셀 기반 필터링 방법을 이용한 고차원 색인 기법

(A High-dimensional Indexing Scheme using Cell-based Filtering Technique)

장 재 우 † 한 성 근 †† 김 현 진 †††

(Jae-Woo Chang) (Sung-Geun Han) (Hyun-Jin Kim)

요 약 최근 이미지 특징 벡터와 같은 고차원 벡터 데이타에 관한 색인 기법들이 많이 연구되고 있다. 하지만, 기존의 색인 기법들은 저차원의 데이타에 대해서는 검색 성능이 우수하지만, 차원이 증가함에 따라 검색 성능이 급격히 저하되는 소위 'dimensional curse' 문제를 안고 있다. 따라서, 본 논문에서는 이러한 문제점을 최소화하기 위해 필터링을 이용한 새로운 색인 기법을 제안한다. 제안하는 셀 기반 필터링 기법은 셀 중심에서 객체까지의 거리값을 저장하여 필터링 효과를 증대시킨다. 또한, 고차원 공간을 셀 단위로 분할하며, 각각의 셀을 시그니처로 표현한다. 검색을 수행하기 위해, 셀 기반 필터링 기법은 데이타 파일을 직접 접근하기 전에 전체 시그니처들을 탐색하여 필터링을 수행함으로써 후보 셀들을 얻는다. 성능 실험을 통해 제안하는 기법이 VA-파일보다 검색 시간에 있어서 약 20%의 성능 향상을 보인다.

Abstract There have been many researches on indexing schemes for high-dimensional vector data such as image feature vectors. However, the conventional indexing schemes work well for low-dimensional data, but perform poorly as dimension increases, called 'dimensional curse' problem. Therefore, we propose a new indexing scheme using filtering technique in order to alleviate the dimensional curse problem. Our Cell-Based Filtering(CBF) scheme stores the distance of an object from the center of a cell in order to make a better effect on filtering. In our CBF scheme, high-dimensional space is divided into cells, each of which is represented as a signature. For retrieval, our CBF scheme first scans all signatures and does filtering by choosing candidate cells before accessing the data file. We show that our CBF scheme outperforms VA-File about 20% in terms of retrieval time.

1. 서 론

최근 데이타베이스 응용에서 내용 기반 검색 및 유사성에 기반한 검색에 대한 관심이 부각되고 있다. 대용량의 멀티미디어 데이타베이스에서 사용자의 다양한 요구를 보다 효율적으로 지원하기 위해서는 효과적인 내용 기반 멀티미디어 검색 기법과 그에 따른 유사성 기반 검색이 요구된다. 실질적으로, 멀티미디어 데이타는 기존의 텍스트 정보뿐만 아니라 이미지, 비디오 등과 같은 다양한 형태의 자료가 복합적으로 구성되어 있다. 내용 기반 멀티미디어 정보 검색은 데이타베이스에 저장된 대량의 멀티미디어 객체들 중에서 내용을 기반으로 사용자가 원하는 객체를 찾는 검색이다. 이를 위해 멀티미디어 객체로부터 n(>1)개의 특징 벡터(feature vector)를 추출하여 데이타베이스에 저장한 다음, 사용자가 검색하려고 하는 멀티미디어 객체의 특징 벡터와 가장 유사한 값을 갖는 객체를 찾는 방법이다. 즉, 내용 기반 멀티미디어 정보 검색은 n-차원 특징 벡터 공간에서 객체의 특징 벡터간의 유사성에 기반한 검색으로 변환되어 처리되며, 이에 따라 멀티미디어 데이타로부터 추출

· 본 연구는 한국전자통신연구원의 내용 기반 멀티미디어 정보검색 기술 개발사업의 위탁과제로 수행되었음.

† 종신회원 : 전북대학교 컴퓨터공학과 교수
jwchang@dblab.chonbuk.ac.kr
†† 비 회 원 : 전북대학교 컴퓨터공학과
sahan@dblab.chonbuk.ac.kr
††† 비 회 원 : 한국전자통신연구원 연구원
jini@etri.re.kr

논문접수 : 2000년 2월 14일
심사완료 : 2001년 4월 19일

된 n-차원 특징(속성) 벡터를 위한 색인 기법에 대한 연구가 크게 증가하고 있다.

한편, 내용-기반 멀티미디어 정보 검색을 위한 유사성에 기반한 사용자의 질의 형태는 크게 3가지로 나누어진다. 첫째, 주어진 질의와 정확하게 일치하는 객체를 검색하는 포인트 질의(point query), 둘째, 주어진 질의와 임의의 유사성 범위 내에 포함되는 객체를 검색하는 범위 질의(range query), 마지막으로 주어진 질의 객체와 가장 유사한 객체 k개를 찾는 k-최근접 탐색 질의(k-nearest neighbor search query) 이다. 객체간의 유사성은 일반적으로 특징벡터 공간에서 유클리디언 거리를 이용한다. 이러한 질의 형태는 내용 기반 검색에서 매우 중요하며 고차원 색인 구조는 이를 효과적으로 지원하기 위한 형태를 취하고 있다. 하지만 멀티미디어 객체로부터 추출된 특징벡터의 차원이 증가함에 따라 기존의 색인 기법의 검색 성능은 급격히 저하된다. 차원 증가에 따른 이러한 문제를 'dimensional curse' 라 한다[1][2]. 이러한 문제를 해결하기 위한 많은 연구들이 진행되고 있으나 아직까지는 미흡한 실정이다. 따라서 본 연구에서는 차원이 증가함에 따라 발생하는 기존의 고차원 색인 기법의 비효율성을 극복하기 위해 필터링(filtering) 에 기반한 새로운 고차원 색인기법을 제안한다. 이를 위해 새로운 거리에 근거한 셀-기반 필터링 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 고차원 색인기법에 대한 기존 연구를 분석한다. 3장에서는 본 연구에서 제안하는 셀-기반 필터링 기법에 대하여 기술한다. 4장에서는 제안한 고차원 색인 기법의 성능 평가를 기술한다. 마지막으로 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1 기존 고차원 색인 기법

대용량의 멀티미디어 데이타베이스를 다루는데 있어서 주된 문제는 검색의 효율성으로, 이를 위한 적절한 색인 기법을 설계해야 한다[3]. 그 동안 다수의 다차원 색인 기법들에 대한 연구가 진행되어 왔으나, R*-트리를 비롯한 색인 기법들은 차원의 수가 증가함에 따라 시간 또는 공간 요구량이 지수적으로 증가되어 고차원의 경우 순차 탐색보다 성능이 뒤지는 결과를 초래하여 색인 구조로서 효력을 상실하게 된다[4][5]. 최근 이러한 문제 해결을 위해 고차원 특징 벡터를 수용할 수 있는 다수의 색인 기법들이 연구되었다[6][7][8][9][10][11]. 본 장에서는 기존의 대표적인 고차원 색인 기법인 X-트리와 VA-파일에 대해 논한다.

X-트리는 기존의 색인 구조들이 차원에 증가함에 따라 겹침 영역이 증가하여 검색 성능이 현저히 저하되는 문제점을 방지하기 위해 제안된 고차원 색인 구조이다[12]. X-트리는 디렉토리에서의 겹침 영역을 피하기 위한 분할 알고리즘과 수퍼 노드 개념을 이용한다. X-트리 구조는 분할 시 겹침 영역이 최소가 되지 못할 때는 분할하지 않고 노드의 크기가 가변적으로 확장될 수 있는 수퍼노드를 사용한다. X-트리 구조는 저차원에서는 계층적인 디렉토리 구조를 사용하고, 고차원으로 갈수록 겹침 영역이 증가되기 때문에 기억공간이 절약되고 빠른 접근이 가능한 선형적인 디렉토리 구조를 사용한다. [그림 1]은 X-트리의 전체적인 디렉토리 구조를 보여준다. 노드는 크게 MBR(Minimum Bounding Rectangle) 정보를 갖는 중간 노드와 실제 특징 벡터 데이타를 갖는 데이타 노드로 구분된다. X-트리는 고차원 특징벡터 공간에서 기존의 색인 구조에 비해 k-최근접 탐색 질의와 같이 유사성에 기반한 검색을 효율적으로 지원하는 적절한 색인 구조로 평가된다.

