R-tree 계열의 인덱싱 구조에서의 효율적 질의 처리를 위한 VP 필터링

(VP Filtering for Efficient Query Processing in R-tree Variants Index Structures)

김 병 곤 † 이 재 호 †† 임 해 철 †††

(Byung-Gon Kim) (Jaeho Lee) (Haechull Lim)

요 약 정보사회가 인터넷의 보급과 더불어 복잡해짐에 따라 데이타베이스의 흐름은 문자나 숫자와 같은 일차원적인 데이타가 아닌 지리정보, 멀티미디어 데이타와 같은 다차원의 데이타를 저장하고 이에 대한 질의를 처리할 수 있는 시스템을 요구하고 있다. 따라서, 다차원적인 특성을 지니는 데이타에 대한 효율적 검색을 위한 다차원 인덱싱 구조에 대한 연구가 활발히 진행되어 왔으며, 그와 동시에 이러한 인덱싱 구조하에서 효율적인 질의 처리를 위한 연구도 병행되고 있다. 다차원 데이타는 그 다양한 응용분야에 따라 요구되는 질의의 형태가 각각 다르므로 이에 대응할 수 있는 알고리즘의 연구가 필요하다. 현재, 많은 다차원 데이타 처리 시스템이 R-트리계열의 인덱싱구조를 근간으로 구성되었으나, 현재까지의 질의처리 기법은 질의처리시에 필터링 특성을 지니지 않으므로, 객체들간의 다차원 거리계산으로 인하여 많은 질의처리시간을 소요한다.

본 논문에서는 다차원 데이타를 처리하기 위한 R-트리 계열의 다차원 인덱싱 구조에서의 효율적인 질의처리를 위하여 질의처리 대상 객체를 줄이기 위한 필터링 기법을 소개하였다. 필터링을 수행하기 위하여 VP-트리와 MVP-트리에서 사용되었던 VP(Vantage Point)를 이용한다. 먼저, VP 필터링의 개념을 소개하고, VP 필터링을 영역질의와 포인트 질의의 일종인 추가객체요구질의에 각각 적용한 알고리즘을 제시하였다. VP 필터링을 적용하기 위하여 요구되는 삽입 객체와 VP간의 거리계산 시간은 객체의 삽입시 수행되며, 질의 처리를 수행할 때에는 다시 계산되지 않는다. 논문에서는 제안된 알고리즘의 효율성을 실험을 통하여 증명하였다.

키워드 : 다차원 인덱싱, R-트리, VP, 필터링

Abstract With the prevalence of multi-dimensional data such as images, content-based retrieval of data is becoming increasingly important. To handle multi-dimensional data, multi-dimensional index structures such as the R-tree, R*-tree, TV-tree, and MVP-tree have been proposed. Numerous research results on how to effectively manipulate these structures have been presented during the last decade. Query processing strategies, which is important for reducing the processing time, is one such area of research.

In this paper, we propose query processing algorithms for R-tree based structures. The novel aspect of these algorithms is that they make use of the notion of VP filtering, a concept borrowed from the MVP-tree. The filtering notion allows for delaying of computational overhead until absolutely necessary. By so doing, we attain considerable performance benefits while paying insignificant overhead during the construction of the index structure. We implemented our algorithms and carried out experiments to demonstrate the capability and usefulness of our method. Both for range query and incremental query, for all dimensional index trees, the response time using VP filtering was always shorter than without VP filtering. We quantitatively showed that VP filtering is closely related with the response time of the query.

Key words : Multidimensional Indexing, R-tree, VP, Filtering

† 종신회원 : 부천대학 사무자동화과 교수
bgkim@bc.ac.kr
†† 종신회원 : 인천교육대학 컴퓨터교육과 교수
jhlee@mail.inue.ac.kr
††† 종신회원 : 홍익대학교 컴퓨터공학과 교수
lim@cs.hongik.ac.kr
논문접수 : 2002년 1월 7일
심사완료 : 2002년 8월 31일

1. 서 론

인터넷의 확산과 웹 기술의 발전으로 지리정보, 이미지, 음성과 같은 다차원 자료를 누구나 쉽게 이용할 수 있게 되었다. 다차원 데이타에 대한 초기의 연구에서는 데이타 자체의 내용이 아닌 데이타에 대한 설명들을 데이타 생성 시에 같이 생성하고 이를 기초로 인덱싱을 하는 기법을 사용하였다. 이를 설명 기반 인덱싱 기법이라고 한다. 이와는 반대로 데이타의 내용을 직접 검색하는 방법을 내용 기반 인덱싱 기법이라 한다. 내용 기반 인덱싱 기법이 필요한 이유는 다차원 공간에 분포하는 데이타는 실제 내용에 의하여 분석 구조화하지 않으면 대용량의 수많은 데이타들을 구분하는데 한계가 있기 때문이다.

