구형 피라미드 기법: 고차원 데이타의 유사성 검색을 위한 효율적인 색인 기법

(Spherical Pyramid-Technique: An Efficient Indexing Technique for Similarity Search in High-Dimensional Data)

이 동 호 † 정 진 완 †† 김 형 주 †††

(Dong-Ho Lee) (Chin-Wan Chung) (Hyoung-Joo Kim)

요 약 피라미드 기법[1]은 d-차원의 공간을 2d개의 피라미드들로 분할하는 특별한 공간 분할 방식을 이용하여 고차원 데이타를 효율적으로 색인할 수 있는 새로운 색인 방법으로 제안되었다. 피라미드 기법은 고차원 사각형 형태의 영역 질의에는 효율적이나, 유사성 검색에 많이 사용되는 고차원 구형태의 영역 질의에는 비효율적인 면이 존재한다. 본 논문에서는 고차원 데이타를 많이 사용하는 유사성 검색에 효율적인 새로운 색인 기법으로 구형 피라미드 기법을 제안한다. 구형 피라미드 기법은 먼저 d-차원의 공간을 2d개의 구형 피라미드로 분할하고, 각 단일 구형 피라미드를 다시 구형태의 조각으로 분할하는 특별한 공간 분할 방법에 기반하고 있다. 이러한 공간 분할 방식은 피라미드 기법과 마찬가지로 d-차원 공간을 1-차원 공간으로 변환할 수 있다. 따라서, 변환된 1-차원 데이타를 다루기 위하여 B+-트리를 사용할 수 있다 본 논문에서는 이렇게 분할된 공간에서 고차원 구형태의 영역 질의를 효율적으로 처리할 수 있는 알고리즘을 제안한다. 마지막으로, 인위적 데이타와 실제 데이타를 사용한 다양한 실험을 통하여 구형 피라미드 기법이 구형태의 영역 질의를 처리하는데 있어서 기존의 피라미드 기법보다 효율적임을 보인다.

Abstract The Pyramid-Technique[1] was proposed as a new indexing method for high-dimensional data spaces using a special partitioning strategy that divides d-dimensional space into 2d pyramids. It is efficient for hypercube range query, but is not efficient for hypersphere range query which is frequently used in similarity search. In this paper, we propose the Spherical Pyramid-Technique, an efficient indexing method for similarity search in high-dimensional space. The Spherical Pyramid-Technique is based on a special partitioning strategy, which is to divide the d-dimensional data space first into 2d spherical pyramids, and then cut the single spherical pyramid into several spherical slices. This partition provides a transformation of d-dimensional space into 1-dimensional space as the Pyramid-Technique does. Thus, we are able to use a B+-tree to manage the transformed 1-dimensional data We also propose the algorithm of processing hypersphere range query on the space partitioned by this partitioning strategy. Finally, we show that the Spherical Pyramid-Technique clearly outperforms the Pyramid-Technique in processing hypersphere range queries through various experiments using synthetic and real data.

· 본 연구는 한국과학재단의 특정 기초 연구 과제(No 95-0100-23-04-3), "다중 매체 요구형 시스템 기술에 관한 연구"와 과학기술처의 국가 지리정보 시스템 기술개발 사업(No. 97-187)에서 지원 받았음

† 비 회 원 : 서울대학교 컴퓨터공학과
dhlee@oopsla.snu.ac.kr
†† 종신회원 한국과학기술원 전산학과 교수
chungcw@ngis.kaist.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학과 교수
hjk@oopsla snu ac kr

논문접수 1998년 12월 16일
심사완료 1999년 8월 31일

1. 서 론

고차원 데이타를 위한 색인 기법은 다음 세대의 데이타베이스 응용들을 위하여 데이타베이스 시스템이 지원해야 할 가장 중요한 기술 중에 하나이다[2].

최근 다양한 새로운 데이타베이스 응용들은 기존의 데이타베이스 응용들과는 다르게 개발되고 있다. 예를 들어, 데이타 웨어하우징(data warehousing)과 같은 응용은 데이타에 대한 다차원 뷰(view)를 요구하는 대규모의

테이블을 생성한다. 또한, 멀티미디어 데이타베이스에서 가장 많이 사용되는 응용 중에 하나인 내용 기반 검색(content-based retrieval) 응용은 대부분 멀티미디어 데이타에서 추출한 특징 벡터를 이용한 유사성 검색(similarity search)을 수행한다[1]. 이러한 종류의 새로운 응용들은 대부분 고차원 데이타를 사용하고 있으며, 하위에 존재하는 데이타베이스 시스템에서 이러한 고차원 데이타에 대한 효율적인 색인을 제공해야만 대규모 데이타베이스에서도 신속한 검색을 지원할 수 있다.

최근의 연구 결과들에 의하면 저차원이나 중차원 데이타에 대하여 효율적인 색인을 제공하는 어떤 색인 구조도 고차원 데이타에 대해서는 효율적인 색인을 제공하지 못하고 있다[1,3-5]. 이러한 문제를 일반적으로 "차원의 저주(curse of dimensionality)"[6]라 하며, 최근 데이타베이스 관련 연구 과제에서는 이러한 문제를 해결하고자 하는 노력을 수행하고 있다. 이러한 노력의 일환으로 다양한 새로운 색인 구조[5,7,8]와 비용 모델[4,9], 질의 처리 알고리즘[10]들이 제시되고 있다. 그러나, 대부분의 새로운 색인 구조는 고차원 데이타에 대한 색인을 제공하기 위하여 기존의 다차원 색인 구조(multi-dimensional index structure)를 확장한 것으로, 그 구조나 공간 분할 방식에 있어서 특별한 제한을 가지고 있다. 또한, 다차원 색인 구조의 잘 알려진 단점인 삽입, 삭제에 고비용이 들어간다는 단점과 동시성 제어(concurrency control)나 회복(recovery)을 거의 제공하지 못한다는 단점을 가지고 있다[1].

피라미드 기법[1]은 이러한 단점을 극복하기 위하여 고차원 데이타를 1-차원 공간으로 변환한 후에 빠른 삽입과 갱신, 삭제 및 검색을 지원하는 B+-트리를 이용하는 방법을 제안하였다. [1]의 연구에서는 고차원 공간을 피라미드 형태로 분할하는 특별한 공간 분할 방식을 제시하였으며, 이렇게 분할된 공간상에서 고차원 사각형 형태의 영역 질의(hypercube range query)를 처리하는 알고리즘을 제시하였다. 그러나, 멀티미디어 데이타베이스의 가장 중요한 응용 중에 하나인 내용 기반 검색의 경우에는 대부분 유사 검색을 사용하고 있으며, 이러한 유사 검색은 기본적으로 유사성을 측정하는 함수를 이용하고 있다. 따라서, 질의의 형태가 고차원 사각형이 아니라, 고차원 구형태(hypersphere range query)가 된다[6,11].

피라미드 기법을 이용하여 구형태의 질의를 처리하는 경우에는 기존의 경계 사각형(bounding rectangle) 기반의 색인 구조를 이용하는 경우와 마찬가지의 단점이 존재한다. 즉, 구형태의 질의를 처리하기 위하여 이러한 구를 둘러싸고 있는 최소 영역 사각형(minimum boundary rectangle)을 이용하여 질의를 수행한 후에 결과로 얻어진 객체들의 집합에 대하여, 다시 한번 구형태의 질의 영역 안에 존재하는지를 검사하는 작업을 수행해야 한다. 이러한 작업은 불필요한 데이타 페이지를 접근해야 하며, 따라서 질의 응답시간이 느려지는 경향이 존재하게 된다[6].

