셀기반 시그니쳐 트리 : 고차원 데이터의 유사성 검색을 위한 효율적인 색인 구조

송광택 장재우
전 북 대 학 교 컴 퓨 터 공 학 과
{ktsong, jwchang}@dblab.chonbuk.ac.kr

Cell-based Signature Tree : Efficient Indexing Structures for Similarity Search in High-Dimensional Feature Space

Kwang-Taek Song○ Jae-Woo Chang
Dept. of Computer Engineering, Chonbuk National University

요 약

본 논문에서는 고차원의 특징 벡터 공간에서의 객체에 대한 효율적인 검색을 지원하는 셀기반 시그니쳐 트리 색인 구조(CS-트리, CI-트리)를 제안한다. 특징 벡터 공간을 셀로써 분할하고 특징 벡터는 셀의 시그니쳐로 표현되며 트리에 저장된다. 특징 벡터 대신 시그니쳐를 사용하여 트리의 깊이가 낮아짐으로서 검색을 효율적으로 수행할 수 있다. 또한 셀에 적합한 새로운 가지치기 거리를 이용한 유사성 검색 알고리즘을 제시한다. 마지막으로 우수한 고차원 색인 기법으로 알려져 있는 X-트리와 성능 비교를 수행하여, 성능비교 결과 본 논문에서 제안하는 CS-트리와 CI-트리가 검색 시간 측면에서 최대 30%의 검색 성능이 개선됨을 보인다.

1. 서론

최근 내용-기반 검색 시스템에 대한 관심이 증가하면서 멀티미디어 데이터베이스에서 유사성에 기반한 질의 검색의 중요성이 두드러지고 있다. 멀티미디어 데이터베이스에서 유사성에 기반한 검색 질의를 효율적으로 제공하기 위한 접근 방식은 멀티미디어 객체를 d-차원의 특징 벡터 공간에서의 점(point)으로 표현하는 것이다. 점으로써 표현되는 특징 벡터는 데이터베이스에 저장되며 질의객체와의 유사성은 특징벡터 공간에서의 거리로써 측정된다. 이러한 유사성에 기반한 질의로는 범위질의(range query), k-최근접 탐색 질의(k-nearest neighbor query)등이 있다. 내용-기반 멀티미디어 데이터베이스 검색에서는 k-최근접 데이터 탐색 질의와 같은 유사성 질의가 매우 중요하다. 고차원 데이터에 대한 효율적인 색인 기법의 제공을 통해 대규모 데이터베이스에서도 이러한 유사성 질의를 보다 효율적으로 지원할 수 있다. 하지만 기존의 트리-기반 색인 구조는 멀티미디어 객체로부터 추출된 특징벡터의 차원이 증가함에 따라 MBR(Minimum Bounding Region)를 위한 노드 정보로 인하여 트리의 팬-아웃(fan-out)이 급격히 감소함으로써 검색 성능이 저하된다. 따라서 본 연구에서는 차원이 증가함에 따라 발생하는 기존의 트리-기반 고차원 색인 기법의 비효율성을 극복하기 위해 특징 벡터의 시그니쳐(signature)를 이용한 새로운 고차원 색인 구조를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 고차원 색인기법에 대한 기존 연구를 분석한다. 3장에서는 본 연구에서 제안한 시그니쳐를 이용한 새로운 고차원 색인 구조에 대하여 서술한다. 4장에서는 제안한 색인 구조의 성능 평가를 기술하고, 마지막으로 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련연구

대용량의 멀티미디어 데이터베이스를 다루는데 있어서 주된 문제는 검색의 효율성으로, 고차원 특징 벡터 공간에서 보다 효율적인 유사성 질의를 제공하기 위한 색인 기법에 대한 많은 연구들이 이루어지고 있다. SS-트리, VAMSplit R-트리, TV-트리, SR-트리, X-트리는 기존의 공간 색인 기법인 R-트리를 확장한 트리-기반 색인기법들이다[BBKK97][BK98].

고차원 특징벡터 공간에서 기존의 색인 구조에 비해 k-최근접 탐색 질의와 같이 유사성에 기반한 검색을 효율적으로 지원하는 색인 구조의 하나로 평가되는 X-트리[BKK96]는 기존의 색인 구조들이 차원에 증가함에 따라 겹침 영역이 증가하여 검색 성능이 현저히 저하되는 문제점을 방지하기 위해 제안된 트리-기반 고차원 색인 구조이다. 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. 노드는 크게 MBR(Minimum Bounding Rectangle) 정보를 갖는 중간 노드와 실제 특징 벡터 데이터를 갖는 데이터 노드로 구분된다. 한편, X-트리는 차원이 증가함에 따라 MBR을 위한 저장공간의 증가로 인하여 노드의 팬-아웃(fan-out)이 급격히 감소함으로써 검색성능이 감소하는 트리-기반 색인 구조의 문제점을 갖는다.