다차원 데이타의 내용 기반 인덱싱을 위하여 다차원 데이타 인덱싱 구조에 대한 연구가 활발히 진행되었다. 그와 동시에 이러한 인덱싱 구조하에서 효율적인 질의처리를 위한 연구도 병행되고 있다. 다차원 데이타는 그 다양한 응용분야에 따라 요구되는 질의의 형태가 각각 다르므로 이에 대응할 수 있는 질의처리 알고리즘의 연구가 필요하다고 할 수 있다. 이와 관련하여 R-트리[1], R*-트리[2], X-트리[3], TV-트리[4], MVP-트리[5]와 같은 많은 다차원 인덱싱 방법이 발표되었으며, 이를 기반으로 효율적인 질의 처리를 수행하기 위한 여러 가지 질의 처리 전략도 연구되어 왔다. 다차원의 데이타에 대한 질의는 대표적으로 영역질의(Range Query)와 최근접객체검색질의(Nearest Neighbor Query)로 나눌 수 있다. 영역질의는 질의 객체로부터 일정 거리내에 존재하는 객체를 검색하는 질의이며, 최근접객체검색질의는 점질의(Point Query)라고도 하며, 질의 객체로부터 가장 근접한 혹은 가장 유사한 객체를 반환하는 것이다. 최근접객체검색질의의 변형으로 k-근접객체검색질의(k-Nearest Neighbor Query)는 질의 객체로부터 가장 근접한 k개의 객체를 반환하는 것이다. 다차원 데이타 검색 시 중요한 또 하나의 질의 형태는 추가객체요구질의(Incremental Query)이다. 이 또한 포인트 질의의 일종이며, 멀티미디어 데이타와 같은 다차원데이타의 검색 시에는 처음 검색한 검색결과가 만족스럽지 못한 경우에는 추가적으로 다음 근접 유사 객체를 반환하도록 요구하는 경우가 빈번하다. 그러므로 이러한 경우에 신속히 대처하기 위해서는 추가객체요구질의를 효율적으로 처리할 수 있어야 한다.

위에서 언급한 질의들은 빠른 시간 내에 수행되어야 하지만 다차원 객체간의 거리계산 시간으로 인하여 많은 시간을 요구한다. 더구나, 가장 일반적으로 많이 사용되고 있는 R-트리 계열의 인덱싱 구조에서는 데이타의 차원이 늘어날수록 중간노드간의 겹침이 증가하여 질의처리 시간이 오히려 순차검색에 비하여 우수하지 못한 경우가 발생할 수 있다. 그러므로 이를 극복할 수 있는 효율적인 검색 기법이 절실히 요구된다고 할 수 있다. 또한, R-트리 계열의 인덱싱 구조에서 질의를 처리하기 위해서는 트리 노드내의 모든 객체에 대하여 거리 계산을 수행한 후 그 결과를 알 수 있다. 그러나 다차원의 객체간의 거리계산은 많은 오버헤드를 초래하고 시스템 전체에 부담을 줄 수 있다. 그러므로, 검색 시에 객체들간의 거리계산 횟수를 줄이는 것이 신속한 질의처리를 위한 중요한 요소가 된다.

본 논문에서는 이와 같이 R-트리 계열의 다차원 인덱싱 구조에서 질의처리를 효율적으로 수행하기 위한 필터링 기법을 소개한다. 필터링을 수행하기 위하여 VP-트리[6]와 MVP-트리[5]에서 사용되었던 VP(Vantage Point)를 이용한다. VP 필터링을 사용하여 객체간의 거리계산 횟수를 최소화하여 짧은 시간 내에 질의가 처리되도록 한다. 논문에서는 VP 필터링의 개념을 소개하고, VP 필터링을 영역질의와 추가객체요구질의에 각각 적용한 알고리즘을 소개하였다. 또한 실험을 통하여 제시된 알고리즘의 성능의 우수함을 증명하였다.

2. 관련연구

2.1 R-트리 계열 다차원 트리에서의 질의 처리

현재 가장 많이 사용되고 있는 다차원 색인 구조는 R-트리 계열의 트리들이다. R-트리 계열의 트리 들은 객체의 삽입과 삭제가 최소한의 오버헤드만을 가지고 가능하다는 점에서 동적인 구조라 할 수 있다. 그러나, 현재까지의 질의처리 기법은 단말 노드에서의 필터링 능력이 없으므로 인하여, 거리계산을 위한 오버헤드가 존재한다. 특히, 다차원 색인 구조로 표현하고자 하는 객체들이 노드간에 많은 겹침이 발생하는 경우에는 단순히 순차 검색을 수행하는 경우보다 낮은 검색 효율을 나타낼 수 있다.

