적응 분할과 벡터 근사에 기반한 고차원 이미지 색인 기법

(High-Dimensional Image Indexing based on Adaptive Partitioning and Vector Approximation)

차 광 호 † 정 진 완 ††
(Guang-Ho Cha) (Chin-Wan Chung)

요 약 이 논문은 고차원 이미지 데이타의 효율적인 색인을 위한 LCP*-file을 제시한다. 멀티미디어 데이타의 사용이 증가하면서 고차원 이미지 데이타의 색인과 검색의 지원에 대한 요구가 증가하고 있다. 최근에 고차원 데이타의 색인을 위해 벡터 근사에 기반한 LPC-file [5]이 개발되었다. LPC-file은 특히, 데이타 집합이 균일하게 분포할 때는 좋은 성능을 나타내지만 클러스터(cluster)를 이룰 때는 성능이 하락한다. 본 논문은 강하게 클러스터를 이루는 이미지 데이타 집합에 대해 LPC-file의 성능을 향상시킨 LCP*-file을 제시한다. 기본 아이디어는 고밀도 클러스터를 갖는 부분 공간을 찾기 위해 데이타 공간을 적응적으로 분할하고, 그 공간에 대해 벡터 근사의 식별 능력을 향상시키기 위해 더 많은 수의 비트를 할당한다. 그러나 분할된 공간이 비트들을 공유하기 때문에 사용되는 전체 비트 수는 오히려 줄어든다. 실험 결과에 따르면 LCP*-file은 강하게 클러스터를 이루는 이미지 데이타 집합에 대해 LPC-file의 성능을 크게 향상시킨다.

키워드 : 고차원 색인, 이미지 색인, 벡터 근사, LPC*-파일

Abstract In this paper, we propose the LPC*-file for efficient indexing of high-dimensional image data. With the proliferation of multimedia data, there is an increasing need to support the indexing and retrieval of high-dimensional image data. Recently, the LPC-file [5] that based on vector approximation has been developed for indexing high-dimensional data. The LPC-file gives good performance especially when the dataset is uniformly distributed. However, compared with for the uniformly distributed dataset, its performance degrades when the dataset is clustered. We improve the performance of the LPC-file for the strongly clustered image dataset. The basic idea is to adaptively partition the data space to find subspaces with high-density clusters and to assign more bits to them than others to increase the discriminatory power of the approximation of vectors. The total number of bits used to represent vector approximations is rather less than that of the LPC-file since the partitioned cells in the LPC*-file share the bits. An empirical evaluation shows that the LPC*-file results in significant performance improvements for real image datasets which are strongly clustered.

Key word : high-dimensional indexing, image indexing, vector approximation, LPC*-file

1. 서 론

이미지 데이타베이스는 이미지 객체를 종종 d 개의 수치적 벡터로 표현하고, 그 특성 벡터(feature vector)와 유사 척도(similarity measure)를 이용하여 저장하고 검색한다. 특성 벡터는 d-차원 벡터 공간 상의 점으로, 유사 척도는 그 공간 내에서 거리의 척도로 볼 수 있다. 이러한 시스템에서 중요한 문제는 주어진 질의 이미지와 가장 유사한 k 개의 이미지를 찾는 것이다. 이 문제는 먼저 질의 이미지를 대응하는 특성 벡터로 바꾸고, 그 벡터에 대한 k 개의 최근접 이웃(nearest neighbors: NNs)을 찾아서 해결한다. 기본 문제는 k 개의 NNs을 어떻게 효율적으로 찾는가 하는 것이다.

10차원 이하의 다차원 특성 벡터를 갖는 응용에서는

· 본 연구는 정보통신연구진흥원 대학기초연구지원사업(과제번호:2001-116-3)을 통해 수행되었음.
† 종신회원 : 숙명여자대학교 멀티미디어학과 교수
ghcha@sookmyung.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
chungcw@islab.kaist.ac.kr
논문접수 : 2001년 5월 11일
심사완료 : 2002년 2월 18일

R*-tree[2], X-tree[4], HG-tree[6], SR-tree[11] 같은 기존의 다차원 색인 기법(multidimensional index- ing methods: MIMs)을 이용하여 이 문제를 훌륭하게 해결할 수 있다. 그러나 지금까지, 예를 들어 100 차원 이상의, 고차원 특성 벡터를 갖는 응용에서는 효과적인 해결책이 발견되지 못했다. 고차원에서는 기존의 MIMs의 성능은 질의 객체와 전체 데이타베이스 객체를 하나씩 비교하는 순차 검색보다 성능이 더 나빠진다[1, 9, 12, 18]. 따라서 주된 문제는 차원의 저주(*dimen- sionality curse*) [16] - 차원이 높아짐에 따라 검색 성능이 급격히 하락하는 현상 - 를 해결하는 것이다. 이러한 요구에 따라 최근에 우리는 k-최근접(k-nearest neighbor: k-NN) 질의를 처리하기 위해 벡터 근사(vector approximation)에 기반한 색인 기법인 *LPC (Local Polar Coordinate)-file* [5]을 개발했다. LPC-file은 고차원 벡터 데이타 집합에서 순차 검색 및 VA-file [18]과 비교할 때 검색 성능을 크게 향상시켰다. VA-file은 차원의 저주를 극복하기 위해 개발된 최초의 벡터 근사 기법이다. LPC-file은 현재의 기법들보다 훨씬 개선된 성능을 보이지만, 공간 분할 기법이 간단하고 또한 분할된 공간에 균일하게 비트를 할당하기 때문에 이미지와 같이 아주 클러스터링되는 데이타 집합에서는 성능이 감소한다.

이 논문은 고차원 이미지 데이타 집합을 효율적으로 색인하는 방법에 대해 논의한다. 우리는 데이타 분포에 따라 데이타 공간을 적응적으로 분할하고 분할된 부분을 최소의 비트로 표현하는 알고리즘을 제안하고, 이에 기반한 새로운 색인 기법인 LPC*-file을 제시한다. 실제 이미지 데이타 집합을 이용한 실험 결과에 따르면 적응 분할을 이용한 LPC*-file은 LPC-file의 성능을 크게 향상시킨다.