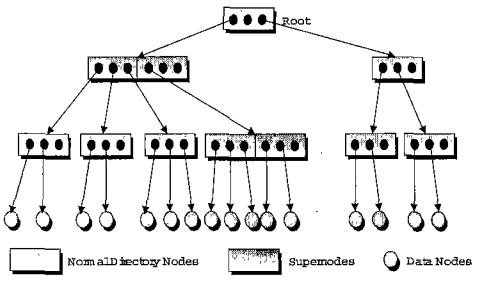
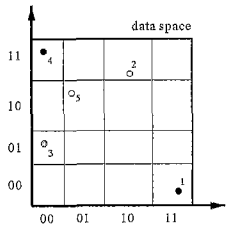


그림 1 X-Tree의 구조

기존의 트리-기반 고차원 색인 기법은 차원이 증가함에 따라 성능이 급격히 저하하여 순차 탐색보다 성능이 뒤지는 결과를 초래한다. 예를 들면, X-트리의 경우 16 차원 이상의 특징벡터에 대해서 순차 탐색보다 못한 실험 결과를 보였다[13]. VA-파일은 이러한 점을 감안하여 차원이 증가함에 따라 성능이 급격히 저하하는 문제를 개선하기 위해 순차 탐색을 개선할 수 있는 방법을 제안하였다[14]. VA-파일은 벡터의 요약값(approximation)을 사용함으로써 순차탐색의 탐색 비용을 줄이고자 하였다. 이 방법은 데이타의 공간을 여러 구역(region)으로 나누고, 각 영역에 대한 요약값을 생성한 후, 모든 요약값들을 순차 파일에 저장한다. [그림 2]는 이를 나타내고 있다. 탐색 과정은 먼저 요약값에



		vector data	VA-File
1	●	0.9 0.1	11 00
2	○	0.6 0.8	10 11
3	○	0.1 0.4	00 01
4	●	0.1 0.9	00 11
5	○	0.3 0.7	01 10

그림 2 벡터의 요약값 표현 예

기반한 검색으로 순차 파일을 순차 탐색하여 1차 후보를 찾고, 후보들의 실제 벡터 값을 검색하게 된다.

2.2 유사성에 기반한 질의 처리

유사성에 기반한 질의 형태로는 범위 질의, 최근접 탐색 질의, k-최근접 탐색 질의 등이 있다. 데이타베이스 시스템에서 이러한 복잡한 유사성 질의를 효율적으로 지원하는 것은 매우 중요하다. 범위 질의는 주어진 질의 점을 중심으로 임의의 거리를 갖는 영역으로써 표현되며, 그 영역에 포함되어 있는 모든 객체를 검색하는 형태로 처리된다. 이를 위해 범위 질의 영역과 겹치게 되는(intersect) 모든 영역이 탐색 영역에 포함된다. k-최근접 탐색 질의는 이보다 좀더 복잡한 형태를 취한다. 기존의 고차원 색인기법에서 지원하는 k-최근접 탐색 질의는 Roussopoulos가 제안한 최근접 데이타 탐색 알고리즘[15]에 기반하고 있다. 이 알고리즘에서는 다차원 검색 공간에서 주어진 질의 점으로부터 객체 또는 부검색 공간을 나타내는 최소경계사각형까지의 최소거리(minimum distance; MINDIST)와 최대탐색거리(minmax distance: MINMAXDIST) 개념을 이용하여 트리에서 검색되는 노드의 수를 줄이고자 하였다. [그림 3]은 질의 점으로부터 각각의 최소경계사각형까지의 MINDIST와 MINMAXDIST를 나타내고 있다.

최근접 데이타 탐색 알고리즘에서는 트리를 검색하는 동안 방문할 필요가 없는 노드들을 방문 대상에서 제외하기 위해 MINDIST와 MINMAXDIST를 조합하여 다음과 같은 세 단계의 노드제거 단계를 거쳐 수행된다.

STEP 1. 질의 점 P로부터 최소경계사각형 R까지의 MINMAXDIST(P, R)보다 더 큰 MINDIST(P, R) 값을 갖는 최소경계사각형 R이 존재하면, R은 k-최근접 객체를 포함하지 않기 때문에 R에 해당하는 노드는 검색 대상에서 제외한다.

STEP 2. 질의 점 P로부터 객체 O까지의 거리가 최소경계사각형 R에 대한 MINMAXDIST(P, R)보다 크다면, 객체 O를 k-최근접 객체 대상에서 제외한다.

STEP 3. 질의 점 P로부터 객체 O까지의 거리보다 더 큰 MINDIST(P, R)를 갖는 모든 최소경계사각형 R은 검색 대상에서 제외한다.

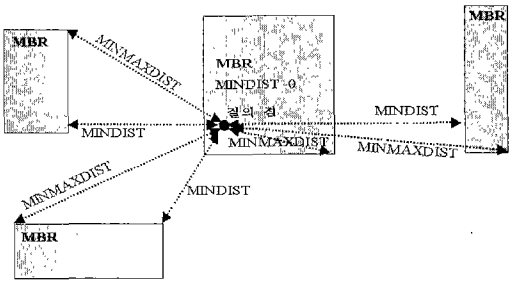


그림 3 MINDIST, MINMAXDIST의 개념

[그림 4]는 2차원 검색 공간에서 노드(MBR)를 제거하는 예이다. k=1일 경우, 즉 질의 점 P로부터 가장 가까운 객체를 찾는 경우, 질의 점으로부터 각각의 최소경계사각형 R1, R2, R3까지의 MINDIST를 구한다. 또한, 검색 대상에서 제외할 노드를 찾기 위해 질의 점으로부터 첫번째 탐색 노드인 R1까지의 MINMAXDIST를 구하여, 이 값보다 큰 MINDIST를 가지는 최소경계사각형을 검색 대상에서 제외하게 된다. 즉, 그림에서 질의 점으로부터 R1 사이의 MINMAXDIST보다 큰 MINDIST를 가지는 R3는 검색 대상에서 제외된다. 왜냐하면, 최소경계사각형 R3에 포함되어 있는 객체들 중에서 질의 점으로부터 가장 가까운 객체보다 더 가까운 객체가 R1 또는 R2에 존재하기 때문이다.