본 논문에서는 내용 기반 검색 응용에서 많이 사용되는 유사 검색을 효율적으로 지원할 수 있는 새로운 색인 구조로 구형 피라미드 기법을 제안한다. 구형 피라미드 기법은 d-차원의 공간을 2d개의 구형 피라미드들로 분할한 후에, 각 구형 피라미드를 다시 부채꼴 모양의 구형조각들로 나누는 특별한 공간 분할 방식에 기반을 두고 있다. 또한, 이렇게 분할된 공간상에서 구형태의 질의를 효율적으로 처리하는 알고리즘을 제시한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 그 동안 수행되어 왔던 고차원 색인 구조에 대한 관련 연구를 소개하고, 3장에서는 기존의 피라미드 기법에 대하여 좀 더 자세히 살펴본다. 4장에서는 본 논문에서 제안하는 구형 피라미드 기법의 공간 분할 방식을 설명하고, 5장에서는 구형태의 영역 질의를 처리하는 알고리즘을 제안한다. 6장에서는 인위적 데이타와 실제 데이타를 이용한 다양한 실험 및 분석에 대하여 설명하고, 마지막으로 7장에서는 결론 및 향후 연구 방향에 대해서 설명한다.

2. 관련 연구

최근 몇 년 동안 여러 가지 종류의 고차원 색인 구조들이 제안되었다.

TV-tree[8]는 Jagadish 와 Faloutsos가 제안한 R-tree[12]를 확장한 색인구조이다. TV-tree는 데이타의 모든 차원을 색인에 사용하는 것이 아니라, 중요한 차원만을 선택하여 색인에 이용하는 방법을 제시하였다. TV-tree의 가장 큰 단점은 고차원 데이타를 구성하는 각 차원에 대한 정보가 있어야 한다는 것이다. 즉, 어떤 차원의 값을 색인에 이용할 것인가를 결정할 수 있어야 한다. 이것은 일반적으로 각 차원의 중요도를 고려해야 하는 문제가 된다.

SS-tree[3]는 TV-tree와 마찬가지로 R-tree를 확장한 색인 구조이다. SS-tree는 디렉토리 구조에서 사각형이 아니라 구형태를 사용함으로써 R*-tree[13]의 성능을 개선하였다. 그러나, 디렉토리 구조에서 구형태를 이용하게 되면, 중첩이 많이 발생할 가능성이 존재한다

SR-tree[14]는 SS-tree의 단점을 극복하기 위하여

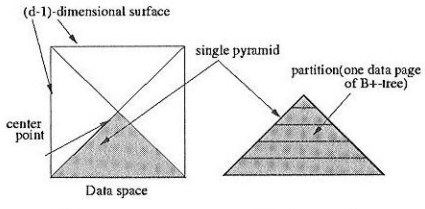
R-tree와 SS-tree를 통합한 방식을 이용하고 있다. SR-tree의 디렉토리는 사각형 형태와 구형태를 동시에 사용함으로써R*-tree와 SS-tree의 성능을 향상시켰다.

X-tree[5]는 두 가지 기법을 이용하여 R*-tree 알고리즘을 확장함으로써, 고차원 데이타에 대한 효율적인 색인을 제공한다. 첫 번째는 겹침 없는 분할(overlap-free split) 방법이며, 두 번째는 슈퍼노드의 개념이다. 겹침이 없는 분할 방법은 K-D-B 트리처럼 탐색 공간을 서로 분리된 다른 공간으로 분리하여 검색을 효율적으로 할 수 있도록 하는 방법이다. 그리고, 슈퍼노드는 이러한 겹침이 없는 분할을 이룰 수 있도록 사용하는 확장 가능한 디렉토리 노드이다. 즉, 만약 분할로 인하여 겹침이 발생하게 되며, 이러한 분할을 생략하고 해당 디렉토리 노드를 확장하는 것이다.

VAMSplit R-tree[7]는 또 다른 최적화된 R-tree계열의 색인 구조로, 주어진 데이타들의 집합을 가지고 하향식(top-down manner)으로 트리를 생성하게 된다. 즉, 모든 데이타들은 색인을 생성하는 시점에 이용 가능해야 한다. 따라서, 동적인 삽입이나 갱신이 불가능하며 주기억 장치(main memory)를 기반으로 하기 때문에 색인의 크기가 주기억 장치의 용량에 제한 받는다는 단점이 있다.

이러한 접근 방법들은 일정한 저장 장치 이용률(storage utilization)을 보장하기 위하여 데이타 페이지를 균등하게 분할하는 방식(balanced splits)을 사용한다. 그러나, 균등 분할 방식을 이용하여 고차원 데이타를 색인할 경우, 차원이 조금만 증가해도 페이지 접근률이 거의 100% 수준에 이르기 때문에 매우 비효율적이다. [1]의 연구에서는 균등 분할을 이용할 경우, 차원의 증가에 따른 페이지 접근률을 수학적으로 분석하였다.

피라미드 기법[1]에서는 지금까지 관련 연구에서 다루었던 고차원 색인 구조의 단점들을 극복하기 위하여 특별한 공간 분할 방식과 고차원 사각형 영역 질의 알고리즘을 제안하였다. 피라미드 기법에 대해서는 다음 장에서 좀 더 자세히 살펴보기로 한다.



(a) First step (b) Second step

그림 1 피라미드 기법의 공간 분할 전략

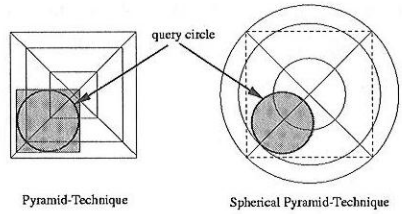


그림 2 피라미드 기법과 구형피라미드 기법의 공간 분할 전략

3. 피라미드 기법

피라미드 기법의 기본적인 개념은 d-차원의 공간을 2d개의 피라미드들로 분할하고 이를 다시 여러 개의 조각들로 나누는 공간 분할 방법을 사용하여 d-차원 데이타를 1-차원 값으로 변환한 후, 이를 1차원 색인인 B+-트리를 이용하여 색인하는 것이다. d-차원의 점을 1-차원의 값으로 변환하는 과정은 다음과 같이 두 단계로 이루어진다.

첫 번째 단계에서는 데이타 공간의 중앙점(0.5, 0.5, ... , 0.5)를 상단점(top)으로 하고, 데이타 공간의 (d-1)-차원 평면을 그들의 기저(base)로 가지는 2d개의 피라미드들로 분할한다. 두 번째 단계에서는 2d개의 각 피라미드를 그들의 기저와 평행하게 여러 개의 조각들로 분할한다. 이때 한 조각이 B+-트리에서 하나의 데이타 페이지를 구성하게 된다. 그림 1은 2차원 공간에서 피라미드 기법의 공간 분할 방식을 보여주고 있다. 그림 1의 (a)에서 데이타 공간은 4개의 삼각형으로 분할되며, 각 삼각형은 공간의 중앙점을 상단점으로 가지고, 한 개의 선분(edge)을 기저로 가지게 된다. 그림 1의 (b)에서 각 삼각형은 다시 기저에 평행한 몇 개의 조각들로 분할된다. 이때, 하나의 조각이 B+-트리에서 하나의 데이타 페이지가 된다.

피라미드 기법에서는 이러한 공간 분할 방식을 기반으로 하여 점 질의(point query)와 고차원의 사각형 영역 질의를 처리하는 알고리즘을 제안하였다. 그러나, 내용 기반 검색에서 많이 사용되는 유사 검색을 피라미드 기법을 이용하여 처리할 경우, 서론에서도 언급했듯이 불필요한 데이타 페이지들을 접근해야 하며, 따라서 전체적인 성능이 저하된다.

그림 2는 2차원 공간상에서 피라미드 기법의 공간 분할 방식과 나무의 나이테 모양과 유사한 구형 피라미드 기법의 공간 분할 방식을 비교한 것이다. 그림 2에서처

럼 원형태의 질의가 주어질 경우, 피라미드 기법의 공간 분할 방식을 이용할 경우, 총 12 페이지 중에서 8 페이지를 접근하게 된다. 즉, 전체 페이지 중에서 약 66% 정도의 페이지를 접근해야 한다. 그러나, 구형 피라미드 기법의 경우에는 4 페이지만을 접근하면 됨으로, 전체 페이지의 약 33%만 접근하면 질의를 처리할 수 있다.

