구형 피라미드 기법을 이용한 최근접 질의 처리 기법

(Nearest Neighbor Query Processing using the Spherical Pyramid Technique)

이 동 호 † 김 형 주 ††
(Dong-Ho Lee) (Hyoung-Joo Kim)

요 약 구형 피라미드 기법[1,2]은 d-차원의 공간을 2d개의 구형 피라미드들로 분할하는 특별한 공간 분할 방식을 이용하여 고차원 데이타를 효율적으로 색인할 수 있는 새로운 색인 방법으로 제안되었다. 구형 피라미드 기법은 구형태의 영역질의를 처리하는 알고리즘을 제안하였으나 유사 검색에 많이 사용되는 또 다른 종류의 질의인 최근접 질의를 처리하는 알고리즘을 제안하지 못했다. 본 논문에서는 점진적 최근접 질의 처리 알고리즘을 확장하여 구형 피라미드 기법 상에서 효율적으로 최근접 질의를 처리하는 알고리즘을 제안한다. 마지막으로, R*-tree와 X-tree상에서 구현된 점진적 k-최근접 질의 처리 방법과 다양한 비교 실험을 통하여 구형 피라미드 기법을 이용한 k-최근접 질의 처리 방법이 더 효율적임을 보인다.

Abstract The Spherical Pyramid-Technique[1,2] was proposed as a new indexing method for high-imensional data spaces using a special partitioning strategy that divides d-dimensional space into 2d spherical pyramids. Although the authors of [1,2] proposed an efficient algorithm for processing hyperspherical range queries, they did not proposed the algorithm for processing k-nearest neighbor queries that are frequently used in similarity search. In this paper, we propose an efficient algorithm for processing nearest neighbor queries on the Spherical Pyramid-Technique by extending the incremental nearest neighbor algorithm. Finally, we show that our method clearly outperforms the related methods in processing k-nearest neighbor queries through various experiments by comparing it to the R*-tree and X-tree.

1. 서 론

특징에 기반한 유사 검색은 데이타베이스 시스템의 중요한 검색 기법 중에 하나로 대두되고 있다. 이는 특정 데이타를 고차원 공간상의 하나의 점으로 변환하여, 다차원 색인 구조를 이용하여 색인하고 검색하는 기법이다. 이러한 검색 기법이 가장 많이 사용되는 응용으로는 내용 기반 멀티미디어 정보 검색을 들 수 있다. 내용 기반 멀티미디어 정보 검색은 멀티미디어 데이타로부터 내용으로 추정되는 특징 데이타를 추출하여 이를 기반으로 검색하는 기법인데, 일반적으로 멀티미디어 데이타에서 추출한 특징 데이타들은 고차원 벡터 형태로 표현된다[3,4]. 또한, 이러한 응용에서는 데이타베이스에 저장된 객체들 중에서 질의 객체와 가장 유사한 객체들을 검색하는 질의로 k-최근접 질의와 구형태의 영역 질의를 사용한다[5].

k-최근접 질의는 질의 객체와 가장 유사한 k-개의 객체를 검색하는 것이고 구형태의 영역 질의는 질의 객체와 일정한 유사성 허용 오차를 만족하는 모든 객체들을 검색하는 방법이다. 효율적인 유사 검색을 지원하기 위해서는 고차원 데이타를 효율적으로 색인할 수 있는

· 본 연구는 교육부 BK21 정보기술 사업단에서 지원 받았음
† 비 회 원 : 서울대학교 컴퓨터공학부
dhlee@oopsla.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjkim@oopsla.snu.ac.kr
논문접수 : 2000년 6월 28일
심사완료 : 2000년 11월 15일

색인 구조와 이러한 색인 구조상에서 위와 같은 유사 질의를 효율적으로 처리할 수 있는 알고리즘이 필수적이다. 그러나 최근 연구 결과에 의하면 저차원이나 중차원 데이타에 대하여 효율적인 색인을 제공하는 어떤 색인 구조도 고차원 데이타에 대해서는 효율적인 색인을 제공하지 못하고 있다[3,5,6,7]. 이러한 문제를 일반적으로 "차원의 저주(curse of dimensionality)"[7]라 하며, 최근 데이타베이스 관련 연구 과제에서는 이러한 문제를 해결하고자 하는 노력을 수행하고 있다.

본 논문의 초기 연구[1,2]에서는 고차원 특징 데이타를 효율적으로 색인하고 유사 검색에 많이 사용되는 구형태의 영역질의를 효율적으로 처리할 수 있는 알고리즘으로 구형 피라미드 기법(SPY-TEC)을 제안하였다. 그러나, 유사 검색에 많이 사용되는 또 다른 종류의 질의인 k-최근접 질의를 처리하기 위한 알고리즘을 제안하지 못했다. 본 논문에서는 구형 피라미드 기법을 이용하여 질의 객체와 가장 유사한 k-개의 객체를 검색하는 k-최근접 질의 처리를 위한 알고리즘을 제안한다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 그동안 수행되어 왔던 최근접 질의 처리에 대한 관련 연구를 소개하고, 3장에서는 구형 피라미드 기법에 대하여 간단히 설명한다. 4장에서는 본 논문에서 제안하는 k-최근접 질의 처리 알고리즘을 설명하고, 5장에서는 인위적 데이타와 실제 데이타를 이용한 다양한 실험 및 분석에 대하여 설명하고, 마지막으로 6장에서는 결론 및 향후 연구 방향에 대해서 설명한다.

