한 제안하는 기법의 정확성을 보이기 위하여 재현율(recall)과 정확율(precision)을 측정하였다[19]. 재현율은 검색한 결과 중 정답이 누락된 정도, 즉 착오기각이 얼마나 발생했는지를 알 수 있는 척도로써 식 (12)으로 계산 할 수 있다. 정확률은 정답이 아닌데 검색된 정도, 즉 착오알림(false alarm)이 얼마나 발생했는지를 알 수 있는 척도로써 식 (13)으로 계산 할 수 있다. 착오기각을 크게 완화시키기 위하여 k개 보다 많은 k'개의 정적 객체를 검색할 경우 정확율이 낮게 측정된다. 정확율이 낮다는 것은 착오알림이 많이 발생하고 있다는 것이지만, 일부 정답을 검색하지 못하는 착오기각에 비하여 착오알림은 추후에 정답이 아닌 것들을 제거할 수 있기 때문에 허용 가능하고 할 수 있다.

$$\text{재현율} = \frac{\text{검색된 정답의 개수}}{\text{실제정답의 개수}} \quad (12)$$

$$\text{정확율} = \frac{\text{검색된 정답의 개수}}{\text{검색된 개수}} \quad (13)$$

다양한 실험을 통해 재현율과 정확율을 측정하기 위하여 다음과 같은 매개 변수를 설정하였다. 표 1은 정확도 실험에 사용된 매개 변수들과 그 값을 정리한 것이다. 변환 차원은 정적 객체를 사상시킬 유클리드 공간의 차원 수이며, k의 개수는 k-최근접 이웃질의에서 요구하는 k개의 정적 객체수이다. 또한 k'/k 비율은 k값을 기준으로 한 k'의 비율을 의미한다. 표 1에서 굵게 표시된 값은 해당 매개 변수를 대표하는 값이다. 하나의 매개 변수 값을 변화키는 경우, 다른 매개 변수 값들은 이 대표 값으로 고정된다. 예를 들어, 한 실험에서 변환 차원을 2, 5, 10, 15, 20으로 변화시킬 때, k의 개수와 k'/k 비율은 각각 10과 1.5로 고정된다. 실험의 정확한 측정을 위하여 100개의 질의 점을 균등하게 분포시켰으며, 이를 처리한 결과의 정확율과 재현율의 평균을 구하였다.

표 1 실험 환경에 사용된 매개 변수

매개 변수	매개 변수 값
변환 차원	2, 5, ***10***, 15, 20
k의 개수	1, 5, ***10***, 15, 20
k'/k 비율	1, ***1.5***, 2, 2.5, 3

4.2 실험 결과 및 분석

실험은 크게 네 가지를 수행하였다. 실험 1)에서는 변환 차원의 수(m)의 증가에 따른 오차율의 변화를 측정하였다. 실험 2)에서는 변환 차원의 수(m)의 증가에 따른 재현율과 정확율의 변화를 측정하였다. 실험 3)에서는 k의 변화에 따른 재현율과 정확율의 변화를 측정하였고, 실험 4)에서는 k'의 비율이 증가함에 따른 재현율과 정확율의 변화를 측정하였다.

그림 5는 실험 1)의 결과를 나타낸 것이다. 실험 1)에서는 m이 증가함에 따라 질의 점으로부터 임의의 두 정적 객체까지 거리의 순서가 바뀌는 정도인 오차율을 측정하였다. 각 데이타 집합의 오차율은 ORN에서 최소 2.5%, CRN에서 최소 2.6%로 나타났다. 차원이 증가함에 따라 대체로 오차율은 감소하였다. 이는 차원의 증가로 인하여 모든 정적 객체간의 m-차원 유클리드 공간상의 상대적 거리가 도로 네트워크 공간상의 상대적 거리를 잘 반영하기 때문이다. 그러나 특정 차원 수 이후(그림 4에서 10차원 이후)로는 오차율이 현상 유지하는