2. 관련 연구

데이타베이스 시스템에서 k-NN 질의 처리를 위한 다양한 접근법이 연구되어 왔다. 직관적으로는 객체들이 다차원 공간상의 점으로 표현되기 때문에 MIMs을 사용하여 k-NN 질의 처리의 속도를 높이는 것이 자연스러워 보인다. 그러나, 대부분의 MIMs은 고차원 데이타 집합에 대해서 성능이 좋지 못하다. MIMs은 데이타 공간을 분할하고, 그 분할에 기초하여 데이타를 분류하고, 클러스터를 만들어 질의 처리 시에 탐색 공간을 잘라낸다. 그러나, 불행히도, 일반적으로 저차원 또는 최소한 중차원 (예를 들어, 10차원 이하) 공간에서는 MIMs은 훌륭한 성능을 발휘하나 차원이 높아질수록 그 성능이 크게 하락한다. 현재의 MIMs이 고차원 공간에서 검색 공간을 잘라낸다는 것은 매우 힘들다. 질의 벡터가 주어졌을 때 그 벡터를 중심으로 해서 예상되는 k-NN 거리 (k-NN^{dist})를 계산할 수 있다. 이 거리가 주어지면 k-NN 질의와 같은 수의 디스크 페이지를 방문하는 영역 질의를 형성하기 위해 k-NN 질의를 k-NN^{dist}를 반경으로 하는 구형 영역 질의(spherical range query)로 재구성할 수 있다. 이렇게 하면 k-NN 질의 처리를 위한 평균 비용은 해당 구와 교차하는 디스크 페이지의 수를 계산하여 측정할 수 있다. 그런데, 고차원에서는 데이타 공간의 희박성때문에 질의 벡터 $\boldsymbol{q}$와 그것의 k-번째 NN 간의 거리 k-$NN^{dist}(\boldsymbol{q})$가 데이타 공간의 각 차원의 길이보다 훨씬 커진다. 따라서 $\boldsymbol{q}$를 중심으로 반경 k-$NN^{dist}(\boldsymbol{q})$을 갖는 구가 데이타 공간의 대부분의 분할들과 교차하게 되고, (이 현상은 현재 대부분의 MIMs에 공통적인 현상이다.) 결국, 기존의 MIMs은 전체 색인 파일뿐만 아니라 전체 데이타 파일을 다 읽어야 하며, 따라서 고차원 공간에서는 대부분의 MIMs이 데이타 파일만 읽는 순차 검색보다 성능이 나을 수 없게 된다.

고차원 공간에서 이러한 MIMs이 갖는 한계를 해결하기 위해 최근에는 다른 방향에서의 접근이 시도되었고, 우리는 크게 다음의 4가지 부류로 이들을 분류한다:

- 차원 축소 접근법(Dimensionality reduction approach) [7, 10, 13],
- 근사 최대 근접 이웃 접근법(Approximate nearest neighbor approach) [1, 9, 12],
- 다중 공간 채움 곡선 접근법(Multiple space-filling curves approach) [14, 17],
- 벡터 근사 접근법(Vector approximation-based approach.) [5, 18].

2.1 차원 축소 접근법

차원 축소 접근법은 먼저 특이값 분해(Singular Value Decomposition), 이산 코사인 변환(Discrete Cosine Transform) 또는 이산 웨이블릿 변환(Discrete Wavelet Transform) 등을 적용하여 데이타 집합에 있는 대부분의 정보를 몇개의 차원으로 압축한 다음, 압축된 차원의 데이타를 MIMs을 이용하여 색인한다. 차원 축소 기법은 차원의 저주에 대한 한가지 해결책이지만 몇가지 결점을 갖고 있다: (1) 차원 축소는 질의 처리 결과의 정확도의 손실을 가져온다; (2) 데이타 변환은 전체 데이타에 대해 수행해야 하고, 또한 그 계산 비용이 높으므로 동적(dynamic) 데이타베이스에 쉽게 적용되지 않는다; (3) 축소된 차원이 MIMs을 적용할 만큼

충분히 작지 않을 수 있다. MIMs를 적용하려면 우리의 경험으로는 최소한 10 차원 정도로는 축소되어야 한다. 그렇지 않으면 검색 공간을 잘라내기 어렵다.

2.2 근사 최근접 이웃 접근법

근사 최근접 이웃 접근법의 기본 아이디어는 정확한 k 개의 NNs을 찾는 대신에 주어진 오차 한계 ε 내에서 k 개의 근사 최근접 이웃(approximate NNs)을 찾자는 것이다. 질의 벡터 $\boldsymbol{q}$와 거리 오차 $\varepsilon(>0)$이 주어졌을 때, $\|\boldsymbol{q}-\boldsymbol{p}\| \le (1+\varepsilon)\|\boldsymbol{q}-\boldsymbol{p}'\|$인 벡터 $\boldsymbol{p}$를 다른 데이타베이스 벡터 $\boldsymbol{p}'$에 대해 $\boldsymbol{q}$의 $(1+\varepsilon)$-근사 최대 근접 이웃이라고 한다. 근사 최근접 이웃 접근법에서도 차원 축소 접근법과 마찬가지로 필연적으로 질의 처리 결과의 정확도의 손실을 가져온다

2.3 다중 공간 채움 곡선 접근법

다중 공간 채움 곡선 접근법도 질의 처리 결과에서 어느 정도의 정확도를 포기한다는 의미에서 근사 접근법이라고 할 수 있다. 이 접근법은 $R^d \rightarrow R^1$ 의 매핑을 수행하는 힐버트 곡선(Hilbert curve)같은 공간 채움 곡선들의 집합을 가지고 많은 방법으로 d-차원 공간을 순서화한다. 이 매핑은 데이타 집합의 모든 점들에 대해 일차원적인 순서를 제공한다. 따라서 질의 벡터를 공간 채움 곡선으로 매핑한 후에는 데이타 공간에서 이웃한 점들을 찾기 위해 공간 채움 곡선을 따라 인접한 점들에 대한 영역 검색을 수행할 수 있다. 그러나, $R^d \rightarrow R^1$ 의 특성 때문에 질의 벡터의 가까운 이웃 중에 어떤 것들은 하나의 공간 채움 곡선을 따라서는 멀리 떨어져 있게 된다. 따라서 이러한 점들이 검색에서 누락되지 않도록 하기 위해서 $R^d \rightarrow R^1$을 위한 여러개의 다른 공간 채움 곡선을 사용한다. 후보 최대 근접 이웃의 집합은 모든 곡선으로부터 나온 작은 이웃 영역에 속한 점들의 합집합으로 구성된다.

비록 이 접근법이 k-NN 검색에 대한 성능을 향상시키지만 다른 근사 접근법과 마찬가지로 검색 중에 실제 가까이 있는 이웃을 누락시킬 가능성이 있다. 모든 k 개의 최근접 이웃을 찾을 가능성을 높이기 위해서는 더 많은 공간 채움 곡선을 사용하던지 각 곡선위에서 더 넓은 영역의 영역 검색을 수행하는 것이다. 그러나, 이것은 후보 집합의 크기를 증가시키게 되어 근사 접근법의 장점을 감소시킨다.

