연속적 I/O와 클러스터 인덱싱 구조를 이용한 이미지 데이타 검색 연구

김 진 옥[†] · 황 대 준[††]

요 약

이미지, 비디오, 오디오와 같은 멀티미디어 데이터들은 텍스트기반의 데이터에 비하여 대용량이고 비정형적인 특성때문에 검색이 어렵다. 또한 멀티미디어 데이터의 특징은 행렬이나 벡터의 형태로 표현되기 때문에 완전일치 검색이 아닌 유사 검색을 수행하여 원하는 이미지와 유사한 이미지를 검색해야 한다. 본 논문에서는 멀티미디어 데이터 검색에 클러스터링과 인덱싱 기법을 같이 적용하여 유사한 이미지는 인접 디스크에 클러스터하고 이 클러스터에 접근하는 인덱스를 구축함으로써 이미지 근처의 클러스터를 찾아 빠른 검색 결과를 제공하는 유사 검색방법을 제시한다. 본 논문에서는 트리 유사 구조의 인덱스 대신 해싱 방법을 이용하며 검색시 I/O시간을 줄이기 위해 오브젝트를 가진 클러스터 위치를 찾는데 한번의 I/O를 사용하고 이 클러스터를 읽기 위해 연속적인 파일 I/O를 사용하여 클러스터를 찾는 비용을 최소화한다. 클러스터 인덱싱 접근은 클러스터링을 생성하는 알고리즘과 해싱 기법의 인덱싱을 이용함으로써 고차원 데이터가 갖는 차원의 문제를 해결하며 클러스터링 또는 인덱싱 만을 이용하는 내용기반의 이미지 검색보다 효율적인 검색 적합성을 보인다

A study on searching image by cluster indexing and sequential I/O

Jin Ok Kim[†] · Dae Joon Hwang[††]

ABSTRACT

There are many technically difficult issues in searching multimedia data such as image, video and audio because they are massive and more complex than simple text-based data. As a method of searching multimedia data, a similarity retrieval has been studied to retrieve automatically basic features of multimedia data and to make a search among data with retrieved features because exact match is not adaptable to a matrix of features of multimedia. In this paper, data clustering and its indexing are proposed as a speedy similarity-retrieval method of multimedia data. This approach clusters similar images on adjacent disk cylinders and then builds indexes to access the clusters. To minimize the search cost, the hashing is adapted to index cluster. In addition, to reduce I/O time, the proposed searching takes just one I/O to look up the location of the cluster containing similar object and one sequential file I/O to read in this cluster. The proposed schema solves the problem of multi-dimension by using clustering and its indexing and has higher search efficiency than the content-based image retrieval that uses only clustering or indexing structure.

키워드 : 이미지 데이터(Image Data), 클러스터링(Clustering), 다차원 인덱싱(Multi-dimensional Indexing), 유사도 검색(Similarity Search)

1. 서 론

인터넷 기술의 발전은 예측할 수 없을 정도로 급격한 정보의 생성과 분포를 가져왔으며 원하는 텍스트와 이미지 데이터를 빨리 검색하려는 이용자의 요구는 갈수록 다양해지고 있다. 이에 따라 이미지 검색 방법은 사람이 주관적으로 이미지 데이터를 설명한 색인을 텍스트 검색으로 찾아내는 방법에서 점차 필요한 이미지 데이터를 제시하고 이와 유사한 이미지를 빨리 찾는 내용기반 검색 기술[1-3]로 발전하고 있다.

† 정 회 원 : 성균관대학교 대학원 전기전자 및 컴퓨터공학부
†† 정 회 원 : 성균관대학교 전기전자 및 컴퓨터공학과 교수
논문접수 : 2002년 4월 18일, 심사완료 : 2002년 8월 19일

내용 기반의 이미지 검색 기술에서는 이미지를 고차원 특징벡터로 정규화하여 고차원 공간의 근접지역에서 주어진 이미지 벡터와 가장 유사한 벡터를 검색한다. 가장 단순한 유사 질의 방법은 데이터베이스에 있는 모든 이미지를 읽고 질의 이미지까지의 거리를 계산하여 최근접 이미지를 선택하는 것이다. 하지만 이 방법의 실행시간은 데이터의 차원(D), 데이터 집합의 크기(N), 알고리즘의 복잡도($O(ND)$)와 비례하기 때문에 검색에 많은 어려움이 있다. 그래서 검색공간을 줄임으로써 검색 실행 시간을 단축하는 트리 구조가 제안되었으나 최근 연구 결과는 트리 구조가 차원의 저주 문제를 야기시킨다는 것을 지적하고 있다[4,5]. 트리 구조에서는 근접 블록을 찾기 위해 인덱스 구조를 왕래하

는 시간이 과도하게 소요되고 유사 이미지를 찾을때 근접 블록 수가 데이터 차원과 함께 지수적으로 커지기 때문에 성능 은 지수적으로 낮아지고 고차원 데이터일 경우 유사 오브젝트를 디스크에 랜덤하게 분산 배치하여 높은 I/O를 유발하기 때문에 유사 검색을 실행하는데 비효율적인 단점이 있다. 인덱싱 기법과는 별도로 클러스터링 기법이 이미지 추출 방안으로 연구되어 왔다. 데이터 분류의 정확성을 최적화하는 것을 목적으로 하는 클러스터링에 대한 연구는 대량의 이미지 집합이 데이터 공간을 단일하게 점유하지 않는 성질을 고려하여 데이터 집합의 분포 패턴을 확인하고 유사한 이미지끼리 클러스터링 한 후 주어진 질의 이미지와 유사한 오브젝트를 추출함으로써 이미지 데이터 추출 분야에 다양하게 적용되고 있다. 하지만 클러스터링 기법은 일정 규모의 데이터베이스 크기를 벗어나 웹과 같은 거대한 데이터 저장소에서 이미지를 찾아내는 방법으로는 처리 속도의 문제 때문에 적용하기 어렵다. 이렇게 각각 다른 접근 방법으로 이미지 데이터 검색분야를 연구하고 있는 인덱싱과 클러스터링 기법은 다양화 하고 있지만 두 기법의 통합 방법론에 대한 연구는 더딘 상태이다. 따라서 본 논문에서는 이미지 데이터의 다차원 특성을 효과적으로 추출하는 클러스터링과 대량의 데이터를 검색하는 최적의 기법인 인덱싱을 통합하여 디스크에 유사한 이미지의 클러스터를 연속적으로 저장하고 그 이미지의 위치를 알아내는 인덱스를 구축함으로써 효율적으로 유사 검색을 수행하는 새로운 클러스터 인덱싱 구조의 이미지 검색 방법을 제안한다. 이를 위해 이미지 데이터끼리의 클러스터링은 CLIQUE 알고리즘 기반의 구성 방법을 적용하고 이 클러스터의 인덱싱에는 해싱기법을 이용하는 클러스터 인덱싱 구조는 단순하고 간략하게 이미지 유사 검색을 처리하는 장점이 있다.