4. 구형 피라미드 기법

구형 피라미드 기법의 기본적인 아이디어는 유사 검색을 위해서는 구형태의 공간 분할이 피라미드 기법의 직각형태 분할보다 적합할 것이라는 관찰에 기반한다. 이러한 관찰은 유사 검색에 사용되는 질의의 형태가 사각형이 아니라 구형태라는 사실에 기인한다. 구형 피라미드 기법에서도 d-차원의 점을 1-차원 값으로 변환하여 B+-트리와 같은 효율적인 1-차원 색인 구조를 사용한다. 또한, d-차원 점과 해당 1-차원 값을 B+-트리의 단말 노드(leaf node)에 함께 저장할 수 있음으로 이 변환에 대한 역변환은 필요하지 않다. 이러한 변환은 d-차원의 데이타 공간을 2d개의 구형 피라미드들로 분할하는 것에 기초하고 있음으로 공간 분할 방식에 대하여 설명한다.

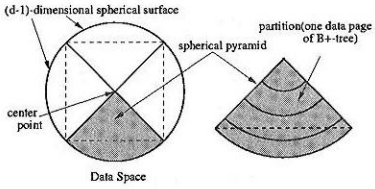


그림 3 구형 피라미드 기법의 공간 분할 전략

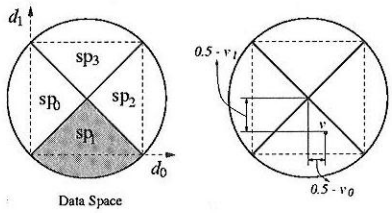


그림 4 구형 피라미드의 번호 결정 과정

4.1 공간 분할 방식

구형 피라미드 기법은 데이타 공간을 다음과 같은 2단계로 나눈다.

첫 번째 단계는 [1] 연구의 공간 분할 방식의 첫 번째 단계와 동일한 방법을 사용한다. 즉, 데이타 공간의 중앙점(0.5, 0.5, ..., 0.5)을 상단점으로 하고 (d-1)-차원의 구형 평면을 기저로 가지는 2d개의 구형 피라미드들로 공간을 분할한다. 두 번째 단계는 각 단일 구형 피라미드의 상단점을 중심으로 갖는 여러 개의 구형 조각들로 나눈다. 이러한 구형태의 조각은 B+-트리의 한 페이지에 상응하게 된다.

그림 3은 2차원 공간상에서 구형 피라미드 기법의 공간 분할을 보여주고 있다. 먼저, 2차원 공간상에서 4개의 구형 피라미드들로 분할되며, 각 구형 피라미드는 동일하게 공간의 중앙점을 상단점으로 가지고 하나의 곡선을 기저로 가진다. 이것은 d-차원으로 동일하게 확장될 수 있으며 d-차원에서는 기저가 곡선이 아니라 (d-1)-차원의 구형 평면이 된다. 두 번째 단계에서 각 구형 피라미드는 상단점을 중심으로 갖는 여러 개의 구형 조각들로 분할된다. 그림 3의 오른쪽 그림을 보면, 하나의 구형 피라미드가 4개의 구형 조각들로 분할된 것을 볼 수 있다. 각 조각은 B+-트리에서 하나의 데이타 페이지가 된다. d-차원의 구는 2d개의 (d-1)-차원의 구형 평면을 가지기 때문에 2d개의 구형 피라미드를 얻을 수 있다.

구형 피라미드 기법에서 각 구형 피라미드의 번호를 결정하는 과정은 [1]의 연구와 동일하다. 즉, 어떤 점 v가 주어졌을 때, 점 v가 속해 있는 구형 피라미드를 결정하기 위해서는 점 v의 각 차원들 중에서 중앙(0.5)으로부터 가장 멀리 떨어진 차원을 찾아야 한다. 즉, $|0.5 - v_i|$의 값이 최대가 되는 i가 이 점이 속해 있는 구형 피라미드가 된다. 만약 v_i의 값이 0.5보다 작으면, 점 v가 속해 있는 구형 피라미드는 sp_i가 되고, 0.5보다 크거나 같으면 $sp_{(i+d)}$가 된다.

그림 4의 오른쪽 그림에서 2차원 점 v는 $|0.5 - v_1|$의 값이 $|0.5 - v_0|$보다 크다. 따라서, 중심으로부터 가장 멀리 떨어진 차원은 d_1이 되며, 그 차원의 값(v_1)이 0.5보다 작기 때문에 구형 피라미드 sp_1에 속하게 된다. 이 과정에 대한 형식적인 정의는 [1]의 연구와 동일하지만 본 논문의 전체적인 이해를 돕기 위해서 다시 한 번 정리하면 다음과 같다.

[정의 1] : (점 v가 속해 있는 구형 피라미드) d-차원의 점 v는 구형 파리미드 sp_i에 존재한다고 정의된다.

$$i = \begin{cases} j_{\max} & \text{if } v_{j_{\max}} < 0.5 \\ (j_{\max} + d) & \text{if } v_{j_{\max}} \geq 0.5 \end{cases}$$

$$j_{\max} = (j \mid (\forall k,\ 0 \leq (j,k) < d,\ j \neq k : |0.5 - v_j| \geq |0.5 - v_k|))$$

[정의 1]에서 $j_{\max}$는 d-차원의 점 v의 각 차원들 중

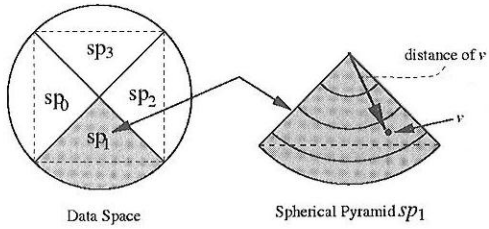


그림 5 구형피라미드 안에 존재하는 점의 거리

에서 중앙(0.5)으로부터 가장 멀리 떨어진 차원을 의미하며, i는 점 v가 속해 있는 구형 피라미드 번호가 된다. 그림 4의 왼쪽 그림은 2차원 공간에서 [정의 1]에 의해 각 구형 피라미드의 번호를 결정하는 것을 보여준다. d-차원의 점을 1-차원의 값으로 변형하기 위해서는 그 점이 속해 있는 구형 피라미드뿐만 아니라, 해당 구형 피라미드 안에서의 위치를 결정해야한다. [1] 연구에서는 중앙점으로부터의 높이로 점의 위치를 결정하였으나, 구형 피라미드 기법에서는 중앙점으로부터 점 v까지의 거리로 위치를 결정한다. 그림 5의 오른쪽 그림은 점 v의 거리를 결정하는 것을 보여주고 있다. 거리는 유사 검색에서 가장 많이 사용되는 유클리드 거리(Euclidean distance)를 사용한다고 가정한다.

[정의 2] : (점 v의 거리) d-차원의 점 v가 주어졌을 때, 점 v의 거리는 다음과 같이 정의된다.

$$d_v=\sqrt{\sum_{i=0}^{d-1}(0.5-v_i)^2}$$

[정의 1]과 [정의 2]를 이용하여 d-차원의 점 v를 1-차원의 값($i\cdot\lceil\sqrt{d}\rceil+d_v$)으로 변환할 수 있다. 여기서 i는 점 v가 속해 있는 구형 피라미드의 번호이고, d_v는 해당 구형 피라미드의 상단점(공간의 중앙점)으로부터 점 v까지의 거리가 된다. 이를 좀더 형식적으로 표현하면 다음과 같다.