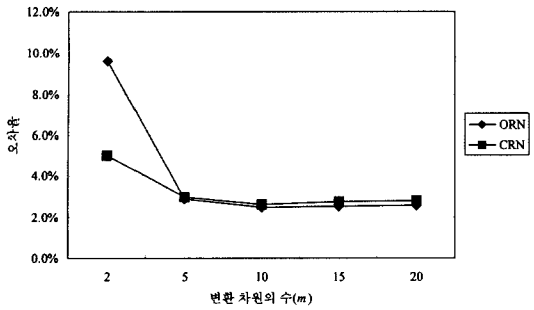


그림 5 변환 차원에 따른 오차율

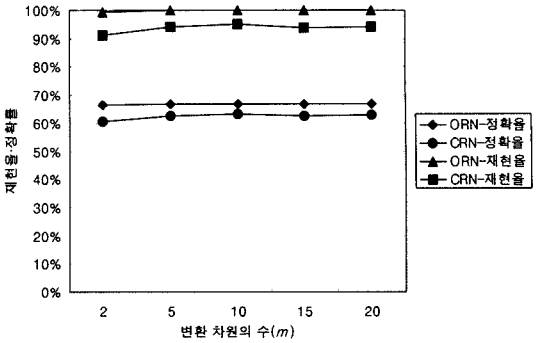


그림 6 차원 변화에 따른 재현율과 정확율

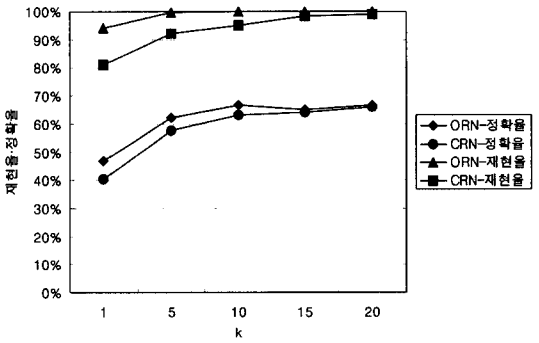


그림 7 k의 변화에 따른 재현율과 정확율.

경향을 보였는데 이는 차원 수가 높아지더라도 Fast-Map이 가지고 있는 고유의 오차가 크게 개선되지 않기 때문이다. 이 결과를 통하여 적은 수의 차원만을 이용하여 도로 네트워크 공간상의 임의의 객체들을 표현 가능하다는 것을 알 수 있다.

그림 6은 변환 차원의 수의 증가에 따른 재현율과 정확율의 변화를 측정한 실험 2)의 결과를 나타낸 것이다. 각 데이타 집합의 재현율은 ORN의 경우 최대 100%, CRN의 경우 최대 95%로 나타났다. 10차원까지는 변환 차원 수가 커짐에 따라 재현율이 상승하였으나, 그 이후부터는 거의 비슷한 재현율을 보였다. 그 이유는 실험 1)에서 보인 오차율이 차원의 증가에 따라 0이 되지 않고 그 전에 수렴하는 경향을 보이기 때문이다. 다음으로 각 데이타 집합의 정확율은 ORN의 경우 최대 67%이며, CRN의 경우 최대 63%이다. 검색한 정적 객체의 수 k'이 k의 1.5배이기 때문에 최대 정확율은 1/1.5 즉 최대 67%을 넘을 수 없다. 따라서 그림 6의 정확율은 매우 높은 결과라 할 수 있다. 이러한 결과의 이유는 검색 시 정적 객체간의 순서가 바뀌는 오차가 발생하지만 그 오차가 크지 않아서 k'/k 비율을 1.5로 할 때 정답인 정적 객체를 대부분 검색했기 때문이다.

그림 7은 k의 개수가 변함에 따른 재현율과 정확율의 변화를 측정한 실험 3)의 결과를 나타낸 것이다. 각 데이타 집합의 재현율은 ORN의 경우 최대 100%, CRN의 경우 최대 99%로 매우 높게 나타났다. 다음으로 각 데이타 집합의 정확율은 ORN의 경우 최대 67%, CRN의 경우 최대 66%로 나타났다. 실험 2)와 마찬가지로 정확율은 최대 67%를 넘을 수 없으므로 ORN과 CRN 모두에서 매우 정확한 결과를 얻음을 알 수 있다. 그림 7에서 보면, k가 증가함에 따라 전체적으로 재현율과 정확율이 증가하는 추세를 보였다. 그 이유는 제안하는 기법은 사상 시 정확한 네트워크 거리를 유지 하도록 사상시키는 것이 아니라 질의 점으로부터 상대적으로 가까운 정적 객체들이 근접하게 위치하도록 사상시키기 때문이다. 따라서 k가 작으면 질의 점에서부터 정확히 네트워크 거리가 가까운 정적 객체를 검색해야 하지만, k가 크면 질의 점으로부터 정확하진 않지만 비교적 가까운 정적 객체만은 검색하기 때문에 재현율과 정확율은 높아지게 된다.

