고차원 멀티미디어 데이터 검색을 위한 벡터 근사 비트맵 색인 방법

황지익○, 손대온, 낭종호
서강대학교 컴퓨터학과
{ziegh○, maxson}@mlneptune.sogang.ac.kr, jhnang@ccs.sogang.ac.kr

Vector Approximation Bitmap Indexing Method for High Dimensional Multimedia Database

Jeeik Hwang○, Daeon Son, Jongho Nang
Department of Computer Science, Sogang University

요 약

기존의 다차원 색인 기법들이 고차원의 특징 벡터를 갖는 멀티미디어 컨텐츠 검색 분야에서 만족할 만한 성능을 보이지 못하므로, 이를 해결하기 위해 VA-File, LPC-File 등의 벡터 근사 방법이 개발 되었다. 이러한 방법들은 데이터의 접근에 소요되는 시간이 전체 검색시간의 대부분을 차지하는 경우에 효과적으로 사용될 수 있다. 그러나 고차원의 멀티미디어 데이터 검색에서 객체 간의 거리 계산 시간은 데이터 접근 시간에 비해 무시할 만큼 작지 않으므로 이 방법들을 그대로 적용하기는 어렵다. 본 논문에서는 객체 간의 거리 계산 시간을 줄이기 위한 새로운 색인 기법을 제안하고 실험을 통해 이 방법이 기존의 방법들에 비해 우수한 검색 성능을 가진다는 것을 보인다.

1. 서 론

멀티미디어 콘텐츠 검색과 같이 고차원 특징 벡터를 사용하는 응용에서는 기존의 다차원 색인 방법이 효과적인 해결책이 되지 못하므로 차원의 저주를 극복하면서도 정확한 k개의 최근접 이웃들을 찾는 벡터 근사 접근법이 제안되었다. 이러한 방법들은 객체의 특징 벡터를 근사값으로 표현하고, 검색 시에 이를 이용하여 객체를 필터링 한 후, 남은 검색 후보 객체들에 대해서만 실제 특징 벡터를 이용하여 질의와 비교한다. 즉, 특징 벡터를 읽기 위한 디스크 접근 회수를 최소화 하여 검색 속도를 향상시킨다. 그러나 디스크의 속도가 빠른 환경에서는 필터링을 위한 계산 시간이 입출력 시간만큼 커질 수 있으므로 이러한 방법들을 적용하기 어렵다. 이 경우에는 계산 시간을 줄임으로써 검색 성능을 향상시킬 수 있다.

본 논문에서는 계산 시간을 줄이기 위해 벡터 근사 접근 방법에 기반을 둔 새로운 비트맵 색인 방법을 제안한다. 이 방법에서 각 객체는 그 객체가 존재하는 특징 벡터 공간상의 위치에 대응하는 비트 패턴으로 표현되며, 이 비트 패턴에 XOR 연산을 적용하여 기존의 방법들보다 더욱 빠르게 객체 간의 근사 거리를 구할 수 있다.

본 논문의 구성은 다음과 같다. 2장의 관련 연구에서는 기존의 고차원 데이터 검색 방법에 대해 설명한다. 3장에서는 벡터 근사 비트맵 색인을 사용한 새로운 검색 방법을 제안하고, 4장에서는 실험을 통해 기존의 방법들과의 성능을 비교 분석하며, 마지막으로 5장에서는 제안한 방법의 장단점과 향후 연구 방향에 대해 기술한다.

2. 관련 연구

고차원의 특징 벡터를 갖는 데이터의 처리 기법은 차원 축소 접근법, 근사 최대 근접 이웃 접근법, 벡터 근사 접근법 등의 세 가지로 분류할 수 있다.

차원 축소 접근법[1]은 전체 데이터에 대하여 특이치 분해, 이산 코사인 변환, 웨이블릿 변환 등을 적용하여 데이터에 포함된 정보를 몇 개의 차원으로 압축한 후, 압축된 차원의 데이터를 다차원 색인 기법에 이용한다. 이러한 방법들은 높은 계산 비용과 차원의 축소로 인한 검색 결과의 정확도 감소 등의 문제점을 가진다.

근사 최대 근접 이웃 접근법[2]은 정확한 k개의 근접 이웃을 찾는 대신, 주어진 오차한계 내에서 k개의 근사 근접 이웃을 찾는 방법으로 차원 축소 접근법과 마찬가지로 검색 결과의 정확도가 감소하는 문제점을 가진다.

