최근접 질의를 위한 고차원 인덱싱 방법
(A High-Dimensional Indexing Method for Nearest Neighbor Queries)

김 상 욱 † Charu Aggarwal†† Philip Yu††
(Sang-Wook Kim)

요 약 최근접 질의(nearest neighbor query)는 멀티미디어 데이타베이스에서 주어진 질의 객체와 가장 유사한 객체를 찾기 위한 매우 중요한 연산으로 사용된다. 대부분의 최근접 질의 처리 기법들은 객체의 효과적인 인덱싱을 위하여 다차원 인덱스(multidimensional index)를 사용한다. 그러나 N차원 사각형 혹은 원을 사용하여 객체 클러스터의 캡슐을 표현하는 기존의 다차원 인덱스들은 차원 수가 높아짐에 따라 검색 성능이 크게 떨어진다. 본 논문에서는 이러한 단순한 캡슐 표현 방식이 최근접 질의 처리의 성능을 저하시키는 주요 원인임을 지적하고, (1) 클러스터에 적합한 새로운 축 시스템(axis system)의 채택, (2) 원과 사각형의 조합에 의한 다양한 캡슐 형태의 표현, (3) 아웃 라이어(outlier)의 별도 관리 등의 해결 방안을 제안한다. 또한, 이러한 개념들을 채택하는 인덱싱 구조를 제시하고, 이를 이용하는 최근접 질의 처리 방안을 제안한다. 끝으로, 다양한 실험에 의한 성능 평가를 통하여 제안된 기법의 우수성을 검증한다.

Abstract The nearest neighbor query is an important operation widely used in multimedia databases for finding the object that is most similar to a given query object. Most of techniques for processing nearest neighbor queries employ multidimensional indexes for effective indexing of objects. However, the performance of previous multidimensional indexes, which use N-dimensional rectangles or spheres for representing the capsule of the object cluster, deteriorates seriously as the number of dimensions gets higher. In this paper, we first point out the fact that the simple representation of capsules incurs performance degradation in processing nearest neighbor queries. For alleviating this problem, we propose (1) adopting new axis systems appropriate to a given cluster, (2) representing various shapes of capsules by combining rectangles and spheres, and (3) maintaining outliers separately. We also verify the superiority of our approach through performance evaluation by performing extensive experiments.

1. 서 론

유사한 객체의 검색을 효과적으로 지원하는 것은 멀티미디어 데이타베이스에서 추구하는 가장 중요한 문제의 하나이다[1, 2]. 기존의 많은 연구에서는 객체로부터 추출한 색상, 질감, 명암 등의 특징 벡터(feature vector)를 대상으로 인덱싱하고, 이를 이용하여 사용자가 요구하는 다양한 질의를 지원한다[3, 4, 5].

최근접 질의(nearest neighbor query)는 유사한 객체를 찾기 위하여 가장 널리 사용되는 질의의 하나이다[6, 7, 8, 9]. 각 객체는 N개의 특징으로 구성되는 N차원 벡터 공간(vector space)내의 한 점으로 간주된다. 이를 기반으로 최근접 질의는 "N차원의 벡터 공간 내에 객체 점들의 집합과 질의 점이 주어질 때, 질의 점으로부터 유클리드 거리(Euclidean distance)가 최소인 객체를 찾는 질의"로 정의된다.

기존의 최근접 질의 처리 기법들은 대부분 객체들의 효과적인 인덱싱을 위하여 R-트리[10], R*-트리[11], R+-트리[12]와 같은 트리 구조를 가지는 다차원 인덱스(multidimensional index)를 사용한다. 그러나 기존의 다차원 인덱스들은 차원 수가 낮은 경우 매우 좋은 성

· 본 연구는 강원대학교 멀티미디어 연구센터를 통한 정보통신 우수시범학교 지원사업과 한국과학재단의 99 해외 Post-Doc 방문 프로그램 및 2000 목적기초연구 중 지역대학 우수과학자 지원 프로그램(과제 번호: 2000-1-51200-006-1)의 연구비 지원을 받았습니다.
† 종신회원 : 강원대학교 컴퓨터정보통신공학부 교수
wook@kangwon.ac.kr
†† 비 회 원 : IBM T.J. Watson Research Center
논문접수 : 2000년 7월 27일
심사완료 : 2001년 7월 18일

능을 나타내지만, 차원 수가 높아짐에 따라 그 성능이 크게 떨어지는 것으로 알려져 있다[13, 14][1].

본 논문에서는 객체 클러스터(cluster)의 캡슐(capsule)을 표현하는 새로운 방식을 채택함으로써 최근접 질의를 효과적으로 처리할 수 있는 새로운 인덱싱 방법을 제시하고자 한다. 기존의 다차원 인덱스에서는 N차원 사각형 혹은 N차원 원을 사용하여 클러스터의 캡슐을 표현한다. 그러나 이러한 방식을 사용하는 경우, 캡슐 내의 죽은 영역(dead space)[12]의 크기가 커지게 되므로 최근접 질의 처리의 성능이 저하된다. 본 논문에서는 이를 해결하기 위하여 채택하는 방안은 다음과 같다.

(1) 클러스터 내에 상관 관계가 있는 경우, 해당 클러스터에 적합한 새로운 축 시스템(axis system)을 채택하고, 이를 기준으로 클러스터의 캡슐을 결정한다.
(2) 원과 사각형을 조합함으로써 죽은 영역을 최소화하는 다양한 캡슐 형태를 지원한다.
(3) 클러스터 내에서 죽은 영역의 크기를 지나치게 증가시키는 아웃 라이어를 검출하여 클러스터와 별도로 관리한다.

또한, 이러한 개념들을 채택하는 인덱싱 기법을 제안하고, X-트리, 순차 검색 등 기존 기법들과의 다양한 실험을 통한 성능 평가에 의하여 제안된 기법의 우수성을 검증한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 기존의 연구들을 간략히 요약하고, 장단점을 지적한다. 제 3장에서는 클러스터를 위한 캡슐의 형태로서 사각형 혹은 원을 채택하는 기존의 다차원 인덱스 기법들의 공통적인 문제점을 지적한다. 제 4장에서는 이러한 문제점을 해결하는 새로운 인덱싱 기법을 제시한다. 제 5장에서는 다양한 실험을 통하여 제안된 기법의 성능을 검증한다. 제 6장에서는 결론을 내리고, 향후 연구 방향을 제시한다.

2. 관련 연구

본 장에서는 관련 연구로서 고차원 인덱싱과 최근접 질의 처리에 관한 기존의 주요 연구 결과에 관하여 간략히 요약한다.

SS-트리[15]는 캡슐의 형태로서 N차원 사각형을 사용하는 R*-트리[12]와는 달리 N차원 원을 사용함으로써 저장 공간의 효율과 최근접 질의 처리의 성능을 동시에 개선한다. 또한, R*-트리의 재 삽입 알고리즘을 개선함으로써 보다 효과적인 트리 구조를 구성할 수 있도록 해 준다.