2.4 벡터 근사 접근법

벡터 근사 접근법은 기존의 MIM들은 검색 공간을 잘라내지 못하므로 검색 중에 모든 특성 벡터들을 디스크에서 읽어야만 한다는 가정하에, 이 특성 벡터들을 근사값으로 표현하고 이 근사값들을 차례로 읽어 특성 벡터 자체는 이 근사값을 통해 필터링하여, 그 중 적은 일부만 실제로 디스크에서 읽도록 하는 것이다. 벡터 근사 접근법에서 색인은 이 벡터 근사들의 배열로 이루어진 평면 파일(flat file)의 형태를 취한다. 따라서 삽입 및 삭제 등의 연산이 아주 간단하며, 전통적인 MIM의 트리 구조에서의 노드 등의 개념은 존재하지 않는다. 차원의 저주를 극복하기 위해 제안된 방법들 중에 이 벡터 근사 접근법만이 모든 k 개의 최근접 이웃을 정확히 찾는 방법이다. VA-file [18]과 LPC-file [5] 및 본 논문에서 제안하는 LPC$^+$-file 이 벡터 근사 접근법에 속한다.

VA-file은 데이타 공간을 2^b 개의 사각형 셀로 분할하고, 각 셀을 길이 b의 비트로 표현한다. 이 셀을 이용해서 VA-file은 그 셀에 놓이는 벡터를 근사한다. VA-file 자체는 이 벡터 근사의 집합으로 구성된 배열로 표현된다. VA-file에서의 최근접 이웃 검색은 두 단계로 이루어 진다. 첫 번째 단계에서, 모든 벡터 근사를 순차적으로 검사하고, 질의 벡터로 부터 각 셀까지의 거리의 하한 경계d_{min}과 상한 경계 d_{max}를 계산한다. 만약 어떤 근사(셀)에 대해, 그의 d_{min}이 지금까지 발견된 k-번째 최소 상한 경계를 초과한다면 그 벡터를 제거할 수 있다. 이 단계의 끝에서, 최소 경계를 갖는 벡터들이 질의 벡터에 대한 k 개의 최근접 이웃에 대한 후보로 결정된다. 두번째 단계에서는 d_{min}의 오름 차순으로 실제 벡터를 읽어서 후보 집합을 걸러낸다. 이 단계에서 후보들 중에서 그것의 d_{min}이 k-번째 최근접 이웃까지의 거리와 같거나 초과하는 근사를 만나면 검색을 마칠 수 있고, 지금까지의 k 개의 최근접 이웃이 검색 결과가 된다.

LPC-file은 최소한의 비트를 추가하여 벡터 근사의 필터링 능력을 높인다. 즉, 벡터의 지역 극좌표(local polar coordinates)를 근사에 추가한다. LPC-file은 벡터 공간을 사각형 셀로 분할하고 벡터를 분할된 지역 셀 내에서의 극좌표로 근사한다. LPC-file에서, 벡터 $\boldsymbol{p}$의 근사 $\boldsymbol{a}$는 다음과 같이 생성된다. 먼저 각 차원에 같은 수의 비트를 b 개 할당하고, 전체 데이타 공간을 2^{bd} 개의 셀로 분할한다. 여기서 d는 차원의 수이고, b는 차원의 수와 데이타 분포에 따라 보통 4에서 8까지의 수이다. 셀은 각 차원을 나타내는 비트들을 단순히 차례로 연결하여 표현된다. 다음에, 벡터 $\boldsymbol{p}$를 $\boldsymbol{p}$가 놓이는 셀 내부에서의 극좌표 (r, θ)를 사용하여 표현한다. 극좌표의 거리 r은 각 셀의 지역 원점 O와 벡터 $\boldsymbol{p}$간의 거리로 계산된다. 각도 θ는 벡터 $\boldsymbol{p}$와 지역 원점에서 반대편 모서리까지의 대각선 간의 각도로 계산된다. 이 근사의 결과로, 벡터 $\boldsymbol{p}$는 $\boldsymbol{a} = <c, r, \theta>$로 표현된다, 여기서 c는 근사 셀이다.

3. LPC$^+$-file

이 장에서는 우리는 VA-file과 LPC-file을 포함하여 기존 벡터 근사 기법의 문제점을 언급하고, 이러한 문제를 해결하고 이미지 데이타 집합에서 더 효율적인 k-NN 검색을 가능하게 하는 새로운 벡터 근사 접근 기법인 LPC$^+$-file을 제안한다. LPC$^+$-file은 이를 통해 범위 질의(range query)의 수행도 가능하지만 효율적인 k-NN 검색을 위해 개발된 기법이다.

이미지 색인 연구에 대한 우리의 시험장(testbed)은 IBM의 QBIC(Query By Image Content) 시스템 [8, 16]이다. 우리는 QBIC의 칼라 유사 척도(similarity measure)를 사용하며, 256-차원의 칼라 히스토그램을 이용하여 이미지를 검색하는 문제에 초점을 맞춘다. 각 이미지의 칼라 히스토그램 $\boldsymbol{p}$와 질의 이미지의 칼라 히스토그램 $\boldsymbol{q}$ 간의 거리 $||\boldsymbol{p}-\boldsymbol{q}||$는 다음과 같이 계산된다:

$$||\boldsymbol{p}-\boldsymbol{q}||=(\boldsymbol{p}-\boldsymbol{q})^T\boldsymbol{A}(\boldsymbol{p}-\boldsymbol{q})$$

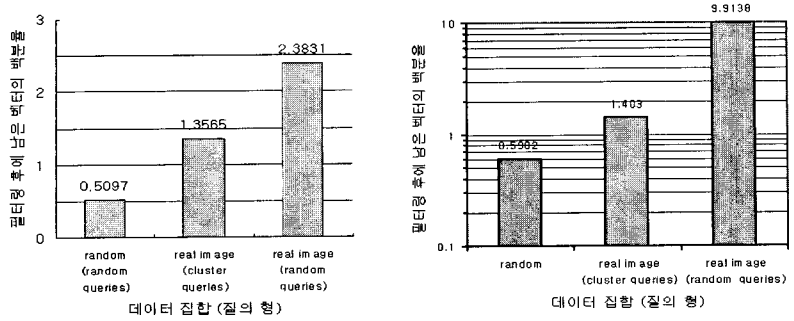
히스토그램 $\boldsymbol{p}$와 $\boldsymbol{q}$는 그것의 i-번째 요소가 해당 이미지에서 칼라 i의 백분율을 나타내는 256-차원 벡터이다. $\boldsymbol{A}$는 그것의 (i, j)-번째 요소 a_{ij}가 다음과 같이 정의되는 대칭 칼라 유사 행렬(symmetric color similarity matrix)이다: $a_{ij}=1-d(c_i, c_j)/d_{max}$, 여기서 c_i와 c_j는 칼라 히스토그램에서 각기 i-번째와 j-번째 칼라이고, $d(c_i, c_j)$는 MTM(Mathematical Transform to Munsell [15]) 칼라 거리이며, d_{max}는 칼라 간의 최대 거리이다.