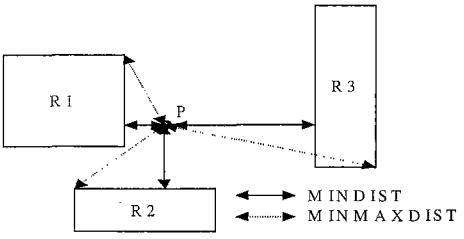


그림 4 노드(MBR) 제거의 예

3. 셀 기반 필터링 기법

3.1 전체 구조

대부분의 고차원 색인 기법은 dimensional curse 즉,

차원이 높아짐에 따라 성능이 저하되는 문제가 발생하는데, 실제로 X-트리는 16차원 이상이 되면 데이타베이스에 저장된 모든 객체들의 특징 벡터들을 순차 탐색하는 방법보다도 성능이 떨어지게 된다[13]. 따라서, 고차원(20차원 이상)에서의 검색 성능을 개선하기 위하여 시그니쳐에 의한 필터링을 수행하는 셀 기반 필터링(Cell-Based Filtering; CBF) 기법을 제안한다. 셀 기반 필터링 기법은 객체들의 특징 벡터들에 대한 순차 탐색을 수행하기 전에, 각각의 특징 벡터에 대한 시그니쳐를 탐색하여 필터링함으로써 탐색 성능을 향상시킨다. 또한, 특징 벡터에 대한 시그니쳐뿐만 아니라 셀 중심에서 객체까지의 거리 정보를 이용함으로써, 필터링 효과를 증대시킨다. 고차원 공간에서는 차원이 크면 클수록 객체에 대한 특징 벡터의 전체 데이타 크기 또한 늘어나게 된다. 따라서, 하나의 버퍼 안에 로딩할 수 있는 객체의 특징 백터 수가 줄어들게 되며, 순차 탐색을 할 경우 그 만큼 디스크 I/O 수가 많이 필요하게 된다. 그러나, 시그니쳐를 사용하게 되면, 실제 특징 벡터에 비해 시그니쳐의 크기가 상대적으로 매우 작기 때문에 버퍼에 로딩할 수 있는 객체의 시그니쳐 수가 많게 되고, 디스크 I/O 수를 줄여 검색 성능을 향상시킬 수 있다.

고차원 데이타에 대한 시그니쳐는 데이타 각각의 차원에 따른 특징 벡터의 요약들의 집합이므로, 각 차원에 해당하는 벡터의 성질을 유지하도록 표현한다. 또한, 질의를 처리하기 위해서, 질의 벡터를 시그니쳐로 표현한다. 따라서, 특징 벡터를 시그니쳐로 변환하는 방법은 VA-파일[14]의 연구와 마찬가지로 데이타 공간을 셀(cell) 단위로 나누고, 셀에 대한 시그니쳐를 사용한다. 이렇게 만들어진 시그니쳐는 각각의 셀을 대표하는 값이 되며, 셀에 대한 정보를 포함한다. 또한, 차원에 따라 데이타 공간을 균등하게 분할함으로써, 셀에 대한 시그니쳐는 데이타 공간에서의 어떤 영역의 범위를 가지게 되고, 시그니쳐를 통해 쉽게 이 범위값을 구할 수 있다.

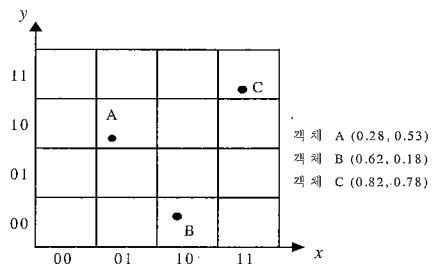


그림 5 2차원 공간에서의 시그니쳐 생성 과정

[그림 5]는 각각의 차원에 대해서 2-비트 시그니쳐를 사용할 경우, 2차원 공간에서의 시그니쳐를 만드는 과정을 나타낸다. 그림에서 각 차원에 따른 특징 벡터의 시그니쳐 크기를 2비트로 할당하므로 각 차원에 대한 데이타 공간은 4개의 부분으로 나뉘어져, 2차원 공간에서 총 16개의 셀을 가지게 된다. 차원에 따른 데이타 공간에서 표현할 수 있는 객체들의 실제값이 0에서 1사이인 경우, 각각의 셀을 나누는 점의 좌표는 0, 0.25, 0.50, 0.75, 1 이 된다.

따라서, 객체의 벡터가 0에서 0.25 사이에 있는 값은 시그니쳐 '00'으로 표현되고, 0.25~0.50 사이의 값은 '01'로, 0.50~0.75 사이의 값은 '10', 0.75~1 사이의 값은 '11'로 각각 표현된다. 따라서, 객체 A의 실제값은 (0.28, 0.53)이고, 1차원(x축)의 특징 벡터가 0.28이므로, 0.25~0.5 사이의 범위를 나타낼 수 있는 시그니쳐 '01'로 표현할 수 있다. 또한 2차원(y축)의 특징 벡터는 0.53이므로 시그니쳐 '10'로 표현하여, 최종적으로 두 개의 시그니쳐를 합성하여 만든 시그니쳐 '0110' 이라는 값으로 표현된다. 같은 방법으로 객체 B는 '1000', 객체 C는 '1111'로 표현된다. 이렇게 생성된 시그니쳐는 각 차원의 순서를 유지할 뿐만 아니라, 차원에 따른 특징 벡터의 실제 범위를 나타낼 수 있으므로, 질의 처리시 상대적으로 크기가 작은 시그니쳐를 접근함으로 인해 검색 성능을 향상시킬 수 있다.

셀 기반 필터링 기법은 두 개의 파일로 구성된다. 즉, 객체의 특징 벡터들을 저장하고 있는 데이타 파일과 벡터에 대한 시그니쳐 및 객체와 셀 중심간의 거리 정보를 저장하는 시그니쳐 파일이다. 새로운 객체의 특징 벡터를 저장하는 경우, 우선 객체와 셀 중심사이의 거리를 구하여, 객체의 특징 벡터와 함께 시그니쳐로 변환하고, 변환된 시그니쳐를 시그니쳐 파일에 추가하게 된다. 그런 다음, 실제 특징 벡터를 데이타 파일에 추가한다. 이때, 저장된 시그니쳐와 특징 벡터는 각각의 파일에서 똑같은 저장 순서를 갖도록 함으로써, 각각의 시그니쳐 정보는 데이타 파일에서의 시그니쳐에 대한 특징 벡터가 저장된 위치 정보를 포함할 필요가 없도록 한다. 이렇게 생성된 시그니쳐에 대해 질의는 다음과 같이 처리된다. 질의가 주어지면, 질의 벡터에 대한 시그니쳐 변환을 수행하여 질의 시그니쳐를 생성한다. 생성된 질의 시그니쳐를 사용하여 시그니쳐 파일에 저장된 각각의 시그니쳐들을 순차적으로 탐색하여 필터링을 수행한다. 시그니쳐를 통해 각 셀의 범위값을 구하고, 이 값을 이용하여 질의 벡터의 검색 범위 안에 있는 시그니쳐들만을 다음의 검색 대상이 되는 후보 시그니쳐로 선택한다. 또한,