VAMSplit R-트리[16]는 미리 주어진 객체 집합에 대하여 탑-다운 방식으로 트리를 구성함으로써 최적화된 R-트리[10] 구조를 지향한다. 또한, 최적화된 K-D-트리[17]에서 사용하는 분할 알고리즘을 채택함으로써 할당되는 디스크 블록의 수를 최소화할 수 있도록 분할점을 결정한다. 캡슐의 형태로는 N차원 사각형을 사용한다. 참고 문헌 [16]에 의하면, VAMSplit R-트리가 SS-트리 및 R*-트리에 비하여 좋은 성능을 갖는다고 알려져 있다.

TV-트리[18]는 차원 축소(dimensionality reduction)과 액티브 차원의 시프트(shift of active dimensions)를 이용하여 R*-트리의 고차원 문제를 해결한다. 또한, 캡슐의 형태로는 N차원 원을 사용한다. 그러나 이 방식은 (1) 차원들이 중요도에 따라 순서화 되어야 하고, (2) 액티브 차원의 시프트를 허용해야 한다는 두 가지 조건을 만족하는 경우에 한하여 효과적이다. 따라서 TV-트리는 적용 가능한 응용 분야가 매우 제한적이다.

X-트리[14]는 R*-트리의 변형으로서 수퍼 노드(super-node)의 개념을 이용한다. 수퍼 노드는 다수의 연속된 블록으로 구성되는 큰 노드이다. 수퍼 노드는 두 노드간의 교차 영역(overlap)이 지나치게 커지는 것을 방지하며, 이 결과 질의 처리 성능을 개선하는 효과를 갖는다. 캡슐의 형태로는 N차원 사각형을 사용한다. 현재까지 제안된 고차원 인덱스들 중 좋은 구조로 알려져 있다.

MVP-트리[19]는 밴티지 점(vantage point)로부터의 거리를 기준으로 영역을 분할하는 VP-트리[20]의 개념을 확장한 구조이다. MVP-트리는 트리의 각 단계에서 두 개 이상의 밴티지 점을 사용함으로써 트리의 팬 아웃(fan-out)을 증대시킨다. 또한, 미리 계산된 밴티지 점과 객체 점간의 거리를 저장함으로써 검색 성능을 개선시킨다. 캡슐의 형태로는 N차원 원을 사용한다.

SR-트리[21]는 캡슐의 형태를 표현하기 위하여 N차원 사각형과 원을 모두 사용한다. 즉, SR-트리의 영역은 해당 영역을 커버하는 사각형과 원의 교집합으로 표현된다. 따라서 사각형 혹은 원만으로 표현하는 방식에 비하여 캡슐의 크기가 작아지며, 이 결과 질의 처리 성능을 개선하는 효과를 갖는다.

또한, 최근접 질의 처리를 위한 근사 알고리즘(approximate algorithm)들이 제안된 바 있다[22, 23].

1) 특히, 참고 문헌 [13]에 의하면, 차원 수가 10이상인 경우 이러한 트리 구조의 인덱스는 순차 검색(sequential scan)보다도 떨어지는 결과를 가지는 것으로 나타났다.

이 알고리즘들은 최근접 질의 결과의 정확도를 약간 희생함으로써 질의 처리 성능을 획기적으로 개선하기 위한 것이다. 그러나 참고 문헌 [24]의 분석 결과에 의하면, 이러한 근사 처리 방식은 실제 응용에서 많은 중요한 최근접 질의 결과를 잃을 위험이 큰 것으로 나타났다.

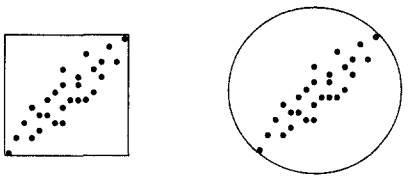
3. 연구 동기

최근접 질의를 처리하기 위한 기존의 기법들은 브랜치-앤드-바운드(branch and bound)를 기반으로 한다. 브랜치-앤드-바운드는 전체 탐색 공간에서 주어진 상한값(upper bound) 이상인 영역들을 이후의 조사 대상에서 제외시킴으로써 탐색을 효과적으로 수행하는 전통적인 방법이다[25]. 기존의 기법들[6, 7, 8, 9]은 다차원 인덱스를 이용하여 전체 객체들을 다수의 객체 클러스터(cluster)들의 집합으로 구분하고[2], 주어진 질의 점으로부터 현재까지 구한 가장 가까운 최근접 객체(nearest neighbor object)보다 먼 곳에 있는 클러스터는 미리 조사 대상에서 제외하는 방식을 사용한다.

다차원 인덱스내의 각 엔트리는 고유의 클러스터와 대응되며, 클러스터내의 모든 객체 점들을 포함하는 N차원 캡슐(capsule)을 사용하여 클러스터를 표현한다. 좋은 브랜치-앤-바운드 효과를 얻기 위해서는 캡슐이 클러스터내의 객체 점들을 모두 포함하되, 내부의 죽은 영역(dead space)의 크기가 가능한 작게 형성되어야 한다. 죽은 영역이란 캡슐 내에서 객체들이 실제로 존재하지 않는 영역을 의미한다. 현재, 제안된 대부분의 다차원 인덱스에서는 캡슐의 형태로서 N차원 사각형(rectangle)을 사용하며, 몇몇 기법에서는 N차원 원(sphere)을 사용하기도 한다[15, 21, 26].

캡슐의 형태로 사각형 혹은 원을 사용하는 기존의 기법에서는 클러스터 내에 상관 관계(correlation)[27]가 커질수록 캡슐내의 죽은 영역의 크기가 커진다. "클러스터 C가 축 X와 Y에 상관 관계를 갖는다"라는 의미는 클러스터내의 객체의 서로 다른 두 축 X와 Y의 값은 서로 연관성을 갖는다는 의미이다. 그림 1은 2차원 공간에서 상관 관계를 가지는 한 객체 클러스터를 투영한 것이다. 클러스터가 축들 간에 상관 관계를 갖는 경우, 사각형과 원의 형태를 가지는 기존의 캡슐 표현 방식에서는 죽은 영역이 매우 커짐을 볼 수 있다.

축들간의 상관 관계가 커질수록 이러한 죽은 영역은



(a) 사각형 형태의 캡슐 (b) 원 형태의 캡슐

그림 1 상관 관계가 있는 클러스터에서의 캡슐의 형태와 죽은 영역의 관계

커지게 된다. 이러한 죽은 영역의 증가는 최근접 질의의 처리 성능을 저하시키는 주요 원인이 된다. 특징 벡터의 차원 수가 증가할수록 임의의 두 축 사이에 상관 관계가 존재할 가능성이 커지므로 고차원 벡터 공간을 대상으로 하는 멀티미디어 응용에서는 이러한 죽은 영역의 크기를 최소화 할 수 있는 캡슐 표현 전략이 필요하다.