클러스터 인덱싱 구조는 먼저 디스크 맵을 작은 하위영역으로 나누는 것과 같이 이미지가 저장된 특징 공간을 작은 셀 형태의 하위영역으로 나눠 셀간의 유사한 레코드를 클러스터링하고 인덱싱하는데 이용한다. 클러스터링 단계에서는 각 셀에 위치한 이미지 수를 기록하고 이미지가 위치한 셀과 주변 셀을 같은 클러스터로 통합한다. 디스크 그리드의 해상도를 변경함으로써 클러스터를 다단계로 구축할 수도 있다. 그리드가 더 섬세할수록 셀 크기는 작아지고 클러스터의 결과도 더 상세해 진다. 클러스터를 형성한 후에는 클러스터를 인덱싱하는 사상 테이블을 만든다. 유사도 질의에 답하기 위해 질의 이미지의 특성 벡터를 해쉬하여 셀 ID를 구한 다음 셀 ID를 이용한 사상 테이블로 질의 이미지가 위치한 클러스터를 탐색한다. 유사도 질의에 사용하는 I/O는 클러스터를 찾는 I/O와 연속적 파일 형태의 클러스터를 읽는 I/O이다. 클러스터 인덱싱 구조는 이와 같은 방법으로 웹 또는 데이터베이스에 이미지를 포함하는 클러스터를 형성하여 원하는 이미지를 찾아낸다.

본 논문의 구성은 다음과 같다. 제2장에서는 이미지 데이터 유사 검색 및 클러스터링과 인덱싱 구조에 관한 관련 연구를 기술한다. 제3장에서는 클러스터 인덱싱 구조를 설명하면서 클러스터링 형성 방법과 클러스터 구성시 클러스터의 수와 크기에 영향을 미치는 계수를 제시하고 계수를 조율하는 절차를 설명하며 고차원 공간에서 빠른 검색을 지원하는 동시에 저장공간을 유지하는데 필요한 인덱스를 구축하는 방안을 제시한다. 유사도 검색을 처리하는 방법과 연속적 I/O 적용 방법 역시 제3장에서 제시한다. 제4장에서는 다양한 환경에서의 성능평가 결과를 제시하고 제5장에서는 결론과 향후 연구 계획을 기술한다.

2. 관련 연구

2.1 유사도 검색과 클러스터링 연구

고차원 공간에서 유사도 검색을 구현하는 데에는 최근접 검색 기술이 주로 이용되어 왔다. 그러나 최근 연구에서는 차원의 저주때문에 개략적으로 유사도 검색을 처리하는 방안을 제안하고 있다. White와 Jain[4]은 질의 대상이 위치하는 데이터의 버켓만을 돌려주는 방법을 제안했다. 이 방법은 빠르지만 재현율이 낮다. Kleinberg[6]는 차원을 줄이고 병렬 자원을 사용하여 차원의 저주를 처리하는 방법으로 $O((D\log^2 D)+((N+\log^3 N)$과 $O((N+D\log^3 N)$ 시간 안에 돌아가는 두 가지 근접 알고리즘을 제안했다. 이 방법은 아주 높지 않은 고차원 응용에는 적용할 수 있지만 고차원의 이미지 검색 문제를 해결하기는 어렵다.

또한 이미지 정보 검색을 위한 다양한 내용 기반 이미지 검색 기술이 제안되어 왔다. QBIC[14]은 사용자가 다양한 형태의 특징값을 사용하지만 영상분할에 사용자의 수작업이 필요한 단점이 있다. Netra[7]는 분할된 영상을 이용하여 영상을 검색하지만 실제적으로 분할된 영역을 사용자가 랜덤으로 입력해야 한다. VisualSeek[8]은 색상 외에 질감이나 모양 정보는 사용하지 않고 Virage[9]는 색상, 질감, 위치정보를 이용하지만 한 영상 분할에 너무 많은 시간이 걸린다. 이 시스템들은 전통적인 트리 구조를 이용하여 구현되었으므로 차원의 저주 문제를 야기할 수 있다.

클러스터링 기술은 통계학, 기계학습, 데이터베이스 분야에서 연구되어 왔지만 각각 다른 측면에 중점을 두고 있다. 기계학습에서 자체 조직맵(SOM : Self-Organizing Map)[10]과 k-Means[11]는 비지도 학습에서 가장 보편적인 기술로

개발되었다. SOM은 주어진 크기의 지도를 만들고 k-Means는 k 값(클러스터 수)이 주어진 클러스터를 만든다. 그러나 주어진 데이터 집합에서 최적의 맵 크기와 k 값은 학습하기 어렵다. 결과적으로 이 기술은 데이터 집합의 자연적 클러스터를 호출할 경우 클러스터를 깰수 있다. 예를 들어 자연적 클러스터의 수가 k와 다를 경우에 k 클러스터를 생성하는 알고리즘을 적용하면 만들어진 클러스터는 자연 발생한 클러스터를 깬다. 현재 클러스터링 연구가 데이터베이스 분야에서 다양하게 이루어지고 있으며 이 연구들은 대형 데이터 집합에서 클러스터를 구성하는데 효과가 있다. 그러나 이 방법들은 데이터 검색과 추출을 효과적으로 처리하지 못한다. BIRCH[12]는 클러스터를 형성하는데 트리 구조를 이용한다. 하지만 각 말단은 페이지 크기로 제한되기 때문에 클러스터는 디스크의 한 부분에 랜덤하게 저장될 때 분할되어야 한다. CLARANS[13]는 모든 오브젝트가 주기억 장치 크기에 맞아야 하고 결과는 입력순서에 민감하다. 전단적 모양의 클러스터를 찾기 위해 집적도 기반의 개념을 이용한 DBSCAN[14]은 속도와 확장성 측면에서 R-tree에 기반하여 최근접 검색질의를 수행하지만 거리공간에서 데이터를 클러스터할 수 없다. CURE[15]는 샘플링 기반의 다단계 클러스터링 알고리즘으로 전단적 모양의 클러스터를 찾아내지만 벡터작용에 근거하여 거리공간에서 데이터를 클러스터링 할 수 없다.

2.2 고차원 인덱스 구조

고차원 데이터를 인덱스하는데 지금까지 R-tree[16], R*-tree[17], SR-tree[18], X-tree[19], M-tree[20], VA-file[21]과 같은 많은 트리 유사구조가 제안되었다. 트리 유사구조는 고차원 공간을 디스크 블럭에 저장된 오브젝트의 하위집합을 포함한 하위영역으로 분할한다. 오브젝트를 표현하는 벡터가 주어졌을때 유사도 질의는 다음 절차로 이루어진다.