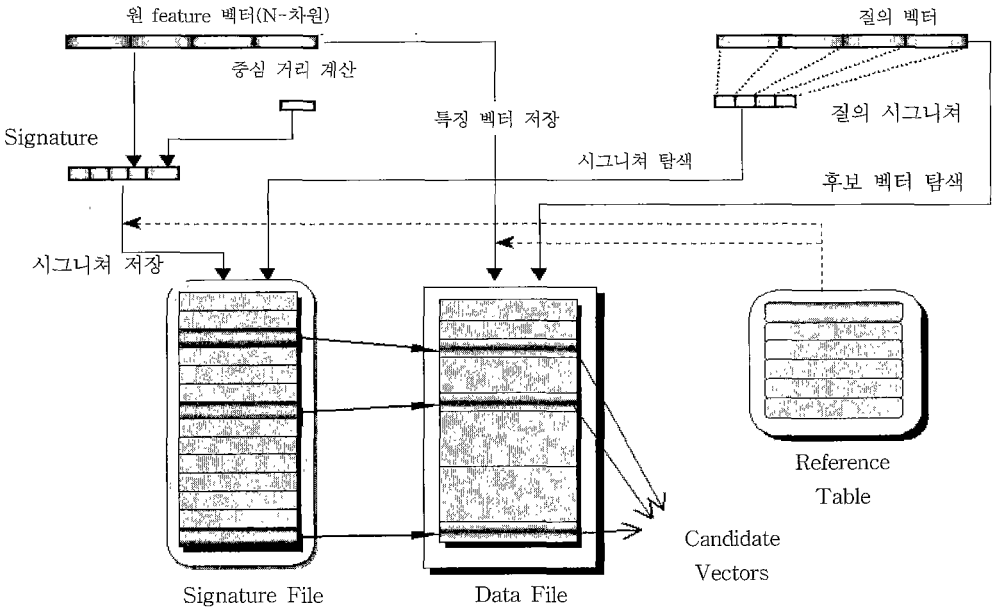


그림 6 셀 기반 필터링 기법을 사용한 전체 구조

각 시그니처 정보와 함께 저장된 거리 정보를 사용함으로써 보다 효율적인 필터링을 수행한다. 시그니처 탐색을 통해 선택된 각 후보 시그니처의 실제 특징 벡터들을 데이타 파일에서 탐색함으로써 최종적으로 주어진 질의를 만족하는 특징 벡터를 검색한다. 시그니처와 거리 정보에 의한 필터링 효과가 클수록 검색 성능은 우수하게 된다. [그림 6]은 셀 기반 필터링 기법을 사용한 전체 구조이다.

3.2 새로운 최소, 최대거리 정의

시그니처를 사용하여 각각의 객체들이 셀 단위로 구성되는 셀-기반 필터링 기법에서는 각각의 셀 안에 저장되어 있는 객체들을 효율적으로 필터링하기 위하여 질의 점과 객체 사이의 최소 거리(MINDIST) 및 최대 거리(MAXDIST)를 새로이 정의한다. 일반적으로 최소 및 최대 거리는 질의 처리시 후보 대상 객체들을 찾는데 사용한다. 즉, 최소, 최대 거리는 질의 점과 임의의 객체 사이에서 객체가 존재할 가능성이 있는 거리의 범위로써, 실제 객체와의 거리를 구하지 않고도 객체의 위치를 짐작할 수 있는 수단을 제공한다. 따라서, 실제 객체와의 거리와 가장 근접한 최소, 최대 거리를 구함으로써 필터링을 효율적으로 수행할 수 있다. 객체(O)가 임의의 셀에 저장되어 있는 경우, VA-파일에서는 질의 점(Q)와 객체가 저장된 셀 사이의 가장 가까운 거리를 최소 거리로, 가장 먼 거리를 최대 거리로 사용한다. 즉, 셀 안에 저장된 객체는 셀의 임의의 위치에 있다고 가정하며, 질의 처리시 질의 점과 셀 사이의 거리 범위를 사용하여 후보 객체들을 찾는다. [그림 7]은 VA-파일에서 사용하는 최소, 최대 거리이다.

본 논문에서 제안하는 셀-기반 필터링 기법에서는 보다 효율적으로 후보 객체들을 필터링하기 위하여 다

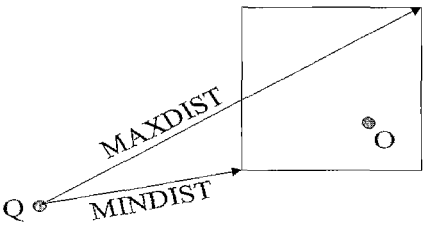


그림 7 VA-파일의 최소, 최대 거리

음과 같이 정의된 최소, 최대 거리를 사용한다. 최소, 최대 거리를 구하기 위하여 객체와 객체가 저장된 셀의 중심 사이의 거리(r)를 이용한다. 질의 점(Q)과 셀 중심 사이의 거리($\overline{QC}$)에서 객체와 셀 중심 사이의 거리(r)를 뺀 값을 최소 거리로, 질의 점(Q)과 셀 중심 사이의 거

리($\overline{QC}$)에서 객체와 셀 중심 사이의 거리(r)를 더한 값을 최대 거리로 정의한다.

- $MINDIST_{CBF} = CENTERDIST - RADIUS$
  $= \overline{QC} - r$
  $= \overline{Qm}$
- $MAXDIST_{CBF} = CENTERDIST + RADIUS$
  $= \overline{QC} + r$
  $= \overline{QM}$

$(\therefore MINDIST_{CBF} > MINDIST_{VA-FILE},$
$MAXDIST_{CBF} < MAXDIST_{VA-FILE})$

새롭게 정의한 최소 거리($MINDIST_{CBF}$)는 VA-파일의 최소 거리($MINDIST_{VA-FILE}$)보다 큰 값을, 그리고 최대 거리($MAXDIST_{CBF}$)는 VA-파일의 최대 거리($MAXDIST_{VA-FILE}$)보다 작은 값을 가지게 된다. 따라서, 임의의 객체(O)가 존재할 범위 영역을 축소시킴으로써 k-최근접 질의나, 범위 질의 처리시 좀더 효율적으로 후보 셀들을 필터링할 수 있다. [그림 8]은 CBF에서 새로이 정의된 최소, 최대 거리를 나타낸다.

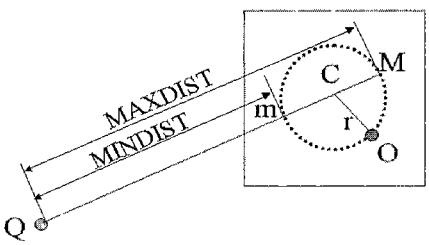


그림 8 CBF의 최소, 최대 거리

VA-파일이나 셀 기반 필터링 기법의 탐색 알고리즘에서는 저장된 시그니쳐들을 순차 탐색하게 되는데, 탐색 과정에서 얻어진 k-번째 최대 거리값이 후보 셀 선택의 기준값이 된다. 즉, k-번째 최대 거리값보다 작은 최소 거리값을 가진 셀들은 모두 후보 셀로 선택되게 된다. 따라서, 현재 k-번째 최대 거리가 하나의 셀 A의 최대 거리값이고, 다음으로 셀 B를 탐색하여 후보 셀인지를 결정한다고 가정하자. 이 때 k-번째 최대 거리와 셀 B의 최소 거리를 비교하는데, 최소 거리가 최대 거리보다 작으면 후보 셀로 선택하고 최소 거리가 최대 거리보다 크면 후보 셀에서 제외된다. [그림 9]는 k-최근접 질의시 VA-파일 및 셀 기반 필터링 기법에서의 필터링 영역을 나타낸다. VA-파일에서는 현재 k-번째 최대 거리값(a)보다도 셀 B의 최소 거리값(c)이 작으므로, 셀 B는 후보 셀로 선택되게 된다. 그러나, 셀 기반 필터링 기법에서는 셀 B의 최소 거리값(d)이 k-번째 최대 거리값(b)보다 크기 때문에 셀 B는 후보 셀에서 제외된다. 이와 같이, 새롭게 정의된 최소, 최대 거리를 사용함으로써 셀 기반 필터링 기법이 VA-파일보다 효율적인 필터링을 수행하게 된다. 범위 질의는 k-최근접 질의보다 단순하기 때문에 본 논문에서는 범위 질의에 대한 설명은 생략하고, k-최근접 질의를 위한 탐색 알고리즘을 자세히 다룬다.