4. 제안하는 기법

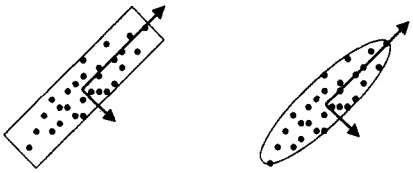
본 장에서는 제 3장에서 지적한 문제점을 해결하는 새로운 인덱싱 기법을 제안한다. 제 4.1절에서는 현재 고정된 축 시스템을 변환함으로써 죽은 영역을 줄이는 방안에 대하여 설명하고, 제 4.2절에서는 원과 사각형을 조합함으로써 캡슐의 크기를 최적화 하는 방안에 대하여 설명한다. 제 4.3절에서는 아웃 라이어로 인하여 캡슐의 크기가 비정상적으로 커지는 것을 방지하기 위한 방안에 대하여 논의한다. 제 4.4절에서는 이러한 개념들을 통합한 구체적인 인덱스 구조를 제시한다.

4.1 축 시스템 변환

본 논문에서는 클러스터 내에 상관 관계가 있는 경우, 죽은 영역이 증가하는 문제를 해결하는 근본적인 방법으로서 해당 클러스터에 맞는 새로운 축 시스템(axis system)을 사용하는 것을 제안한다. 즉, 해당 클러스터내의 객체들의 분포를 조사함으로써 죽은 영역을 최소화할 수 있는 축 시스템을 채택하고, 이를 기준으로 해당 클러스터를 포함하는 캡슐을 결정하는 것이다. 그림 2(a)는 그림 1에 나타난 클러스터를 새로운 축 시스템과 사각형 캡슐을 이용하여 표현한 것이다. 두 화살표는 새롭게 채택된 새로운 축 시스템을 의미한다. 이 축 시스템을 사용하는 경우, 사각형 캡슐은 원 축 시스템을 사용할 때와 마찬가지 방식으로 우상점(upper right corner)과 좌하점(lower left corner)으로 쉽게 표현된다. 이렇게 새로운 축 시스템을 사용하는 경우, 죽은 영역의 크기가 현저하게 감소함을 볼 수 있다. 차원의 수

2) 대부분의 다차원 인덱스들은 트리 구조를 사용하므로, 인덱스의 상위 단계에서 하위 단계로 내려갈수록 보다 세분화된 클러스터들을 만나게 된다.

가 커질수록 이러한 죽은 영역의 감소 효과는 더욱 커지게 된다.



(a) 사각형 캡슐의 사용 (b) 타원 캡슐의 사용

그림 2 변환된 축 시스템의 적용

4.2 캡슐 형태의 표현

캡슐의 형태로서 널리 사용되는 사각형이나 원 외에도 타원을 고려할 수 있다. 그림 2(b)는 타원을 이용하여 해당 클러스터의 캡슐을 표현한 것을 나타낸 것이다. 그림에서 나타난 바와 같이, 캡슐의 형태로서 타원을 이용하는 경우 죽은 영역의 크기를 더욱 줄일 수 있는 경우가 발생한다. 그러나 브랜치-앤드-바운드를 위해서는 주어진 질의 점과 캡슐간의 최소 거리를 계산해야 하는데, 타원은 사각형 혹은 원과는 달리 주어진 질의 점으로부터의 최소 거리의 유추가 매우 복잡하며 계산의 오버헤드가 매우 크다. 따라서 본 논문에서는 타원을 이용하여 캡슐을 표현하는 방식을 제외하였다.

이 대신, 본 논문에서는 원과 사각형을 조합함으로써 죽은 영역의 크기를 최소화하는 다양한 형태의 N차원 캡슐을 표현하는 방식을 이용한다. 먼저, $\{r_1, r_2, ..., r_k\}$를 구한다. 여기서, r_i는 변환된 축 시스템에서 클러스터 내의 객체들이 갖는 i-번째 축의 평균값이다. 이후부터는 클러스터의 분포 형태에 맞도록 변환된 새로운 축 시스템을 기준으로 논의를 전개한다. 또한, 설명의 편의상 $r_1 \le r_2 \le r_3 ... \le r_k$ 라 가정한다[3]. 다음에는 $\{ r_i \}$를 m($\le$k)개의 그룹 $[r_1, .., r_{f(1)}]$, $[r_{f(1)+1}, .., r_{f(2)}]$, ..., $[r_{f(m-1)+1}, r_{f(m)}]$ 로 분할한다. 여기서, f(i)는 $r_{f(i)}/r_{f(i-1)+1} < 2$를 만족하도록 선택된다. 예를 들어, $\{ r_i \}$가 {1, 2, 3, 5, 8, 12, 15, 22, 26, 40}인 10차원 공간 클러스터의 경우, 위의 방법을 적용하면, [1, 2], [3, 5], [8, 12, 15], [22, 26, 40]의 네 그룹이 형성된다.

클러스터를 위한 캡슐은 다음의 식들 표현하는 다차원 원들의 교집합으로 구성되는 공간에 의하여 결정된다. $x_1^2 + ... + x_{f(1)}^2 \le a_1^2$, $x_{f(1)+1}^2 + ... + x_{f(2)}^2 \le a_2^2$, $x_{f(2)+1}^2 + ... + x_{f(3)}^2 \le a_3^2$, ..., $x_{f(m-1)+1}^2 + ... + x_{f(m)}^2 \le a_m^2$. 여기서, a_i는 i-번째 그룹에 의하여 표현되는 다차원 원의 지름을 의미하며, 구체적인 값을 결정하는 방법에 대해서는 제 4.3절에서 논의한다.

그림 3은 3차원 공간에서 클러스터의 분포에 따라 가능한 모든 분할 방식과 이에 의한 캡슐의 형태를 도면화 한 것이다. Case 1은 클러스터내의 객체들의 각 축의 평균값이 유사하여 모든 축이 같은 그룹에 포함된 경우를 나타낸다. Case 2는 객체들의 x, y 축의 평균값은 유사하지만, z 축의 평균값은 상이한 경우를 나타내며, Case 3은 반대로 객체들의 y, z 축의 평균값은 유사하지만, x 축의 평균값은 상이한 경우를 나타낸다. 마지막으로 Case 4는 객체들의 모든 축의 평균값이 모두 상이한 경우를 나타낸다.

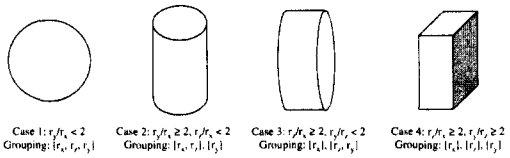


그림 3 3차원 공간에서 캡슐 형태의 결정

4.3 아웃 라이어 처리

캡슐의 기본 형태는 분할의 결과에 의하여 결정되지만, 각 그룹의 형태를 결정하는 원의 지름 혹은 사각형의 변은 a_i 값에 의하여 결정된다. 가장 간단한 방법의 하나는 중심점과 이로부터 가장 멀리 떨어진 객체와의 거리를 a_i 값으로 이용하는 것이다. 이 경우, 클러스터 내의 모든 객체들은 결정된 캡슐 내에 포함된다.