1) 주어진 벡터가 어느 하위영역에 위치하는지 *where-am-I* 검색을 수행한다.
2) 유사벡터가 위치한 근접영역을 찾는 최근접(*nearestn-eighbor*) 검색을 수행한다. 검색은 검색범위가 중첩된 모든 지역을 찾아가는 범위검색을 이용하여 수행된다.
3) 전 단계에서 구한 근접지역의 벡터와 주어진 벡터간의 유클리디안 거리를 계산한다. 검색 결과는 주어진 벡터로부터 거리 내에 있는 모든 벡터를 포함한다.

유사도 검색의 병목은 1, 2단계에서 발생한다. 첫 번째 단계에서 인덱스 구조가 주기억 장치에 맞지 않고 검색 알고리즘이 부적절하면 인덱스 구조의 많은 부분을 디스크로부터 가져와야 한다. 두 번째 단계에서 근접한 하위영역의 수가 특성벡터의 차원에 대해 지수적으로 커진다. 만약 D가 차원의 수라면 근접한 하위영역의 시간 복잡도는 $O(2^D)$ [22]가 된다. 그래서 인접한 영역의 데이터 읽기는 지수적 I/O를 취한다. 이와 같은 I/O는 랜덤하기 때문에 비용 소모가 크다. 대부분의 인덱스 구조는 하향 방식으로 공간을 동일한 포인트를 가진 하위영역으로 나눈다. 디스크 블록에서 공간을 하위영역을 나누는데 만약 각 하위영역 경계선 근처에 위치한 질의 오브젝트가 주어지면 유사도 질의는 인접한 모든 영역의 오브젝트를 추출한다. 이 경우 인덱스 구조에서 인접 포인트를 찾는데 필요한 하위 영역 수는 데이터 차원과 같이 지수적으로 커진다. 또한 고차원 공간에서 주어진 질의 오브젝트에 대해 하위영역을 연속적 방법으로 처리하지 못하면 I/O는 랜덤하게 이루어져 데이터의 차원이 크면 클수록 주어진 오브젝트의 근접 블록들은 디스크에 랜덤하게 분포한다. 그래서 고차원 공간을 인덱싱하는 트리 유사 구조는 인덱스 구조가 큰 D(차원의 수)와 큰 N(데이터베이스 크기)때문에 메모리에 맞지 않을때 검색 비용이 비싸지며 I/O가 랜덤하고 지수적으로 커지기 때문에 인접하는 오브젝트를 포함하는 블록을 추출하는 비용이 많이 소요된다.

따라서 본 논문에서는 랜덤 위치 신드롬을 피하는 동시에 I/O를 연속적으로 처리하기 위해 같은 실린더나 디스크 트랙에서 유사한 오브젝트끼리 클러스터링하고 클러스터를 해싱하는 인덱스 방법을 적용한다. 이미지 데이터의 특징 정보를 고려하여 유사 오브젝트끼리 클러스터가 형성되면 데이터의 속성 때문에 발생하는 차원의 문제는 클러스터링 되는 과정에서 해결되고 이렇게 형성된 클러스터들을 인덱싱하여 유사도 검색을 수행한다.

3. 클러스터 인덱싱 구조

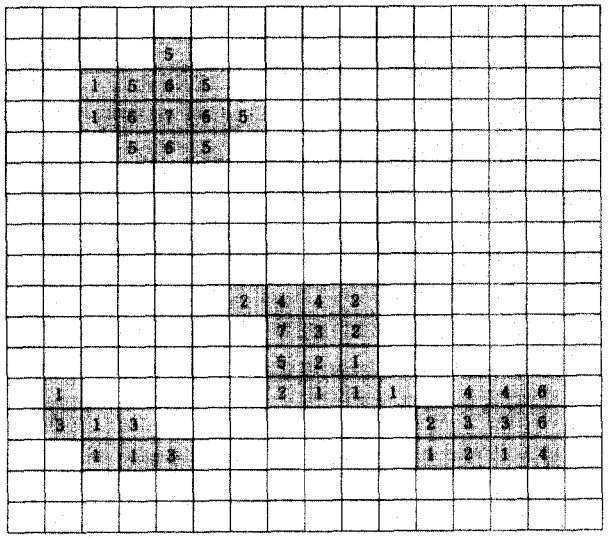
클러스터 인덱싱 구조의 주요 아이디어는 유사 오브젝트를 추출하기 위한 디스크 지연정도를 최소화하도록 디스크에 유사 데이타끼리 클러스터하고 클러스터를 인덱스하는데 해싱을 이용하여 클러스터 탐색 비용을 최소화하는데 있다. 접근 방법은 세 단계로 이루어진다.

1) i번째 차원을 2^k개 영역으로 나누고 최소 비트 수를 사용하여 특성벡터가 속한 셀을 인코딩한다. (그림 1)

은 이 방법으로 x와 y차원을 16개의 같은 크기로 나눈 2D 형태의 그리드에 분포한 오브젝트를 클러스터링한 것이다.

2) 클러스터내로 셀을 통합한다. 셀은 각각 다른 모양의 클러스터를 구축하기 위한 최소 블록으로서 차원을 나눈 하위영역이 작을수록 셀은 더 작아진다. 클러스터는 연속적인 파일형태로 디스크에 저장된다.

4) 클러스터를 참조하는 인덱스 구조를 만든다. 셀은 최소 주소단위이다. 인코딩 스키마는 오브젝트를 셀 ID로 해쉬하고 1회의 I/O에 클러스터를 추출한다.



(그림 1) 클러스터링 생성

3.1 클러스터링 방법

고차원 공간에서 클러스터링을 효율적으로 수행하기 위한 클러스터 구성 알고리즘은 다음 방법으로 이루어진다.

1) 클러스터 구성 알고리즘은 먼저 각 셀의 높이(오브젝트 수)를 기록한다.

2) 클러스터 구성 알고리즘은 포인트 집적도가 가장 높은 셀에서 시작한다. 이 셀은 최초 클러스터의 중심점이다.((그림 1)에서는 7로 표시된 셀에서 시작한다.) 클러스터 구성 알고리즘은 한번에 한 단위씩 내려온다. 셀은 세 가지 조건중의 하나인데 어떤 클러스터에도 인접하지 않거나 오직 한 클러스터에만 인접하거나 한 개 이상의 클러스터에 인접해 있다.