3.3 k-최근접 탐색 알고리즘

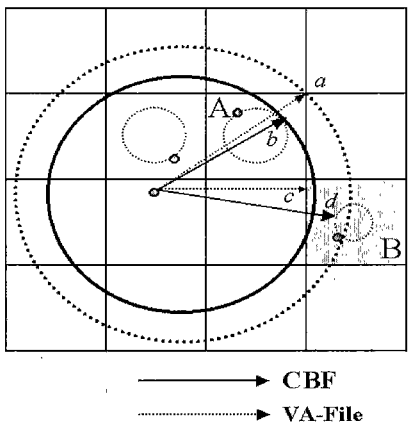


그림 9 VA-파일, CBF 기법의 필터링 영역

셀 기반 필터링 기법에서는 효율적인 k-최근접 탐색을 수행하기 위해 Nick Roussopoulos가 제안한 탐색 알고리즘[15]을 기반으로하여, VA-파일의 최적화된 탐색 알고리즘[14]인 VA-NOA를 개선한 탐색 알고리즘을 사용한다. 또한, 탐색 알고리즘에서 효율적인 필터링을 수행하기 위해 앞 절에서 새로이 정의한 최소, 최대 거리를 사용한다. 셀 기반 필터링 기법의 k-최근접 탐색 알고리즘은 세 단계의 탐색 과정을 거쳐 질의 처리를 수행한다. 첫 번째 단계에서는 시그니쳐 파일을 접근하여 저장된 전체 시그니쳐들을 순차적으로 탐색하여 필터링을 수행하는 단계로, 객체와 셀 중심간의 거리값을 이용한 최소, 최대 거리를 사용하여 후보 셀을 얻는다. 이렇게 얻어진 후보 셀들에 대해 최종적으로 k-번째 최대 거리값으로 필터링하는 과정이 두 번째 단계이다. 마지막으로, 세 번째 단계에서는 현재까지 얻어진 시그니쳐 정보를 가지고 실제 객체의 특징 벡터가 저장되어 있는 데이타 파일을 접근함으로써, 최종적으로 질

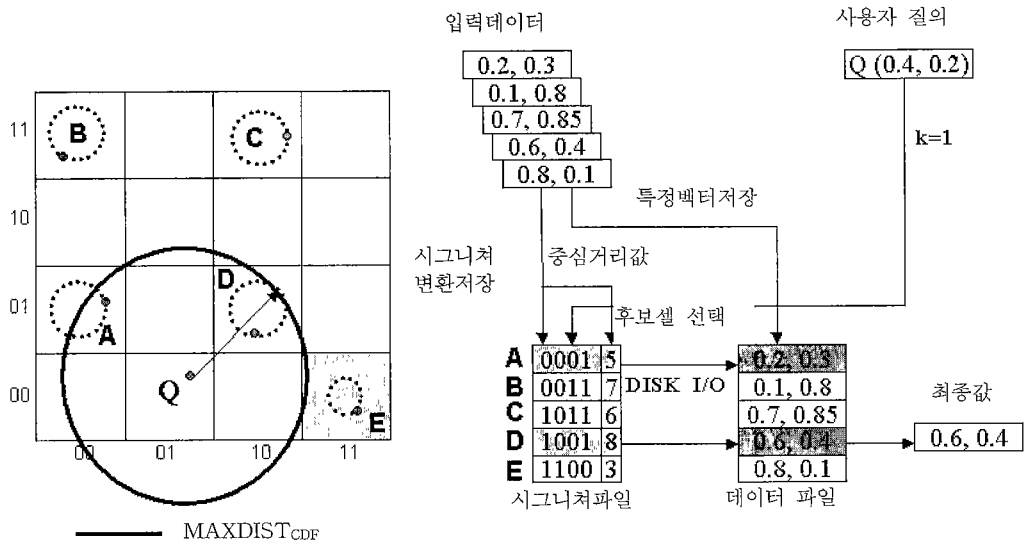


그림 10 k-최근접 질의 처리 예

의에 대한 결과를 얻는다. [그림 10]은 [그림 6]에서 제시한 전체 CBF 구조에 실제값들을 저장하여 k-최근접 질의 검색을 수행하는 예를 나타낸다.

위의 예는 2차원 공간 상에서 2비트 시그니처를 사용할 경우를 나타낸다. 특징 벡터 (0.2, 0.3)을 갖는 객체를 삽입하는 경우, 먼저 특징 벡터에 대한 시그니처 변환을 수행하여 객체가 저장될 셀 A의 시그니처(0001)를 얻는다. 또한, 객체와 셀 A의 중심간의 거리값을 구한 후, 이 값 또한 1바이트의 시그니처(5_{10}=00000101_2)로 변환한다. 이렇게 얻어진 시그니처(0001)와 거리값(5)을 시그니처 파일에 저장하고, 객체의 특징 벡터(0.2, 0.3)는 데이타 파일에 저장한다. 나머지 객체들도 이와 같은 방식으로 시그니처 파일 및 데이타 파일에 삽입한다. 위와 같이 저장된 객체들에 대하여 사용자 질의(Q)가 (0.4, 0.2)로 주어졌을 경우, k-최근접 질의(k=1)는 다음과 같이 수행된다. 먼저 시그니처 파일에 저장된 모든 시그니처들(A,B,C,D,E)을 순차적으로 탐색하여 후보 셀들을 얻는다. 즉, 각각의 시그니처들을 순차적으로 탐색하면서 현재까지 얻어진 k-번째 MAXDIST와 현재 탐색 중인 셀의 MINDIST를 비교하여 필터링을 수행한다. 위의 그림에서 시그니처들을 탐색하면서 얻어진 k-번째 MAXDIST 값은 질의 점(Q)과 셀 D 사이의 거리($MAXDIST_{CBF}$)이다. 따라서, 이 값보다 큰 MINDIST를 갖는 셀들(B,C,E)은 후보 셀로부터 제외되며, 이 값보다 작은 MINDIST를 갖는 셀들(A,D)은 후보 셀로 선택된다. 이렇게 선택된 후보 셀들에 대해서, 데이타 파일을 접근하여 객체의 특징 벡터와 질의 점(Q) 사이의 거리를 비교하여 가장 가까운 거리를 갖는 객체(0.6, 0.4)를 최종적으로 검색한다.

```
int CBF_KNN(q_vector, k_value, nlist_result)
float       *q_vector,          // 질의 벡터
int               k_value,              // k 값
cNSDLL      *nlist_result          // 결과값 저장 리스트
{
    // 초기화
    k_maxdist = MAXREAL;
    // PHASE-1 : 시그니처 탐색
    for(sig_node=sigbuf_list->getHead();
                        sig_node=sig_node->next) {
      r_dist = sig_node->distance;  // 거리값(r)
      // PHASE-1 Pruning
      PHASE1(q_vector, sig_node->data, r_dist, nlist_
          maxdist,list_mindist, sig_node->index);
    }
    k_maxdist = nlist_maxdist->getLastValue()->mbrdist;
    // PHASE-2 Pruning
    PHASE2(k_maxdist, list_mindist);
    // PHASE-3 Pruning
    PHASE3(q_vector, list_mindist, nlist_result);
    return (nlist_result->num);
}
```