[정의 3] : (점 v의 구형 피라미드 값) d-차원의 점 v가 주어졌을 때, sp_i가 [정의 1]에 의해 얻어진 점 v가 속해 있는 구형 피라미드이고, d_v가 [정의 2]에 의해 얻어진 점 v의 거리라고 할 때, 점 v의 구형 피라미드 값은 다음과 같이 정의된다.

$$spv_v=(i\cdot\lceil\sqrt{d}\rceil+d_v)$$

여기서, i는 $[0,2d)$의 범위를 가지는 정수이고, d_v는 $[0,0.5\sqrt{d}]$의 범위를 가지는 실수이다. 그러므로, 각 구형 피라미드 sp_i에 속해 있는 모든 d-차원의 점들은 $[i\cdot\lceil\sqrt{d}\rceil,\ i\cdot\lceil\sqrt{d}\rceil+0.5\sqrt{d}]$ 사이의 구형 피라미드 값을 가진다. 만약 i에 $\lceil\sqrt{d}\rceil$을 곱하지 않으면, 구형 피라미드 sp_i가 가질 수 있는 구형 피라미드 값의 범위는 $[i,\ i+0.5\sqrt{d}]$가 된다. 이럴 경우, 차원이 4차원 이상이 되면, 두개의 구형 피라미드 sp_i와 sp_j에서 가지는 두개의 구형 피라미드 값의 범위가 서로 겹칠 수 있다. 이것은 많은 수의 B+-트리 키값들이 서로 중복되는 현상을 초래할 수 있다. 이러한 현상을 방지하기 위하여 i에 $\lceil\sqrt{d}\rceil$를 곱하게 된다. 또한, 이 변환은 단사(injective) 함수가 아니다. 즉, 두개의 점 v 와 v'는 서로 같은 구형 피라미드 값을 가질 수 있다. 그러나, 전에도 설명하였듯이 d-차원의 점과 해당 구형 피라미드 값을 B+-트리의 단말 노드에 함께 저장하기 때문에 이 변환에 대한 역변환은 필요하지 않다.

4.2 색인의 생성

구형 피라미드 기법에 의해 색인을 생성하는 과정은 간단하다. 먼저, d-차원의 점 v가 주어지며 이것의 구형 피라미드 값(spv_v)을 결정한 후에, 이 값을 B+-트리의 키값으로 하여 점 v를 B+-트리에 삽입한다. 그리고 점 v와 구형 피라미드 값 spv_v를 B+-트리의 해당 데이타 페이지에 저장한다. 갱신이나 삭제도 B+-트리를 이용하여 이와 유사한 방법으로 할 수 있다.

5. 질의 처리 알고리즘

유사성 검색에 사용되는 질의들은 질의 객체와 가장 유사한 k개의 객체를 검색하는 k-최근접 질의와 질의 객체와 일정한 유사성 허용 오차(tolerance)안에 존재하는 모든 객체를 검색하는 영역 질의가 있으며 고차원 공간상에 이 질의들은 모두 고차원 구형태가 된다[3].

본 논문에서는 이 두 가지 질의 중에서 영역 질의를 처리하는 알고리즘을 제시한다. 구형 피라미드 기법을 이용하여 이러한 질의를 처리하는 과정은 삽입이나 삭제, 갱신과는 다르게 복잡하다. 간단한 점 질의는 기존의 피라미드 기법에서 점 질의를 처리하는 방법과 동일한 방법으로 처리할 수 있다[1]. 영역 질의는 실제로 다음과 같은 질의 객체를 의미하는 질의 점과 그 점으로부터의 거리의 한계를 의미하는 유사성 허용 오차로 ε과 같은 두개의 파라미터로 주워진다.

질의 점(query point): $[q_0, q_1, \ldots, q_{d-1}]$,거리(distance): ε

구형 피라미드 기법을 이용하여 이러한 질의를 처리하기 위해서는 d-차원의 질의를 B+-트리에 적용할 수 있는 1-차원 범위 질의(1-dimensional interval query)로 변환해야한다. 그러나, 2차원 공간 예제에서처럼(그

림 6의 왼쪽), 원 형태의 질의 영역이 몇 개의 구형 피라미드와 겹칠 수 있으며, 겹치는 부분에 대한 범위를 구하는 것은 쉬운 일이 아니다. 고차원 구형태의 영역 질의를 처리하기 위해서는 다음과 같은 두 가지 과정이 필요하다.

첫 번째 과정은 먼저 어떤 구형 피라미드가 질의 영역과 겹치는지를 검사해야 한다. 두 번째 과정은 겹치는 구형 피라미드에서 질의에 의해 영향을 받는 부분의 범위를 결정해야 한다. 이때, 어떤 점 v가 해당 범위 안에 존재하는지를 검색하는 것은 d_v가 두개의 값 사이에 존재하는지를 검사하면 된다. 따라서, 이것은 1차원 색인 문제가 된다. 이러한 범위밖에 있는 객체는 해당 질의 원안에 존재하지 않는다는 것이 보장된다. 일단 범위 안에 존재하는 객체는 실제로 질의 영역 안에 존재하는지를 검사해야되는 후보 객체(candidate object)가 된다. 그림 6의 왼쪽 그림을 보면 질의 원은 두개의 구형 피라미드와 겹치게 되며, 이 두개의 구형 피라미드들 안에서 질의 영역에 의해 영향을 받는 부분을 각각 구해야 한다. 질의 점이 속해있지 않은 구형 피라미드에서는 빗금친 부분, 즉, d_{low}와 d_{high}의 범위 안에 존재하는 객체들이 바로 후보객체들이 된다. 물론, 질의 점이 속해있는 구형 피라미드에서도 d_{low}와 d_{high}를 구해야 하며, 그 범위 안에 존재하는 객체들이 후보 객체들이 된다. 이러한 후보 객체들 중에서 실제 질의 원안에 존재하는 객체는 적합한 객체(hits)이고, 그렇지 않은 객체는 적합하지 않은 객체(false hits)가 된다. 이를 검사하기 위해서는 질의 점과 후보 객체를 나타내는 점들과의 거리를 계산하여 이것이 질의 원의 반지름(ε)보다 적은지를 검사해야 하는 정제 과정(refinement step)을 거쳐야 한다. 본 연구의 실험을 통하여, 실제 이 작업을 수행하는데 많은 양의 CPU 시간이 소비된다는 것을 발견하였다. 그래서, 모든 후보 객체 점들과 질의 점과의 거리를 구하기 전에, 많은 양의 후보 객체들을 간단한 비교 연산을 통해

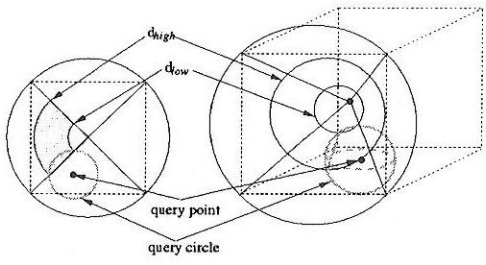


그림 6 구형태의 영역질의의 변환

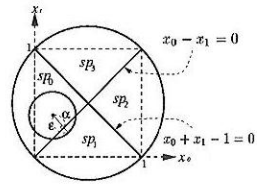


그림 7 구형 피라미드와 질의 영역의 겹침

사전에 제거하는 여과 작업(filtering)을 수행한다. 즉, 후보 객체의 i-번째 좌표가 범위 $[q_i-\varepsilon,\ q_i+\varepsilon]$안에 존재하는지를 검사하면 된다. 이러한 여과 작업은 비교 연산자만을 이용하여 수행할 수 있음으로 CPU 시간을 거의 소모하지 않으면서 많은 양의 후보 객체를 정제 과정 전에 제거할 수 있다. 후보 객체들의 수가 많고, 차원이 증가할수록 여과 작업에 의해 많이 양의 CPU 사용 시간을 줄일 수 있었다.