3) 클러스터 구성 알고리즘이 취하는 방법은

a) 만약 셀이 어떤 클러스터에도 인접하지 않으면 셀은 새로운 클러스터를 생성하는 중심이 된다.

b) 셀이 한 개 클러스터에만 인접하면 셀을 클러스터에 조인한다.

c) 셀이 한 개 이상의 클러스터에 인접하면 클러스터 결정 알고리즘은 어떤 클러스터에 셀이 속하는지 또는 셀이 속한 클러스터끼리 통합되는 것을 결정하기 위해 셀 결정 알고리즘을 이용한다.

5) 클러스터 구성 알고리즘은 셀의 높이가 임계값이 되면 끝난다. 어떤 클러스터에도 속하지 않는 셀은 외곽 클러스터에 통합되고 한 개의 연속적 파일로 저장된다. 임계값은 각 클러스터들의 전체 높이를 평균하여 조정한다.

클러스터 구성 알고리즘의 입력은 데이터 차원 D, 각 차원을 인코딩하는 비트 수, 클러스터링 알고리즘을 끝내는 임계값, 데이터 집합이고 출력은 클러스터 집합, 클러스터를 인덱스하는 셀 ID로 정렬한 힙 구조이다. 데이터를 가진 각 셀에 셀 ID, 셀의 포인트 수, 셀이 속하는 클러스터를 할당하고 셀은 힙에 삽입된다.

힙에서의 셀 수를 $|H|$라고 정의하면 $|H|$의 값은 N개의 오브젝트 포인트가 어떻게 클러스터링 되느냐에 달려 있고 데이터 차원 D와는 관련이 없다. 만약 셀이 한 개 이상의 클러스터에 인접해 있으면 알고리즘은 셀이 속할 클러스터를 결정한다. 셀이 인접한 클러스터의 최고점 약간 아래 높이라면 알고리즘은 이 클러스터를 통합한다. 다른 클러스터들의 근접 셀들이 같은 높이라면 오브젝트 수가 작은 클러스터들을 통합한다. 그렇지 않으면 랜덤하게 셀을 인접 클러스터에 연결한다.

3.2 클러스터 인덱싱

클러스터를 형성한 후의 마지막 단계는 클러스터에 빨리 접근할 수 있는 인덱스를 구축하는 것이다. 오브젝트를 클러스터에 사상하기 위해 사상 테이블을 생성하고 클러스터에 대한 정보를 기록하기 위해 클러스터 디렉토리를 구축한다. 사상 테이블의 크기는 힙의 크기에 비례한다. 힙은 각 셀에 대한 정보를 기록한다. 최악의 경우 각 포인트에 대해 셀을 한 개씩 갖게 되어 N개의 셀 구조가 된다. 각 셀 구조는 셀 ID, 포인트 수, 클러스터 ID, 힙 유지를 위한 포인터로 이루어지며 사상 테이블은 셀 ID와 클러스터 ID로 구성된다. 포인트 수, 클러스터 ID, 포인터에 대한 스토리지 소요량은 불변하며 셀 ID의 크기는 데이터 차원 D에 비례한다. 본 논문의 제안 알고리즘은 트리 유사 인덱스 구조와는 달리 검색시 메모리에 셀을 포함하고 있는 블록을 읽을때 외에는 메모리 용량을 크게 필요로 하지 않는다. 테

이블의 디스크 스토리지 소요는 트리 유사 인덱스 구조와 비교 가능하다. 클러스터 디렉토리는 각 클러스터 해당 ID와 클러스터가 오브젝트를 저장한 파일의 이름, 최고 높이의 포인트 셀인 클러스터 중심 셀 ID를 가지고 있다.

3.3 제어계수

클러스터의 입도는 네 가지 계수로 제어한다

- D : 데이터 차원 또는 오브젝트 특징
- b : 각 차원을 인코딩하는데 사용하는 비트의 수
- N : 오브젝트의 수
- θ : 수평계수

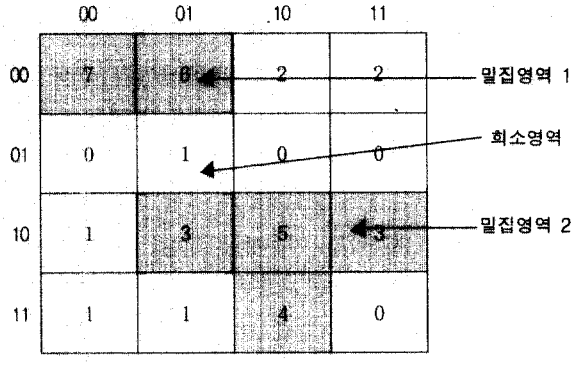
셀의 수는 D, b 계수로 결정되며 $2^{D\times b}$로 쓴다. 셀의 평균 포인트 수는 $2^{(N/(D\times b))}$ 이다. 두 인자는 원하는 포인트 밀도를 결정한다. 낮은 포인트 밀도는 많은 수의 셀을 만들어 내면서 많은 수의 작은 클러스터들을 생성하기 때문에 좋지 않다. 반면 높은 포인트 밀도는 클러스터 수는 적지만 대형 크기의 클러스터를 만들어 내기 때문에 역시 좋지 않다. b 값의 변화는 그리드의 해상도에 영향을 미치고 클러스터의 수와 크기도 결정한다. θ 값 역시 클러스터의 크기와 수에 영향을 미친다. 임계값이 낮아지면 포인트 밀집 영역의 수와 크기는 증가하는 대신 포인트가 없는 희소영역의 크기는 감소한다. 희소영역이 감소하면 I/O의 효율성을 높일 수 있지만 밀집영역의 수와 크기가 증가해 포인트 밀집영역 클러스터들의 경계선이 불분명해 진다. 그래서 적절한 데이터 분포와 원하는 클러스터 크기를 고려하여 b 와 θ 의 값을 조절해야 한다.

3.4 알고리즘의 시간 복잡도

N은 데이터 집합에서 오브젝트 수이고 M은 오브젝트가 할당된 셀의 수이다. 클러스터 형성 알고리즘의 시간 복잡도는 오브젝트의 셀 ID를 계산하는데 $O(D)$ 가 걸리고 두 개의 셀이 인접하는지 확인하는데가 $O(D)$ 걸릴때 첫 번째 단계에서 셀 ID와 높이를 유지하는데 힙을 이용하며, 셀 ID가 주어지면 셀이 힙에 위치하는지 확인하는 시간복잡도는 $O(D\times \log M)$ 이다 그래서 클러스터 형성 알고리즘의 첫 번째 단계의 시간 복잡도는 $O(N\times D\times \log M)$ 이다. 두 번째 단계에서 인접한 셀을 찾는 시간은 각 차원에서 주어진 셀은 적어도 세개의 하위영역과 인접하기 때문에 $O(\min(3^D, M))$ 이다. 두개 고차원 공간에서 $M \ll 3^D$ 이기 때문에 두 번째 단계의 시간 복잡도는 $O(D\times M^2)$ 이다. 그래서 총체적인 시간 복잡도는 $O(N\times D\times \log M) + O(D\times M^2)$ 가 된다.