그림 11 CBF 기법의 k-최근접 탐색 알고리즘

질의가 주어지면 PHASE1(), PHASE2(), PHASE3() 세 단계의 필터링 단계를 거쳐 원하는 결과를 얻는다. PHASE1()을 수행하기 위해 시그니쳐 파일에 저장된 전체 시그니쳐들을 탐색한다. 실제 검색 효율을 높이기 위해 탐색을 수행하기 전에 시그니쳐 파일로부터 시그니쳐 정보들을 메모리에 로딩하여 시그니쳐 리스트(sigbuf_list)를 유지한다. 따라서, 매 질의마다 시그니쳐 파일을 접근함으로 인한 I/O를 줄일 수 있다. 메모리 상의 시그니쳐 정보에는 특징 벡터에 대한 시그니쳐 뿐만 아니라 셀 중심에서 객체까지의 거리값(r_dist)도 포함되어 있으므로 이를 사용하여 쉽게 질의 점으로부터의 최소 거리를 구할 수 있다. [그림 11]은 셀 기반 필터링 기법의 k-최근접 탐색 알고리즘을 나타낸다.

```
void PHASE1(q_vector, sig_vector, r_dist, nlist_maxdist,
                          list_mindist, int rec_index)
float     *q_vector,          // 질의 벡터
char      *sig_vector,     // 시그니쳐
float      r_dist,            // 중심 거리값
cNSDLL    *nlist_maxdist,  // MAXDIST 리스트
cUSDLL    *list_mindist,      // 후보 셀 저장 리스트
int       rec_index       // 레코드 인덱스
{
    k_maxdist = nlist_maxdist->getLastValue()
                                    ->mbrdist;
    c_mindist = getMindist(q_vector, sig_vector, r_dist);
    // 필터링 수행
    if (c_mindist <= k_maxdist)    {
       list_mindist->insert(mindist_data->mbrdist,
                                    mindist_data);
       // k-번째 MAXDIST 갱신
       c_maxdist = getMaxdist(q_vector,
                              sig_vector, r_dist);
       if (c_maxdist < k_maxdist) {
            nlist_maxdist->insert(maxdist_
                    data->mbrdist,  maxdist_data);
            k_maxdist = nlist_maxdist->
                        getLastValue()->mbrdist;

       }
    }
}
```

그림 12 첫번째 단계의 필터링 : PHASE1

[그림 12]는 첫 번째 단계의 필터링을 수행하는 PHASE1() 알고리즘을 나타낸다. 여기서는 입력된 셀(sig_vector)이 다음 단계의 후보 셀인지를 검사하기 위해서 셀 중심에서의 거리값(r_dist)을 이용하여 질의 점과 셀 사이의 최소 거리값(c_mindist)을 구한다. 그런 다음, 가장 최근까지 얻은 k-번째 최대 거리값과 현재 구한 최소 거리값을 비교하여 최소 거리값이 작으면 후보 셀로 선택하고, 그렇지 않으면 제거하게 된다. 이렇게 해서 얻어진 후보 셀 리스트(list_mindist)는 다음 단계로 넘겨 진다.

```
void PHASE2(k_maxdist, list_mindist)
float              k_maxdist,     // k-번째 upper bound
cUSDLL *list_mindist            // 후보 셀 리스트
{
       for (pos = list_mindist->getHead(); pos !=
                                   NULL; pos = next_pos)
{
            next_pos = pos->next;
            // k-번째 upper bound보다 큰
                     lower bound를 갖는 셀 제거
            if (k_maxdist < pos->value->mindist)
                     list_mindist->deleteNode(pos);

       }
}
```

그림 13 두번째 단계의 필터링 : PHASE2

[그림 13]은 두 번째 단계의 필터링을 수행하는 PHASE2() 알고리즘이다. 첫 번째 단계에서 얻어진 후보 셀 리스트 중에서 최종적으로 얻어진 k-번째 최대거리값(k_maxdist)을 사용하여 이 값보다 큰 최소 거리값을 갖는 셀들을 후보 셀들로부터 제거한다. [그림 14]는 마지막 단계의 필터링을 수행하는 PHASE3() 알고리즘을 나타낸다. 필터링을 수행하기 전에 먼저 이전 단계에서 얻어진 후보 셀들을 최소 거리의 오름차순으로 정렬한다. 그 이유는 최소 거리값이 작은 셀을 먼저 탐색하기 위함이다. 즉, 질의 점과 가장 가까운 셀들을 먼저 탐색함으로써 검색 성능을 높일 수 있기 때문이다. PHASE3()에서 필터링을 수행하기 위해 후보 셀의 최소 거리값(sorted_mindist[idx].mindist)과 현재까지 얻어진 k-번째 객체 거리값(k_dist)를 비교하여 k_dist보다 크면 결과 리스트로부터 제거하고, 작으면 데이타 파일을 접근하여 질의 점과 실제 객체와의 거리(obj_dist)를 구하여 비교한 후, k 개의 가장 가까운 객체를 검색한다.

```
void PHASE3(q_vector, list_mindist,nlist_result)
float       *q_vector,                  // 질의 벡터
cUSDLL          *list_mindist,              // 후보 셀 리스트
cNSDLL          *nlist_result               // 결과값 저장 리스트
{
    // 후보 셀들을 lower bound값의 오름차순으로 정렬
    qsort(sorted_mindist, list_mindist->getTotalNum(), sizeof(sDATA_MBR), QS_Compare);

      for (idx = 0; idx < list_mindist->getTotalNum(); idx++)
      {
            if (k_dist != MAXREAL)
            {
                // k-번째 객체 거리보다 큰 셀들을 제거
                if (k_dist < sorted_mindist[idx].mindist)
                    break;
            }

            // k-번째 객체 거리보다 셀 들에 대해서 데이터 파일 접근
            index = sorted_mindist[idx].index;
            fseek(vec_fptr, index*vec_rec_size+sizeof(char), SEEK_SET);
            fread((char*)object_vector, sizeof(float)*header->dimension, 1, vec_fptr);
            fread((char*)id, header->id_length, 1, vec_fptr);
            obj_dist = getObjectdist(q_vector, object_vector);

            k_dist = nlist_result->getLastValue()->distance;
            // k-번째 객체 거리보다 작은 객체 거리를 갖는 객체 선택
            if (obj_dist < k_dist)     {
                nlist_result->insert(obj_dist, result_data);
            }
      }
}
```

그림 14 세번째 단계의 필터링 : PHASE3

4. 성능 평가

본 장에서는 필터링 기법에 기반한 새로운 고차원 색인 기법인 셀 기반 필터링 기법(CBF)을 구현하여 성능 평가를 수행한다. 구현된 시스템 환경은 [표 1]과 같다.

표 1 구현 시스템 환경

OS	Windows NT 4.0
CPU	450 MHz
Main Memory	256 MB
Compiler	Visual C++ 6.0

실험 데이타는 10차원, 20차원, 50차원, 80차원의 synthetic 데이타 100,000 건과 TV-트리에서 제시된 Hadamard 알고리즘[16]을 사용하여 논문 인스턴스로부터 추출한 리얼 데이타 100,000 건을 사용하였으며, 성능 비교 대상은 특징 벡터를 순차 탐색하는 방법(SeqScan), X-트리, VA-파일에서 제안한 VA-NOA 방법, 셀 기반 필터링 기법(CBF)이다. 또한 성능 평가는 삽입 시간, k-최근접 질의 검색 시간, 범위 질의 검색 시간 및 부가 저장 공간에 대해서 수행한다.