그러나 이 방법의 큰 문제점은 죽은 영역이 매우 커질 수 있다는 것이다. 그림 4에 나타난 클러스터 c1의 예를 살펴보자. 대부분의 객체들은 2차원 공간의 인접한 위치에 밀집해 있는 반면, 하나의 객체 o1은 이와 매우 떨어진 위치에 존재한다. 모든 객체를 포함하도록 캡슐 1과 같은 형태를 취하는 경우 죽은 영역이 지나치게 많아지고, 이 결과 최근접 질의의 성능이 크게 저하된다. 반면, 지나치게 동떨어진 객체를 클러스터부터 제외하고, 캡슐 2와 같은 형태를 취하면 대부분의 죽은 영역을 제거할 수 있다. 이때, 클러스터로부터 제외되는 객체를 아웃 라이어(outlier)라 정의한다. 이러한 아웃 라이어는 클러스터와 별도로 관리되어야 한다.

본 연구에서는 a_i 값을 먼저 결정함으로써 해당 캡슐의 형태를 결정하고, 이 캡슐 밖에 존재하는 객체들은 모두 아웃 라이어라 간주한다. a_i 값의 결정을 위하여 다음의 정리 1을 사용한다.

3) 물론, 일반적인 경우에는 이 조건이 만족하지 않으므로 먼저 r_i의 값에 따라 정렬해야 한다.

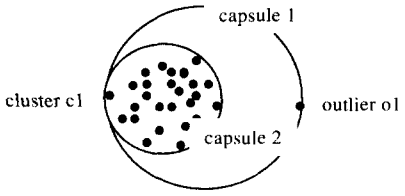


그림 4 클러스터, 아웃 라이어, 캡슐의 관계

정리 1: N차원 공간의 반지름이 R인 원 S에서 n개의 객체 점들이 균일하게 분포될 때, 중심점으로부터 객체 점들의 거리의 평균값 μ과 표준 편차 σ는 다음과 같다.

$$\mu = R \times \frac{N}{(N+1)}$$

$$\sigma = R \times \frac{1}{(N+1)} \times \sqrt{\frac{N}{N+2}}$$

증명:

N차원 공간에서 반지름이 r인 *임의의* 원은 표면적이 $K_N \times r^{N-1}$이며 체적 V는 $\frac{K_N \times r^N}{N}$이다. 여기서, K_N은 π를 포함하는 상수이며, 차원에 따라 다르게 결정된다 [28]. 원의 중심으로부터 임의의 거리 r만큼 떨어진 곳의 매우 얇은 두께 dr를 갖는 껍질의 부피는 $K_N \times r^{N-1} \times dr$ 이며, 이 껍질 내에 존재하는 객체 점들의 수 n_r은 $K_N \times \frac{n}{V} \times r^{N-1} \times dr$ 이다. 따라서 중심으로부터 객체 점들의 평균 지름은 다음과 같다.

$$\mu = \frac{\int_{r=0}^{R} r \times n_r dr}{n} = K_N \times \frac{1}{V} \times \frac{R^{N+1}}{N+1} = R \times \frac{N}{(N+1)}$$

같은 방법을 적용하여 표준 편차의 제곱을 다음과 같이 구할 수 있다.

$$\sigma^2 = \frac{\int_{r=0}^{R} (r-\mu)^2 \times N^r dr}{n} = R^2 \times \left(\frac{N}{N+2} - \frac{N^2}{(N+1)^2} \right)$$

$$= R^2 \times \frac{1}{(N+1)^2} \times \frac{N}{N+2}$$

따라서 정리 1은 성립한다.

□

정리 1은 차원 수가 커짐에 따라 $\mu+\sigma$가 R 값에 가까워짐을 의미한다[4]. 아웃 라이어는 클러스터와 독립적으로 관리되므로 질의 점의 위치에 관계없이 항상 액세스해야 하는 객체들임을 의미한다. 따라서 아웃 라이어 수가 지나치게 많아지면, 이를 별도로 처리하는 부분에서 병목 현상이 발생하게 된다. 만일, a_i 값으로서 정리 1의 $\mu+\sigma$를 그대로 사용하는 경우, 중심으로부터 R 만큼 떨어진 거리에 산재하는 많은 수의 객체들이 아웃 라이어로 간주될 가능성이 커진다. 따라서 본 연구에서는 정리 1을 기반으로 하되 클러스터내의 대부분의 객체들을 가능한 모두 포함할 수 있는 a_i 값으로서 $\mu+2\sigma$를 이용하기로 결정하였다[5].

4.4 인덱스 구조 및 질의 처리 방안

본 절에서는 지금까지 기술한 캡슐의 형태 표현 방식을 채택하는 인덱스의 구조와 이를 이용한 최근접 질의의 처리 방안을 제안한다.

그림 5에 나타난 바와 같이 제안된 인덱스 구조는 다수의 엔트리들로 구성되는 디렉토리와 페이지들의 집합으로 구성된다. 디렉토리는 주기억장치 내에서 관리되며, 페이지는 디스크 내에서 관리된다. 각각의 디렉토리 엔트리는 클러스터와 일대일 대응되며, 클러스터에 대한 다음과 같은 정보를 포함한다.

- 축 시스템 정보: 제 4.1절과 관련하여 이 클러스터가 어떤 축 시스템을 채택하는가에 대한 정보를 포함한다. 축 시스템은 N개의 벡터로 구성된다.
- 그룹 정보: 제 4.2절과 관련하여 이 클러스터에서 채택한 캡슐의 모양을 표현하기 위하여 전체 축들이 몇 개의 그룹으로 나누어지며, 각 축이 어떤 그룹에 속하는가에 대한 정보를 포함한다. 각 축은 그 축이 속하는 그룹 번호를 갖는다.
- 크기 정보: 제 4.3절과 관련하여 이 클러스터 채택한 각 그룹이 어떤 a_i 값을 갖는가에 대한 정보를 포함한다. 각 그룹은 하나의 a_i 값을 갖는다.

실제 객체들은 디스크 내에 존재하는 페이지 내에 저장된다. 즉, 하나의 페이지는 다수의 객체들을 저장하는 객체의 저장 단위로서, 개념적으로는 객체 배열이라 간주할 수 있다. 같은 클러스터 내에 포함되는 객체들의 페이지들은 연결 리스트(linked list) 형태로 관리되며, 이 연결 리스트의 시작 페이지를 해당 디렉토리 엔트리가 가리키게 된다. 디렉토리 엔트리 수는 응용 환경에 따라 조절할 수 있다.