3.1 연구 동기

VA-file과 LPC-file을 포함하여 현 벡터 근사 기법에서는 같은 셀에 여러 개의 점들이 들어가는 경우는 거의 없다는 묵시적인 가정을 하고 있다. 데이타 분포가 무작위일 때는 실제로 같은 셀에 여러 개의 점들이 들어가는 경우는 거의 없다고 할 수 있다. 단위 하이퍼 큐브(unit hyper-cube) $\Omega = [0, 1]^d$ 내에 놓이는 d-차원 데이타 집합을 생각해보자. 데이타 점들과 질의 점들이 이 데이타 공간에서 균일하게 분포한다고 가정하자. 데이타 공간을 사각형 셀로 분할할 때, 한 점이 한 셀에 놓이는 확률은 그 셀의 크기에 비례한다: $Volume(\text{셀}) = 1/2^{bd}$, 여기서 b는 각 차원을 나타내기 위해 각 차원에 할당된 비트 수, d는 차원의 수이다. 한 벡터가 특정 셀에 놓였을 때, 다른 벡터가 같은 셀에 놓일 확률은 다음과 같다: $1-(1-1/2^{bd})^{N-1} \approx N/2^{bd}$, 여기서 N은 데이타 집합에 있는 전체 점들의 수이다. b=8, d=100, N=1,000,000 $\approx 2^{20}$을 가정할 때, 위의 확률은 2^{-780}이다. 이것은 이상과 같은 가정 하에서는 같은 셀에 여러 개의 점들이 놓일 확률은 매우 희박하다는 것을 의미한다. 나아가, 셀의 수 2^{bd} 가 전체 데이타 집합의 크기 N보다 훨씬 크기 때문에 대부분의 셀은 비어있다.

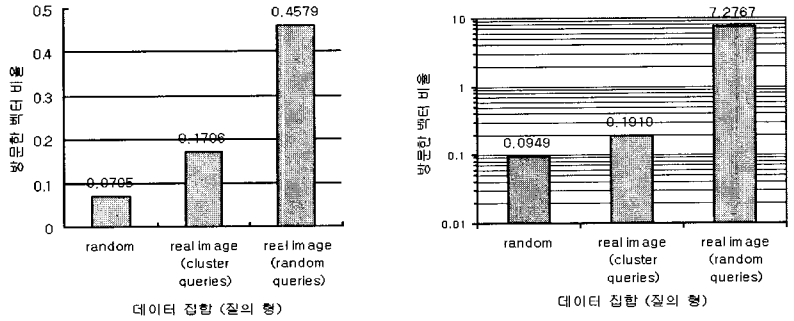
그러나, 만약 이미지 데이타 집합처럼 데이타 점들이 아주 클러스터를 이루고, 한 셀에 여러 개의 점들이 놓일 확률이 증가하면 동일한 근사를 사용하는 벡터(점)들이 늘어난다. 이것은 각 근사의 구분 능력을 떨어지게 하고, 따라서 검색 알고리즘의 각 단계에서 근접 이웃에 대한 후보를 필터링하는 능력을 떨어지게 만들어, 결국 디스크 접근 회수를 증가시키게 된다.

그림 1과 2는 실험을 통해 각기 벡터 근사 기법에서 첫번째 필터링 단계와 두번째 정제 단계에서 무작위 데이타와 실제 이미지 데이타를 사용했을 때의 벡터 선택률(vector selectivity)을 비교한 그림이다. 벡터 선택률은 데이타 집합에 있는 전체 데이타의 수에 대한 실제 방문한 벡터의 수의 비율로 정의한다. k-NN 검색은 디스크를 방문한 회수에 크게 좌우되고, 그것은 벡터 선택률에 크게 영향을 받기 때문에 벡터 선택률은 k-NN 검색의 좋은 성능 평가 기준이 된다. 벡터 근사 기법의 첫번째 검색 단계에서의 선택률은 전체 벡터 수에 대한 첫번째 단계 후에 필터링되지 않고 남은 후보의 수의 비율을 의미한다. 두번째 단계에서의 선택률은 전체 벡터 수에 대해 실제로 방문한 벡터의 비율을 나타낸다. 그림 1에서, x-축은 우리가 실험에서 사용한 데이타 집합과 질의 형(query type)을 나타내고, y-축은 첫번째 필터링 단계 후에 남아 있는 벡터의 비율(즉, 첫번째 단계의 벡터 선택률)을 나타낸다. 클러스터 질의(cluster query)란 실제 이미지 데이타 집합에 있는 벡터에서 질의 벡터를 선택한 것을 의미하고, 무작위 질의(random query)는 질의 벡터를 무작위로 선택한 것을 의미한다. 벡터 선택률 실험에서 우리는 13,724 개의 256-차원의 미국 우표 이미지와 사진 이미지를 사용하였으며, 이 이미지들은 IBM QBIC 시스템의 이미지 데이타베이스에서 가져온 것이다. 우표는 종종 공통적인 칼라와 디자인을 갖는 시리즈로 발행되고, 사진도 공통적인 주제를 갖고 있다. 따라서 이 이미지 데이타 집합은 강하게 클러스터를 이루는 분포를 나타낸다. 실험에 사용된 무작위 데이타 집합은 256-차원의 13,724 개의 무작위 벡터로 구성된다. 무작위 데이타 집합에 대해 1,000 개의 무작위 10-NN 질의를 수행하였고, 이미지 데이타 집합에 대해, 각각 1,000 개의 무작위 10-NN 질의와 클러스터 10-NN 질의를 수행하였고, 그 결과를 평균해서 제시하였다. 그림 1(가)의 LPC-file의 경우에, 이미지 데이타 집합에 대한 실험 결과를 살펴보면, 첫번째 단계의 필터링 후에 남은 후보 벡터의 수가 무작위 데이타 집합의



(가) LPC-file (나) VA-file

그림 1 필터링 단계에서의 무작위 데이타 집합과 이미지 데이터 집합 간의 선택률 비교



(가) LPC-file (나) VA-file

그림 2 정제 단계에서의 무작위 데이타 집합과 이미지 데이터 집합 간의 선택률 비교

경우와 비교할 때 2.66 배에서 4.68배까지 많다. 그림 1 (나)의 VA-file의 경우에도 이미지 데이타 집합에 대한 실험에서 첫번째 단계의 필터링 후에 남은 후보 벡터의 수가 무작위 데이타 집합의 경우보다 2.34 배에서 16.57 배까지 많다. 이것은 무작위 데이타 집합과 비교하여 이미지 데이타와 같이 클러스터를 이루는 데이타 집합의 경우에, 현존하는 벡터 근사 기법의 필터링 능력이 현저히 떨어짐을 나타낸다.