3. 셀-기반 시그니쳐 트리

3.1 시그니쳐 변환

고차원 데이터에 대한 시그니쳐는 데이터 각각의 차원에 따른 특징 벡터에 대한 요약들의 집합이므로, 각 차원에 해당하는 벡터의 성질을 유지하도록 표현한다. 이를 위해 데이터 공간을 셀(cell) 단위로 나누고, 셀에 대한 시그니쳐를 생성한다. 이렇게 만들어진 시그니쳐는 각각의 셀을 대표하는 값이 되며, 셀에 대한 정보를 포함하게 된다. 또한, 차원에 따라 데이터 공간을 균등하게 분할함으로써, 셀에 대한 시그니쳐는 데이터 공간에서의 어떤 영역의 범위를 가지게 되고, 시그니쳐를 통해 쉽게 이 범위값을 구할 수 있다. 즉, 데이터를 검색하는 경우, 직접 실제 데이터를 접근하기 전에, 먼저 시그니쳐로 표현되는 셀을 접근하여 필요한 셀만을 선택하고, 선택된 셀에 대해서만 직접 실제 데이터를 접근함으로써, 시그니쳐에 의한 필터링 효과를 얻을 수 있다. 그림1은 각각의 차원에 대해서 2-비트 시그니쳐를 사용할 경우, 2차원 공간에서의 시그니쳐를 만들어내는 과정을 나타낸다.

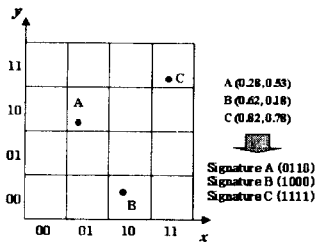

그림 1. 2차원 공간에서의 시그니쳐 생성 과정

3.2 셀 기반 시그니쳐 트리

본 논문에서 제안하는 셀-기반 시그니쳐 트리는 기존의 트리 구조 색인 구조에 시그니쳐를 적용한 방법으로, 트리 내부 노드에 실제 데이터값이 아닌 데이터값에 대한 시그니쳐를 사용한 방법이다. 셀-기반 시그니쳐 트리를 위해 두 가지 구조를 제안한다. 첫 째는 트리의 중간 노드(internal node)와 리프 노드(leaf node)에 특징 벡터 대신 시그니쳐를 저장하는 구조(Tree with Cell-based Signatures in all nodes ; CS-트리)이다. CS-트리 기법에서는 두 개의 파일로 구성되는데, 트리를 구성하는 노드 정보를 가지고 있는 시그니쳐 인덱스 파일과 객체의 실제 특징 벡터를 가지고 있는 데이터 파일이다. 새로운 객체의 N-차원 특징 벡터를 저장하는 경우, 우선 객체의 특징 벡터를 객체가 포함되는 셀을 나타내는 시그니쳐로 변환된다. 이렇게 변환된 시그니쳐는 트리의 노드에 저장되고, 루트 노드로부터 트리를 순차적으로 탐색하여, 해당되는 리프 노드에 저장되게 된다. 이때 탐색 경로에 해당되는 내부 노드(internal node)에는 리프 노드에 저장된 시그니쳐를 포함하는 MBR 정보를 나타내는 시그니쳐를 지닌다. 일단, 객체의 시그니쳐가 리프 노드에 저장되면, 각각의 리프 노드가 가리키는 데이터 파일의 해당 페이지 위치에 객체의 실제 특징 벡터를 저장하게 된다. 그림 2는 CS-트리의 구조를 나타낸다. 둘 째는 중간 노드(internal node)는 셀을 포함하는 MBR에 대한 시그니쳐를 저장하고 리프 노드(leaf node)에는 실제 특징 벡터를 저장하는 구조(Tree with Cell-based signatures in Internal nodes; CI-트리)이다. CI-트리는 CS-트리와는 달리 시그니쳐 인덱스 파일만을 요구하며 트리를 구성하는 노드 정보뿐만 아니라 객체의 특징 벡터를 저장한다. 중간노드(internal node)에서는 MBR을 시그니쳐로 변환하여 저장하고 리프노드(leaf node)에서는 객체로부터 추출한 특징벡터를 저장하게 된다. 그림 3은 CI-트리의 구조를 나타낸다.