질의 처리 과정을 좀 더 단순화시켜 설명하기 위하여 질의 점이 속해 있는 구형 피라미드 번호가 차원 d보다 작은 경우만을 설명한다(차원 d보다 큰 경우도 유사한 방법으로 확장이 가능하다). 질의 점과 ε이 주어지면, 먼저 이 구형태의 질의 영역과 겹치는 구형 피라미드를 결정해야 한다. 이것은 한 점과 평면에 이르는 거리를 구하는 수학 공식을 이용하면 쉽게 해결된다. 그림 7의 2차원 공간 예처럼 질의 점으로부터 인접한 구형 피라미드의 측평면까지의 거리(α)가 ε보다 적으면 질의 영역과 해당 구형 피라미드(sp_1)는 겹치게 된다. 물론, 질의 점이 속해 있는 구형 피라미드는 질의 영역과 겹치게 된다. 이를 좀더 형식으로 표현하면 다음과 같다.

[정리 1] : (구형 피라미드와 질의 영역의 겹침) 질의 점($[q_0, q_1, \ldots, q_{d-1}]$)과 ε이 주어졌을 때, 질의 점이 속해 있는 구형 피라미드를 $sp_j(j<d)$라하고 질의 영역과 겹침 여부를 검사해야 하는 해당 구형 피라미드를 sp_i라고 하면 다음과 같은 경우에 질의 영역과 구형 피라미드 sp_i는 겹치게 된다.

경우 1 : ($i = j$)

경우 2 : ($|i-j| = d$) 데이타 공간의 중앙점과 질의 점까지의 거리를 β라고 할 때,

$$\beta - \varepsilon \le 0$$

경우 3 : ($i < d$)

$$\frac{|q_j - q_k|}{\sqrt{2}} \le \varepsilon$$

경우 4 : ($i \ge d$)

$$\frac{|q_j + q_i - 1|}{\sqrt{2}} \le \varepsilon$$

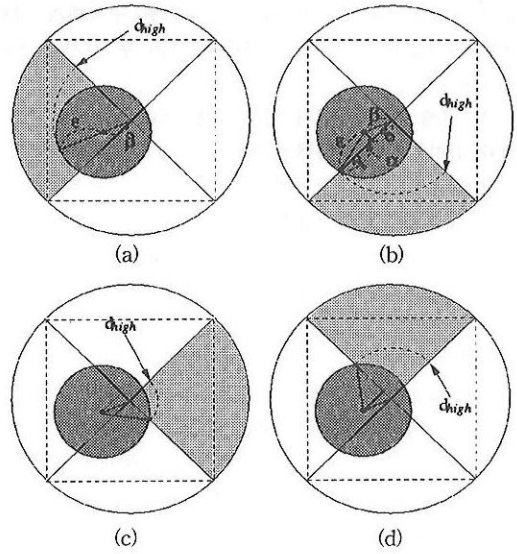
증 명 : 위상 수학에서는 한 점($[q_0, q_1, \ldots, q_{d-1}]$)과 한 평면($k_0x_0+k_1x_1+,\ldots,k_{d-1}x_{d-1}+C=0$)이 주어졌을 때, 이 점과 평면 사이의 거리는 다음과 같이 정의된다.

$$\text{거리} = \frac{|k_0q_0+k_1q_1+\ldots+k_{d-1}q_{d-1}+C|}{\sqrt{k_0{}^2+k_1{}^2+\ldots+k_{d-1}{}^2}} \quad (1)$$

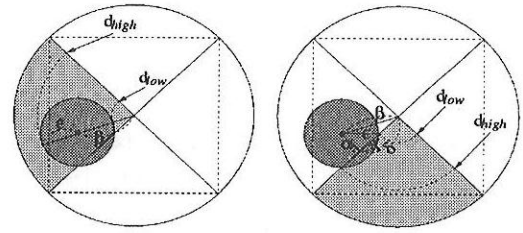
우리는 이 공식을 이용하여 **경우 3**과 **경우 4**를 증명할 수 있다.

1. $i = j$이면, sp_i는 질의 점이 속해 있는 구형 피라미드이다. 따라서, 질의 영역과 구형 피라미드 sp_i는 겹친다.
2. $|i-j| = d$이면, sp_i는 질의 점이 속해 있는 구형 피라미드 sp_j의 맞은 편에 있는 구형 피라미드가 된다. 만약, $\varepsilon \geq \beta$가 되면 중앙점이 질의 영역에 속하게 된다. 따라서, sp_i는 질의 영역과 겹치게 된다.
3. 단위 공간을 기반으로 하기 때문에 식 (1)의 색인 k_n와 상수 C 는 [-1,0,1]의 값 중에 하나를 갖는다. $i < d$ 이면, 그림 7의 2차원 공간의 예에서처럼 질의 점과 인접하는 구형 피라미드 sp_i의 한 측면의 방정식은 $k_jx_j+k_ix_i=0$이 된다. 이것은 d차원 이상의 공간에 대해서도 일관되게 확장이 가능하다. 2차원 공간인 경우에는 직선이지만, d-차원 공간인 경우에는 (d-1)-차원 평면이 된다. 이 (d-1)-차원 평면의 방정식은 k_i와 k_j를 제외한 모든 색인이 0이 되는 일반적인 특성을 가진다. 이때, $k_j=1$이고, k_i는 $i < d$이므로 -1이다. 따라서, 질의 점과 인접한 구형 피라미드 sp_i의 한 면과의 거리는 $|q_j-q_i|/\sqrt{2}$가 된다. 그러므로, 만약 $|q_j-q_i|/\sqrt{2} \leq \varepsilon$ 이면, 질의 영역의 일부가 구형 피라미드 sp_i에 존재하게 된다.
4. $i \geq d$이므로, 질의 점과 인접하는 구형 피라미드 sp_i의 한 측면의 방정식은 $k_jx_j+k_ix_i-1=0$이 된다 (그림 7 참고). 이때, $k_j=1$이고, k_i는 $i \geq d$이므로 1이다. 따라서, 질의 점과 인접한 구형 피라미드 sp_i의 한 면과의 거리는 $|q_j+q_i-1|/\sqrt{2}$가 된다. 그러므로, 만약 $|q_j+q_i-1|/\sqrt{2} \leq \varepsilon$ 이면, 질의 영역의 일부가 구형 피라미드 sp_i에 존재하게 된다. **[Q.E.D]**

만약 **경우 2**에서와 같이 구형태의 질의 영역이 중앙점을 포함하고 있다면, 모든 구형 피라미드와 겹치게 된다. 질의 점이 속해 있는 구형 피라미드가 차원 d 보다 큰 경우도, 이와 유사한 방법으로 확장이 가능하다.



(I) 중앙점이 질의 영역에 포함되는 경우



(II) 중앙점이 질의 영역에 포함되지 않는 경우

그림 8 d_{low}와 d_{high}를 결정하는 과정

두 번째 단계에서는 질의 영역에 의해 영향을 받는 구형 피라미드 sp_i안에서 어떤 구형 피라미드 값이 질의 영역 안에 존재하는지를 결정해야 한다. 즉, $[0, 0.5\sqrt{d}]$의 범위 안에서 $[d_{low}, d_{high}]$를 구해야 한다. 따라서, 질의 원과 구형 피라미드 sp_i가 겹치는 부분에 존재하는 모든 점들의 구형 피라미드 값은 $[i \cdot \lceil\sqrt{d}\rceil + d_{low},\ i \cdot \lceil\sqrt{d}\rceil + d_{high}]$범위 안에 존재하게 된다. 그림 6은 2차원과 3차원에서 이러한 범위를 보여주고 있다. 우리는 피타고라스의 정리를 이용하여 d_{low}와 d_{high}를 구할 수 있다.