그림 2의 두 번째 단계에 대한 실험에서도, 이미지 데이타 집합의 경우에 LPC-file은 무작위 데이타 집합의 경우와 비교할 때 2.43 배에서 6.49 배까지 더 많은 벡터를 방문한다. 이것은 실제로 이만큼 더 무작위로 디스크 접근 회수가 많아짐을 나타낸다. VA-file의 경우에도, 무작위 데이타 집합의 경우와 비교할 때 이미지 데이타 집합의 경우에 2.02 배에서 76.67 배까지 더 많은 벡터를 읽는다. 이것은 클러스터 데이타 집합의 경우에 k-NN 검색 시에 드는 입출력 비용이 무작위 데이타 집합의 경우와 비교할 때 훨씬 크다는 것을 의미한다.

클러스터 데이타 집합에서의 성능 저하는 정도의 차이는 있지만 현존하는 벡터 근사 기법에 공통적인 문제이다. 이러한 성능 감소의 원인은 데이타 분포가 분할된 셀에 따라 일정하지 않고 각 셀의 밀도의 차가 극심함에도 불구하고, 공간을 일정한 셀로 분할하고, 분할된 셀에 할당하는 (즉, 각 벡터를 근사하는데 사용하는) 비트의 수가 일정하기 때문이다. 이 문제를 극복하기 위해 데이타 공간의 지역적 통계 특성에 따라 공간을 적응적으로 분할하고, 분할된 셀에 할당하는 비트의 수를 달리하는 것이 LPC*-file의 기본 개념이다.

3.2 데이타 공간의 적응 분할

LPC*-file은 데이타 공간을 분할하고, 분할된 셀에 할당할 비트의 수를 결정하기 위해 분할된 셀의 밀도에 기반한 접근법을 사용한다. 데이타 집합의 밀도를 근사하기 위해 LPC*-file은 데이타 공간을 사각형 셀로 분할하고, 각 셀에 속하는 점의 수를 계산한다. 이렇게 하기 위해 LPC-file에서처럼 각 차원 j에 균일하게 b 개의 비트를 할당하고, 전체 데이타 공간을 2^{nd} 개의 셀로

분할한다. 각 셀이 같은 크기를 갖고 있으므로 셀 내부의 점의 수를 그 셀의 밀도를 근사하는데 사용할 수 있다. 전체 데이타 수에 비하여 전체 셀의 수 2^{bd}가 훨씬 크므로 대부분의 셀은 비어 있게 되나 이미지 데이타 집합의 군집 특성에 따라 특정 셀에 많은 점들이 존재하게 된다.

한 셀의 선택률은 전체 데이타 수에 대한 한 셀에 포함된 데이타의 수의 비율로 정의한다. 한 셀 c 의 선택률이 어떤 밀도 임계치(density threshold) τ 보다 크면, 그 셀 c를 조밀하다고 하고, 그렇지 않으면 희박하다고 한다. 우리는 조밀한 셀을 클러스터(cluster)라고 하고, 희박한 셀에 포함된 점들을 아웃라이어(outliers)라고 부른다.

희박한 셀에 포함되는 아웃라이어들은 근사 $<c, r, \theta>$로 표현한다. 여기서 c, r, θ는 LPC-file에서 정의된 것과 같다. 초기 아웃라이어들을 위한 셀 c는 초기 고정 분할 자체의 비트 패턴으로 표현한다. 즉, 초기 아웃라이어는 bd 개의 비트로 표현한다. 클러스터가 발견되면 그 속에 있는 점들이 아웃라이어가 될 때까지 클러스터를 2^d 개의 더 작은 셀로 계속 분할한다. 하나의 공통 분할에서 생성된 아웃라이어들은 그 분할의 비트 표현을 공유한다. 따라서 각 아웃라이어의 셀은 그 셀이 속하는 분할의 공통 비트 패턴과 자신이 속하는 셀에 대한 d 개의 비트로 표현된다. 즉, 하나의 공통 분할에서 생성된 아웃라이어들은 d 개의 비트만으로 표현되고, 공통 비트 패턴은 공유한다. 실제로 이것은 셀을 표현하는데 드는 비트의 수를 크게 줄여주고, 따라서 근사 화일을 읽는 입출력 비용을 크게 줄일 수 있다. 그림 3은 3.1 절에서 기술한 실험에서 LPC$^+$-file, LPC-file, VA-file에 대한 근사 화일을 읽는데 사용된 디스크 접근 회수를 보여준다. 즉, 이것은 벡터 근사 기법에서 k-NN 검색의 첫번째 단계에서의 입출력 비용을 나타낸다. 실험에서 사용된 디스크 페이지 크기는 4 KB 이고, 13,724 개의 256-차원의 이미지 데이타 집합에 대해 10-NN 검색을 수행하였다. 그림 3의 결과에서, LPC$^+$-file에서의 디스크 접근 회수는 LPC-file과 VA-file의 디스크 접근 회수의 거의 50%밖에 되지 않음을 볼 수 있다. 이것은 LPC$^+$-file에서 셀 근사를 표현하는데 사용된 비트의 수가 LPC-file과 VA-file의 경우에 비해 거의 절반 이었음을 나타낸다.