2. 관련 연구

최근접 질의 혹은 k-최근접 질의를 처리하기 위한 알고리즘은 GIS응용이나 패턴 인식, 서류 검색, 학습 이론 등의 분야에서 그 필요성에 의해 많은 연구가 이루어졌다. 이러한 알고리즘은 d-차원의 벡터 공간에서 점 객체나 임의의 공간 객체에 대하여 질의 객체와 가장 유사한 객체들을 찾기 위해 개발되었으며, 대부분 일정한 공간 자료 구조를 기반으로 하고 있다. 예를 들어, k-d tree[9]에 기반한 알고리즘과 quad-tree[10]에 기반한 알고리즘, 그리고, R-tree[11]에 기반한 알고리즘 등이 있으며 이러한 알고리즘들은 대부분 수정을 통하여 다른 공간 자료 구조상에서도 자연스럽게 적용될 수 있다.

[12]의 연구에서는 ***mindist***와 ***minmaxdist***를 이용하여 R-tree상에서 ***branch&bound*** 알고리즘으로 질의 객체와 가장 유사한 k개의 객체를 검색하는 알고리즘을 제안하였다. 이 알고리즘의 기본적인 아이디어는 R-tree를 깊이 우선 방식으로 탐색하면서 후보가 될 수 있는 객체들을 활성 리스트(Active Branch List)에 유지하는 것이다. 이 알고리즘은 k가 이미 결정된 상태에서 알고리즘이 시작된다. 즉, 사전에 k가 이미 결정되기 때문에 만약 사용자가 (k+1)번째 객체를 얻고자 하다면 알고리즘을 처음부터 다시 시작해야 하는 단점이 존재한다.

Hjaltason과 Samet[13]은 이러한 단점을 제거하고자 사전에 k 값을 결정하는 것이 아니라 사용자의 요구에 따라 한 개씩 점진적으로 질의 결과를 얻어내는 점진적 최근접(incremental nearest neighbor) 질의 처리 알고리즘을 제안하였다. 공간 데이타베이스에서 사용자는 질의 객체와 가장 가까운 객체들을 하나씩 차례로 얻고자 하는 '거리 브라우징(distance browsing)' 요구를 많이 하게 된다. 점진적 최근접 알고리즘은 이러한 응용을 위하여 고안되었다. 이 알고리즘은 순위 큐(priority queue)를 사용하여 항상 질의 객체와 가장 가까운 객체가 큐의 맨 앞에 오게 함으로써, 사용자가 원할 때마다 큐의 맨 앞에 있는 원소를 반환하여 차례로 가장 가까운 객체들을 질의 결과로 보여준다. 따라서, 사용자는 알고리즘을 처음부터 재시작할 필요 없이, 항상 자신이 원하는 만큼의 객체를 얻을 수 있다. 즉, k개의 객체를 검색한 후, (k+1)번째 객체를 얻기 위해 알고리즘을 처음부터 재시작할 필요 없이 순위 큐에 남아 있는 다음 객체를 반환하면 된다. [13]의 연구에서는 점진적 최근접 알고리즘을 이용하여 k-최근접 질의를 처리할 경우에도 기존의 R*-tree상의 k-최근접 질의 처리 알고리즘보다 효율적임을 보였다. 그러나, 이 알고리즘도 고차원 데이타에 대해서는 효율적이지 못하다. 이것은 알고리즘 자체의 문제라기 보다는 R-tree라는 공간 자료 구조가 고차원 공간상에서 효율적이지 못하기 때문이다.

본 연구에서는 최근 고차원 데이타를 위한 효율적인 색인 구조로 제안된 구형 피라미드 기법을 이용하여 점진적 최근접 질의를 처리하기 위한 알고리즘을 제안한다.

3. 구형 피라미드 기법

[1,2]의 연구에서 구형 피라미드 기법의 기본적인 아이디어와 공간 분할 방법 및 색인 방법, 구형태 영역질의 처리 방법을 자세히 설명하였으나 본 논문에서 제시하는 알고리즘의 이해를 돕기 위하여 다시 한번 간단히 설명한다.

구형 피라미드 기법은 d-차원의 점을 1-차원 값으로 변환하여 B+-트리와 같은 효율적인 1-차원 색인 구조를 사용하여 1-차원 값들을 저장하고 접근한다. 이러한 변환은 2단계로 이루어진다. 첫 번째 단계에서는 d-차

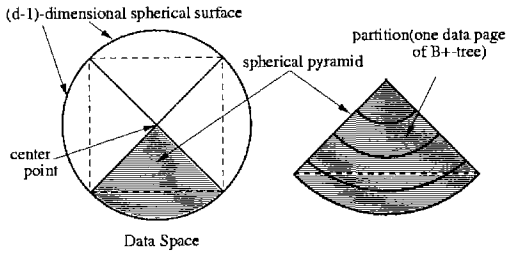


그림 1 구형 피라미드 기법의 공간 분할 전략

원의 데이타 공간을 2d개의 구형 피라미드들로 분할한다. 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 1은 2차원 공간상에서 구형 피라미드 기법의 공간 분할을 보여주고 있다. 먼저, 2차원 공간은 4개의 구형 피라미드들로 분할되며, 각 구형 피라미드는 동일하게 공간의 중앙점을 상단점으로 가지고 하나의 곡선을 기저로 가진다. 이것은 d-차원으로 일관되게 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 1의 오른쪽 그림을 보면, 하나의 구형 피라미드가 4개의 구형 조각(bounding slice)들로 분할된 것을 볼 수 있다. 각 구형 조각은 B+-트리에서 하나의 데이타 페이지를 구성하게 된다. d-차원의 구는 2d개의 (d-1)-차원의 구형 평면을 가지기 때문에 2d개의 구형 피라미드를 얻을 수 있다[2,6].