2.1.1 영역질의의 처리

R-트리에서의 영역질의는 다음과 같이 진행된다. [1][2] 먼저, R-트리의 루트노드의 각 엔트리에 대하여 질의 영역과의 겹침을 조사한 후, 겹침이 있는 엔트리의 자식 노드들에 대하여 재귀적으로 검색을 수행한다. 이와 같이 재귀적으로 검색을 수행하면 마침내 단말노드에 이른다. 검색공간에 대한 영역 정보만을 지니는 중간노드와는 달리 단말노드의 엔트리는 실제 객체에 대한 정보를 지닌

다. 이때, 각 객체 엔트리에 대하여 검색영역과의 겹침을 조사하여 최종적인 해답 객체를 얻을 수 있다.

```
Procedure RangeQuery(tree R, query object Q, range r)
R1  if R is a intermediate node //search intermediate node of the
        tree
R2      for each entry E of R
R3          if MINDIST(Q, E) ≤ r
R4              RangeQuery(E, Q, r);
R5  if R is a leaf node //search leaf nodes
R6      for each entry object O of R
R7              if d(Q, O) ≤ r
R8                  report O as a solution;
```

그림 1 영역질의 처리 알고리즘

그림 1은 영역질의 처리 알고리즘을 나타내며, MINDIST (Q,E)는 질의객체 Q와 현재 노드 엔트리 E와의 최소거리를 나타낸다. 그러나, 이와 같은 방식으로 영역질의를 수행하면, 단말노드에 존재하는 많은 객체들과의 겹침을 조사하여야 하고, 이는 해당 객체를 표현하고 있는 n-차원의 수치정보와의 거리계산을 요구하게 된다. 예를 들어, 그림 2에 나타난바와 같이 MBR A, B ,C에 각각 객체들이 존재한다. 이때, 질의 영역 X에 해당하는 질의를 처리하기 위해서는 MBR A, B ,C에 존재하는 모든 11개의 객체들에 대하여 거리계산을 수행하여야 한다. 그러나 실제로 질의 영역 내에 존재하는 객체는 하나이다. 이와 같이, 겹침을 조사한 객체들 중 실제로는 일부의 객체들만이 실제 해답에 해당하는 경우가 대부분이므로, 질의 영역과 겹침이 존재하지 않는 객체와의 거리계산 시간은 시스템에 있어서 오버헤드로 작용하며, 결과적으로 질의 처리 시간을 늦추는 역할을 하게 된다. 더구나, 차원이 커지고, 데이타베이스 내의 객체의 수가 기하 급수적으로 늘어날수록 더 커다란 영향을 미치게 된다. 그

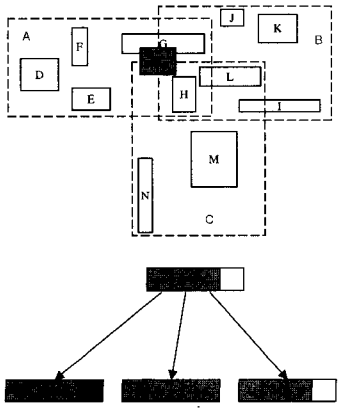

그림 2 영역질의 처리에서의 겹침 발생

러므로, 이러한 오버헤드를 최대한 줄이기 위하여 거리계산의 대상이 되는 객체의 수를 최대한 줄일 수 있는 필터링 기법이 요구된다.

2.1.2 추가객체요구 k-NNQ(Incremental k-NNQ) 질의처리

추가적으로 질의를 처리하는 방식은 최근접객체요구질의(NNQ 혹은 Point Query)를 수행하는 경우에 사용자의 요구에 능동적으로 대처하기 위한 또 하나의 질의처리 방식이다. 추가객체요구질의 처리는 기본적으로 사용자가 주어진 해답에 만족하지 못하고 추가적으로 "하나 더"요구하게 되는 경우를 대비할 수 있도록 한다. 따라서, 이러한 특성을 가지고 질의를 처리하면, 첫 번째 결과에 만족하지 못한 경우에 다음 결과를 구하기 위하여 질의를 처음부터 다시 수행할 필요가 없어진다. 추가적인 질의에 대하여 k-NNQ에 있어서는 단지 다음 k 값을 늘리거나, 영역질의에 있어서는 영역 r을 늘리기만 하면 되는 것이다.

```
Algorithm Incremental_ranking(query object Q)
01 Priorityqueue queue;
02 queue.insert(0, root);
03 wait until get_next is called
04 while not queue.isempty() do
05       Element first = queue.pop();
06       case (first)
07            intermediate node :
08                 for each child entry in first do
09                      queue.insert(MINDIST(Q, child entry rectangle),
                        child entry)
10            leaf node :
11                 for each child object in first do
12                           queue.insert(MINDIST(Q, child object),
                             child object)
13            object :
14                 report first
15                      wait until get_next is called
16     endcase
17 endwhile
```

그림 3 추가객체요구질의 알고리즘[7]

추가객체요구질의 처리 알고리즘은 PMR Quadtree[8] 상에서 2차원의 지리정보 시스템을 구현하면서 처음 소개되었다. Seidl과 Kriegel은 다단계 k-NNQ 알고리즘[7]을 구현하면서 PMR-Quadtree상에서 구현된 추가객체요구질의 처리 알고리즘을 R-트리 환경으로 변환하여 사용하였다. [7]에서의 추가객체요구질의 알고리즘은 그림 3에 나타난바와 같이 트리의 루트 노드로부터 하나의 우선 순위 큐에 삽입하게된다. 이때, 먼저 루트노드의 엔트리노드들에 대하여 질의객체와의 거리를 계산하고 거리순서에 따라 가까운 노드가 큐의 앞에 오도록 삽입한다. 다시 가장 앞에 위치한 노드의 엔트리 노드들에 대하여 거리계산을 하고 다시 질의객체와의 거리 순