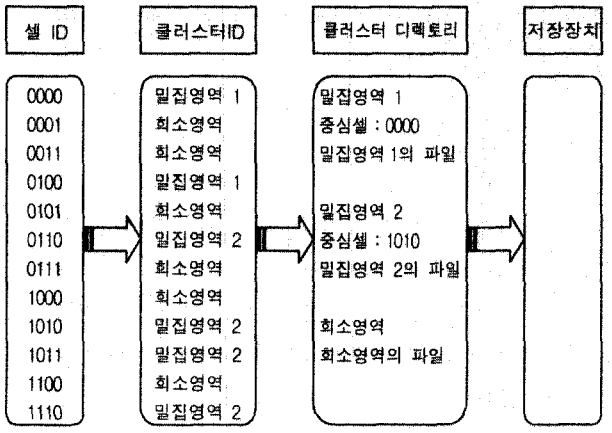
3.5 유사도 검색

I/O의 효율성을 개선하기 위해 가장 촘촘하게 입도된 밀집 클러스터를 개별적인 연속적 파일로 저장하고 성글게 입도된 희소 클러스터를 촘촘한 클러스터와 같은 실린더 그룹에 저장한다. 질의 오브젝트가 주어지면 먼저 사상 테이블을 이용하여 오브젝트를 클러스터에 사상한다. (그림 2)는 2D 예를 보여준다. 각 셀은 x 축으로부터 시작해 y 축으로 가는 이진코드로 표현한다.



(그림 2) 클러스터 예

예를 들어 셀 0001은 x 축의 좌측 첫째 부분과 y 축의 위로부터 두 번째 부분의 결합이다. (그림 3)은 인덱스 구조 예로 그림의 셀 ID와 클러스터 ID는 셀을 클러스터에 사상하는 테이블이다. 클러스터를 찾은 다음 클러스터가 저장된 연속적 파일을 찾기 위해 클러스터 디렉토리를 이용한다. 이 예에서 θ 값은 3이다. 3 이하의 셀은 희소 영역으로 클러스터링한다. 여기서 두개의 밀집영역과 한개의 희소 영역으로 이루어진 세개 클러스터가 만들어진다. 밀집영역 1은 0000과 0100 셀로 구성되고 밀집영역 2는 0110, 1010,1110, 1011 셀로 구성된다. 클러스터 디렉토리는 세 종류의 클러스터에 의해 생성된다. 밀집영역 1의중심과 2의중심은 0000과 1010이다. 클러스터를 저장하고 있는 파일의 이름 역시 디렉토리에 기록된다. 질의 이미지가 주어지면 먼저 x 와 y 속성을 4비트 코드로 양자화하고 그 다음 오브젝트가 위치한 클러스터를 사상 테이블에서 찾는다. 만약 질의 대상이 밀집영역에 있으면 디스크에서 순차적으로 밀집영역에 속하는 디스크 블록을 읽고 거리함수를 이용하여 포인트를 순위 매김한 후 그 결과를 보여준다. 예를 들어 질의 오브젝트가 0000 셀에 해쉬되면 밀집영역 1의 13개 오브젝트를 정렬순서로 찾아 보여준다.



(그림 3) 인덱스 예

3.5.1 질의 정제

유사도 질의에서 질의 정제 기능은 중요하다. 최초 검색 단계에서 원하는 데이터를 찾지 못한 이용자를 위해 질의는 양방향이어야 한다. 만약 질의 오브젝트가 밀집 영역 클러스터에 속한다면 해당 밀집 영역을 보여 주는 것이 이용자에게 도움이 된다. 질의 오브젝트가 어떤 밀집 영역 클러스터에도 속하지 않는다면 이용자는 원하는 결과를 찾기 위한 시도를 계속한다. 질의 오브젝트가 밀집 영역에 있지 않으면 희소 영역에 있다. (그림 2)에서 0001 셀에 속한 질의 오브젝트는 희소 영역에 있다. 유클리디안 거리 계산 함수들이 희소 영역의 셀에서 오브젝트를 찾는 동시에 타 클러스터의 중심 셀을 구해 밀집 영역 근처에서도 질의 이미지와 유사한 오브젝트를 찾는다. 예를 들어 질의 오브젝트가 0001 셀에 있으면 희소 영역의 모든 포인트와 밀집지역 1의 셀 0000, 밀집 지역 2의 1010 셀의 포인트를 이용자에게 보여 주는 것이다. 그리고 이용자가 제시된 밀집 영역 1에 위치한 오브젝트를 선택하면 연속적인 질의 정제는 밀집 지역 1에 속한 모든 포인트를 이용자에게 보여준다.

3.5.2 I/O 분석

이용자가 첫 번째 시도에서 원하는 결과를 찾을 수 있는 가능성을 p라고 하고 마지막 시도한 클러스터에서 읽는 I/O 시간을 T_{read}, 질의 정제 I/O시간을 T_{refine}라고 하면 검색 시간 T_{search}은 식 (1)로 모델링한다.

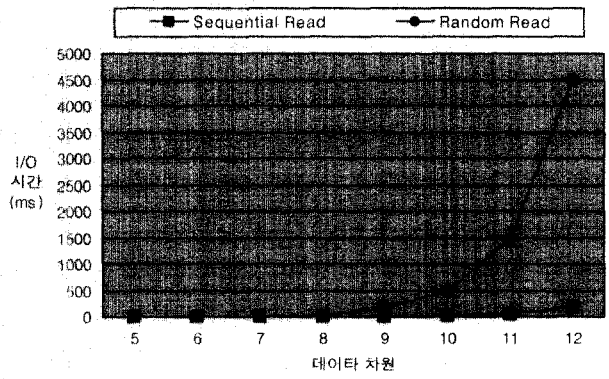
$$T_{search} = (1-p)\,T_{refine} + T_{read} \qquad (1)$$

T_{search}를 최소화하기 위해서는 I/O시간을 최소화하고 p를 최대화한다. p값은 질의 오브젝트가 속한 클러스터를 정확하게 찾는 가능성이다. 검색비용은 한 개 연속적 파일 I/O 비용과 비슷하며 클러스터의 평균 크기에 달려 있다.