다음의 정의 1과 2는 구형 피라미드 기법의 공간 분할 전략의 첫 번째와 두 번째 단계에 상응한다. 즉, 그림 2에서와 같이, 먼저 정의 1을 이용하여 어떤 점 v가 속해 있는 구형 피라미드의 번호를 결정한다. 그리고, 정의 2에 의해 점 v의 위치를 결정한다.

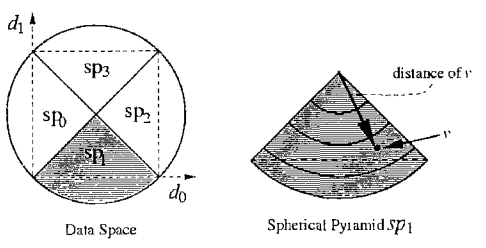


그림 2 구형 피라미드의 번호 결정 과정과 존재하는 점의 거리

[정의 1] (점 v가 속해 있는 구형 피라미드) d-차원의 점 v는 구형 피라미드 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 2] (점 v의 거리) d-차원의 점 v가 주어졌을 때, 점 v의 거리는 다음과 같이 정의된다.

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

마지막으로 정의 3은 정의 1과 2를 이용하여 d-차원의 점 v를 1차원 값($i \cdot \lceil \sqrt{d} \rceil + d_v$)으로 변환한다.

[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 할때, 점 v의 구형 피라미드 값은 다음과 같이 정의된다.

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

예를 들어, 2차원 점 $v=(0.4, 0.8)$가 주어질 경우, $j_{\max}$는 1이고 $v_1(=0.8)$이 0.5보다 크기 때문에 정의 1에 의하여 점 v는 $sp_{(1+2)}$에 속하게 된다. 또한, 정의 2에 의해 중앙으로부터 점 v까지의 거리는 $\sqrt{0.1}$이 된다. 따라서, 점 v의 구형 피라미드 값은 [정의 3]에 의해 $(3 \cdot \lceil \sqrt{2} \rceil + \sqrt{0.1})$이 된다. 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.

4. 점진적 최근접 질의 처리

그림 3은 2차원 공간상에서 구형 피라미드의 예를 보여 주고 있다. 각 구형 조각에는 1개의 객체만 들어간다고 가정하자. 그림 3에서 질의 객체(q)는 구형 피라미드 sp_1상의 구형 조각 BS_4에 존재함을 알 수 있다. 대부분의 최근접 질의 처리 알고리즘이 그렇듯이 SPY-TEC상에서의 최근접 질의 처리 알고리즘도 질의 객체가 속해 있는 데이타 페이지에 있는 객체들을 먼저 검색한다. 그러나, 효율적인 검색을 위해서는 질의 점과

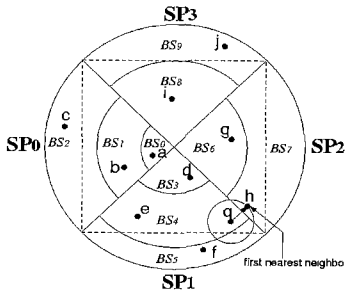


그림 3 10개의 객체가 존재하는 SPY-TEC의 예

구형 피라미드 사이의 최소 거리와 질의 점과 구형 조각 사이의 최소 거리를 측정할 필요가 있다. 이러한 거리들을 순서화함으로써 검색시 불필요하게 방문하는 페이지들의 수를 줄일 수 있다. 다음의 [정리 1]은 질의 점과 구형 피라미드 사이의 최소 거리를 위한 정리이다. 이러한 거리 측정 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[1]).

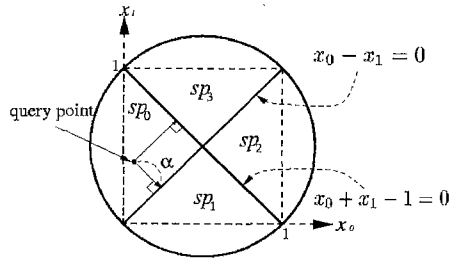


그림 4 질의 점과 인접 구형피라미드 사이의 최소 거리

[정리 1] (질의 점과 구형 피라미드 사이의 거리) 질의 점($q=[q_0, q_1, \ldots, q_{d-1}]$)이 주어졌을 때, 질의 점이 속해 있는 구형 피라미드를 $sp_j(j<d)$라 하면, 질의 점과 구형 피라미드 sp_i와의 최소 거리 $MINDIST(q, sp_i)$는 다음과 같이 정의된다.

$$MINDIST(q, sp_i) = \begin{cases} 0 & \text{if } i=j \\ d_q & \text{if } |i-j|=d \\ \dfrac{|q_j - q_i|}{\sqrt{2}} & \text{if } i<d \\ \dfrac{|q_j + q_i - 1|}{\sqrt{2}} & \text{if } i>d \end{cases}$$

증명 : 위상 수학에서 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 최소 거리는 점에서 평면에 이르는 수직선으로 다음과 같이 정의된다.