예 1: 데이타 공간의 차원은 2이고, 각 차원에 할당된 초기 비트 수는 2, 밀도 임계치 τ는 $2/N$이라고 가정한다. 여기서 N은 데이타베이스에 있는 전체 점들의 수이다. 그림 4에서, 2-차원 데이타 공간은 차원 당 2 비트를 사용하여 초기에 4×4 개의 셀로 분할되어 있다. 아웃라이어, 즉, 선택률이 τ보다 작거나 같은 셀 c_i에 속하는 데이타 점들은 $<c_i, r, \theta>$로 근사된다. 점 $x1$, $x2$, $x3$, $x4$는 그들이 놓이는 셀 c_i와 c_i에서의 그들의 극좌표로 근사된다. 클러스터들은, 즉, 선택률이 τ보다 큰 셀, 예를 들어, 셀 C4,은 2^2 개의 더 작은 셀로 분할된다. 셀 C4는 4 개의 셀로 분할되고, 셀 C5에 있는 점들은 $<c_5, r, \theta>$로 근사된다. 셀 C6는 더 작은 셀로 분할되고 최종적으로 모든 점들이 근사된다. 셀 C1, C2, C3의 셀 표현은 각기 (10 00), (01 01), (11 11)이다. 셀 C5에 있는 점들은 셀 표현 (1 1)과 공통 셀 표현 (00 11)을 갖는다. 셀 C6 내의 두 셀은 셀 C6의 공통 셀 표현 (000 110)과 각기 셀 표현 (0 1), (1 0)을 갖는다. 실제 이미지 데이타는 특정 셀들에 모이는 경향이 있으므로 실제로 공통의 셀 표현을 공유하는 많은 점들이 존재한다. 이것은 그림 3에 나타낸 것처럼 디스크 입출력 비용을 줄인다.

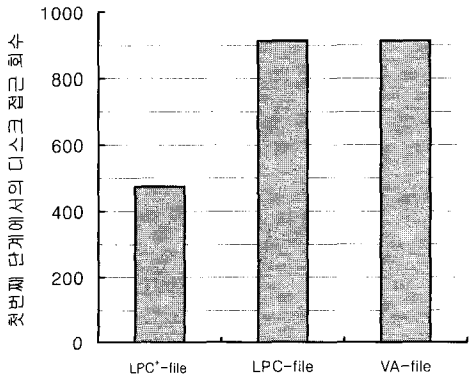


그림 3 필터링 단계에서의 디스크 접근 회수의 비교

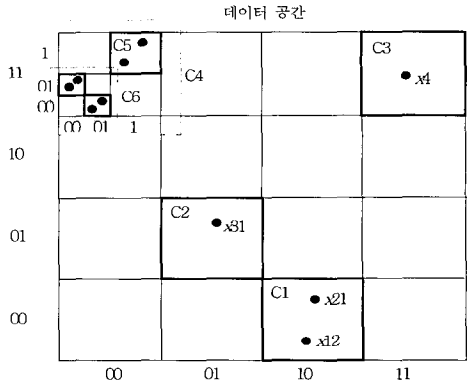


그림 4 LPC$^+$-file의 비균일 데이타 공간 분할

3.3 알고리즘

이 절에서는 각 분할 셀에 비트를 비균일하게 할당하기 위해 데이타 공간을 적응적으로 분할하는 LPC$^+$-file의 알고리즘을 설명한다. 공간 분할 알고리즘은 제일 먼저 차원 당 b 개의 비트를 가지고 데이타 공간을 분할한다. 이 초기 분할에서 아웃라이어와 클러스터를 결정하고, 아웃라이어들의 셀들은 bd 개의 비트로 표현한다. 클러스터들은 그들을 표현할 적합한 수의 비트를 찾기 위해 계속 분할하고, 후속적인 아웃라이어들을 발견하고 근사해 나간다. 다음 알고리즘 **Partition**은 한 클러스터의 분할을 수행한다. 함수 **MakeOutliers**는 주어진 한 레벨에서 아웃라이어들을 모으고 그들을 LPC-file의 근사를 사용하여 근사한다.

Algorithm Partition(dataset S, dataset_size s, split_dim j, split_location l_j, global_cell c_g, local_cell c_l)

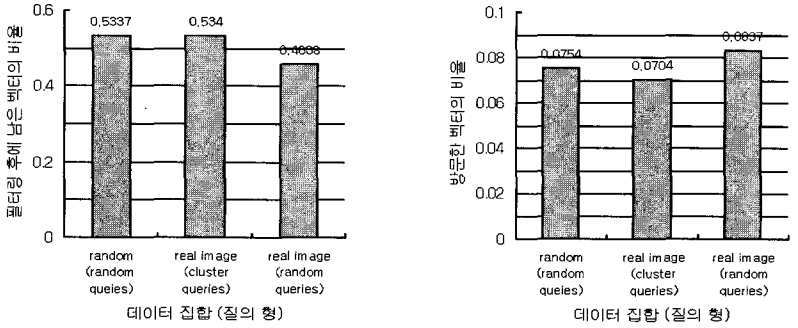
// S: 분할할 한 클러스터의 데이터 집합
// s: 데이터 집합 S의 크기
// j: 분할할 차원
// d: 데이터 공간의 차원
// l_j: 차원 j에서 분할할 위치
// S_1, S_2: j와 l_j 에 기초하여 S에서 분할된 두 데이터 집합
// $size(S_1)$, $size(S_2)$: 각기 S_1과 S_2 에 있는 점들의 수
// c_g: 데이터 집합 S 에 대한 초기 전체 셀
// c_l: c_g로부터 분할된 지역 셀, 처음에, c_g는 c_l와 같다.
// n: 밀도 임계치 τ · 데이터 집합에 있는 전체 점들의 수
{
1. 데이터 집합 S를 분할 차원 j와 분할 위치 l_j에 기초하여 두 개의 부분 집합 S_1과 S_2로 분할한다.
2. If ($size(S_1) > 0$) {
  If ($j < d$) {
    2.1 지역 셀 c_l과 새로운 분할 차원 j+1에 기초하여 새로운 분할 위치 l_{j+1}을 결정한다.
    2.2 분할에 기초하여 지역 셀 c_l 를 조정한다.
    2.3 **Partition(S_1, size(S_1), j+1, l_{j+1}, c_g, c_l)**을 호출한다.
  }
  Else {
    If ($size(S_1) < n$)
      MakeOutliers(S_1, size(S_1), c_g)를 호출한다.
    Else {
      2.1 지역 셀 c_l와 첫번째 차원 1에 기초하여 새로운 분할 위치 l_1을 결정한다.
      2.2 **Partition(S_1, size(S_1), 1, l_1, c_l, c_l)**을 호출한다.
    }
  }
}
3. If ($size(S_2) > 0$) {
  If ($j < d$) {
    3.1 지역 셀 c_l과 새로운 분할 차원 j+1에 기초하여 새로운 분할 위치 l_{j+1}을 결정한다.
    3.2 분할에 기초하여 지역 셀 c_l 를 조정한다.
    3.2 **Partition(S_2, size(S_2), j+1, l_{j+1}, c_g, c_l)**를 호출한다.
  }
  Else {
    If ($size(S_1) < n$)
      MakeOutliers(S_2, size(S_2), c_g)를 호출한다.
    Else {
      3.1 지역 셀 c_l와 첫번째 차원 1에 기초하여 새로운 분할 위치 l_1을 결정한다.
      3.2 **Partition(S_2, size(S_2), 1, l_1, c_l, c_l)**를 호출한다.
    }
  }
}
}