3.6 연속적 I/O 대 랜덤 I/O

트리 구조에서는 근접 블록을 찾기 위해 인덱스 구조를 왕래하는 시간이 과도하게 소요되고 유사 이미지를 찾을때 근접 블록 수가 데이터 차원과 함께 지수적으로 커지기 때문에 성능은 지수적으로 낮아지고 고차원 데이터일 경우 유사 오브젝트를 디스크에 랜덤하게 분산 배치하여 높은 I/O를 유발하기 때문에 유사 검색을 실행하는데 비효율적이다. 따라서 본 연구의 클러스터 인덱싱 구조는 디스크에서 클러스터 파일을 찾아 읽을 때 연속적 I/O방법을 취함으로써 전통적 트리 유사구조보다 많은 데이터를 읽기 때문에 데이터의 정확율은 떨어지지만 재현율은 높다. 데이터 검색시 I/O가 메모리 거리 계산보다 빈번하게 병목효과를 일으키므로 연속적 방법으로 한번에 많은 데이터를 읽어오는 것이 랜덤하게 데이터를 읽어 빈번하게 I/O를 행해야 하는 랜덤 I/O 방법보다 효과가 높은 것이다.

(그림 4)는 데이터 차원 D의 값이 다를때 연속적 I/O 대 랜덤 I/O의 효과를 비교한 내용이다.



(그림 4) 연속적 I/O와 랜덤 I/O 시간

(그림 4)의 x 축은 5부터 12까지의 데이터 차원 수이며 y 축은 3^D 블록수 검색에 필요한 IO시간이다. 랜덤 I/O가 연속적 I/O보다 더 비용이 높다는 것이 명백하다. 클러스터 인덱싱 방법이 취한 연속적 파일 I/O는 한번에 많은 데이터를 읽어냄으로써 질의 오브젝트에 가깝지 않은 포인트를 포함하기 때문에 정확율은 낮지만 높은 차원의 멀티미디어 데이터를 검색하는데 전통적 인덱스 구조보다 빠른 I/O 시간을 보인다.

4. 실 험

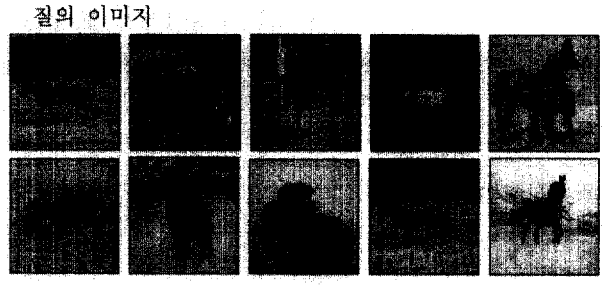
실험에서는 주어진 질의에 대해 가장 유사한 k개 오브젝트를 찾아내거나 k최근접 오브젝트를 찾아낸다. 최근접 데이터는 10개로 하여 먼저 데이터 집합을 스캔한다. 검색의

효율성에 있어서 검색 범위를 넓게 해야 하는 경우에는 재현율에 중점을 두어야 하고 검색을 깊게 해야 하는 경우에는 정확율에 중점을 두어야 하므로 본 논문에서는 넓은 범위의 이미지 유사도 검색을 위해 재현율과 재현된 오브젝트 중 최근접 오브젝트를 찾아내는 R-정확율로 질의 결과를 확인한다.

- X번의 I/O 후 재현율 : 전체 오브젝트 대비 X번의 I/O가 이루어진 후 추출된 오브젝트
- X번의 I/O 후 정확율 : X번의 I/O가 이루어진 후 추출된 오브젝트 대비 적합한 오브젝트
- R-정확율 : 추출된 오브젝트 대비 찾아낸 최근접 데이터 수

평가를 수행하기 위해 3,000개의 이미지 데이터베이스를 구축했다. 검색을 수행하기 위해 세 단계를 거쳐 이미지의 특징벡터를 추출한다. 이미지 특징벡터 추출과정은 내용기반 이미지 검색시스템에서 주로 적용하는 Daubeches wavelet transform[23]을 적용한다. 먼저 정규 크기와 형식으로 이미지를 바꾼다. 그리고 이미지에 선처리과정으로 Daubeches wavelet transform을 적용하여 계수를 추출한다. 마지막으로 이미지의 특징벡터로 선택한 HSV 명도, 채도, 색도의 세가지 색상 특징치, 모멘트, 대비, 연관성, 분산, 혼잡도의 5가지 질감 특징치, 영역 크기 정보, 푸리에 기술자를 이용한 모양 및 위치정보 특징치를 특징공간에 48 차원의 웨이블릿 계수로 저장한다. 그런 다음 데이터 집합을 적절하게 클러스터링하기 위해 클러스터링 알고리즘을 적용하고 데이터 집합에서 패턴을 찾도록 d와 θ계수를 조정하여 d를 2로 하고 θ를 1로 하면 평균 16개 이미지 수를 갖는 25개 클러스터를 찾는다. 대부분 유사 이미지는 같은 클러스터에 위치한다. 질의 이미지가 클러스터 중심에 있을 때는 검색이 잘되지만 포인트 밀집 지역의 주변이나 희소지역에 있을 때는 잘 감지되지 않는다. 이 경우에는 질의 이미지의 주변 포인트들을 근접 클러스터로 별도 분류해 검색한다.

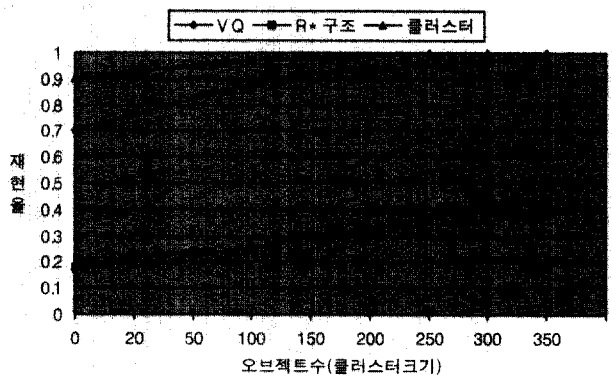
제안 구조의 성능 평가를 위해 클러스터 인덱싱 구조와 VQ(Vector Quantization) 알고리즘[24, 25], R*-tree[17, 26] 구조를 비교했다. 클러스터링과 인덱싱 구조를 동시에 적용한 사례가 많지 않아 클러스터링 구조 및 인덱싱 구조와 직접 비교했다. VQ는 고차원 공간에서 데이터를 클러스터링 하기 위해 구현된 k-Means 알고리즘을 적용한 방법이며 R*-tree 구조는 고차원 공간에서 유사도 검색을 위해 사용되는 대표적인 인덱스 구조이다. R*-tree 구조는 재현율을 측정하기 위해 교차 유효 기법을 적용하여 10회의 테스트를 했다. 각 회에서 30개의 테스트 이미지를 이용하여 제안 알고리즘과 VQ로 데이터베이스를 검색하였다. 30개 이미지 질의의 재현율로 매회 평균 질의 재현율을 구한 다음 구한 재현율을 평균했다. 재현율은 클러스터링 알고리즘과 클러스터 크기를 고려했다. (그림 5)는 상단 왼쪽의 질의 이미지를 주었을 때 나타나는 검색 결과의 예이다.