중앙점이 질의 영역에 포함된 경우에 d_{low}는 항상 0

이다. 따라서 d_{high}만을 구하면 된다. 그림 8(I)의 (a)와 같이 질의 점이 속해 있는 구형 피라미드에서의 d_{high}는 중앙점과 질의 점까지의 거리와 질의 원의 반지름인 ε을 이용하면 된다. 즉, 중앙점과 질의 점까지의 거리를 β라고 하면 $d_{high}=(\beta+\varepsilon)$가 된다. 그리고, 그림 8(I)의 (b)나 (d)와 같이 질의 점이 속해 있지 않은 구형 피라미드인 경우에는 좀더 복잡하지만 피타고라스의 정리를 이용하여 질의 영역이 해당 구형 피라미드와 만나는 지점을 정확히 구할 수 있다. 질의 점과 해당 구형 피라미드의 측면까지의 거리를 α라 하면 α와 β로 이루어지는 직각 삼각형의 밑변의 길이 δ는 $\sqrt{\beta^2-\alpha^2}$이다. 그리고, α와 ε으로 이루어지는 직각 삼각형의 밑변의 길의 γ는 $\sqrt{\varepsilon^2-\alpha^2}$이다. 이때, $d_{high}=(\delta+\gamma)$가 된다. 마지막으로, 그림 8(I)의 (c)와 같이 질의 점이 속해 있는 구형 피라미드의 맞은 편에 있는 구형 피라미드인 경우($|j-i|=d$)에, $d_{high}=(\gamma-\delta)$가 된다. 그러나, 이 경우에 주의해야할 것은 질의 점과 인접한 구형 피라미드까지의 거리 α는 인접한 모든 구형 피라미드의 측면까지의 거리들 중에서 최대값이 되는 것이어야 한다.

중앙점이 질의 영역 안에 존재하지 않은 경우도 위와 유사한 방법으로 d_{low}와 d_{high}를 구할 수 있다. 그림 8(II)의 (a)와 같이 질의 점이 속해 있는 구형 피라미드인 경우에 $d_{low}=(\beta-\varepsilon), d_{high}=(\beta+\varepsilon)$이 된다. 그림 8(II)의 (b)와 같이 질의 점이 속해 있지 않은 구형 피라미드인 경우에는 그림 8(I)의 (b)나 (d)의 경우와 유사한 방법으로 구할 수 있으며, 그 값은 $d_{low}=(\delta-\gamma), d_{high}=(\delta+\gamma)$가 된다. 이것을 좀더 형식으로 정리하면 다음과 같다.

[정리 2] : (질의 영역과 구형 피라미드가 겹치는 범위) 질의 점이 속해 있는 구형 피라미드 sp_i와 질의 영역에 의해 겹침이 발생하는 구형 피라미드 sp_j가 주어졌을 때 겹치는 범위 $[d_{low}, d_{high}]$는 다음과 같이 결정된다.

경우 1 : 중앙점이 질의 영역 안에 존재할 때,

경우 1.1 : ($j = i$) 질의 점과 중앙점 사이의 거리를 β라하고 질의 영역의 반지름을 ε이라고 할 때,

$$d_{low}=0, \quad d_{high}=\beta+\varepsilon$$

경우 1.2 : ($|j-i| = d$) 질의 점과 모든 구형 피라미드의 측면까지의 거리들 중에서 최대값을 α라하고, α와 β로 이루어지는 직각 삼각형의 밑변의 길이를 δ라고 한다. 그리고, α와 ε으로 이루어지는 직각 삼각형의 밑변의 길이를 γ라고 하면,

$$d_{low}=0, \quad d_{high}=\gamma-\delta=\sqrt{\varepsilon^2-\alpha^2}-\sqrt{\beta^2-\alpha^2}$$

경우 1.3 : (*otherwise*) 질의 점과 인접한 해당 구형 피라미드 측면까지의 거리를 α라하고, δ와 γ를 경우 1.2와 같다고 하면,

$$d_{low}=0, \quad d_{high}=\delta+\gamma=\sqrt{\beta^2-\alpha^2}+\sqrt{\varepsilon^2-\alpha^2}$$

경우 2 : 중앙점이 질의 영역 안에 존재하지 않을 때,

경우 2.1 : ($j = i$)

$$d_{low}=\beta-\varepsilon, \quad d_{high}=\beta+\varepsilon$$

경우 2.2 : ($j \neq i$)

$$d_{low}=\delta-\gamma, \quad d_{high}=\delta+\gamma$$

증 명 : 우리는 어떤 점 v가 질의 영역과 영향을 받은 구형 피라미드 sp_i안에 존재하면, d_v가 질의 범위 $[d_{low}, d_{high}]$안에 존재함을 보여야 한다. 즉, 우리는 다음과 같은 조건이 만족됨을 보여야 한다.

$$d_{low} \le d_v \le d_{high}$$

1. $d_{low} \le d_v$: 질의 영역이 중앙점을 포함하고 있으므로, $d_{low}=0 \le d_v$ 이다.

1.1 $d_v \le d_{high}$: 질의 영역과 영향을 받는 구형 피라미드 sp_i안에 존재한 점들 중에서 최대 거리를 갖는 점 v의 거리는 $d_v \le \beta+\varepsilon$이다. 따라서, $d_v \le d_{high}=(\beta+\varepsilon)$ 이다(그림 8(I)의 (a) 참고).

1.2 $d_v \le d_{high}$: 이 경우에 구형 피라미드 sp_i는 질의 점이 속해 있는 구형 피라미드 sp_j의 맞은 편에 있기 때문에, 질의 점과 직접적으로 인접하는 측면이 존재하지 않는다. 따라서, 질의 점과 모든 구형 피라미드의 측면들과의 거리 중에서 최대값을 α라 하면, α와 β로 이루어지는 직각 삼각형의 밑면 δ는 $\sqrt{\beta^2-\alpha^2}$이 된다. 또한 α와 ε으로 만들어지는 직각 삼각형의 밑면 γ는 $\sqrt{\varepsilon^2-\alpha^2}$이다. 점 v를 질의 영역과 영향을 받는 구형 피라미드 안에 존재하는 점들 중에서 거리가 최대인 점이라고 하면, $d_v \le \gamma-\delta$가 된다. 따라서, $d_v \le d_{high}=(\gamma-\delta)$ 이다(그림 8(I)의 (c) 참고).

1.3 1.1과 1.2의 경우를 제외하면, 질의 점과 인접한 구형 피라미드 측면이 존재한다. 질의 점과 인접한 구형 피라미드의 측면까지의 거리를 α라 하

면, δ와 γ는 1.2의 경우와 마찬가지로 피타고라스의 정리에 의해 구할 수 있다. 만약 점 v를 질의 영역과 영향을 받는 구형 피라미드 안에 존재하는 점들 중에서 거리가 최대인 점이라고 하면, $d_v \le \delta+\gamma$가 된다. 따라서, $d_v \le d_{high}=(\delta+\gamma)$ 이다(그림 8(I)의 (b), (d) 참고).

2. $d_{low} \le d_v \le d_{high}$: 질의 영역이 중앙점을 포함하고 있지 않기 때문에, 질의 점이 속해 있는 구형 피라미드의 맞은 편에 있는 구형 피라미드는 질의 영역에 의해 영향을 받지 않는다. 또한, $d_{low} \neq 0$ 이므로, d_v는 d_{low}보다 크거나 같고, d_{high}보다 작거나 같아야 한다.

2.1 만약, 점 v를 질의 영역과 영향을 받는 구형 피라미드 sp_i안에 존재하는 점들 중에서 거리가 최소인 점이라고 하면, $d_v \ge \beta-\varepsilon$이 된다. 또한, 만약 점 v를 질의 영역과 영향을 받는 구형 피라미드 sp_i안에 존재하는 점들 중에서 거리가 최대인 점이라고 하면, $d_v \le \beta+\varepsilon$이 된다. 따라서, $(\beta-\varepsilon)=d_{low} \le d_v \le d_{high}=(\beta+\varepsilon)$이 된다(그림 8(II)의 (a) 참고).