4.1 삽입 시간

삽입 시간은 각 차원에 따른 특징 벡터들로 구성된 데이타 100,000 건을 삽입하는 경우 소요되는 전체 시간이다. [그림 15]는 synthetic 데이타를 삽입하는 경우 소요되는 시간을 나타내며, 가로축은 차원, 세로축은 삽입 시간을 의미한다. 그림에서는 세로축이 두 개가 나타나는데, 오른쪽에 나타난 세로축은 X-트리에 해당하는 값이다. 다른 방법들에 비해 X-트리에 데이타를 삽입하

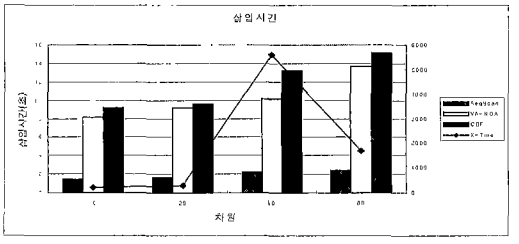


그림 15 삽입 시간

는 시간이 상대적으로 많이 걸리는데, 10차원 데이타일 경우 약 220초, 80차원 데이타일 경우 약 1700초 정도 소요된다. SeqScan 방법은 객체의 특징 벡터들만을 순차적으로 저장함으로써 약 2초 정도의 시간이 걸린다. VA-NOA는 특징 벡터뿐만 아니라 특징 벡터에 대한 시그니쳐를 저장하는 시간이 필요하여 약 14초 정도가 소요되며, CBF 방법은 셀 중심과 객체 사이의 거리를 계산하고 저장하는 시간이 더 요구되어 약 15초 정도 소요된다. 즉, CBF 방법이 약 7%의 삽입시간 오버헤드를 가진다. 이 시간은 상대적으로 미미한 값이라 할 수 있다. 리얼 데이타를 삽입하는 시간도 거의 비슷한 결과를 나타내므로 생략한다.

4.2 k-최근접 질의 검색 시간

k-최근접 질의는 주어진 질의 점으로부터 가장 거리가 가까운 k개의 객체를 찾는 질의이다. [그림 16]은 synthetic 데이타에 대한 k-최근접 질의 검색을 수행한 후 I/O 횟수를 나타낸다. 여기에 사용된 k 값은 100 이다. X-트리는 저차원에서는 좋은 성능을 나타내지만, 차원이 커질 경우 검색 성능이 급격히 저하됨을 알 수 있다. 20차원에서부터는 SeqScan 방법보다 성능이 떨어짐을 알 수 있다. 새로운 거리 개념을 사용하여 필터링을

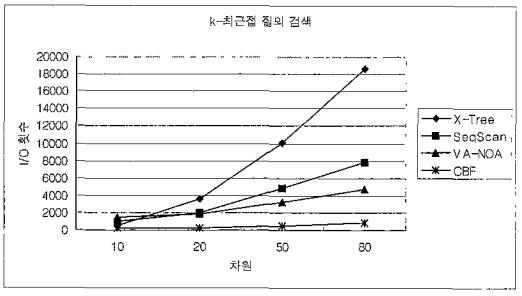


그림 16 k-최근접 질의 검색 : I/O 횟수 (synthetic 데이타)

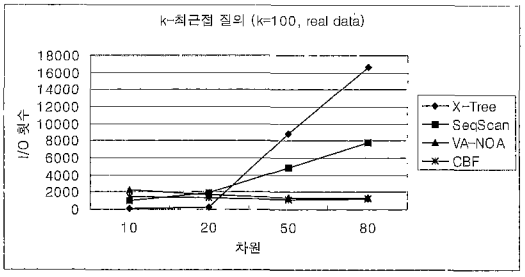


그림 17 k-최근접 질의 검색 : I/O 횟수 (리얼 데이타)

수행한 CBF 방법이 가장 좋은 검색 성능을 나타낸다.

[그림 17]은 리얼 데이타에 대한 검색 성능을 나타낸다. 10차원과 20차원에서는 X-트리가 가장 좋은 성능을 나타내며, [그림 16]과 마찬가지로 차원이 높을수록성능이 급격히 저하된다. CBF 방법은 차원에 거의 영향을 받지 않으며, 가장 좋은 성능을 나타낸다.

표 2 k-최근접 질의 검색에 대한 응답 시간 (단위:초)

액세스 메소드	Synthetic 데이타				리얼 데이타			
	10차원	20차원	50차원	80차원	10차원	20차원	50차원	80차원
Seq Scan	2.7043	3.3575	7.1968	14.0562	2.7407	3.5030	8.0374	17.3982
X-Tree	0.6618	4.7551	11.9133	19.4289	0.3125	0.4355	9.5938	14.2703
VA-NOA	0.2733	0.4454	1.0608	1.8140	0.3813	0.4969	0.9828	1.4390
CBF	0.1906	0.2845	0.6016	0.9421	0.2969	0.3516	0.6297	0.9219

[표 2]는 synthetic 데이타 및 리얼 데이타에 대한 k-최근접 질의 검색을 수행했을 경우 소요된 응답 시간 (real time)을 나타낸다. Synthetic 데이타를 사용한 경우, CBF 방법은 VA-파일보다 20차원에서는 36%, 80차원에서는 48%의 성능 향상을 보이며, 리얼 데이타를 사용한 경우 20차원에서는 29%, 80차원에서는 36%의 성능 향상을 보인다. [그림 18]은 리얼 데이타에 대한 k-최근접 질의 검색의 응답 시간 그래프를 나타낸다.

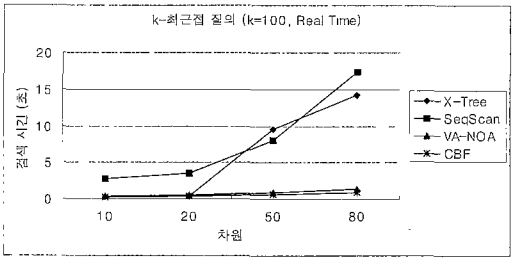


그림 18 k-최근접 질의 검색 : 응답 시간 (리얼 데이타)

4.3 범위 질의 검색 시간

범위 질의는 주어진 질의 점으로부터 일정한 거리 영역 안에 포함된 객체를 찾는 질의이다. [표 3]은 범위

표 3 범위 질의 검색에 대한 응답시간 (단위:초)

액세스 메소드	Synthetic 데이터				리얼 데이터			
	10차원	20차원	50차원	80차원	10차원	20차원	50차원	80차원
SeqScan	79.4625	92.1734	99.2973	115.7938	72.4200	89.2100	101.1200	160.9000
X-Tree	0.0314	4.7129	16.5840	15.7136	0.0047	0.0751	6.7390	15.6406
VA-NOA	0.1734	0.2969	0.6609	1.0827	0.2392	0.4031	0.7265	1.1219
CBF	0.1516	0.2469	0.5047	0.8016	0.2001	0.3484	0.5516	0.8500

질의 검색에 대한 응답 시간을 나타낸다. 범위값은 차원마다 다르게 지정하였으며, 전체 데이타 중에서 0.1%의 데이타를 검색할 수 있는 값을 범위값으로 사용하였다. 10차원에 대해서는 X-트리가 약 0.04초로 가장 성능이 좋지만, 차원이 증가할 수록 성능이 급격히 저하됨을 알 수 있다. Synthetic 데이타를 사용한 경우, CBF 방법은 VA-파일에 비해 20차원에서는 17%, 80차원에서는 26%의 성능이 향상되고, 리얼데이타를 사용한 경우는 20차원에서 14%, 80차원에서 24%의 성능이 향상됨을 알 수 있다. [그림 19]는 리얼 데이타에 대한 범위 질의 검색 시간을 나타낸다. 순차 탐색(SeqScan)이 다른 방법들에 비해 검색 시간이 너무 많이 소요되어 그래프에서 제외하였다. 저차원에서는 X-트리가 좋은 성능을 나타내며, 고차원으로 갈수록 CBF 방법이 가장 좋은 성능을 나타낸다.