(그림 5) 질의 이미지 검색 결과

4.1 재현율

실험을 통해 (그림 6)과 같은 결과를 볼 수 있다. 25개의 클러스터로 나뉜 데이터 집합에서 5회의 I/O에 클러스터 인덱싱 구조는 약 84%의 재현율에 해당하는 오브젝트를 검색했다. 5회를 더 테스트한 후 재현율은 98%까지 높아졌다. 클러스터를 읽으면 읽을수록 재현율은 높아지지만 처리 속도는 느려진다. 15개의 클러스터를 읽은 후 재현율은 100%에 가까워지면서 10개의 최고 유사 오브젝트를 찾는데 데이터 집합에서 16%(4/25)의 데이터를 읽는다. VQ는 5회 I/O에 69%의 재현율을 보이고 I/O 5회를 더 시도했을 때 90% 정도의 재현율을 보였다. R*-tree구조는 5회의 I/O에 18%의 재현율을 보였으며 5회의 I/O를 더 시도한 결과 43%의 재현율을 보였다.



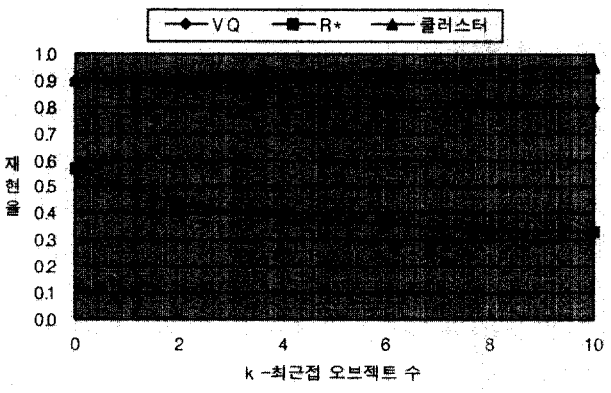
(그림 6) 클러스터 크기 대 재현율/5회 I/O

클러스터 인덱싱 구조는 VQ 및 R*-tree 보다 높은 재현

율을 보이며 클러스터의 모양과 크기에 대해 더 유연하다. VQ는 선형함수로 클러스터를 바운드하는 반면 클러스터 인덱싱은 여기에 제약을 받지 않는다. R*-tree는 인덱스 구조이므로 선처리 단계에서 클러스터를 형성하지 않으면 클러스터링의 영향을 받지 않으므로 재현된 오브젝트수로 결과를 나타냈다.

4.2 클러스터 크기에 따른 재현율

평균 오브젝트 수를 클러스터 크기로 규정하고 재현율에 클러스터 크기가 어떤 영향을 미치는지 보기 위해 각각 다른 클러스터 크기로 재현율 값을 측정했다. 클러스터 크기는 50~400개 오브젝트로 규정하여 실험했다. (그림 6)은 5회의 I/O를 행한 후 다른 클러스터 크기를 적용했을 때의 재현율이다. 클러스터 인덱싱 구조가 VQ 및 R* -tree보다 높은 재현율을 보인다.



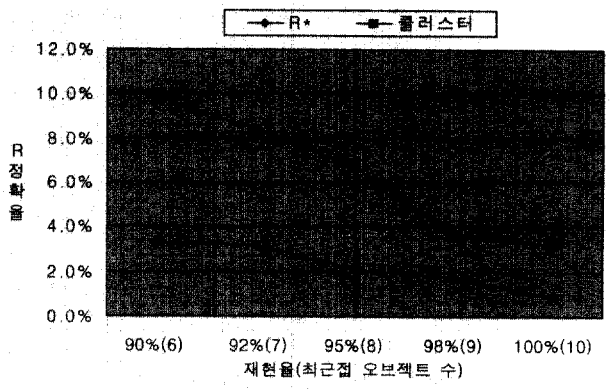
(그림 7) k-최근접 오브젝트 대 재현율

4.3 k 최근접 오브젝트 수에 따른 재현율

최고 10개 최근접 오브젝트를 검색하는데 5회의 I/O를 실행한 결과 (그림 7)의 x 축은 요청한 최근접 오브젝트의 수이고 y는 읽은 횟수이다. 클러스터 인덱싱 구조가 5회의 I/O를 행했을때 95%의 재현율을 보인다.

4.4 최근접 오브젝트 수에 따른 R-정확율

(그림 8)은 찾아낸 최근접 오브젝트 대비 추출 오브젝트 수로 계산한 R-정확율를 보여준다. 클러스터 인덱싱 구조가 R* 구조 보다 높은 R-정확율을 보인다. 제안 알고리즘은 재현율을 높이는데 중점을 두고 있으므로 4.4절의 실험에서는 R-정확율의 실험 결과치가 높지 않으나제안 알고리즘과 기존 알고리즘과의 비교를 설명하기 위해 제시한다. 제안 알고리즘의 정확율을 높이는 방안은 향후 연구에서 수행한다.



(그림 8) 10개 최근접 오브젝트에 대한 R-정확율

5. 결 론

본 논문에서는 고차원 이미지 데이터 유사 검색에 개별적으로 적용되어 온 인덱싱 구조와 클러스터링 기법을 통합한 클러스터링 인덱싱 구조를 제안했다. 데이터 검색에 유효한 인덱싱 구조와 이미지 추출에 효과가 높은 클러스터링 기법을 통합한 제안 구조는 고차원 공간에서 차원의 저주 문제를 피하면서 유사도 검색을 수행하는 새로운 방법이다. 이 방법은 클러스터링 구성 알고리즘을 통해 유사한 오브젝트들을 클러스터링하여 디스크에 저장하고 디스크 셀 ID와 클러스터 ID로 클러스터 디렉토리를 구성하여 해당 클러스터를 찾아가는 인덱스 구조로 이미지 데이터 유사검색을 수행하고 있다. 질의 이미지 오브젝트가 주어지면 클러스터 인덱스를 이용하여 해당 오브젝트 근처의 클러스터를 찾음으로써 이미지 데이터의 유사 검색이 이루어지며 이때 디스크로부터 관련 정보를 연속적으로 추출하기 때문에 기존 인덱스구조의 랜덤 I/O 시간보다 I/O 시간이 줄어든다. 또한 오브젝트를 클러스터링하여 디스크 저장시에는 밀집 데이터 포인트 영역과 희소 데이터 포인트 영역의 클러스터를 같이 저장하여 디스크의 이용 효율성을 높인다. 고차원 공간에서 데이터를 클러스터링 하는 VQ 및 다차원 인덱스 구조인 R*-tree와의 비교 실험에서 본 논문의 클러스터 인덱싱 구조가 VQ보다 클러스터의 크기와 모양에 대해 더 유연하며 R*-tree보다 재현율이 더 높은 결과를 보였다. 전통적 트리 유사구조의 검색방법보다 많은 데이터를 읽어 내기 때문에 데이터 재현율이 높음을 알 수 있다. 제안 클러스터 인덱싱 구조는 클러스터의 중심 정보를 이용하여 인접 영역을 효과적으로 검색하기 때문에 패턴을 클러스터링하는 고차원 데이터 집합에 적용 가능하다. 질의 오브젝트와 연관된 클러스터를 연속적으로 검색해 감으로써 질의 내용을 계속 바꿔가며 검색하는 이용자의 피