그림 8은 k에 대한 k'의 비율이 증가함에 따른 재현율과 정확율의 변화를 측정한 실험 4)의 결과를 나타낸 것이다. 그림 7을 보면, k'의 비율이 증가함에 따라 재현율은 100%에 가까워짐을 볼 수 있다. 그 이유는 k'의 비율이 작을 경우에는 착오기각 되었던 정답이 k'의 비율이 커짐에 따라 검색 결과에 포함될 가능성이 높아지기 때문이다. 따라서 k'의 비율을 충분히 높여서 k-최근접 이웃 질의를 수행한다면, 착오기각의 부담을 상당부분 제거할 수 있을 것으로 기대된다. 다음으로 각 데이타 집합의 정확율은 ORN의 경우 최대 96%, CRN의 경우 최대 83%로 나타났다. k'의 비율이 증가할수록 정확율이 감소하는 이유는 분모인 검색된 정적 객체의 수가 크게 증가하여 최대 정확율이 감소하기 때문이다. 이와 같이 재현율과 정확율 사이에는 트레이드오프 관계가 있으므로 착오기각과 착오알림이 응용에 미치는 영향을 고려하여 k'의 비율을 설정해야 한다. 그림 7의 실험 결과에 따르면, k'/k 비율이 1.5인 경우 비교적 높은 정확율과 재현율을 보이는 것으로 나타났다.

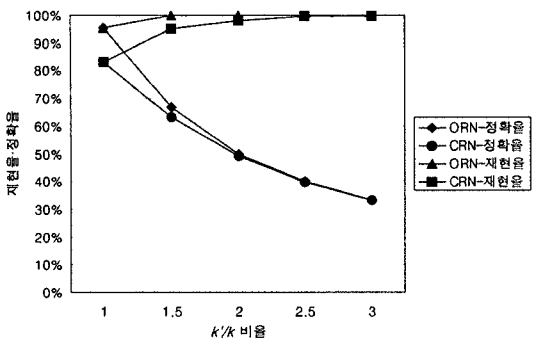


그림 8 k'/k의 비율 변화에 따른 재현율과 정확율

5. 결 론

본 논문에서는 도로 네트워크 데이타베이스에서 정적 객체들 간의 네트워크 거리를 기반으로 한 근사 색인 및 질의 처리 기법을 제안하였다. 도로 네트워크 데이타베이스에서의 질의 처리를 위한 기존의 연구는 인덱스를 사용하지 않아 성능 저하의 문제가 있거나 선계산 결과를 유지하기 위한 저장 공간 오버헤드가 심각한 문제가 있었다. 이에 반해 본 논문에서 제안한 네트워크 거리 기반 근사 색인 방법은 인덱스를 구축하여 높은 성능을 제공함과 동시에 작은 양의 저장 공간만을 필요로 한다는 장점이 있다.

본 논문의 공헌은 다음과 같이 요약할 수 있다. 첫째, 네트워크 공간 상의 객체를 유클리드 공간상으로 사상하기 위한 체계적인 방법을 제시하였다. 이를 위해 우선 두 객체 간의 평균 네트워크 거리라는 새로운 거리 개념을 제시하였다. 그리고 이 평균 네트워크 거리가 FastMap을 사용하기 위한 세 가지 조건을 만족함을 증명하고, FastMap을 사용하여 네트워크 공간 상의 객체들을 유클리드 공간으로 사상시키는 방법을 제시하였다. 둘째, 평균 네트워크 거리와 FastMap을 사용하여 네트워크 공간 상의 정적 객체들로 인덱스를 구축하는 근사 색인 알고리즘을 제시하였다. 제안한 방법은 유클리드 공간으로 사상을 수행하기 때문에 R*-트리를 포함한 여러 다차원 인덱스에 적용이 가능하다. 셋째, 구축한 인덱스를 사용하여 k-최근접 이웃 질의를 효율적으로 수행하는 알고리즘을 제시하였다. 제안한 알고리즘은 k-최근접 이웃 질의뿐 아니라 영역 질의 및 공간 조인 질의 등 다양한 질의 처리에 활용될 수 있다. 넷째, 다양한 실험을 통하여 제안한 근사 색인 기법의 정확성을 검증하였다.

이 같은 결과를 볼 때, 제안한 기법은 도로 네트워크 데이타베이스에서 네트워크 거리를 기반으로 하는 인덱스와 이를 이용하여 질의를 처리하는 첫 번째 연구로서 큰 의미가 있다고 할 수 있다. 특히, 인덱스를 저장하기 위한 작은 양의 저장 공간 오버헤드만으로 질의 처리 성능을 크게 개선시킬 수 있는 기반을 마련한 연구로 큰 의미가 있다 할 수 있다. 향후 연구로는 기존 방법들과의 성능 및 저장 공간 오버헤드를 비교하여 제안한 방법의 우수성을 실험적으로 입증하는 것과 색인 과정의 정보를 사용하여 k-최근접 객체들까지의 경로(path) 정보를 효율적으로 찾아내는 것을 들 수 있다. 또한, 본 논문의 결과를 토대로 질의 점의 방향성을 고려하여 질의 결과에 반영하는 연구와 네트워크 거리 기반 이동 객체 색인 및 질의 처리에 대한 연구로 확장할 수 있다.