2.2 δ와 γ를 1.3의 경우와 마찬가지로 각 직각 삼각형의 밑변들이라고 하자. 점 v를 질의 영역과 영향을 받는 구형 피라미드 sp_i안에 존재하는 점들 중에서 거리가 최소인 점이라고 하면, $d_v \ge \delta-\gamma$가 된다. 또한, 만약 점 v를 질의 영역과 영향을 받는 구형 피라미드 sp_i안에 존재하는 점들 중에서 거리가 최대인 점이라고 하면, $d_v \le \delta+\gamma$가 된다. 따라서, $(\delta-\gamma)=d_{low} \le d_v \le d_{high}=(\delta+\gamma)$이 된다(그림 8(II)의 (b) 참고). **[Q.E.D]**

만약 $\beta+\varepsilon > 0.5\sqrt{d}$이거나 $\delta+\gamma > 0.5\sqrt{d}$이면, d_{high}는 당연히 $0.5\sqrt{d}$가 된다. 구형 피라미드 기법에서 고차원 구형태의 영역 질의를 처리하는 과정은 먼저 [정리 1]을 사용하여 구형태의 질의 영역과 겹치는 구형 피라미드들을 결정하고, [정리 2]를 이용하여 질의 영역에 의해 영향을 받는 구형 피라미드에서 d_{low}와 d_{high}를 구한 후에 이 값을 이용하여 B+-트리에 대하여 1차원 범위 질의를 수행하는 것이다. 그림 9는 고차원 구형태의 질의를 처리하는 과정을 보여주는 알고리즘이다.

6. 실험 및 분석

구형 피라미드 기법의 실제적인 성능을 보이기 위하여 다양한 실험을 수행하였다. [1]의 연구에서 피라미드 기법의 성능이 기존에 존재하는 다른 고차원 색인 기법들에 비하여 모든 측면에서 월등함을 보였다. 따라서, 본 논문에서는 피라미드 기법과의 성능 비교를 통하여 구형 피라미드 기법이 여러 측면에서 더 효율적임을 보인다.

본 실험은 10 기가(Giga) 바이트의 보조 기억 장치를 가지는 SGI Onyx2 웍스테이션 상에서 실험을 수행하였다. 실험은 인위적으로 생성된 데이타의 집합과 실제 데이타 집합을 사용하였으며, 모든 실험에서는 일정

```
Point_Set SP_Tree::hypersphere_query(point qp,
                                     range ep)
{
  Point_Set  res;
  for ( i=0; i<2d; I++ ) {
    if (intersect(p[i], qp, ep)) {
      // 정리 1을 이용
      determine_interval(p[i], qp, ep, dlow, dhigh)
      // 정리 2를 이용
      cs = btree_interval_query( i · ⌈√d⌉ + d low,
                                 i · ⌈√d⌉ + d high)
      for ( c=cs.first; cs.end; cs.next ) {
        if (inside_DiameterOfCircle(c, qp, ep))
          // 필터링 단계
          if (point_in_circle(c, qp, ep))
            // 정제 단계
            res.add(c)
      }
    }
  }
  return res;
}
```

그림 9 고차원 구형태의 질의 처리 알고리즘

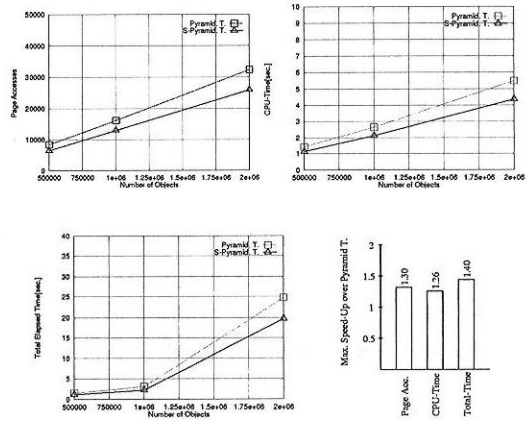


그림 10 데이타 크기에 따른 성능 비교

한 선택도[1]를 가지는 구형태의 영역 질의를 수행하였다.

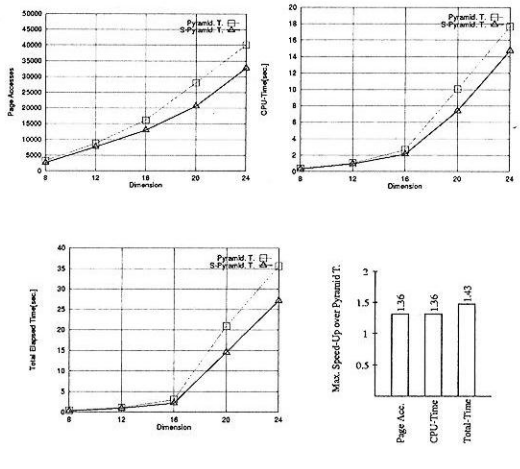


그림 11 차원 크기에 따른 성능 비교

구형태의 질의 영역은 전체 데이타 공간상에서 무작위로 선출되었으며, 질의의 분포는 데이타 자체의 분포와 동일하게 하였다. 즉, 균등하게 분포된 데이타의 집합에 대해서는 균등하게 분포된 고차원 구형태의 질의를 사용하였다. 인위적으로 생성된 데이타들은 500,000~2,000,000개의 균등하게 분포된 데이타들이며 각 데이타의 차원은 4~24이다.

첫 번째 실험은 데이타의 갯수를 변화시키면서 성능의 변화를 측정하는 것이다. 16차원의 데이타 공간에 대하여 0.001%의 선택도를 유지하는 구형태의 질의에 대하여 실험을 하였으며, 데이타의 갯수는 500,000개에서 2,000,000개까지 변화시키면서 실험을 하였다. 이 실험에 사용된 데이타와 색인 화일의 크기는 총 0.9 기가 바이트이다. 페이지 크기는 4096 바이트이며, 효율적인 페이지 용량은 한 페이지 당 37개의 객체가 들어간다. 피라미드 기법과의 비교를 위하여 동일한 크기의 구형태의 질의를 처리하는데 소요되는 페이지 접근 횟수, CPU 사용 시간, 전체 응답 시간을 측정하였다. 전체 응답 시간은 CPU 사용 시간과 I/O처리에 소요되는 시간을 합한 것이 된다. 그림 10은 첫 번째 실험 결과를 보여주는 그림이다. 전반적으로 데이타의 갯수가 증가할수록 구형 피라미드의 성능이 피라미드 기법보다 좋아짐을 알 수 있었다. 데이타의 갯수에 따라서 페이지 접근 횟수는 기존의 피라미드 기법에 비하여 최소 24%, 최고 30% 정도의 페이지를 덜 접근하게 된다. CPU 사용 시간은 피라미드 기법보다 약 24%~26% 정도의 성능 향상이 있었다. 구형 피라미드 기법의 질의 처리 알고리즘 자체는 피라미드 기법보다 복잡하지만, 피라미드 기법이 구형 피라미드 기법보다 후보 객체를 더 많이 생성하기 때문에, 그만큼 정제 단계에서 CPU를 더 사용하게 된다. 마지막으로 전체 질의 응답 시간도 구형 피라미드 기법이 피라미드 기법보다 약 25%~40% 정도 좋아짐을 볼 수 있었다.