드백을 포함하는 방법도 제공 가능하다. 향후 연구에서는 이미지 유사 검색의 중요한 측면인 정확율을 높이는 방안을 계속 진행해 나가야 하며 본 논문의 제안 구조와 현재 가장 효과적인 다차원 색인기법인 X-tree, VA-file 구조의 비교를 통해 제안 알고리즘의 신뢰성을 높이는 것도 필요하다.

참 고 문 헌

[1] P. Aigrain, H. Zang and D. Petkovic, "Content based representaiton and retrieval of visual media : A State-of-the-Art Review," Multimedia Tools and Applications, Vol.3, pp. 179-202, 1996.

[2] James Z. Wang and Jia Li, Gio Wiederhold, "SIMPLIcity : Semantics-sensitive Integrated Matching for Picture LIbraries," IEEE Trans. PAMI, Vol.23, No.9, pp.947-963, 2001.

[3] M. Carson, S. Thomas, J. Belongie, M. Hellerstein, and J. Malik, "Blobworld : A system for region-based image indexing and retrieval," In Proceeding of International Conference Visual Information System, 1999.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, and et al, "Query by image and video content : the QBIC system," IEEE Computer, Vol.28(9), pp.23-32, 1995.

[5] R. Weber, H. Schek and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," In Proceeding of the 24th VLDB, pp.194-205, 1998.

[6] J. M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," Proceeding of 29th Symposium on Theory of Computing, 1997.

[7] W. Y. Ma and B. S. Manjunath, "Netra : A toolbox for navigating large image database," IEEE International Conference on Image Processing, 1997.

[8] J. R. Smith and S. F. Chang, "VisualSeek : A fully automated content-based image query system," Multimedia, Boston, 1996.

[9] A. Hampapur, A. Gupta, B. Horowitz, C. Fuller, J. R. Bach M. Gorkani, and R. C. Jain, "Virage : virage video engine," In Proceeding of SPIE, Vol.30(22), pp.188-198, Februry, 1997.

[10] T. Kohonen, "Sel-Organizing Maps Springer," Berlin, Heidelberg(2nd extended edition), 1997.

[11] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," Applied Statistics 28, pp.100-108, 1979.

[12] T. Zhang, R. Ramakrishnan and M. Liny, "Birch : An efficient data clustering method for very large databases," Proceeding of SIGMOD, June, 1996.

[13] J. Han and M. Kamber, "Data mining, Concepts & Techniques," Morgan Kaufman, 2001.

[14] M. Ester, H. P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," Proceeding of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining, August, 1996.

[15] S. Guha, R. Rastogi and K. Shim, "CURE : An efficient clustering algorithm for large databases," In Proceeding of SIGMOD98, 1998.

[16] A. Guttman, "R-tree : A dynamic index structure for spatial searching," In Proceeding of SIGMOD, June, 1984.

[17] N. Beckmann, H. P. Kriegel, R. Schneider and B. Seeger, "The R*-tree : An efficient and robust access method for points and rectangles," In Proceeding of SIGMOD, May, 1990.

[18] N. Katayama and S. Sotoh, "The SR-tree : An index structure for high-dimensional nearest neighbor queries," In Proceeding of SIGMOD, May, 1997.

[19] S. Berchtold, "The X-tree : An index structure for high-demensional data," Proceedings of the 22nd VLDB, August, 1996.

[20] P. Ciaccia, M. Patella and P. Zezula, "M-tree : An efficient access method for similarity search in metric spaces," Proceedings of the 23rd VLDB, August, 1997.

[21] R. Weber and S. Blott, "A Approximation-based Data Structure for Similarity Search," Technical Report, No.24, ESPRIT project HERMES, No.9141, Oct., 1997.

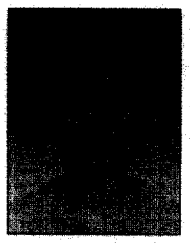
[22] J. Ullman, H. Garcia-Molina and J. Widom, "Database system principles lecture notes," 1998.

[23] 김진옥, 황대준, "클러스터인덱싱을 이용한 이미지 데이타 검색연구", 정보처리학회 2002 춘계학술대회논문집(상), 제9권 제1호, pp.97-100, 2002.

[24] A. Gersho and R. Gray, "Vector quantization and signal compression," Kluwer Academic, 1991.

[25] 장동식, 정세환, 유헌우, 손용준, "VQ를 이용한 영상의 객체 특징 추출과 이를 이용한 내용기반 영상 검색", 정보과학회논문지 : 컴퓨팅의 실제, Vol.7(6), pp.724-732, 2001.

[26] E. Chang, C. Li, J. Wang, P. Mork and G. Wiederhold, "Searching Near-Replicas of Images via Clustering," Proceedings of SPIE Symposium of Voice, Video, and Data Communications, Boston, pp.281-92, September, 1999.



김 진 욱

e-mail : jinny@ece.skku.ac.kr

1989년 성균관대학교(문학사)

1998년 성균관대학교 대학원 정보통신공학과(공학석사)

1992년~1994년 (주)현대전자산업 정보통신사업본부

1994년~1999년 (주)현대정보기술 인터넷사업본부 과장

1999년~2000년 (주)온세통신 온라인사업 팀장

2000년~2001년 (주)유로코넷 기술담당 이사

2002년 성균관대학교 전기전자 및 컴퓨터공학부(공학박사)

현재 성균관대학교 정보통신공학부 박사후 과정

관심분야 : Multimedia, Image Processing, Biometrics, Data mining, Recognition

황 대 준

e-mail : djhwang@skku.ac.kr

1978년 경북대학교 컴퓨터공학과(공학사)

1981년 서울대학교 컴퓨터과학과(이학석사)

1986년 서울대학교 컴퓨터과학과(이학박사)

1981년~1987년 한남대학교 전자계산학과 교수

1990년~1991년 미국 MIT 컴퓨터과학연구소 연구교수

1987년~현재 성균관대학교 전기전자 및 컴퓨터공학부 교수

관심분야 : 멀티미디어, 원격교육, 병렬처리, 가상교육, 지적재산권 보호 시스템