참 고 문 헌

[1] S. Wu and K. Wu, "Effective Location-Based Services with Dynamic Data Management in Mobile Environments," *Wireless Networks*, Vol. 12, No. 3, pp. 369-381, 2006.

[2] S. Duri et al., "Data Protection and Data Sharing in Telematics," Mobile Networks and Applications, MONET, Vol. 9, No. 6, pp. 693-701, 2004.

[3] O. Wolfson et al., "Moving Object Databases: Issues and Solutions," In *Proc. Int'l. Conf. on Scientific and Statistical Database Management*, SSDBM, pp. 111-112, 1998.

[4] N. Weghe et al., "Representation of Moving Objects along a Road Network," In *Proc. Int'l. Conf. on Geoinformatics*, pp. 187-197, 2004.

[5] L. Speicys, C. Jensen, and A. Kligys, "Computational Data Modeling for Network-Constrained Moving Objects," In *Proc. Int'l. Symp. on Advances in Geographic Information Systems*, ACM GIS, pp. 118-125, 2003.

[6] J. Kelley, *General Topology*, Springer-Verlag, 1975.

[7] M. Vazirgiannis and O. Wolfson, "A Spatiotemporal Model and Language for Moving Objects on Road Networks," In *Proc. Int'l. Symp. on Spatial and Temporal Databases*, SSTD, pp. 20-35, 2001.

[8] D. Papadias et al., "Query Processing in Spatial Network Databases," In *Proc. Int'l. Conf. on Very Large Data Bases*, VLDB, pp. 802-813, 2003.

[9] G. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases," *ACM Trans. Database Systems*, ACM TODS, Vol. 24, No. 2, pp. 265-318, 1999.

[10] M. Kolahdouzan and C. Shahabi, "Voronoi-Based K-Nearest Neighbor Search for Spatial Network Databases," In *Proc. Int'l. Conf. on Very Large Data Bases*, VLDB, pp. 840-851, 2004.

[11] C. Faloutsos and K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets," In *Proc. ACM Int'l. Conf. on Management of Data*, ACM

SIGMOD, pp. 163-174, 1995.
[12] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," *Proceedings of the 23rd VLDB Conference*, pp. 426-435, 1997.
[13] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," In *Proc. ACM Int'l. Conf. on Management of Data*, ACM SIGMOD, pp. 47-57, 1984.
[14] N. Beckmann et al., "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," In *Proc. ACM Int'l. Conf. on Management of Data*, ACM SIGMOD, pp. 322-331, 1990.
[15] X. Huang, C. S. Jensen, and S. Saltenis, "The Islands Approach to Nearest Neighbor Querying in Spatial Networks," In *Proc. Int'l. Symp. on Spatial and Temporal Databases*, SSTD, pp. 73-90, 2005.
[16] B. Yi, H. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," In *Proc. IEEE Int'l. Conf. on Data Engineering*, ICDE, pp. 201-208, 1998.
[17] J. Wang et al., "Evaluating a Class of Distance-Mapping Algorithms for Data Mining and Clustering," In *Proc. ACM Int'l. Conf. on Knowledge Discovery and Data Mining*, ACM SIGKDD, pp. 307-311, 1999.
[18] The R-tree Portal, http://www.rtreeportal.org/
[19] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2000.

이 상 철

2005년 2월 한양대학교 소프트웨어학과 졸업(학사). 2007년 2월 한양대학교 전자컴퓨터통신공학과 졸업(석사). 2007년 3월~현재 한양대학교 전자컴퓨터통신공학과 재학(박사 과정). 관심분야는 데이타베이스 시스템, 데이타 마이닝, 공간 데이타베이스/GIS, 이동 객체 데이타베이스/텔레매틱스, 사회 연결망 분석

김 상 욱

1989년 2월 서울대학교 컴퓨터공학과 졸업(학사). 1991년 2월 한국과학기술원 전산학과 졸업(석사). 1994년 2월 한국과학기술원 전산학과 졸업(박사). 1991년 7월~8월 미국 Stanford University, Computer Science Department 방문 연구원 1994년 2월~1995년 2월 KAIST 정보전자연구소 전문연구원. 1999년 8월~2000년 8월 미국 IBM T.J. Watson Research Center Post-Doc. 1995년 3월~2000년 8월 강원대학교 컴퓨터정보통신공학부 부교수. 2003년 3월~현재 한양대학교 정보통신대학 정보통신학부 교수. 관심분야는 데이타베이스 시스템, 저장 시스템, 트랜잭션 관리, 데이타 마이닝, 멀티미디어 정보 검색, 공간 데이타베이스/GIS, 주기억 장치 데이타베이스, 이동 객체 데이타베이스/텔레매틱스, 사회 연결망 분석, 웹 데이타 분석