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

우리는 이 공식을 이용하여 (i>d)인 경우와 (i<d)인 경우를 증명할 수 있다. 먼저, $i=j$이면 sp_i는 질의 점이 속해 있는 구형 피라미드이다. 따라서, $MINDIST(q, sp_i)=0$이다. $|i-j|=d$인 경우, sp_i는 질의 점이 속해 있는 구형 피라미드 sp_j의 맞은 편에 있는 구형 피라미드이다. 따라서, 질의 점과 구형 피라미드 sp_i에 이르는 최소 거리는 sp_i의 상단점 즉, 공간의 중앙점과 질의 점 사이의 거리가 된다. 따라서, $MINDIST(q, sp_i)=d_q$가 된다. 단위 공간을 기반으로 하기 때문에 식 (1)의 색인 k_n와 상수 C 는 [-1,0,1]의 값 중에서 하나를 갖는다. $i<d$이면, 그림 4의 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_i사이의 최소 거리는 $|q_j - q_i|/\sqrt{2}$가 된다. 마지막으로, $i>d$인 경우는 질의 점과 인접하는 구형 피라미드 sp_i의 한 측면의 방정식은 $k_jx_j + k_ix_i - 1 = 0$이 된다(그림 4 참고). 이때, $k_j=1$이고, k_i는 $i>d$이므로 1이다. 따라서, 질의 점과 인접한 구형 피라미드 sp_i사이의 거리는 $|q_j + q_i - 1|/\sqrt{2}$가 된다. □

질의 점과 구형 조각 사이의 최소 거리를 측정하는 것은 질의 점과 구형 피라미드 사이의 최소 거리를 측정하는 것보다 복잡하다. 질의 점이 속해 있는 구형 피라미드 안에 존재하는 구형 조각들과 맞은 편에 있는 구형 조각들, 그리고 인접한 구형 피라미드 안에 존재하는 구형 조각들로 나누어 정리할 수 있다.

[정리 2] (질의 점과 구형 조각 사이의 거리) 질의 점이 주어졌을 때 질의 점이 속해 있는 구형 피라미드를 sp_j라 하면, 질의 점과 구형 피라미드 sp_i안에 존재하는 구형 조각(BS_l)들과의 최소 거리 $MINDIST(q, BS_l)$는 다음과 같다.

경우 1: ($i=j$: 질의 점이 속해 있는 구형피라미드

안에 존재하는 구형 조각인 경우)

$$MINDIST(q, BS_l)=\begin{cases}|d_q-\max(BS_l)| & \text{if } d_q>\max(BS_l)\\ 0 & \text{if } \min(BS_l)\le d_q\le\max(BS_l)\\ |d_q-\min(BS_l)| & \text{if } d_q<\min(BS_l)\end{cases}$$

경우 2: ($|i-j|=d$: 질의 점의 맞은편에 있는 구형 피라미드 안에 존재하는 구형 조각인 경우)

α를 질의 점에서 가장 가까운 구형 피라미드의 한 면에 이르는 거리라 하고, θ를 d_q와 α에 의해 만들어지는 직각 삼각형의 한 각($\sin\theta=\frac{\alpha}{d_q}$)이라고 하면,

$$MINDIST(q, BS_l)=\sqrt{d_q{}^2+\min(BS_l)^2-2d_q\min(BS_l)\cos(\theta+\frac{\pi}{2})}$$

경우 3: (otherwise : 질의 점과 인접한 구형 피라미드 안에 존재하는 구형 조각의 경우)

α와 d_q에 의해 만들어지는 직각 삼각형의 밑변의 길이를 δ라고 하면,

$$MINDIST(q, BS_l)=\begin{cases}\sqrt{|\delta-\max(BS_l)|^2+\alpha^2} & \text{if } \delta>\max(BS_l)\\ \alpha & \text{if } \min(BS_l)\le\delta\le\max(BS_l)\\ \sqrt{|\delta-\min(BS_l)|^2+\alpha^2} & \text{if } \delta<\min(BS_l)\end{cases}$$

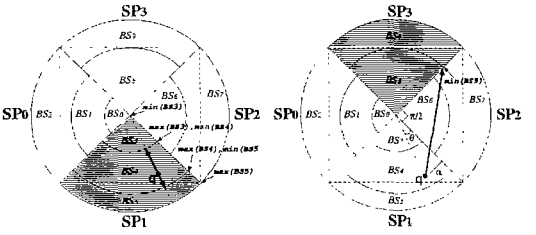
여기서;

$\min(BS_l)=\{d_v|(\forall v', v, v'\in BS_l: d_v\le d_{v'})\}$
$\max(BS_l)=\{d_v|(\forall v', v, v'\in BS_l: d_v\ge d_{v'})\}$

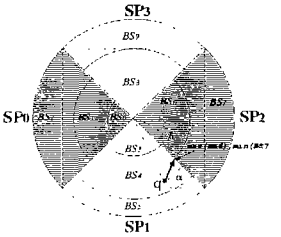
증 명: $\min(BS_l)$는 BS_l에 속해 있는 점들 중에서 d_v값이 가장 적은 것이며, $\max(BS_l)$는 d_v값이 가장 큰 것을 의미한다. $\min(BS_l)$와 $\max(BS_l)$를 이용하여 경우 1, 2, 3을 증명할 수 있다.

1. 질의 점이 BS_l안에 속해 있는 경우, $MINDIST(q, BS_l)$는 질의 점과 BS_l의 안에 존재하는 어떤 점과의 거리보다 작거나 같아야 함으로 0이 된다. $d_q>\max(BS_l)$이면, $MINDIST(q, BS_l)$는 BS_l에 존재하는 점들 중에서 중앙으로부터 가장 멀리 떨어진 점 v의 d_v, 즉 $\max(BS_l)$와 d_q의 차이가 된다. 마지막으로, $d_q<\min(BS_l)$인 경우, $MINDIST(q, BS_l)$는 BS_l에 존재하는 점들 중에서 중앙으로부터 가장 가까운 거리에 있는 점 v의 d_v, 즉 $\min(BS_l)$와 d_q의 차이가 된다. 그림 5의 (a)는 2차원 공간에서 경우 1을 보여주는 예이다.