으로 우선 순위 큐에 삽입하게된다. 이와 같은 절차를 반복하면, 단말노드에 까지 이르게되고 단말노드내의 실제 객체들에 대하여 거리계산을 통하여 실제객체가 큐의 맨 앞에 오게되면 해당 객체를 반환하게 된다. 여기서 알고리즘의 라인 10-12를 살펴보면, 비록 당장 모든 객체에 대한 실제 거리가 필요하지 않더라도 단말노드의 모든 객체에 대한 MINDIST(Q, object) 거리 계산을 수행한다. 이는 가장 가까운 위치에 있는 개체를 결정하기 위한 것이지만 모든 객체에 대한 거리계산은 질의 처리 시에 상당한 오버헤드로 작용한다. 여기서, MINDIST(Q, object) 는 질의점 Q와 객체간의 최소거리를 의미한다.

이와 같이 Seidl과 Kriegel에 의하여 사용된 추가객체 질의처리 알고리즘은 단말노드에서의 필터링 능력을 지니지 못한다는 단점이 있다. 그러므로, 단말노드의 모든 객체는 우선순위큐에 삽입될 때 질의점과의 거리계산이 반드시 수행되어야 한다. 그러나, 추가적인 질의처리는 하나의 객체씩만을 반환하며, 추가적인 질의요구가 있는 경우에만 다음 객체를 반환하므로 단말노드내의 모든 객체에 대한 거리계산은 추후에 쓸모 없는 일이 될 가능성이 많다.

2.2 VP-트리와 MVP-트리에서의 질의 처리

VP-트리[6]와 MVP-트리[5]에서는 R-트리와 달리 해당 영역을 MBR을 통하여 분할하는 방법이 아닌 객체 점들과 VP(Vantage Point)와의 상대적인 거리를 이용하여 트리를 구성하는 거리 기반의 트리 구성 방법을 사용하였다.

VP란 수많은 데이타 객체들을 구분하기 위한 기준점 역할을 하는 것이다. VP는 객체 도메인내에 존재하는 임의의 한 점이며, 일반적으로 객체들 중에서 선택하거나, 임의의 한 점을 선택하기도 한다. 예를 들어, 초등학교 한 반의 학생들을 키를 기준으로 두 그룹을 분류하고자 할 때, 가장 쉬운 방법은 중간정도의 키를 지니는 한 학생을 정해서 그 학생보다 큰 학생들은 왼쪽에 작은 학생들은 오른쪽에 세우는 방식이다. 다소간의 차이점이 있기는 하지만 다차원 공간의 객체들도 임의의 VP를 기준으로 위와 같이 여러 개의 그룹으로 분류될 수 있다.

MVP-트리의 경우에는 거리계산을 줄일 수 있는 필터링 능력을 지니고 있지만, 구조적인 특성 때문에 정적인 데이타 집합에 대하여만 적용할 수 있다는 단점이 있다. 이는 MVP-트리가 하향 방식으로 균형 트리를 구성하며, 트리를 구성하기 전에 데이타 집합에 대한 모든 분석을 요구하기 때문이다. 그러므로, 객체의 삽입과 삭제는 균형 트리의 비 균형화를 초래하게 되어, 동적인 환경에서의 MVP-트리는 적합하지 않다.

본 연구에서는 위 두 가지 트리 구조의 장점, 즉, R-트리의 동적인 특성과 MVP-트리의 VP를 이용한 필터링 능력을 결합하여 R-트리 계열의 트리에 적용할 수 있는 VP 필터링 개념을 이용한 질의처리 기법들을 제안한다. R-트리상에서 VP 필터링을 통한 질의처리를 위하여 추가적으로 요구되는 오버헤드는 최초로 객체가 삽입될 때, 정해진 VP와의 거리계산이다. 여기서 계산된 거리 값은 객체가 삽입될 때 함께 삽입되며, 한번 정해진 VP는 변하지 않는 것이 원칙이므로 이 거리 값 역시 트리의 변화에 상관없이 변하지 않는다. 또한 질의처리 시에는 전혀 영향을 미치지 않는다.

3. VP 필터링을 이용한 질의처리

VP 필터링의 기본적인 개념은 단말노드에서 해당 노드에 속해 있는 객체들과 질의 점과의 거리계산에 소요되는 시간을 줄이자는 것이다. 즉, 모든 객체와의 거리계산을 수행하지 않고, 해답이 될 가능성이 없는 객체들은 거리 계산에서 미리 제외하자는 의미이다. 이를 위해서는 어떤 객체가 해답이 될 가능성이 있는지 없는지의 판단을 위한 정확한 기준이 필요하다. 이때에 기준으로 사용하는 것이 바로 VP(Vantage Point)와 이를 기준으로 하는 각 객체와의 거리값이다. VP와 객체간의 거리값은 객체의 삽입시 계산된다. 먼저, VP필터링을 수행하기 위한 준비 단계인 VP의 선정과 트리의 생성에 관하여 살펴본다.