이러한 인덱스 구조는 이미 존재한 객체들의 집합을

4) 물론, 실제 환경에서는 클러스터 내의 객체 점들이 원의 표면 근처에서 보다 산발적으로 분포할 가능성이 높다.

5) 다차원 객체 집합에서 아웃 라이어를 검출하는 것은 데이타 마이닝 분야의 중요한 문제 중의 하나이다. 참고 문헌 [29, 30, 31, 32] 등에서는 이러한 아웃라이어 검출 문제에 대하여 집중적으로 다루고 있다. 아웃 라이어는 응용의 특성에 따라 그 정의에 차이가 있고, 따라서 그 검출 방법 또한 달라질 수 있다. 응용에 대한 분석을 기반으로 해당 응용에 가장 적합한 아웃 라이어 검출 방법을 사용하여 하는 경우, 제안된 기법의 성능에 긍정적인 영향을 미칠 수 있다. 그러나 이것은 본 논문의 공헌에서 다루고자 하는 주요 논점이 아니므로 이에 대한 자세한 설명은 생략한다.

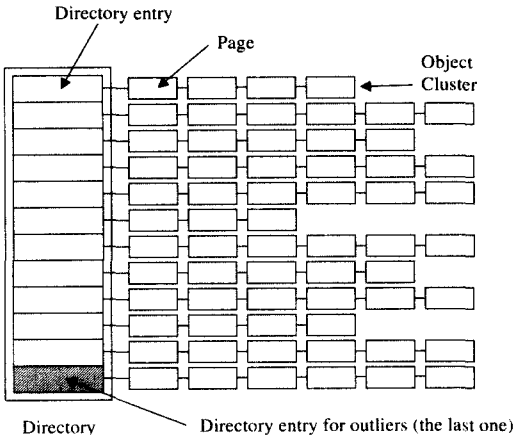


그림 5 인덱스 구조

대상으로 하는 일괄 구성 방식으로 생성된다. 인덱스 구조를 위한 일괄 구성 알고리즘은 Algorithm 1과 같다.

단계 1에서 선택되는 시드 객체는 각 클러스터의 중심점이 된다. 단계 2에서 각 객체가 소속될 클러스터를 파악하는 기준으로는 해당 객체와 시드 객체간의 거리의 역을 사용한다. 단계 3에서 주어진 클러스터에 가장 적합한 축 시스템은 Principal Component Analysis에서 널리 사용되는 SVD(singular value decomposition) 기법[33]을 이용한다. 단계 4에서 캡슐의 형태의 결정은 제 4.2절의 방법을 이용한다.

Algorithm 1 인덱스 구조를 위한 일괄 구성 알고리즘

1. 각 클러스터를 위한 시드 객체를 객체 집합으로부터 무작위로 선택한다.
2. 각 객체를 차례로 읽어들여 소속될 클러스터를 파악한다.
3. 각 클러스터에 적합한 최적의 축 시스템을 결정한다.
4. 각 클러스터를 위한 캡슐의 형태와 a_i 값을 결정한다.
5. 각 클러스터에서 캡슐 밖에 존재하는 아웃 라이어들을 파악하여 마지막 디렉토리 엔트리에 옮긴다.

이러한 인덱스 구조를 기반으로 하는 최근접 질의 처리 알고리즘은 Algorithm 2와 같다.

단계 1에서 마지막 디렉토리 엔트리가 가리키는 아웃 라이어들은 모든 클러스터의 캡슐 밖에 있는 객체들을 수집하여 함께 저장한 것으로서 질의 조건에 관계없이 항상 조사되어야 한다. 이들의 조사를 다른 엔트리들의 조사가 끝난 후, 최종적으로 하는 것을 고려할 수도 있다. 그러나 O_{min}에 해당되는 객체가 아웃 라이어로 존재하는 경우, 클러스터들을 모두 조사해야 하는 경우가 발생하므로 커다란 성능 저하를 유발한다. 반면, 이들을 가장 먼저 조사하는 제안된 기법에서는 작은 D_{min} 값을 조기에 확보함으로써 O_{min}을 포함할 가능성이 없는 클러스터를 미리 조사 대상에서 제외시킬 수 있는 장점이 있다.

Algorithm 2 최근접 질의 처리 알고리즘

1. 마지막 디렉토리 엔트리가 가리키는 아웃 라이어를 조사하여 질의 점과 가장 가까운 객체 O_{min}과 이 객체가 질의 점으로부터 떨어진 거리를 D_{min}을 찾아낸다
2. 마지막 엔트리를 제외한 모든 엔트리들을 질의 점과 해당 엔트리가 표현하는 캡슐간의 거리 D_{entry}를 계산하고, 이 값을 기준으로 정렬한다
3. **FOR** 이 정렬된 각 엔트리에 대하여,
   IF 엔트리가 조건 ($D_{min} > D_{entry}$)를 만족하면,
   THEN
   3.1 그 엔트리와 대응되는 클러스터를 조사하여 새로운 D_{min}과 O_{min}을 찾아낸다.
   ELSE
   3.2 현재까지 찾은 D_{min}과 O_{min}을 반환하고, 질의 처리를 종료한다.

단계 2에서 정렬은 실제 디렉토리 엔트리 자체를 대상으로 하는 것이 아니라 <entryNum, distance>의 쌍으로 구성되는 요약 엔트리를 대상으로 한다. entryNum은 디렉토리 엔트리 번호를 의미하며, distance는 질의 점과 entryNum과 대응되는 클러스터의 캡슐간의 거리를 의미한다. 이와 같이 요약 엔트리를 사용하도록 함으로써 정렬시의 CPU 시간을 최적화 할 수 있다.

단계 2에서 최근접 질의 처리에서 질의 점과 해당 엔트리가 표현하는 캡슐간의 거리를 계산하는 과정에서 고유의 축 시스템으로의 변환이 요구된다. 그러나 해당 엔트리와 대응되는 클러스터 내에 속하는 각 객체와 질의 점간의 거리 계산은 축 시스템 변환을 유발하지 않는다. 따라서 데이타베이스 내의 전체 객체 점들의 수를 고려할 때, 이 축 시스템 변환의 오버헤드는 상대적으로 미미하다.

5. 성능 평가

본 장에서는 다양한 실험에 의한 성능 분석을 통하여 제안된 기법의 우수성을 검증한다. 제 5.1절에서는 실험 환경에 대하여 설명하고, 제 5.2절에서는 실험 결과를 분석한다.

5.1 실험 환경

본 실험에서는 제안된 기법의 비교 대상으로서 X-트리[14]와 순차 검색(sequential scan)을 채택한다. X-트리는 가장 뛰어난 성능을 가지는 다차원 인덱스들 중 하나로 알려져 있으며, 순차 검색은 고차원 벡터 공간에서 다차원 인덱스보다 오히려 나은 성능을 가진다고 알려져 있다[13]. 성능 평가 지수로는 DBMS 연산의 비용에 대부분을 차지하는 페이지 액세스 수를 사용한다[6]. 페이지 크기로는 실제 DBMS에서 가장 널리 사용하는 4K 바이트를 세 가지 기법 모두에서 공통적으로 사용한다. 제안된 기법에서 디렉토리 엔트리의 수는 1,000으로 설정하였다.