2. $|i-j|=d$이면, sp_i는 질의 점이 속해 있는 구형 피라미드의 맞은편에 있는 구형 피라미드가 된다. 이 경우 질의 점과 해당 구형 조각 사이의 최소 거리는 d_q와 $\min(BS_l)$, 그리고 이것들로 이루어지는 삼각형의 밑변의 길이가 된다. 이는 코사인 제 2법칙으로 구할 수 있다. 먼저, 질의 점과 가장 가까운 인접 구형 피라미드의 한 면에 이르는 거리를 α라 하면, d_q와 α에 의해 이루어지는 각 θ는 $\arcsin(\alpha/d_q)$이 된다. 또한, 구형 피라미드의 상단점의 각은 항상 $\frac{\pi}{2}$임으로, $MINDIST(q, BS_l)$는 코사인 제 2법칙에 의해 $\sqrt{d_q{}^2+\min(BS_l)^2-2d_q\min(BS_l)\cos(\theta+\frac{\pi}{2})}$이 된다. 그림 5의 (b)는 경우 2를 보여주는 예이다.



(a) 경우 1 (b) 경우 2



(c) 경우 3

그림 5 질의 점과 각 구형 조각 사이의 최소거리 (MINDIST)

3. 이 경우는 sp_i가 질의 점에 인접해 있는 구형 피라미드이다. α와 d_q에 의해 만들어지는 직각 삼각형의 밑변의 길이를 δ라 하면, $\min(BS_l)\le\delta\le\max(BS_l)$인 경우, BS_l과 q 사이의 최소 거리는 경우 1과 같은 이유로 질의 점 q로부터 sp_i의 수직 거리인 α가 된다. 한편, $\delta>\max(BS_l)$인 경우, $MINDIST(q, BS_l)$는 α와 $|\delta-\max(BS_l)|$에 의해 만들어지는 직각 삼각형의 빗변의 길이가 된다. 마지막으로, $\delta<\min(BS_l)$인 경우는 경우 1과 동일한 이유로 α와 $|\delta-\min(BS_l)|$ 의해 만들어지는 직각 삼각형의 빗변의 길이가 된다.□

4.1 최근접 질의 처리 알고리즘

이 절에서는 앞서 설명한 정리 1과 2를 이용하여 점진적 최근접 질의를 처리하기 위한 알고리즘을 설명한다. 알고리즘에 대한 이해를 돕기 위하여 먼저 알고리즘의 전체적인 동작을 설명하고 간단한 예를 들어 실제동작하는 모습을 보인다. 알고리즘 1의 줄 1~4에서는

Algorithm 1 Processing the incremental nearest neighbor query

```
1:  for i = 0 to 2d-1
2:    dist = Dist_QP_to_SP(qp, sp[i]); /* Using Lemma 1 */
3:    Enqueue(queue, sp, dist);
4:  end for
5:
6:  while not IsEmpty(queue) do
7:    Element = Dequeue(queue);
8:    if Element is a spherical pyramid then
9:      for each bounding slice in a spherical pramid do
10:       dist = Dist_QP_to_BS(qp, bs); /* Using Lemma 2 */
11:       Enqueue(queue, bs, dist);
12:     end for
13:   else if Element is a bounding slice then
14:     for each object in a bounding slice do
15:       dist = Dist_QP_to_OBJ(qp,object);
16:       Enqueue(queue,object,dist);
17:     end for
18:   else /* Element is a object */
19:     report element as the next nearest object
20:   end if
21: end while
```

정리 1을 이용하여 질의 점과 각 구형 피라미드 사이의 거리를 계산하여 큐에 삽입한다. 그리고, 줄 6~21은 큐가 empty될 때까지, 큐의 첫 번째 원소를 추출하여 원소의 타입에 맞는 처리를 해준다. 추출된 원소가 구형 피라미드이면 해당 구형 피라미드 안에 있는 구형 조각들과 질의 점과의 거리를 계산하여 이를 다시 큐에 삽입한다. 만약, 추출한 원소의 타입이 구형 조각이면 해당 구형 조각 안에 있는 객체와 질의 점과의 거리를 계산하여 다시 큐에 삽입한다. 그리고, 마지막으로 추출한 원소가 객체이면 해당 객체를 최근접 질의의 결과로 반환한다. 순위 큐는 항상 최소 거리를 가지고 있는 원소가 맨 앞에 있기 때문에 반환된 객체는 질의 점과 가장 가까이에 있는 객체가 된다.

표 1 그림 3에서 질의 점과 구형 피라미드(SP), 구형 조각(BS), 객체(OBJ) 사이의 ***MINDIST***

SP	Dist.
SP0	21
SP1	0
SP2	4
SP3	33

BS	Dist.
BS0	21
BS1	25
BS2	29
BS3	14
BS4	0
BS5	2
BS6	8
BS7	4
BS8	33
BS9	42

OBJ	Dist.
a	23
b	27
c	45
d	26
e	29
f	12
g	35
h	6
i	39
j	47

표 1은 그림 3의 예에서 질의 점(q)과 각 구형 피라미드, 구형 조각, 및 해당 객체간의 거리를 보여주고 있다. 다음은 그림 3의 예에 대하여 알고리즘 1이 수행되는 동안 순위 큐의 내용을 보여주고 있다.