3.1 VP의 결정 및 인덱싱 트리의 생성

R-트리, R*-트리, X-트리와 같은 R-트리 계열의 인덱싱 구조에서 VP를 이용하여 필터링을 수행하기 위해서는 먼저 VP를 선정하고, 객체와 VP와의 거리를 지니고 있어야 한다. R-트리, R*-트리, X-트리와 같은 R-트리 계열의 인덱싱 구조에서 VP 거리값을 지니는 객체들을 가지고 인덱싱을 구성하는 방법은 기본적으로 VP를 이용하지 않는 경우와 동일하다. 단지, VP를 결정하고, 결정된 VP를 기준으로 각 객체와 VP와의 거리값을 계산하는 과정이 추가된다.

대상 도메인 공간에 있어서 VP를 결정하는 방법은 다음과 같은 몇 가지 방법이 있다. Yianlos에 의하여 발표된 논문[9]에서는 도메인 공간의 구석(Corner Point)에서 VP를 선택하는 것이 VP 거리값의 분포면에서 좋은 변별력을 지닌다고 발표되었다. 본 논문에서는 구석 지점을 즉, 객체의 모든 차원이 0인 지점을 VP로 선정하는 방법을 사용하였다. 이는 VP를 선정하기 위하여

특별한 시간을 소비하지 않아도 되며, 검증된 성능을 나타내기 때문이다.

VP가 선정되면 VP와 각 객체들간의 VP거리를 계산한다. 계산된 거리 값은 객체가 R-트리에 삽입될 때 함께 저장된다. 이 거리 값은 VP와 객체와의 절대 거리이며, 객체가 삽입될 때 단 한번만 계산하면 된다. 즉, 트리의 구조가 바뀌거나, 다른 객체가 삽입되어 MBR이 변화한다고 하여도 VP가 변하지 않으므로 거리 값은 변하지 않는다. 거리 값은 단말노드의 엔트리에 객체 포인터와 함께 저장된다. 그림4는 VP 거리값을 지니는 R-트리의 노드 구조이다

```
struct intermediate_node_entry //중간노드 구조
{
  node_range_description;
  p : child_node_pointer;
 }
struct leaf_node_entry // 단말노드 구조
{
  node_range_description;
  d : distance from vp; //VP로부터 객체와의 거리값
  p : object_pointer; // 실제 객체로의 포인터
 }
```

그림 4 VP 거리를 지니는 트리 노드 구조

이제, VP 거리값과 함께 생성된 R-트리계열의 다차원 인덱싱 구조에서의 VP 필터링을 이용한 질의 처리 알고리즘을 소개한다. 본 논문에서는 영역질의와 추가 객체요구질의에 대한 효과적인 질의 방법을 제안한다.

3.2 VP 필터링을 이용한 영역질의의 처리

VP 거리값을 지니는 R-트리계열의 다차원 인덱싱 구조에서의 영역질의는 다음과 같은 절차를 통하여 처리 시간을 줄일 수 있다.

그림 5에서 MBR A에는 모두 14개의 객체가 존재한다(하나의 점이 객체에 해당한다.). 이때, 질의 점 Q로부터 거리 r사이에 존재하는 객체를 구한다고 가정하자.

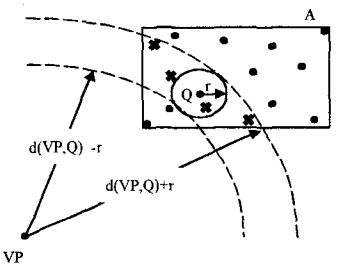

그림 5 VP 필터링을 이용한 영역질의처리

필터링의 개념이 없다면, 질의 점 Q와 14개의 객체간의 거리를 모두 구한 후, 거리가 r 보다 적은 객체가 해답으로 선택된다. 그러나, 그림 5에서 볼 수 있듯이 실제의 해답은 14개의 객체 중에서 단 하나만이 해당한다. 이렇듯, 나머지 13개의 객체에 대한 거리 계산은 시스템에 있어서 오버헤드가 된다. 이러한 오버헤드를 줄이기 위해서는 거리계산의 대상을 줄여야 한다.

VP 필터링을 이용하면 그림에서 'x'로 표시된 객체에 대하여만 거리 계산을 수행할 수 있는데 원리는 다음과 같다. 질의점 Q에 대하여 VP로부터의 거리인 d(VP,Q)를 구할 수 있다. 또한, 질의는 질의점으로부터 거리 r내에 존재하는 객체를 구하는 것이므로, d(VP,Q)-r 와 d(VP,Q)+r 사이에 존재하는 객체만을 거리계산의 고려 대상으로 정할 수 있다. 그림 5에서 두 개의 점선으로 표시된 둥근 트랙이 바로 그 범위에 해당한다.