실험에서 사용된 객체 집합들은 2차원에서 100차원까지의 다양한 차원 수를 가진다. 각 객체 집합은 클러스터들의 집합으로 구성되며, 전체 100,000개의 객체들을 포함한다. 각 클러스터 내에 속하는 객체들이 서로 다른 축들간의 상관 관계를 갖도록 다음과 같은 방식을 이용하여 각 클러스터 내의 객체들을 생성한다. (1) 클러스터 내에 속하게 될 객체 수 X를 선택한다. (2) 클러스터의 축 시스템을 무작위로 결정한다. (3) 각 축에 대하여 X개의 객체들이 평균값 0, 표준편차 값 Y인 정규 분포를 취하도록 객체의 각 축 값을 생성한다. (4) 생성된 클러스터 내의 객체들을 단계 (1)에서 결정한 축 시스템에 맞도록 회전시킨다. (5) 각 축에 대하여 클러스터의 중심점의 값을 -1,000,000,000~1,000,000,000 사이에서 무작위로 선택한 후, 클러스터 내의 객체들을 선택된 중심점에 맞도록 이동시킨다.

이와 같은 객체 집합 생성 방법에서 사용되는 X 값에 따라 다음과 같이 분류한다.

- MC(many clusters): X를 500~1,000 사이에서 무작위로 취하도록 함으로써 객체 집합 내에는 적은 수의 객체들을 가지는 많은 수의 클러스터들이 존재한다.
- FC(few clusters): X를 5,000~10,000 사이에서 무작위로 취하도록 함으로써 객체 집합 내에는 많은 수의 객체들을 가지는 적은 수의 클러스터들이 존재한다.

또한, 객체 집합 생성 방법에서 사용되는 Y 값에 따라 다음과 같이 분류한다.

- LS(large standard deviation): 각 차원에 대한 Y를 1,000,000 ~ 10,000,000 사이에서 무작위로 취하도록 함으로써 클러스터내의 객체들이 넓은 공간 내에서 분포한다.
- SS(small standard deviation): 각 차원에 대한 Y를 10,000~100,000 사이에서 무작위로 취하도록 함으로써 클러스터내의 객체들이 좁은 공간 내에서 분포한다.

이러한 X, Y의 가능한 값의 설정에 따라 실험에서는 MC/LS, MC/SS, FC/LS, FC/SS의 네 가지 조합의 객체 세트들을 사용한다. 또한, 이와 별도로 모든 차원간의 상관 관계가 없으며, 각 차원의 값들이 균일하게 분포하는 UU를 기본적인 객체 세트로서 사용한다.

제안된 기법이 실제 응용에서 잘 동작한다는 것을 규명하기 위하여 인위적으로 생성된 객체 집합 외에도 실제 이미지 데이타를 이용한 실험도 함께 수행한다. Real ImageData는 URL http://core.digitalriver.com에 저장된 68,000여 개의 이미지들로부터 추출한 특징 벡터의 집합이다. 본 실험에서는 이를 이용하여 실제 상황에서 제안된 기법이 어떻게 동작하는가에 관하여 분석한다.

각 객체 세트에 대한 실험에서는 200개의 질의 점에 대한 최근접 질의를 처리하도록 한다. 질의 점들은 객체들의 수와 비례하여 각 클러스터에 할당되며, 클러스터 내에서 질의 점의 분포는 클러스터 내의 객체 분포를 그대로 따르도록 한다[7].

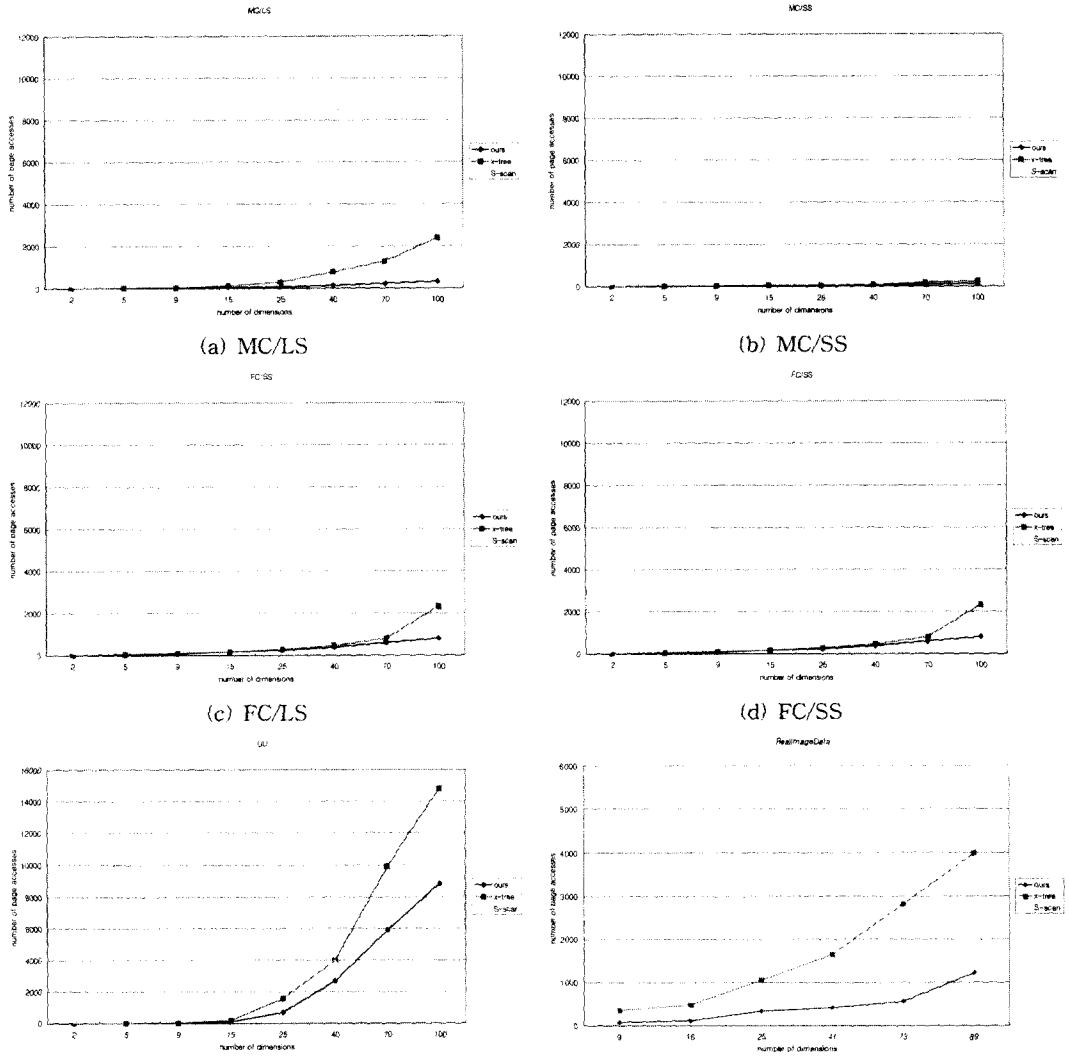
5.2 실험 결과

그림 6은 실험에서 사용된 다섯 가지 객체 집합에 대한 성능 평가 결과를 나타낸 것이다. 각 그래프에서 가로축은 객체 집합의 차원 수를 의미하며, 세로 축은 200개의 최근접 질의를 처리할 때, 발생되는 페이지 액세스 수의 평균값을 나타낸 것이다.