1. Enqueue SP0~SP3; {[SP1,0],[SP2,4],[SP0,21],[SP3,33]}

2. Dequeue SP1, enqueue BS3,BS4,BS5; {[BS4,0],[BS5,2],[SP2,4],[BS3,14],[SP0,21],[SP3,33]}

3. Dequeue BS4, enqueue **e**; {[BS5,2],[SP2,4],[BS3,14],[**e**,19],[SP0,21],[SP3,33]}

4. Dequeue BS5, enqueue **f**; {[SP2,4],[**f**,12],[BS3,14],[**e**,19],[SP0,21],[SP3,33]}

5. Dequeue SP2, enqueue BS6,BS7; {[BS7,4],[BS6,8],[**f**,12],[BS3,14],[**e**,19],[SP0,21], [SP3,33]}

6. Dequeue BS7, enqueue **h**; {[**h**,6],[BS6,8],[**f**,12],[BS3,14],[**e**,19],[SP0,21],[SP3,33]}

7. Dequeue **h**, report h as 1st nearest neighbor

알고리즘 1은 먼저 질의 점과 각 구형 피라미드 $sp_0 \sim sp_3$ 사이의 거리를 측정하여 이를 큐에 삽입하면서 시작된다. 알고리즘 1의 순위 큐는 항상 거리를 키값으로 오름차순으로 정렬됨으로 질의 점이 속해 있는 구형 피라미드 sp_1가 큐의 맨 앞에 있게 된다. 다음, 알고리즘 1의 줄 7에서 큐의 첫 번째 원소를 추출하고 이것이 구형 피라미드 sp_1이므로 sp_1에 속해 있는 구형 조각 BS_3, BS_4, BS_5에 대하여 질의 점과의 거리를 측정하여 큐에 삽입한다. 다시, BS_4를 추출하고 이것이 구형 조각이므로 BS_4에 있는 모든 객체들을 큐에 삽입하게 된다. 이 경우는 하나의 객체만을 가정했으므로 객체 **e**가 큐에 삽입된다. 이런 식으로 알고리즘 1이 진행 되며, 결국 질의 점과 가장 가까운 객체가 큐의 맨 앞에 있게 된다(여기서는 **h**). 점진적 최근접 질의 처리 알고리즘은 순위 큐를 이용하여 사용자가 원하는 최근접 객체들을 차례대로 추출할 수 있다. 알고리즘 1의 **while**-루프에서 최근접 객체로 반환하는 객체의 수를 제어하면 간단히 기존의 k-최근접 질의를 처리할 수 있다.

5. 실험 및 분석

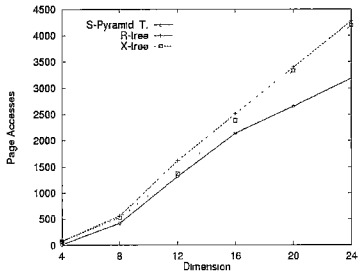
구형 피라미드 기법을 이용한 최근접 질의 처리의 효율성을 보이기 위하여 R*-tree와 X-tree와 비교 실험을 수행하였다. 공정한 실험을 위하여 R*-tree와 X-tree상에서 점진적 최근접 질의 처리 알고리즘을 구현하였으며 k-최근접 질의를 처리할 수 있도록 알고리즘을 수정하였다. 모든 실험은 128M의 주메모리와 10GB의

보조 기억장치를 가진 Sun Sparc 20 웍스테이션 상에서 수행되었으며, 블럭의 크기는 모든 색인 구조에서 4096 Byte이고, 블럭 사용률은 65%로 고정시켜 실험하였다.

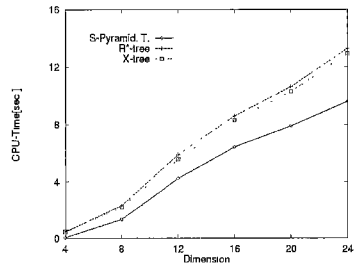
5.1 인위적 데이타를 이용한 실험

인위적으로 생성된 데이타들은 20,000~100,000개의 균등하게 분포된 데이타들이며 각 데이타의 차원은 4,8,12,16,20,24이다.

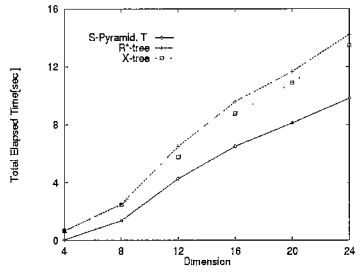
첫 번째 실험은 데이타의 차원을 변화시키면서 10개의 최근접 객체(10-NN)를 찾는데 소요되는 페이지 접근 횟수, CPU 사용 시간, 전체 질의 응답 시간을 측정하였다. 100개의 무작위로 선출된 질의 점을 사용하였으며 모든 결과는 100번의 질의 처리 결과에 대한 평균이다.



(a) 페이지 접근횟수

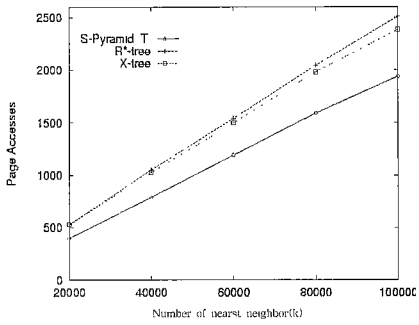


(b) CPU 사용시간

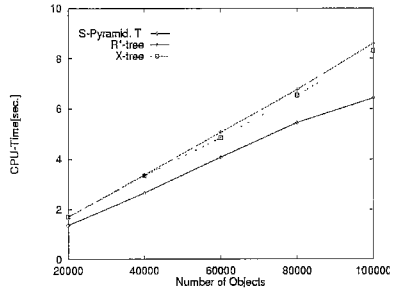


(c) 전체응답시간

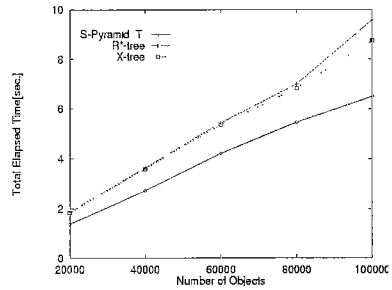
그림 6 데이타 차원에 따른 성능 변화(데이타베이스 크기=100000, k=10)