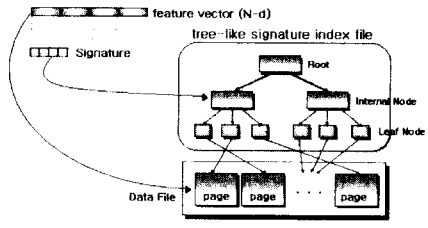

그림 2. CS-트리의 구조

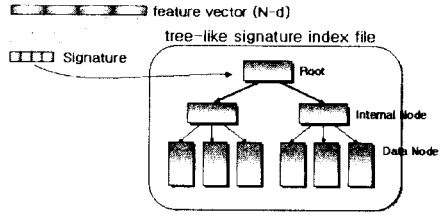

그림 3. CI-트리의 구조

3.3 유사성-기반 질의 처리 알고리즘

본 논문에서 제안하는 CS-트리와 CI-트리는 중간 노드에 셀에 기반한 경계(bounce) 시그니쳐를 저장하고 있다. 따라서 트리의 노드의 MBR(Minimum Bounding Region)은 이러한 셀을 포함하는 영역이 된다. 기존의 MINMAXDIST는 MBR 내의 적어도 하나의 객체를 포함할 수 있는 거리를 의미한다. 따라서 k-최근접 탐색에서 k번째 MINMAXDIST는 질의 객체와 가장 가까운k개의 객체를 보장할 수 있는 거리이다[SC 99]. 셀-기반 시그니쳐 트리에서의 MBR은 객체들의 최소경계영역이 아니라 셀들의 최소경계영역이기 때문에 셀-기반 시그니쳐 트리에서는 기존의 k-최근접 탐색에서 사용했던 가지치기 거리 MINMAXDIST 를 그대로 사용할 수 없다. 본 논문에서는 셀을 저장하고 있는 노드들의 MBR에 대한 새로운 가지치기 거리, CMINMAXDIST를 정의한다.

정의 1. 임의의 점 P와 MBR R 간의 거리 CMINMAXDIST(P, R)은 다음과 같이 정의한다.

$$CMINMAXDIST(P,R) = \min_{1\le k\le n}\left(|p_k - crm_k|^2 + \sum_{1\le i\le n,\ i\neq k} |p_i - crM_i|^2\right)$$

여기에서 crmk 는 MBR R의 평면들(hyper-plains) 중 각 차원 축과 직교하는 두 개의 평면들 중 점 P로부터 가까운 쪽의 평면까지의 거리에 적어도 하나의 셀을 포함할 수 있도록 cd를 고려한 거리이다. cd 는 하나의 셀 폭을 의미한다.

$$crm_k = \begin{cases} s_k + cd & \text{if } p_k \le \frac{s_k+t_k}{2} \wedge p_k < s_k \\ s_k & \text{if } p_k \le \frac{s_k+t_k}{2} \wedge p_k \ge s_k \\ t_k - cd & \text{if } p_k > \frac{s_k+t_k}{2} \wedge p_k > t_k \\ t_k & \text{if } p_k > \frac{s_k+t_k}{2} \wedge p_k \le t_k \end{cases}$$

crMi 는 각 차원 측과 직교하는 두 개의 평면들 중 점 P로부터 먼 쪽의 평면까지의 거리로서 다음과 같이 정의된다.

$$crM_i = \begin{cases} s_i & \text{if } p_i \le \frac{s_i + t_i}{2} \\ t_i & Otherwise. \end{cases}$$

CMINMAXDIST(P, R) 거리는 셀의 특징을 고려하여 MBR R 내의 셀 중에서 객체가 저장되어 있는 적어도 하나의 셀을 포함할 수 있는 거리를 보장한다.

한편, CS-트리에서 k-최근접 탐색 질의를 지원하기 위해 두 단계로 수행된다. 첫 번째 단계에서는 객체를 포함하는 셀의 시그니처가 저장되어 있는 트리 인덱스에서 질의 점과 가장 가까운 셀을 1차 후보 리스트로 선택한다. 1차 후보 리스트에 포함되는 셀은 탐색 과정에서 발생하는 k번째 CMINMAXDIST보다 작은 MINDIST를 갖는 셀들이 포함된다. 이로써 후보 리스트는 정확한 k 개의 객체를 검색할 수 있도록 보장하게 된다. 마지막 단계에서는 1차 후보 리스트에 있는 셀들에 대한 실제 벡터를 순차 탐색에 의해 직접 액세스하게 된다. 이로서 주어진 질의 점과 가장 근접한 k 개의 객체를 찾는다. CI-트리에서는 기존 트리-기반 색인 구조와 비슷하지만 CS-트리와 마찬가지로 MBR은 셀을 포함하는 시그니처이기 때문에 k-최근접 탐색 영역의 제한 영역(upper bound)으로 CMINMAXDIST를 이용한다.