벡터 근사 접근법은 필터링에 기반한 방법으로, 특징 벡터들을 근사 값으로 표현하고, 검색 시에 이들을 이용하여 필터링 한 후, 남은 일부 객체에 대해서만 실제 특징 벡터를 디스크에서 읽도록 한다. 따라서 디스크 입출력 시간이 크게 줄어 전체 검색 시간이 단축된다.

벡터 근사 접근법에 기반한 기존의 방법은 VA-File[3], LPC-File[4] 등이 있으며 이와 유사한 GC-Tree[5]는 LPC^{+}-File을 Tree구조에 접목시킨 방법이다.

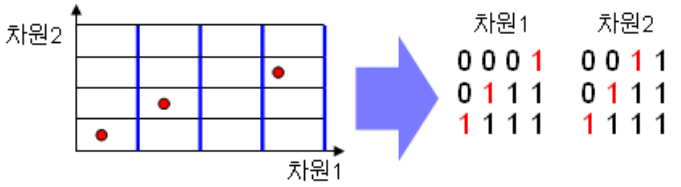
3. 벡터 근사 비트맵 색인을 사용한 검색

벡터 근사 접근법은 객체 사이의 거리 계산 시간이 디

스크 입출력 시간에 비해 매우 작다고 가정한다. 그러나 거리 계산 시간은 벡터 근사의 정확도가 높고 공간 분할 방식이 복잡할수록 증가 하며 고차원의 특징 벡터를 갖는 멀티미디어 데이터의 검색에서는 무시할 수 있을 만큼 작지 않다. 따라서 본 논문에서는 기존의 방법이 간과하고 있는 계산 시간을 크게 줄임으로서 전체 검색 속도를 향상시키는 방법을 제안한다.

3.1. 비트맵 색인 구조

기존의 벡터 근사 접근법은 색인 파일을 사용하여 근사 거리를 계산하는데 큰 오버헤드가 발생한다. 이는 데이터베이스의 모든 객체에 대해 색인 파일의 비트열을 실제 벡터 값으로 변환하고 그 값을 이용하여 거리를 계산하기 때문이다. 본 논문에서는 거리 계산의 오버헤드를 줄이기 위해 색인 파일의 비트열을 실제 벡터로 변환하는 과정 없이 거리를 계산할 수 있도록 새로운 비트맵 색인 방법을 제안한다. 비트맵 색인을 구성하기 위해 <그림 1>과 같이 데이터 공간의 각 차원을 b개의 구간으로 균등하게 분할하고, 각 구간을 b개의 비트로 표현한다. 각 비트는 기본적으로 0의 값을 가지며 객체가 존재하는 구간에 해당하는 비트와 그 오른쪽에 위치한 비트들은 1의 값을 가진다.

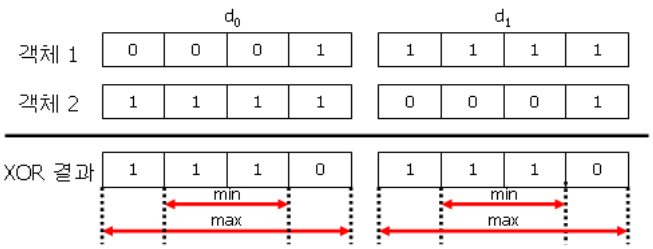


<그림 1. 데이터 객체에 대한 각 차원의 인덱스 생성>

3.2. 근사 거리 예측 방법

벡터 근사 방법에서는 색인 파일만을 이용하여 질의 객체와 데이터베이스의 모든 객체 사이의 근사 거리를 계산하여 최근접 이웃의 후보 객체들을 가려내야 한다. 빠른 속도의 근사 거리를 계산하기 위해 질의 객체의 비트열을 생성한 후 데이터베이스의 모든 객체의 비트열과 XOR 연산을 수행한다. 객체간의 거리 측정 기준이 L_1 거리 측정 방법이고 XOR 연산 결과의 1의 개수를 k, 데이터 객체의 차원을 N, 한 구간 크기를 c 라고 할 경우 두 객체 사이의 거리는 다음과 같이 계산된다.