(a) 페이지 접근횟수



(b) CPU 사용시간



(c) 전체응답시간

그림 7 데이타베이스 크기에 따른 성능변화(차원=16, k=10)

그림 6은 첫 번째 실험의 결과이다. 전반적으로 데이타의 차원이 증가할수록 구형 피라미드 기법을 이용한 최근접 질의 처리가 기존의 R*-tree나 X-tree에 비해 효율적임을 알 수 있다. 그림 6(a)는 페이지 접근 횟수를 보여주고 있는데, R*-tree는 12차원 이상에서는 거의 모든 내부 노드와 단말 노드를 접근해야함을 볼 수 있다. X-tree는 비록 R*-tree보다는 적은 수의 페이지를 접근하나, R*-tree와 마찬가지로 고차원으로 올라갈수록 대부분의 페이지를 접근함을 알 수 있다. 구형 피라미드 기법도 비록 차원이 증가할수록 데이타 페이지

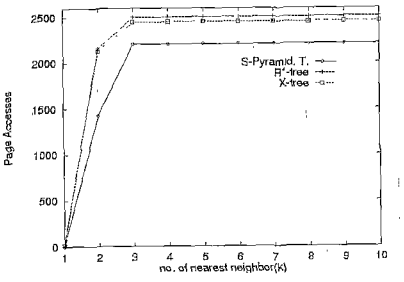
접근 횟수가 선형적으로 증가하나 항상 R*-tree나 X-tree보다 적음을 발견할 수 있었으며, 특히 24차원에서 R*-tree보다 34%, X-tree보다는 31%정도의 성능향상을 발견할 수 있었다. 최근접 질의 처리 알고리즘은 특정 거리 측정 함수(예, R*-tree의 min-max distance)를 이용하여 노드들을 비교하고 정렬하여 처리하기 때문에 조인과 같은 다른 종류의 질의 처리에 비하여 CPU 사용 시간이 많이 소요된다[3]. 그림 6(b)는 CPU 사용 시간을 보여주고 있다. 비록 구형 피라미드 기법을 이용한 최근접 질의 처리 알고리즘은 각 구형 조각들과 질의 점 사이의 최소 거리를 측정하는 과정이 복잡해 보이지만 간단한 비교 연산을 통하여 각 경우를 쉽게 검사하여 최소 거리를 측정할 수 있다. 이에 반하여, R*-tree의 경우에는 고차원 공간으로 올라갈수록 영역들 간의 많은 겹침(overlap)이 발생하며 이에 따라 거리를 비교하고 계산하는데 많은 CPU 시간이 사용되는 단점이 있다. 그러나, 구형 피라미드 기법은 그 구조상 겹침이 없는 공간 분할 방법을 이용하기 때문에 영역 겹침에서 오는 부담이 없다. 24차원의 경우, CPU 사용 시간에 있어서 R*-tree보다는 38%, X-tree보다는 35%정도의 성능향상을 발견할 수 있었다. 마지막으로 그림 6(c)는 전체 질의 응답 시간을 보여주고 있는데, 페이지 접근 횟수나 CPU 사용 시간과 거의 유사한 결과를 보여주고 있으며, 24차원일 때, R*-tree보다는 45%, X-tree보다는 37%정도의 성능 향상이 있었다.

두 번째 실험은 데이타베이스의 크기를 변화시키면서 성능의 변화를 측정한 것이다. 차원은 16차원으로 고정시켜 실험을 수행하였다. 그림 7은 두 번째 실험 결과를 보여주고 있으며 첫 번째 실험과 거의 유사한 결과를 확인할 수 있었다. 16차원 100,000개의 데이타에 대해서 구형 피라미드 기법을 이용한 최근접 질의 처리가 R*-tree에 비하여 47%, X-tree보다는 35%정도 빨리 질의를 처리함을 알 수 있었다.

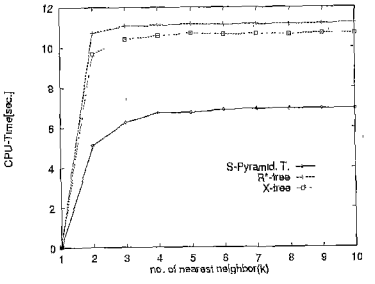
5.2 실제 데이타를 이용한 실험

이 실험에서는 실제 CAD객체의 영역을 표시하는 16차원 푸리에(Fourier) 점 100,000개를 사용하였다[3]. 질의 객체는 실제 데이타에서 무작위로 선출한 100개의 16차원 푸리에 점들을 사용하였으며 모든 결과는 100번의 질의에 대한 평균값이다. 최근접 객체의 갯수(k)를 1에서 10까지 증가시키면서 질의 처리에 필요한 페이지 접근 횟수, CPU 사용 시간, 전체 응답 시간을 측정하였다.