4. 성능 평가

LPC$^+$-file의 성능 평가를 위해 3.1절에서 기술한 13,724 개의 256-차원의 이미지 히스토그램 데이타를 사용하여 실험을 수행하였다. 이 이미지들은 IBM QBIC 시스템의 이미지 데이타베이스에서 가져왔으며, QBIC의 칼라 특성 추출 알고리즘을 사용하여 256-차원의 특성을 추출하였다. 우리는 더 큰 데이타 집합을 형성하기 위해 주어진 데이타 집합에서 n개의 클러스터를 추출한 다음, 각 클러스터 내에서 무작위로 선택한 2개의 데이타 벡터의 평균치를 새로운 데이타 벡터로 만들어서 100,000 개까지 데이타 집합을 확장하였다. 이렇게 하면 원래 데이타 집합의 데이타 분포를 보존하면서 데이타 집합의 크기를 확장할 수 있다. 확장한 데이타 집합과 원래 데이타 집합에 대해 수행한 실험 결과가 크게 다르지 않으므로 본 논문에서는 원래 데이타 집합에 대해 수행한 실험 결과를 기술한다.

모든 실험에서, 유클리드 거리 L_2를 사용하였고, 찾는 최근접 이웃의 수는 10으로 고정하였다. 즉, k = 10이다. 다양한 k (즉, k = 5, 10, 15, 20)에 대해서도 실험을 수행하였지만 상대적인 성능의 차는 발견되지 않아 대표치 k = 10에 대한 결과만 기술한다. 실험에서 사용한 디스크 페이지 크기는 4 KB이다. 1,000 개의 무작위 10-NN 질의와 1,000 개의 클러스터 10-NN 질의를 수행하고 그 결과를 각기 평균하였다. 근사 셀의 각 차원을 위한 초기 비트 수로 6을 사용하였다. 밀도 임계치 τ로는 0.001을 사용하였다.

그림 5는 무작위 데이타 집합과 실제 이미지 데이타 집합에 대한 k-NN 검색의 필터링 단계와 정제 단계에서의 LPC$^+$-file의 벡터 선택률을 비교한 그림이다. x-축은 실험에서 사용된 데이타 집합과 질의 형을 나타낸다. 그림 5(가)의 y-축은 첫번째 단계 후의 남은 벡터의 비율(벡터 선택률)을 나타내고, 그림 5(나)의 y-축은 두 번째 단계에서 방문한 벡터의 비율을 나타낸다. 그림 5의 결과와 그림 1과 2의 결과를 비교해보자. LPC$^+$-file



(가) 필터링 단계 (나) 정제 단계

그림 5 LPC$^+$-file의 선택률 성능 실험

의 경우에, 이미지 데이타 집합에서 첫번째 검색 단계에서 남은 벡터의 수는 무작위 데이타 집합에서의 그것과 비교할 때 작거나 거의 같다. VA-file과 LPC-file에 대한 그림 1과 2를 보면, 이미지 데이타 집합의 경우, 첫번째 검색 단계 후에 남은 벡터의 수는 무작위 데이타 집합에 대한 그것의 수보다 2.34 배에서 16.57 배까지 많다. 두번째 검색 단계에서, 실제 이미지 데이타 집합의 경우, LPC$^+$-file은 클러스터 10-NN 질의를 수행하였을 경우 무작위 데이타 집합에 대해 질의를 수행한 것보다 적은 수의 실제 벡터를 방문한다. 무작위 10-NN 질의를 수행하였을 경우에는 무작위 데이타 집합에 대해 질의를 수행한 것보다 오직 0.0083% 많은 실제 벡터를 방문한다(실제 더 많이 방문한 벡터의 수는 2.79 이다). 반면에, VA-file과 LPC-file의 경우, 이미지 데이타 집합의 경우, 두번째 검색 단계에서 방문한 실제 벡터의 수는 무작위 데이타 집합에 대한 것보다 2.02 배에서 76.77 배까지 많다. 이 결과를 요약해 보면, VA-file과 LPC-file과 달리 LPC$^+$-file은 강하게 클러스터링되는 데이타 집합의 경우에도 성능이 떨어지지 않음을 알 수 있다.

그림 6은 10-NN 질의에 대해서 두번째 검색 단계에서의 평균 디스크 접근 회수를 나타낸다. 첫번째 단계에서의 디스크 접근 형태는 순차적인 반면에 두 번째 단계에서는 무작위이다. 따라서 두번째 단계에서의 디스크 접근이 첫번째 단계에서의 디스크 접근보다 검색 성능에 더 큰 영향을 준다. 그림 6에서 이미 첫번째 단계에서의 LPC$^+$-file, LPC-file, VA-file의 평균 디스크 접근 회수를 보였다. 첫번째 단계에서 LPC$^+$-file의 디스크 접근 회수는 LPC-file과 VA-file의 그것의 약 절반에 불과하다. 그림 6을 살펴보면, LPC$^+$-file의 경우에, 실제

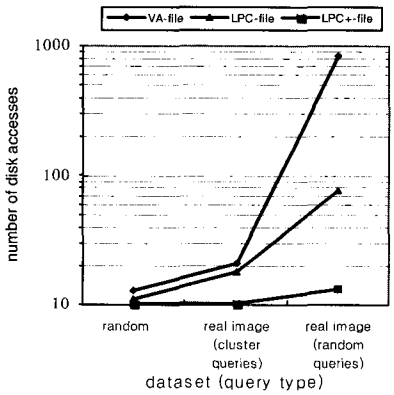


그림 6 두 번재 단계에서의 평균 디스크 접근 회수

이미지 데이타 집합에서의 검색으로 인한 디스크 접근 회수의 증가는 매우 작다. 이미지 데이타 집합에서의 클러스터 질의의 경우, 디스크 접근 회수는 무작위 데이타 집합의 그것과 거의 같다. 이미지 데이타 집합에서의 무작위 질의의 경우 무작위 데이타 집합에서의 질의와 비교할 때, 실제로, 디스크 접근 회수의 증가는 오직 2.79에 불과하다. 반면에, VA-file과 LPC-file의 경우에, 이미지 데이타 집합에 의해 야기된 디스크 접근 회수의 증가는, 특히, VA-file의 경우에, 굉장히 크다. 결론적으로 실제 이미지 데이타 집합에서 LPC$^+$-file의 성능은 LPC-file과 VA-file에 비교할 때 상당히 우월하다.