다음은, MBR A내의 객체가 이 범위 내에 속하는지의 여부를 계산하여야 한다. 하지만, 이는 각 객체가 지니는 자신의 VP 거리 값을 가지고 쉽게 알아낼 수 있다. 즉, 객체 O의 거리 값 d(O,VP)가 거리 d(VP,Q)-r과 d(VP,Q)+r 사이에 존재하는지 계산하면 알 수 있다. 이 계산은 n차원의 거리계산과는 달리 1차원의 산술계산이므로 시스템에 거의 영향을 미치지 않는다. 이 조건을 체크하면 그림 5에서 14개의 객체들 중에서 단지 4개만이 조건을 만족하고, 4개의 객체에 대하여 실제 거리를 계산하면 정확한 해답을 얻을 수 있다. 위에서 설명한 필터링 개념을 정리하면 다음과 같다.

<규칙 1> 질의점 Q로부터 거리 r내에 존재하는 객체를 찾는 영역질의에서, 다음 조건을 만족하는 노드내의 임의의 객체들의 집합은 질의 해답 객체를 반드시 포함한다.

조건 : $d(VP,Q)-r \leq d(O, VP) \leq d(VP,Q)+r$

<규칙 2> 노드의 엔트리 개수가 n이고, 규칙 1의 조건을 만족하는 노드내의 객체가 k개이면, (n-k)개의 객체에 대한 n차원의 거리계산은 수행하지 않는다.

VP 필터링의 개념을 가지고 영역질의를 처리하는 알고리즘은 기존 R-트리에서의 영역질의 방법과 거의 유사하다. 단지 단말노드에서의 필터링 과정이 추가되는 것이다. 그림 6을 참조하면, 질의점 Q와 거리 r이 결정되면 루트노드로부터 시작하여 질의영역과 겹침이 있는 엔트리에 대하여 운행을 시작한다.(그림 6,라인 R1~R4) 운행은 중간노드들을 거쳐 단말노드에 이르면, 필터링을 시작한다. VP 필터링을 수행하지 않는 경우라면 모든 객체에 대하여 R8과 R9를 수행하지만 R7에서 필터링($d(VP,Q)-r \leq d(O,VP) \leq d(VP,Q)+r$)을 수행하므로 처리시간을 줄일 수 있다.

```
Procedure RangeQuery(tree R, query object Q, range r)
R1  if R is a intermediate node //search intermediate node of
        the tree
R2      for each entry E of R
R3          if MINDIST(Q, E) ≤ r
R4              RangeQuery(E, Q, r);
R5  if R is a leaf node //search leaf nodes
R6      for each entry object O of R
R7          if d(VP,Q)-r ≤ d(O, VP) ≤ d(VP,Q)+r
                                  // filtering condition checking
R8              if d(Q, O) ≤ r
R9                  report O as a solution;
```

그림 6 VP 필터링을 이용한 영역질의처리 알고리즘

3.3 VP 필터링을 이용한 추가객체요구질의 처리

관련연구 [7]에서 제안한 추가객체 질의처리 알고리즘은 단말노드에서의 필터링 능력을 지니지 못한다는 단점이 있다. 이러한 단점을 극복하기 위하여, 본 연구에서는 하나의 큐가 아닌 세 개의 큐로 분리하여 관리함으로써, VP 필터링이 가능하도록 하였다.

[7]에서는 오직 하나의 우선순위큐를 이용하여 거리계산이 수행된 객체를 다시 우선순위큐에 삽입하여 가장 가까운 객체를 반환하도록 하였으나, 본 논문에서는 거리계산이 수행된 단말노드의 객체들에 대하여 별도의 추가적인 두 개의 큐로 분리되어 삽입되도록 하였다.

단말노드의 객체들 중에서 앞으로 반환될 가능성이 높은 객체들은 실제거리 계산을 수행한 후 Return_Q에 삽입되도록 하였다. 여기서 Return_Q는 기존 알고리즘이 지니고 있던 우선순위 큐인 Main_Q와 마찬가지로 거리순서에 따라 우선순위큐 형태로 유지한다. 반면, 상대적으로 반환될 가능성이 낮은 객체들은 실제 거리계산을 수행하지 않고 Delay_Q에 삽입되도록 하였다. Delay_Q에 객체가 삽입되는 의미는 시간이 소요되는 실제 거리계산을 거리 값이 필요한 순간까지 최대한 연기할 수 있는 의미가 있다.

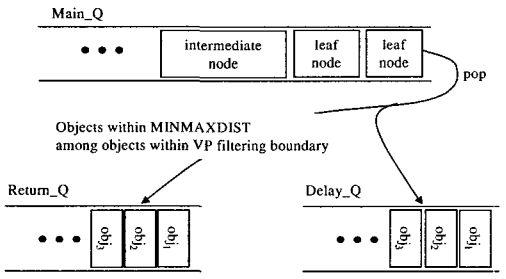

그림 7 VP 필터링을 이용한 추가 객체요구 k-NNQ 처리를 위한 큐 관리

그림 7에는 Main_Q와 Result_Q 그리고 Delay_Q의 관계를 보여준다. Main_Q에서는 루트노드와 중간노드만을 지니며, Return_Q와 Delay_Q에서는 실제 객체만을 지닌다. Return_Q는 질의점과의 실제거리를 기준으로, Main_Q는 질의점으로 부터 MINDIST를 기준으로 순서화 되며, Delay_Q는 실제거리를 계산하기 전이므로 특별한 순서 없이 삽입되는 순서대로 유지된다. k-NNQ가 실행되면 오직 Return_Q로부터 하나씩 객체가 반환되도록 알고리즘을 유지한다. 각 큐의 내용을 정리하면 다음과 같다.