$$d_{\max} = (k+1) \cdot N \cdot c$$
$$d_{\min} = (k-1) \cdot N \cdot c$$



<그림 2. 객체의 비트맵을 이용한 근사 거리 계산>

3.3. k개의 최대 근접 이웃 검색 알고리즘

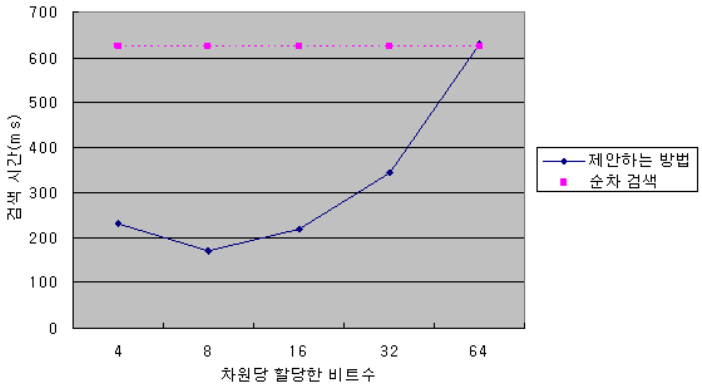
최근접 이웃의 검색은 두 단계로 이루어진다. 첫 번째 단계에서, 모든 벡터 근사를 순차적으로 검사하고, 질의 벡터로부터 각 셀까지의 거리의 하한 경계 d_{min}과 상한 경계 d_{max}를 계산한다. 만약 어떤 근사(셀)에 대해, 그것의 d_{min}이 지금까지 발견된 k 번째 d_{max}를 초과한다면 그 벡터를 제거할 수 있다. 이 단계의 끝에서, 최소 경계를 갖는 벡터들이 질의 벡터에 대한 k개의 최근접 이웃에 대한 후보로 결정된다. 두 번째 단계에서는 d_{min}의 오름차순으로 실제 벡터를 읽어서 후보 집합을 걸러낸다. 이 단계에서 후보들 중에서 그것의 d_{min}이 k번째 최근접 이웃까지의 거리와 같거나 초과하는 근사를 만나면 검색을 마칠 수 있고, 지금까지의 k개의 최근접 이웃이 검색 결과가 된다.

4. 실험 및 결과 분석

이 장에서는 제안한 벡터 근사 비트맵 색인을 이용한 고차원 멀티미디어 데이터의 검색 성능을 측정하고 그 결과를 기존의 벡터 근사 방법들과 비교하고 특징을 분석한다.

4.1. 벡터 근사 정도에 따른 성능 평가

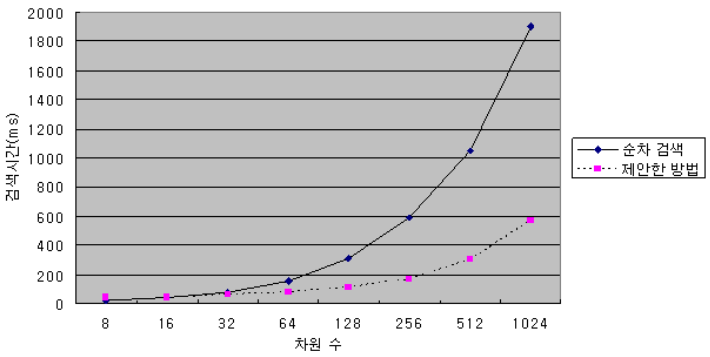
벡터 근사 정도란 색인 파일이 실제 데이터의 정보를 얼마나 정확하게 표현하는 가를 말한다. 제안한 알고리즘의 벡터 근사 정도는 객체의 한 차원을 몇 비트로 나타내는 가에 따라 달라진다. 색인 파일의 근사 정도가 높아지면 그 만큼 필터 아웃시킬 수 있는 객체의 개수도 커질 수 있지만 색인 파일의 크기가 커지기 때문에 적절한 크기의 비트 할당이 필요하다. <그림 3>은 각 차원당 할당된 비트 수에 따른 알고리즘의 성능을 나타낸다.



<그림 3. 벡터 근사 정도에 따른 검색 시간>

위 그래프로부터, 각 차원 당 8 비트를 할당할 경우 가장 성능이 좋은 것을 알 수 있고 순차 검색보다 3배 정도의 성능향상이 있는 것을 확인할 수 있다.