4. 실험 평가

본 논문에서 제안한 CS-트리 및 CI-트리는 Windows NT 4.0 상에서 Visual C++ 6.0 언어를 이용하여 구현되었다. 아울러 성능평가를 위해 애니메이션 MPEG 동영상에서 100,000개의 I-프레임으로부터 추출한 특징 벡터 셋(15차원과 20차원)과 논문요약으로부터 생성한 특징 벡터 100,000 건(10차원과 20차원)을 사용하였다. 애니메이션 데이터는 논문요약 데이터에 비해 보다 클러스터링 되어있는 특징을 지닌다. 검색 성능 평가를 위해 k-최근접 질의 검색에 대한 검색시간(search time)을 이용한다. k-최근접 질의의 검색 인자 k는 100 이며 질의 횟수는 100번으로, 검색 시간은 평균 값을 이용한다.

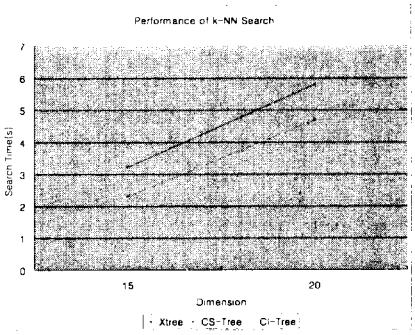

그림 4. 애니메이션 데이터에 대한 성능 평가

그림 4는 애니메이션 동영상에서 추출한 특징 데이터에 대한 k-최근접 질의 검색을 위한 검색시간을 나타낸다. 애니메이션 데이터에 대해 CS-트리와 CI-트리가 X-트리에 비해 검색 시간 측면에서 우수함을 나타내고 있다. X-트리는 3.2~5.8초의 검색시간을 요구한 반면, CS-트리는 2.3~4.7초, CI-트리는 1.8~4.5초만을 요구하였다. 그림 5는 논문의 요약에서 Hadamard 알고리즘[LJF95]에 의해 추출한 특징 데이터에 대해 k-최근접 질의 검색을 수행하는 데 소요된 검색 시간을 나타낸다. CI-트리의 검색 시간은 X-트리와 비슷한 결과를 나타나고 있지만, 애니메이션 데이터와 마찬가지로 차원이 높을수록 CS-트리는 X-트리에 비해서 검색 시간이 감소함을 나타낸다. 20차원일 경우, CS-트리는 X-트리의 검색 시간의 약 60%만을 요구한다.

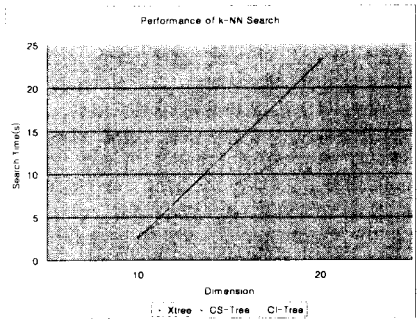

그림 5. 논문 요약 데이터에 대한 성능 평가

5. 결론

본 연구에서는 차원이 증가함에 따라 발생하는 기존 트리 기반 색인 기법의 비효율성을 극복하기 위해, 셀 영역에 기반한 시그니처를 이용한 새로운 고차원 색인구조를 제안하였다. 제안하는 CS-트리와 CI-트리는 기존의 트리 기반 색인 기법이 차원 증가에 따라 팬-아웃(fan-out)이 급격히 떨어지는 문제점을 해결하고자 특징벡터의 시그니처(signature)를 저장한다. 또한 효율적인 k-최근접 탐색 질의를 지원하기 위해 셀-기반 k-최근접 탐색 알고리즘을 제안하였다. 아울러 본 연구에서 제안한 CS-트리와 CI-트리를 기존 고차원 색인 구조인 X-트리와의 성능 비교를 수행한 결과, 최대 30%의 검색 성능이 개선됨을 보였다.

앞으로의 연구방향은 다양한 데이터의 고차원에 대한 성능 실험을 수행하는 것이다.

[참고 문헌]

[BBKK97] 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.

[BK98] Berchtold S., Keim D. A.: "High-Dimensional Index Structures - Database Support for Next Decade's Applications", Tutorial, SIGMOD, 1998.

[BKK96] Berchtold. S, Keim. D. A, Kriegel. H-P, "The X-tree : An Index Structure for High-Dimensional Data, Proceedings of the 22nd VLDB Conference, pp.28-39, 1996.

[LJF95] Lin. H.I, Jagadish. H, and Faloutsos. C, "The TV-tree : An Index Structure for High Dimensional Data", VLDB Journal, Vol. 3, pp.517-542, 1995.

[RKV95] Roussopoulos N., Kelley S., Vincent F., "Nearest Neighbor Queries", Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 71-79, 1995.