큐	객체종류	순서화 기준
Main_Q	루트노드, 중간노드	질의점으로부터 MINDIST
Return_Q	실제 객체	질의점과의 실제거리
Delay_Q	실제 객체	삽입되는 순서대로 유지

그림 8 큐의 내용 정리

이제, 단말노드의 객체들을 어떤 방식으로 두 개의 별도의 큐로 분리하는지에 대하여 논한다. 추가객체요구 k-NNQ에서는 필터링을 위하여 영역질의와는 달리 MINMAXDIST를 사용한다. MINMAXDIST는 질의객체로부터 어떤 MBR의 먼까지의 최대거리 중에서 최소거리를 택한 것이다.

MINDIST와 MINMAXDIST는 다음 정의에 의하여 계산된다.[10] 먼저, n차원의 데이타인 MBR R은 대각선 위치의 S와 T의 두 끝점으로 이루어져 있으며, S와 T는 다음과 같다.

$R=(S,T)$ *where* $S=[s_1,s_2,\cdots,s_n]$,

$T=[t_1,t_2,\cdots,t_n]$ $(s_i \le t_i$ *for* $1 \le i \le n)$

이때 질의객체 Q로부터 MBR R까지의 MINDIST는 MINDIST(Q,R)로 표현할 수 있으며, 다음과 같이 정의할 수 있다.

$$MINDIST(Q,R)=\sum_{i=1}^{n}|q_i-r_i|^2$$

$$where \quad r_i=\begin{cases} s_i, when\ q_i < s_i \\ t_i, when\ q_i > t_i \\ q_i, otherwise. \end{cases}$$

또한, 질의객체 Q로부터 MBR R까지의 MINMAXDIST는 MINMAXDIST(Q,R)로 표현할 수 있으며, 다음과 같이 정의할 수 있다.

$$MINMAXDIST(Q,R)$$

$$= \min_{1\le k\le n}(|q_k-{}_k|^2)+\sum_{1\le i\le n,(i\ne k)}|q_i-rM_i|^2)$$

$$where \quad rm_i=\begin{cases} s_k, when\ q_k \le (s_k+t_k)/2 \\ t_k, otherwise \end{cases}$$

$$rM_i=\begin{cases} s_i, when\ q_i \le (s_i+t_i)/2 \\ t_i, otherwise \end{cases}$$

MINMAXDIST의 가장 중요한 의미는 질의 점 Q로부터 MBR R의 MINMAXDIST까지의 거리 내에 반드시 하나 이상의 객체가 존재한다는 것이다. 이 성질을 이용하면 단말노드의 MBR내에 존재하는 객체 중에서 우선적으로 거리계산이 요구되는 객체를 구분할 수 있는 것이다. 즉, MINMAXDIST 거리 내에 존재하는 객체는 우선적으로 거리계산이 요구되므로, 거리계산을 수행한 후 반환을 위하여 Return_Q로 보내진다. 나머지 객체들은 일단 거리계산을 연기하여도 되므로 거리 계산 없이 Delay_Q로 보내진 후 추후의 추가적인 질의에 대비한다.

다음에 논의할 내용은 MBR R내에 존재하는 객체들이 MINMAXDIST내에 존재하는 것 인지의 여부를 판단하는 방법에 대하여 설명한다. 이를 판단하는 가장 간단한 방법은 실제로 질의 점 Q와 객체들과의 거리계산을 수행하면 파악할 수 있으나, 모든 객체와의 거리계산을 수행하면 질의시간 단축에 아무런 도움이 되지 못한다. 본 논문에서는 VP 필터링 개념을 이용하여, 실제 거리 계산 없이 이를 판단한다. 이를 위하여, 첫 번째 단계로, d(VP,Q) - MINMAXDIST(Q,R) 와 d(VP,Q) + MINMAXDIST (Q,R) 사이에 존재하는 객체를 필터링 하여 거리계산을 수행한다. 여기서 d는 거리함수를 나타내며, Q는 질의 점, R은 단말노드를 나타낸다.

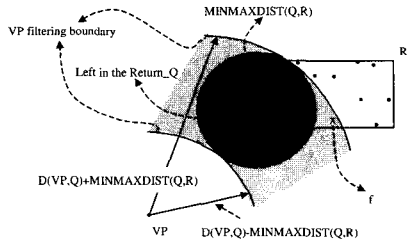

그림 9 추가객체요구 k-NNQ 처리를 위한 VP 필터링 개념

이와 같이 객체를 필터링하고 거리계산을 수행하는 단계를 VP 필터링 단계라고 한다. 필터링 된 객체들은 Return_Q로 옮겨질 가능성이 높은 후보 객체가 된다. 그림 9에서 "VP filtering boundary"라고 표시된 엷은 색으로 표시된 트랙 부분에 위치한 객체들이 여기에 속한다.