먼저, 그림 6(a)~6(d)에 나타난 객체 집합들 MC/LS, MC/SS, FC/LS, FC/SS의 결과를 살펴보자. 순차 검색은 모든 경우에서 제안된 기법 및 X-트리에 비하여 떨어지는 성능을 보였다. 이것은 제안된 기법 및 X-트리에서 사용되는 브랜치-앤드-바운드가 효과적으로 수행되었음을 의미한다. 차원 수가 15 이하인 경우에는 제안된 기법과 X-트리의 성능 차이가 거의 없는 것으로 나타났다. 그러나 차원 수가 25 이상인 고차원 객체 집합에서는 제안된 기법의 성능이 일관적으로 우수한 것으로 나타났으며, 이러한 경향은 차원 수가 커짐에 따라

6) 물론, CPU 시간과 디스크(페이지) 액세스 시간을 모두 포함한 전체 검색 시간을 고려할 수도 있다. 그러나 이러한 실제 시간은 사용되는 하드웨어 플랫폼에 따라 많은 차이가 있으므로 고려 대상에서 제외하였다. CPU 시간은 디스크 액세스 시간에 비하여 상대적으로 무시할 만큼 작으므로, 데이타베이스 관련 연구에서는 디스크 액세스 수를 성능 평가 지수로 삼는 경향이 크다.

7) 이러한 질의 점 생성은 "질의 분포는 객체의 분포를 따른다" [34]는 실제 응용에서의 경향을 반영하기 위한 것이다.



(a) MC/LS (b) MC/SS

(c) FC/LS (d) FC/SS

(e) UU (f) RealImageData

그림 6 차원 수 증가에 따르는 최근접 질의 처리 성능

더욱 커지는 것으로 나타났다. 이것은 제안된 기법에서 채택한 클러스터 캡슐의 표현 방식이 매우 효과적임을 보이는 것이다.

다음으로 그림 6(e)에 나타난 객체 집합 UU의 결과를 살펴보자. 차원 수가 40 미만인 경우에는 X-트리가 순차 검색과 비교하여 나은 성능을 보였으나, 차원 수가 40 이상인 경우에는 오히려 순차 검색의 성능이 오히려 우수한 것으로 나타났다. 이는 "고차원의 균일 데이타 분포에서는 순차 검색이 다차원 인덱스보다 우수한 결과를 갖는다"는 참고 문헌 [13]의 분석 결과와 일치하는 것이다. 그러나 제안된 기법은 모든 차원의 객체 집합에 대하여 X-트리는 물론 순차 검색과 비교해서도 우수한 성능을 가지는 것으로 나타났다.

끝으로, 그림 6(f)에 나타난 실제 이미지 집합 RealImageData의 결과를 살펴보자. 순차 검색은 차원 수에 관계없이 X-트리에 비하여 성능이 떨어지는 것으로 나타났다. 이것은 참고 문헌 [13]의 분석과는 달리 고차원 공간상에서 실제 응용에서는 데이타가 균일하게

분포하지 않음을 의미하는 것이다. 제안된 기법은 모든 차원에 대하여 X-트리 및 순차 검색과 비교하여 우수한 성능을 보였다. 이는 제안된 기법이 실제 응용에 잘 적용할 수 있는 바람직한 인덱싱 기법임을 나타내는 것이다.

6. 결 론

멀티미디어 데이타베이스 응용에서는 최근접 질의의 효과적인 지원이 매우 중요하다. 이를 위하여 제안된 기존의 다차원 인덱스들은 차원 수가 높아짐에 따라서 성능이 급격히 저하되는 문제점을 가진다. 본 논문에서는 이러한 성능 저하가 클러스터 캡슐내의 죽은 영역의 크기 증가로 인한 것임을 밝히고, 이를 해결하기 위한 다음과 같은 방안을 제시하였다.

(1) 클러스터 내에 상관 관계가 있는 경우, 해당 클러스터에 적합한 새로운 축 시스템을 찾아 이를 기준으로 클러스터의 캡슐을 결정한다.

(2) 원과 사각형을 조합함으로써 죽은 영역을 최소화하는 다양한 캡슐 형태를 지원한다.

(3) 클러스터 내에서 죽은 영역의 크기를 지나치게 증가시키는 아웃 라이어를 검출하여 클러스터와 별도로 관리한다.

또한, 이러한 개념들을 채택하는 구체적인 인덱스 구조를 제안하였다. 제안된 기법의 우수성을 규명하기 위하여 다양한 실험을 통한 성능 평가를 수행하였다. 최근접 질의 처리를 대상으로 한 실험 결과에 의하면, 제안된 기법은 15차원 이하의 저차원 객체 집합에 대해서는 X-트리와 유사한 성능을 보였다. 또한, 25차원 이상의 고차원 객체 집합에 대해서는 객체 분포에 관계없이 X-트리 및 순차 검색에 비하여 우수한 성능을 보이는 것으로 나타났다. 이것은 제안된 기법이 고차원 데이타의 인덱싱을 위한 매우 효과적인 방법임을 의미하는 것이다.

제안된 방법은 많은 객체가 존재하는 경우 이들을 대상으로 인덱스를 일괄 구성하는 방식을 사용한다. 따라서 제안된 방법은 응용으로서 객체의 삽입과 삭제가 매우 빈번하지 않은 정적인 환경을 주요 대상으로 하고 있다. 향후 연구 방향으로는 제안된 방법을 동적 환경으로 적용하는 효과적인 방안에 대하여 고려하고 있다.

참 고 문 헌

[1] C. Faloutsos et al., "Efficient and Effective Querying by Image Content, Journal of Intelligent Information Systems, Vol. 3, No. 3, pp. 231-262, 1994.

[2] C. Faloutsos, "Fast Searching by Content in Multimedia Databases," IEEE Data Engineering Bulletin, Vol. 18, No. 4, pp. 31-40, 1995.