5. 결 론

이 논문에서 우리는 클러스터링 이미지 데이타 집합을 위한 LPC$^+$-file이라고 하는 새로운 벡터 근사 기반의 고차원 색인 기법을 제시하였다. VA-file과 LPC-

file을 포함하는 현재의 벡터 근사 기법은 동일한 분할 셀에 여러 개의 점들이 놓이는 경우는 거의 없다는 가정 하에서 균일하게 셀을 분할하고 분할된 셀의 표현을 위해 동일한 수의 비트를 할당한다. 그러나, 이러한 가정과 전략은 데이타가 균일하게 또는 무작위로 분포된 데이타베이스에만 적합하고, 이미지 데이타 집합같이 강하게 클러스터링되는 데이타베이스에는 적합하지 않다.

LPC$^+$-file은 데이타 공간을 분할하고 분할된 셀에 할당하는 비트 수를 결정하기 위해 공간 밀도에 기초한 접근법을 사용한다. LPC$^+$-file의 목적은 벡터 근사 기법의 성능을 향상시키기 위해 최소 수의 비트를 가지고 벡터 근사의 구분(즉, 필터링) 능력을 향상시키는 것이다. LPC$^+$-file의 근본적인 아이디어는 조밀한 지역은 세밀히 분할하고 거기에 있는 벡터 근사의 표현에는 많은 비트를 할당하여 벡터 근사의 구분 능력을 향상시키고, 밀도가 희박한 지역에 있는 벡터 근사에는 적은 수의 비트만 할당하는 것이다. 아울러, 비록 조밀한 지역을 세밀히 분할하여 많은 비트를 할당해도 대부분의 분할 셀의 표현을 위한 비트를 공유하기 때문에 LPC$^+$-file은 전체 벡터 근사 표현에 드는 비트 수는 오히려 기존의 VA-file과 LPC-file보다 훨씬 적다. 결국 LPC$^+$-file은 더 적은 수의 비트로 더 나은 벡터 근사를 생성함으로써 클러스터링되는 고차원 이미지 데이타베이스에서 k-NN 검색 성능을 크게 향상시켰다.

참 고 문 헌

[1] Arya, S. et al., "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," *Journal of the ACM*, 45(6), 891-923, Nov. 1998.

[2] Beckmann, N. et al., "The R*-tree: An efficient and robust access method for points and rectangles," *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, 322-331, 1990.

[3] Berchtold, S., Boehm, C., and Kriegel, H.-P., "The Pyramid-Technique: Towards Breaking the Curse of Dimensionality," *Proc. of the ACM SIGMOD Int'l Conf.*, 142-153, 1998.

[4] Berchtold, S., Keim, D.A., Kriegel, H.-P., "The X-tree: An Index Structure for High-Dimensional Data," *Proc. of the Int'l Conf. on Very Large Data Bases*, 28-39, 1996.

[5] Cha, G.-H., Zhu, X., Petkovic, D., and Chung, C.-W., "An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Databases," *IEEE Transactions on Multimedia*, 4(1), 76-87, March 2002.

[6] Cha, G.-H. and Chung, C.-W., "A New Indexing Scheme for Content-Based Image Retrieval," *Multimedia Tools and Applications*, 6(3), 263-288, May 1998.

[7] Chakrabarti, K. and Mehrotra, S., "Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces," *Proc. of the Int'l Conf. on VLDB*, 89-100, 2000.

[8] Flickner, M., et al., "Query by image and video content: the QBIC system," *IEEE Computer*, 28, 23-32, 1995.

[9] Indyk, P. and Motwani, R., "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," *Proc. of the ACM Symp. on the Theory of Computing*, 604-613, 1998.

[10] Kanth, K.V.R., Agrawal, D. and Singh, A., "Dimensionality Reduction for Similarity Searching in Dynamic Databases," *Proc. of the ACM SIGMOD Int'l Conf.*, 166-176, 1998.

[11] Katayama, N. and Satoh, S. "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," *Proc. of the ACM SIGMOD Int'l Conf*, 369-380, 1997.

[12] Kushilevitz, E., Ostrovsky, R. and Y. Rabani, "Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces," *Proc. of the ACM Symp. on the Theory of Computing*, 614-623, 1998.

[13] Lin, K.-I., Jagadish, H.V., and Faloutsos, C., "The TV-tree: An Index Structure for High-Dimensional Data," *The VLDB Journal*, 3(4), 517-542, 1994.

[14] Megiddo, N. and Shaft, U., "Efficient Nearest Neighbor Indexing Based on a Collection of Space-Filling Curves," *Technical Report RJ 10093, IBM Almaden Research Center*, Nov. 1997.

[15] Miyahara, M. and Yoshida, Y., "Mathematical Transform of (R,G,B) Color Data to Munsell (H, V, C) Color Data," *Visual Communication and Image Processing*, 1001, 650 - 657, *SPIE*, 1992.

[16] Niblack, N. et al., "The QBIC Project: Querying Images By Content Using Color, Texture, and Shape," *Proc. of the SPIE Conf.*, 173-187, 1993.

[17] Shepherd, J., Zhu, X. and Megiddo, N., "A Fast Indexing Method for Multidimensional Nearest Neighbor Search," *Proc. of the SPIE Conf.*, 350 - 355, 1999.

[18] Weber, R., Schek, H.-J., and Blott, S., "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," *Proc. of the Int'l Conf. on VLDB*, 194-205, 1998.

차 광 호

1992년 9월 ~ 1997년 8월 한국과학기술원 정보및통신공학과(박사). 1987년 3월 ~ 1989년 2월 한국과학기술원 전산학과(석사). 1980년 3월 ~ 1984년 2월 부산대학교 계산통계학과(학사). 2002년 3월 ~ 현재 숙명여자대학교 멀티미디어학과(조교수). 1997년 3월 ~ 2002년 2월 동명정보대학교 멀티미디어공학과(조교수). 1998년 12월 ~ 2000년 2월 Digital Media Management Group, IBM Almaden Research Center (Visiting Scientist), 1989년 2월 ~ 1997년 2월 데이콤 부가통신사업본부(선임연구원), 1986년 1월 ~ 1987년 2월 삼성반도체통신(연구원). 관심분야는 내용기반 이미지/비디오/음악 검색, XML 데이타베이스, 내용 기반 원격 학습/교육

정 진 완

1973년 서울대학교 공과대학 전기공학과(학사). 1983년 University of Michigan 컴퓨터공학과(박사). 1983년 ~ 1993년 미국 GM 연구소 전산과학과 선임연구원 및 책임연구원. 1993년 ~ 현재 한국과학기술원 전산학과 부교수 및 교수. 관심분야는 XML, 멀티미디어 데이타베이스, GIS.