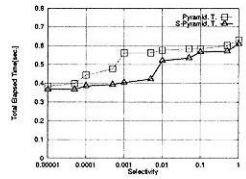


그림 12 실제 데이타를 이용한 실험

두 번째 실험은 데이타의 차원을 변화시키면서 질의 처리에 들어가는 비용을 측정하였다. 데이타의 갯수는 1,000,000개로 고정시켰으며, 차원은 8, 12, 16, 20, 24로 변경하면서 실험을 수행하였다. 데이타 화일과 색인 화일의 크기는 1.44 기가 바이트이며, 페이지의 크기는 첫 번째 실험과 마찬가지로 4096 바이트이다. 효율적인 페이지 용량은 차원의 크기에 따라서 페이지당 25~74개가 된다. 데이타의 차원에 따라 질의 영역의 크기를 변경함으로써 0.001%의 선택도를 유지하도록 하였다. 그림 11은 실험 결과를 보여주고 있다. 전반적으로 차원이 증가할수록 구형 피라미드 기법이 피라미드 기법보다 더 좋은 성능을 보였다. 데이타의 차원에 따라서 페이지 접근 횟수는 피라미드 기법보다 약 22%~36% 정도 덜 접근하며, CPU 사용 시간은 약 22%~36% 정도 좋아졌다. 마지막으로, 전체 응답 시간은 피라미드 기법보다 약 26%~43% 정도의 성능 향상이 있음을 알 수 있었다.

마지막 실험은 균등 분포를 따르는 인위적인 데이타가 아닌 실제 데이타들을 이용한 실험이다. 이 실험을 위하여 실제 내용 기반 이미지 검색 시스템 SCARLET [15]에서 사용되는 이미지들의 특징 벡터들을 이용하였다. 이 특징 벡터들은 56,230개의 이미지들로부터 웨이브릿 변환(wavelet transform)을 이용하여 자동으로 추출되었으며, 각 특징 벡터는 16차원으로 구성되어 있다.

그림 12는 10^{-5}에서 1까지의 선택도를 가지는 고차

1) 선택도 = (질의를 만족하는 객체의 수)/(데이타베이스에 저장된 전체 객체의 수)

원 구형태의 질의를 처리할 때 소요되는 총 시간을 측정한 것이다. 선택도가 아주 작거나 클 경우에 구형 피라미드 기법과 기존의 피라미드 기법이 거의 차이가 없음을 알 수 있었다. 이것은 선택도가 아주 작은 경우에는 페이지 접근 횟수가 아주 적고, 후보 객체의 수도 아주 적기 때문이다. 또한, 선택도가 아주 클 경우에는 구형 피라미드 기법이나 피라미드 기법이 거의 모든 데이타 페이지를 접근하게 되며, 따라서 거의 모든 객체들을 후보 객체로 가지기 때문이다. 선택도가 100%인 경우, 즉, 모든 객체를 검색해야 할 경우에는 구형 피라미드 기법이나 기존의 피라미드 기법이 거의 동일한 시간을 필요로 한다는 사실을 알 수 있었다. 그러나, 합리적인 선택도라고 할 수 있는 0.01%~5%인 경우에는 구형 피라미드 기법이 기존의 피라미드 기법보다 더 좋은 성능을 보임을 알 수 있었다. 특히, 선택도가 0.05%~0.5%인 경우에는 성능 차이가 더 심했으며, 최고 40% 정도의 성능 향상이 있었다.

실제 데이타를 이용한 실험을 통하여 구형 피라미드 기법이 균등하게 분포된 인위적인 데이타를 이용한 실험과 마찬가지로 기존의 피라미드 기법보다 효율적임을 알 수 있었다.

7. 결 론

본 논문에서는 기존의 피라미드 기법을 확장하여 내용 기반 검색 응용 등에 많이 사용되는 유사 검색에 적합한 새로운 색인 방법으로 구형 피라미드 기법을 제안하였다.

구형 피라미드 기법은 고차원 구형태의 영역 질의에 적합한 새로운 공간 분할 방식에 기반을 두고 있다. 이 방법은 피라미드 기법과 마찬가지로 d-차원의 공간을 1-차원 공간으로 변환하여 B+-트리를 이용함으로써 B+-트리의 모든 장점을 이용할 수 있다. 본 논문에서는 또한 구형 피라미드 기법이 구형태의 질의를 처리하는데 있어서 기존의 피라미드 기법보다 효율적임을 다양한 실험을 통하여 보였다.

최근 고차원 공간상에서 구형태의 질의를 처리하는데 소요되는 비용 모델에 관한 몇몇 연구가 수행되어 왔다[4,9]. 비록 이러한 연구들이 고차원 공간상에서 구형태의 질의를 처리할 경우, 빈번히 발생하는 경계 효과(bound effect)를 적절히 고려한 비용 모델을 제시하였으나, 색인 페이지의 모양이 사각형이나 구형태인 경우만을 고려한 모델들이기 때문에, 구형 피라미드 기법의 비용 모델을 설정하는데 이용할 수 없다. 구형 피라미드 기법을 위해서는 새로운 비용 모델이 필요하며, 이것은 매우 복잡하고 어려운 작업이다.

본 연구에서는 향후 연구 과제로 구형 피라미드 기법의 비용을 정확히 예측할 수 있는 새로운 비용 모델에 관한 연구와 구형 피라미드 기법을 이용하여 또 다른 형태의 유사성 질의인 k-최근접 질의를 처리할 수 있는 효율적인 알고리즘에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] S. Berchtold, C. Bohm, H.-P Kriegel. "The Pyramid-Technique: Towards Breaking the Curse of Dimensionality". Proc. ACM SIGMOD Int Conf. on Management of Data, 1998.

[2] S. Berchtold and D. A. Keim. "High-Dimensional Index Structures Database Support for Next Decade's Applications". Tutorial, Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998.

[3] D. A. White, and R. Jain. "Similarity Indexing with the SS-tree". Proc 12th Int. Conf on Data Engineering, pages 516--523, 1996.

[4] S. Berchtold, C. B"ohm, D. A. Keim, and H.-P. Kriegel "A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space". ACM PODS Symposium on Principles of Database Systems, Tucson, Arizona, 1997.

[5] S. Berchtold, D. A. Keim, and H.-P. Kriegel "The X-tree: An Indexing Structure for High-Dimensional Data". Proc. 22nd Int. Conf. on Very Large Database, pages 28--39, September 1996.

[6] C. Faloutsos. "Fast Searching by Content in Multimedia Databases". Data Engineering Builetin, 18(4), 1995.

[7] D. A. White, and R Jain. "Similarity Indexing: Algorithms and Performance". Proc. SPIE Storage and Retrieval for Image and Video Databases IV Vol. 2670, San Diego, USA, pages 62--75, 1996.

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

[9] R. Weber, H.-J. Schek, and S. Blott. "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces". Proc. 24nd Int. Conf. on Very Large Database, pages 194--205, August 1998.

[10] S. Berchtold, D. Keim, H.-P Kriegel, and T. Seidl. "Fast Nearest Neighbor Search in High-Dimensional Spaces" Proc. 14th Int. Conf on Data Engineering, Orlando, 1998.

[11] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W Equiz. "Efficient and Effective Querying by Image Content" Journal of Intelligent Information System(JIIS), 3(3):231--262,

July 1994.
[12] A. Guttman. "R-trees: a dynamic index structure for spatial searching". Proc. ACM SIGMOD Int. Conf. on Data Management of Data, pages 47--57, June 1984.
[13] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. "The R*-tree: an efficient and robust access method for points and rectangles". Proc. ACM SIGMOD Int. Conf. on Data Management of Data, pages 322--331, May 1990.
[14] N. Katayama and S. Satoh. "The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries". Proc. ACM SIGMOD Int. Conf. on Data Management of Data, pages 517--542, May 1997.
[15] 이 동호, 송 용준, 김 형주. "SCARLET: 웨이브릿 변환을 이용한 내용 기반 이미지 검색 시스템의 설계 및 구현". 한국정보과학회 논문지(C), 3(4), 1997.

이 동 호
제 26 권 제 1 호(B) 참조

정 진 완
제 26 권 제 1 호(B) 참조

김 형 주
제 26 권 제 1 호(B) 참조