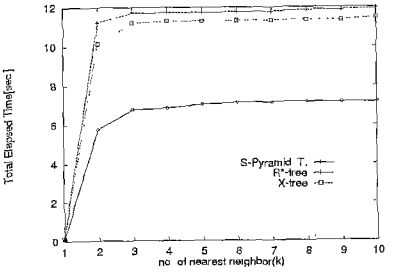
그림 8은 이러한 실험의 결과를 보여준다. k=1일 때, 구형 피라미드 기법을 이용한 최근접 질의 처리 알고리



(a) 페이지 접근횟수



(b) CPU 사용시간



(c) 전체응답시간

그림 8 실제 데이타를 이용한 실험(차원=16, 데이타베이스 크기=100,000)

즘은 단지 1개의 데이타 페이지만을 접근함으로써 질의 객체와 가장 가까운 객체(즉, 질의 객체 자체)를 검색함을 알 수 있었다. 그러나, X-tree는 13.45개의 페이지를 접근하고, R*-tree는 16.43개의 페이지를 접근해야 질의를 처리할 수 있었다. 또한, k=3이상인 경우에 X-tree나 R*-tree는 거의 모든 데이타 페이지를 접근해야 함을 알 수 있었다. 구형 피라미드의 경우에도 k=3이상이 되면 k값이 1이나 2인 경우보다는 많은 수의 데이타 페이지를 접근해야 하지만, 항상 X-tree나 R*-tree보다는 적은 수의 데이타 페이지를 접근함을 알 수 있었다. CPU 사용 시간이나 전체 질의 응답 시간도 역시 페이

지 접근 횟수와 유사한 경향을 보여주고 있다. k=10일 때, 구형 피라미드 기법을 이용한 최근접 질의 처리 알고리즘은 전체 질의 응답 시간에 있어서 R*-tree보다는 67%, X-tree보다는 60%정도 더 효율적임을 알 수 있었다. 인위적 데이타를 이용한 실험과 비교해 보면 구형 피라미드 기법을 이용한 최근접 질의 처리 알고리즘은 균등하게 분포된 데이타 집합보다 실제 데이타 집합에서 더 좋은 성능을 보임을 알 수 있다.

6. 결 론

본 논문에서는 기존의 구형 피라미드 기법을 이용하여 점진적 최근접 질의 처리 알고리즘을 제안하였다. 또한, 다양한 실험을 통하여 이 알고리즘이 기존의 X-tree나 R*-tree의 점진적 최근접 질의 처리 알고리즘보다 효율적임을 보였다. 구형 피라미드 기법은 그 구조가 기존의 R*-tree 기반의 다른 색인 구조에 비하여 단순하며 B+-tree를 그대로 이용할 수 있기 때문에 B+-tree가 가지는 빠른 삽입과 검색, 삭제의 장점을 그대로 이용할 수 있다. 또한, 현재 사용되는 대부분의 상용/학술 데이타베이스는 R*-tree에 대한 동시성 제어나 회복 기법을 제대로 지원하지 못하고 있으나, B+-tree에 대해서는 이러한 기법들을 대부분 지원해 주고 있다. 구형 피라미드 기법은 현재 사용되는 어떤 종류의 데이타베이스 시스템 상에서도 B+-tree를 이용하여 쉽게 구현될 수 있으며, 따라서 동시성 제어나 회복 기법도 그대로 이용할 수 있는 장점이 있다. 본 논문의 향후 연구로는 구형 피라미드 기법을 이용한 점진적 최근접 질의 처리 알고리즘을 확장하여 근사 최근접 질의 처리 알고리즘을 개발하고 실험할 계획이다.

참 고 문 헌

[1] 이 동호,정 진완,김 형주. "구형 피라미드 기법: 고차원 데이타의 유사성 검색을 위한 효율적인 색인 기법", 한국정보과학회 논문지(B), 25(11), 1999.

[2] Dong-Ho Lee, Hyoung-Joo Kim. "SPY-TEC : An Efficient Indexing Method for Similarity Search in High-Dimensional Data Spaces". Data & Knowledge Engineering, 34(1), 2000.

[3] 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.

[4] 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.

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

[6] 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.

[7] S. Berchtold, C. Bohm, 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.

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

[9] J. T. Robinson."The K-D-B-tree: a Search Structure for Large Multidimensional Dynamic Indexes". Proc. ACM SIGMOD, Ann Arbor, USA, pages 10-18, April 1981.

[10] H. Samet. "The Design and Analysis of Spatial Data Structures". Addison-Wesley, 1989.

[11] A. Guttman. "R-trees: a dynamic index structure for spatial searching". Proc. ACM SIGMOD Int. Conf. on Management of Data, pages 47-57, June 1984.

[12] N. Roussopoulos, S. Kelley, F. Vincent. "Nearest Neighbor Queries". Proc. ACM SIGMOD, San Jose, CA, pages 71-79, 1995.

[13] Gisli R. Hjaltason, Hanan Samet. "Distance Browsing in Spatial Databases". ACM Transaction on Database Systems, 24(2):265-318, 1999.

이 동 호

1995년 2월 홍익대학교 컴퓨터공학과(학사). 1997년 2월 서울대학교 컴퓨터공학과(석사). 1997년 3월 ~ 현재 서울대학교 컴퓨터공학부 박사과정 재학중. 관심분야는 객체지향 시스템, 멀티미디어 정보 검색, 데이타베이스

김 형 주

1982년 2월 서울대학교 컴퓨터공학과 졸업. 1985년 8월 Univ. of Texas at Austin, 전자계산학 석사. 1988년 5월 ~ Univ. of Texas at Austin, 전자계산학 박사. 1988년 5월 ~ 9월 Univ. of Texas at Austin. Post-Doc. 1988년 9월 ~ 1990년 12월 Georgia Institute of Technology, 부교수. 1991년 1월 ~ 현재 서울대학교 컴퓨터공학부 부교수. 관심분야는 객체지향 시스템, 사용자 인터페이스, 데이타베이스