[3] M. Arya et al., "QBISM: Extending a DBMS to Support 3D Medical Images, In Proc. Intl. Conf. on Data Engineering, IEEE, pp. 314-325, 1994.

[4] H. V. Jagadish, "A Retrieval Technique for Similar Shapes," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 208-217, 1991.

[5] W. Niblack et al., "The QBIC Project: Querying Images by Content, Using Color, Texture, and Shape," In Proc. Intl. Conf. Storage and Retrieval for Image and Video Databases, pp. 173-187, 1993.

[6] S. Berchtold et al., "Fast Nearest Neighbor Search in High-Dimensional Space, In Proc. Intl. Conf. on Data Engineering, IEEE, pp. 209-218, 1998.

[7] F. Korn et al., "Fast Nearest Neighbor Search in Medical Image Databases, In Proc. Intl. Conf. on Very Large Data Bases, VLDB, pp. 215-226, 1996.

[8] N. Roussopoulos, S. Kelley, F. Vincent, "Nearest Neighbor Queries," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 71-79, 1995.

[9] T. Seidl and H.-P. Kriegel, "Optimal Multi-Step k-Nearest Neighbor Search," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 154-165, 1998.

[10] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 47-57, 1984.

[11] T. Sellis, N. Roussopoulos, C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects, In Proc. Intl. Conf. on Very Large Data Bases, VLDB, pp. 507-518, 1987.

[12] N. Beckmann et al., "The R*-tree: an Efficient and Robust Access Method for Points and Rectangles," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 322-331, May 1990.

[13] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces, In Proc. Intl. Conf. on Very Large Data Bases, VLDB, pp. 194-205, 1998.

[14] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," In Proc Intl. Conf. on Very Large Data Bases, VLDB, pp. 28-39, 1996.

[15] D. A. White and R. Jain, "Similarity Indexing with the SS-tree," In Proc. Intl. Conf. on Data Engineering, IEEE, pp. 516-523, 1996.

[16] D. A. White and R. Jain, "Similarity Indexing: Algorithms and Performance," In Proc. Intl. Conf.

Storage and Retrieval for Image and Video Databases, SPIE, pp. 62-73, 1996.
[17] Sproull, "Refinements to Nearest Neighbor Searching in K-Dimensional Trees," Algorithmica, Vol. 6, No. 4, pp. 579-589, 1991.
[18] K. Lin, H. Jagadish, and C. Faloutsos, "The TV-Tree: An Index Structure for High Dimensional Data," The VLDB Journal, Vol. 3, No. 4, pp. 517-542.
[19] T. Bozkaya, and Z. Ozsoyoglu, "Distance-Based Indexing for High-Dimensional Metric Spaces," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 357-368, 1997.
[20] J. K. Uhlmann, "Satisfying General Proximity and Similarity Queries with Metric Trees," Information Processing Letters, Vol. 40, pp. 175-179, 1991.
[21] N. Katayama and S. Satoh, "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 369-380, 1997.
[22] E. Kushilevitz, R. Ostrovsky, and Y. Rabani, "Efficient Search for Approximate Nearest Neighbor Queries," In Proc. ACM Intl. Symp. on Theory of Computing, pp. 614-623, 1998.
[23] S. Pramanik, S. Alexander, and J. Li, "An Efficient Searching Algorithm for Approximate Nearest Neighbor Queries in High Dimensions," IEEE Multimedia Systems, pp. 865-869, 1999.
[24] K. Beyer et al., "When Is Nearest Neighbor Meaningful?," In Proc. Intl. Conf. on Database Theory, pp. 217-235, 1998.
[25] E. Horowitz, S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, 1978.
[26] P. Ciaccia, M. Patella, P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," In Proc Intl. Conf. on Very Large Data Bases, VLDB, pp. 426-435, 1997.
[27] A. M. Law and W. D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Company, New York, 1982.
[28] W. Fleming, Functions of Several Variable, 2nd Edition, 1977.
[29] E. M. Knorr and R. T. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," In Proc. Intl. Conf. on Very Large Data Bases, VLDB, pp. 392-403, 1998.
[30] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 427-438, 2000.
[31] M. M. Breunig et al., "LOF: Identifying Density-Based Local Outliers," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, pp. 93-104, 2000.
[32] C. Aggarwal and P. S. Yu, "Outlier Detection for High Dimensional Data," In Proc. Intl. Conf. on Management of Data, ACM SIGMOD, 2001. (accepted to appear)
[33] I.T. Jolliffe, Principal Component Analysis, Springer-Verlag 1986.
[34] B.-U. Pagel, H.-W. Six, and M. Winter, "Window Query-Optimal Clustering of Spatial Objects," In Proc. Intl. Conf. on Principals of Database Systems, pp. 86-94, 1995.

김 상 욱
정보과학회논문지 : 데이타베이스
제 28 권 제 2 호 참조

Charu C. Aggarwal received his B. Tech. degree in Computer Science from the Indian Institute of Technology (1993) and his Ph. D. degree in Operations Research from Massachusetts Institute of Technology (1996). He has been a Research Staff Member at the IBM T. J. Watson Research Center since June 1996. He has applied for or been granted 39 US patents, and has published in numerous international conferences and journals. He has been designated Master Inventor at IBM Research. His current research interests include algorithms, data mining, and information retrieval. He is interested in the use of data mining techniques for web and ecommerce applications.

Philip S. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is with the IBM Thomas J. Watson Research Center and currently manager of the Software Tools and Techniques group. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, disk arrays, computer architecture, performance modeling and workload analysis. Dr. Yu has published more than 300 papers in refereed journals and conferences. He holds or has applied for 244 US patents. Dr. Yu is a Fellow of the ACM and a Fellow of the IEEE. He is the Editor-in-Chief of IEEE Transactions on Knowledge and Data Engineering. He is also an associate editor of ACM Transactions on the Internet Technology and that of

Knowledge and Information Systems. He is a member of the IEEE Data Engineering steering committee and is also on the steering committee of IEEE Conference on Data Mining. He was an editor and advisory board member of IEEE Transactions on Knowledge and Data Engineering and also a guest co-editor of the special issue on mining of databases. In addition to serving as program committee member on various conferences, he was the program co-chair of the 11th Intl. Conference on Data Engineering and the program chairs of the 2nd Intl. Workshop on Research Issues on Data Engineering: Transaction and Query Processing, the PAKDD Workshop on Knowledge Discovery from Advanced Databases, and the 2nd Intl. Workshop on Advanced Issues of E-Commerce and Web-based Information Systems. He served as the general chair of the 14th Intl. Conference on Data Engineering and will be serving as the general co-chair of the 2nd Intl. Conference on Data Mining. He has received several IBM and external honors including Best Paper Award, 2 IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, 2 Research Division Awards and the 68th plateau of Invention Achievement Awards. He also received an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. Dr. Yu is an IBM Master Inventor and was recognized as one of the IBM's ten top leading inventors in 1999.