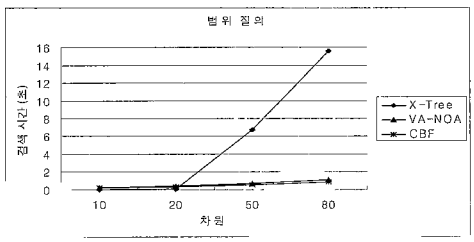


그림 19 범위 질의 검색 : 응답 시간 (리얼 데이타)

4.4 부가 저장 공간

인덱스 파일에 100,000 건의 synthetic 데이타 또는 리얼 데이타를 저장할 경우 부가 저장 공간 비율은 [표 4]와 같다. X-트리는 데이타에 따라 전체 노드의 구조가 영향을 받으므로 부가 저장 공간이 synthetic 데이타 및 리얼 데이타에 대해서 다르지만, 타 기법들은 같은 부가 저장 공간을 나타낸다. X-트리는 10차원 synthetic 데이타에 대해서 데이타 파일을 기준으로 93%, 리얼 데이타에 대해서 84%로써 가장 많은 부가 저장 공간을 차지한다. VA-파일은 특징 벡터뿐만 아니라 객체의 시그니쳐도 저장하므로 약 28%의 부가 저장 공간이 필요하며, CBF의 경우 셀의 중심으로부터 객체까지의 거리 정보까지 저장하므로 약 31%의 부가 저장 공간이 필요하다. 즉, CBF 방법은 10차원 데이타에 대해서 VA-파일보다 3%의 부가 저장 공간이 더 필요하고, 20차원 데이타에 대해서는 1%, 그리고 80차원 데이타에 대해서는 두 방법이 거의 같은 부가 저장 공간이 필요하다. 즉, 일반적으로 거리 정보가 차지하는 부가 저장 공간은 미미하며, 차원이 커지면 거리 정보에 대한 부가 저장 공간은 거의 없다고 할 수 있다.

표 4 부가 저장 공간 비율

액세스 메소드	데이타 종류	10차원	20차원	50차원	80차원
SeqScan	synthetic real	0 %	0 %	0 %	0 %
X-Tree	synthetic	93 %	86 %	112 %	142 %
	real	84 %	78 %	102 %	119 %
VA-File	synthetic real	28 %	20 %	16 %	15 %
CBF	syntehtic real	31 %	21 %	17 %	15 %

5. 결 론

유사성에 기반한 검색 질의는 데이타베이스에 저장된 대량의 멀티미디어 객체들 중에서 사용자가 원하는 객체와 유사한 객체를 찾는 검색이다. 객체간의 유사성은 일반적으로 특징벡터 공간에서 유클리디언 거리를 이용한다. 범위 질의, k-최근접 탐색 질의와 같은 유사성에 기반한 검색 질의는 내용 기반 검색에서 매우 중요하며,

기존의 고차원 색인 구조는 이를 효과적으로 제공하기 위한 형태를 취하고 있다. 하지만 멀티미디어 객체로부터 추출된 특징벡터의 차원이 증가함에 따라 기존의 색인 구조의 검색 성능은 급격히 저하된다.

본 연구에서는 차원이 증가함에 따라 발생하는 기존 트리 기반 색인 기법의 비효율성을 극복하기 위해, 필터링(filtering) 에 기반한 새로운 고차원 색인기법을 제안하였다. 제안하는 새로운 고차원 색인기법인 셀-기반 필터링 기법은 새로운 거리 개념을 사용하여 효율적인 필터링을 수행한다. 아울러 본 연구에서 제안한 새로운 고차원 색인기법을 Windows NT 환경에서 구현하고 타 기법과 성능 비교를 수행하였다. 고차원(10~80차원) 특징 데이터를 통한 성능비교 결과 삽입 시간은 X-트리가 10차원에서 200초 이상으로 가장 많이 소요되었으며, CBF 방법은 10초 정도가 소요되었다. synthetic 데이타 및 리얼 데이타에 대한 k-최근접 질의 검색 및 범위 질의 검색 결과 필터링에 기반한 CBF 방법이 약 0.5초 정도로 매우 우수한 성능을 나타내었다.

향후 연구로는 다중 사용자를 위한 동시성 제어 및 회복 기법을 적용하는 작업과 검색 성능을 보다 개선하기 위해 병렬 처리(*parallel processing*) 기법을 연구하는 것이다.

참 고 문 헌

[1] Arya S., Mount D.M., Narayan O., "Accounting for Boundary Effects in Nearest Neighbor Searching," Proc. 11th Annaual Symp. on Computational Geometry, Vancouver, Canada, pp. 336-344, 1995

[2] Berchtold S., Bohm C., Keim D., Kriegel H. -P, "A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space," ACM PODS Symposium on Principles of Databases Systems, Tucson, Arizona, 1997.

[3] Bentley J. L., "Multidimensional Search Trees Used for Associative Searching," Comm. of the ACM, Vol. 18, No. 9, pp. 509-517, 1975.

[4] Guttman A., "R-trees :A Dynamic Index Structure for Spatial Searching," Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA, pp. 45-57. 1984.

[5] N. Beckmann, H.P. Kriegel, R. Schneider and B. Seeger, "The R*-tree : An Efficient and Robust Access Method for Points and Rectangles," ACM SIGMOD, pp.322-331, May 1990.

[6] Robinson J. T., "The K-D-B-tree : A Search Structure for Large Multidimensional Dynamic Indexes," Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 10-18, 1981.

[7] Henrich, A., "The LSDh-tree : An Access Structure for Feature Vectors," Proc. 14th Int. Conf. on Data Engineering, Orlando, 1998

[8] D. A. White and R. Jain, "Similarity Indexing : Algorithms and Performance," InProc. Of the SPIE : Storage and Retrieval for Image and Video Databases IV, Vol. 2670, pp.62-75, 1996.

[9] D. A. White and R. Jain, "Similarity Indexing with the SS-tree," In Proc. 12th Intl. Conf. On Data Engineering, New Orleans, pp.516-523, 1996.

[10] Katayama N., Satoh S., "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 369-380, 1997.

[11] Berchtold S., Bohm C., Kriegel H.-P., "The Pyramid-Tree : Indexing Beyond the Curse of Dimensionality," Proc. ACM SIGMODE Int. Conf. on Management of Data, Seattle, 1998

[12] S. Berchtold, D. A. Keim, H-P. Kriegel, The X-tree : An Index Structure for High-Dimensional Data, Proceedings of the 22nd VLDB Conference, pp.28-39, 1996.

[13] Roger Weber, Hans-Jorg Schek, Stephen Blott: A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. VLDB 1998: 194-205

[14] Roger Weber, Stephen Blott, "An Approximattion -Based Data Structure for Similarity Search," Technical report Nr. 24, ESPRIT project HERMES (no. 9141), October 1997.

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

[16] H.I. Lin, H. Jagadish, and C. Faloutsos, "The TV-tree : An Index Structure for High Dimensional Data," VLDB Journal, Vol. 3, pp.517-542, 1995.

장 재 우

1984년 서울대학교 전자계산기공학과(공학사). 1986년 한국과학기술원 전산학과(공학석사). 1991년 한국과학기술원 전산학과(공학박사). 1996년 ~ 1997년 Univ. of Minnesota, Visiting Scholar. 1991년 ~ 현재 전북대학교 컴퓨터공학과 부교수. 관심분야는 멀티미디어 데이타베이스, 멀티미디어 정보검색, 하부저장구조

한 성 근

1998년 전북대학교 컴퓨터공학과 (공학사). 2000년 전북대학교 컴퓨터공학과 (공학석사). 2000년 ~ 현재 전북대학교 컴퓨터공학과 (박사과정). 관심분야는 멀티미디어 데이타베이스, XML, 정보 검색

김 현 진

1997년 부산대학교 전자계산학과 석사. 1997년 ~ 현재 한국전자통신연구원 지식정보검색팀 연구원. 관심분야는 멀티미디어 데이타베이스, 정보검색, 한글처리, HCI 등.