4.2. 차원 및 데이터 개수에 따른 성능 평가



<그림 4. 차원의 변화에 따른 검색 시간>

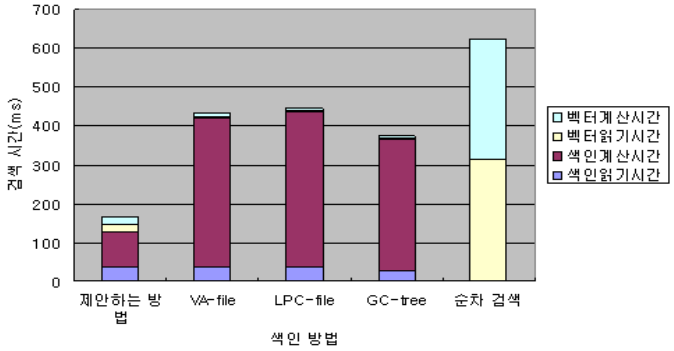
<그림 4>는 차원의 증가에 따른 검색 성능의 변화를 평가하기 위해 특성 벡터의 차원의 수를 확장시키며 검색 시간을 측정한 결과를 보여준다. 제안한 방법은 10차원 내외의 데이터에서는 순차검색과 큰 차이를 보이지 않지만 차원이 증가할수록 검색 시간의 차이가 커지는 것을 확인할 수 있다.

4.3. 기존 방법들과의 성능 비교

기존의 방법들과 제안한 방법의 검색 성능을 비교하기 위해 50,000개의 이미지들에 대해 256차원을 갖는 MPEG-7 Color Structure 특성 벡터를 추출하고 이를 이용한 검색 시스템을 구성하였다. 각 차원 당 8비트를 할당하였을 경우 제안한 방법과 기존의 방법들에 의해 생성된 색인 파일의 크기는 <표 2>와 같다.

<표 2. 객체 수에 따른 색인 파일의 크기(KB)>

객체수	실제 벡터크기	제안한 방법	VA-File	LPC-File	GC-Tree
25,168	50,337	6,311	6,805	6,854	4,115
50,000	100,002	12,427	12,921	12,971	7,429
100,000	200,003	24,251	24,745	24,795	14,210
200,000	400,005	48,174	48,668	48,718	26,999



<그림 5. 각 방법에 대한 검색 시간의 비교>

실험 결과는 <그림 5>에 표시되어 있다. 실험으로부터 실제 벡터를 읽고 계산하는 시간이 색인 파일을 읽고 계산하는 시간에 비해 매우 작음을 알 수 있으며 제안한 방법이 기존의 방법들과 비교해 2배 이상의 우수한 성능을 가진다는 것을 확인할 수 있다.

5. 결론 및 향후 연구 방향

본 논문에서는 필터링을 위한 계산 시간을 줄이기 위해서 질의 객체와 전체 데이터 사이의 빠른 거리 계산을 위한 새로운 비트맵 색인 구조를 제안하였다. 제안한 방법은 각 객체를 특성 벡터 공간상의 위치를 나타내는 비트맵 패턴으로 저장하고, 객체 사이의 거리를 구하는 연산을 XOR 비트 연산으로 대체하였으므로 빠른 속도의 거리 계산이 가능하다. 실험을 통해 순차 검색 보다는 약 4배, 기존의 벡터 근사 접근 방법들보다는 약 2배의 성능이 향상됨을 확인할 수 있었다.

향후, 데이터의 분포에 따라 적응적으로 공간을 분할하면서도 비트 연산을 통해 계산 속도를 향상시킬 수 있는 방법에 대한 연구가 지속되어야 할 것이다.

6. 참고 문헌

[1] K. Chakarabarti and S. Mehrotra, "Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces," in *Proceedings of the International Conference on VLDB,* pp.89-100, 2000

[2] E. Kuchilevitz, R. Ostrovsky, and Y. Ravani, "Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces," in *Proceedings of the ACM Symposium on the Theory of Computing,* pp. 614-623, 1998

[3] R. Weber, H. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," in *Proceedings of the Conference on VLDB*, pp. 194-205, 1998

[4] G. Cha, X. Zhu, D. Petkovic, and C. Chung, "An Efficient Indexing Methods for Nearest Neighbor Searches in High-Dimensional Image Databases," in *IEEE Transactions on Multimedia*, *Vol. 4, No. 1*, pp. 76-87, 2002.

[5] G. Cha, and C. Chung, "The GC-Tree: A High Dimensional Index Structure for Similarity Search in Image Databases," in *IEEE Transactions on Multimedia*, *Vol. 4, No. 2*, pp. 235-247, 2002.