다음 단계에서는 이 후보들 중에서 실제로 MINMAX DIST에 속하는 객체들을 구별해야 한다. 후보들 중에는 MINMAXDIST 거리안에 속하지 않는 객체들이 포함될 수 있다. 예를 들어, 그림 9에서 'f'라고 표시된 객체의 경우에는 후보에는 포함되었지만 실제로는 MINMAX DIST의 범위를 벗어난다. 이러한 객체들은 후보에서 제외된다. 이를 체크하기 위한 조건은 d(f,Q)≥MIN MAXDIST(Q,R)이며 이 조건을 만족하는 객체들은 Delay_Q에 삽입한다. 물론, 이 객체들은 일단 거리계산이 수행된 상태이므로 이 객체들이 다시 Return_Q로 삽입되는 경우에는 다시 거리계산을 수행할 필요가 없다. 이 단계의 조건체크를 끝내면 Return_Q에는 그림 9에서 어두운 색깔로 표시된 원안의 객체들만이 남게 된다.

단 하나의 단말노드만을 고려한 경우에는 앞에서 설명한 두 가지의 단계를 거치면 항상 가장 가까운 거리에 위치한 객체가 반환된다. 그러나, Return_Q에 객체가 있는 상태에서 Main_Q로부터 추가적인 단말노드의 필터링이 수행되는 경우에 아래와 같은 경우를 대비하여야 한다. 다음과 같은 상황을 고려해보자. 현재, Return_Q에 질의점 Q로부터의 거리가 각각 2, 5, 8인 객체 O_2, O_5, O_8 가 있다고 가정하자. 이때 추가적인 질의요구에 의하여 Main_Q에서 MINDIST가 1인 단말노드 R이 있는 경우에, R내에 있는 객체들에 대하여 필터링을 수행한다. MINMAXDIST(Q,R)이 3이라고 하고, R내에 거리가 각각 1, 3, 4인 객체 O_1, O_3, O_4 가 있다면, 앞에서 설명한 두 단계의 필터링을 거치면 O_1과 O_3는 Return_Q로, O_4 는 Delay_Q로 가게 된다. 그러면, 추가적인 반환에 의하여, Return_Q로부터 O_1, O_2, O_3, O_5, O_8의 순서로 반환되게 되지만, O_4 는 여전히 반환되지 못하고 Delay_Q에 남게된다.

이 문제는 두 개의 변수 Qmax와 Rmax를 선언하여 다음과 같이 해결할 수 있다. Qmax는 Return_Q내에 존재하는 객체들의 거리 중에서 최대거리를 나타내며, Rmax는 Return_Q에 삽입될 객체들의 거리 중에서 최대거리를 나타낸다.

이제, 이 두 변수를 가지고 단말노드의 객체들을 Retrun_Q에 삽입할 때, Min(Qmax, Rmax)보다 먼 거리를 지니는 객체는 모두 Return_Q가 아닌 Delay_Q로 보내도록 한다. 앞의 예에서는, Qmax 가 8이고 Rmax는 3이다. 그러므로 Min(Qmax, Rmax)인 3보다 먼 거리를 지니는 객체인 O_5와 O_8는 Return_Q로부터 제거되고, Delay_Q에 삽입하게 된다. 이렇게 하면, 앞에서 지적한 O_4에 대한 오류를 막을 수 있다. 위에서 설명한 질의 처리 개념을 정리하면 다음과 같다.

<규칙 3> 추가객체요구 NNQ질의에서, 다음 조건들을 만족하는 노드내의 임의의 객체들의 집합은 최근접 객체를 반드시 포함한다.

조건 1: (d(VP,Q)-MINMAXDIST(Q,R)
≤d(VP,child object)

≤d(VP,Q)+MINMAXDIST(Q,R))

조건 2: (d(child object,Q)≥MINMAXDIST(Q,R))

<규칙 4> 노드의 엔트리 개수가 n이고, 규칙 3의 조건 1을 만족하는 노드내의 객체가 k개이면, (n-k)개의 객체에 대한 n차원의 거리계산은 Delay_Q에서 Return_Q로 이동할 때까지 보류된다.

이러한 절차가 그림 10의 알고리즘에 표현되어 있다. 이 알고리즘은 더 이상의 추가적인 질의가 발생하지 않거나, 세 개의 큐에 더 이상의 노드나 객체가 존재하지 않을 때까지 계속된다. k-NNQ에 대한 해답 반환은 라인P9에 나타난바와 같이 언제나 Return_Q로부터 주어진다. 라인 P12에서 P17은 Main_Q가 Return_Q보다 더 가까운 거리의 단말노드를 지니는 경우에 새로운 객체를 생성하는 단계이다. 단말노드의 객체들은 필터링단계를 거친 후 Return_Q 나 the Delay_Q에 삽입된다. VP 필터링 단계와 앞에서 해결한 두 가지 문제점에 대한 부분은 라인 M8에서 M18사이에 언급되어 있다. 객체가 Return_Q로 보내지는 경우는 라인M13에 언급되었고, Delay_Q로 보내지는 경우는 라인M15에 언급되어 있다.

```
Procedure Incremental_NNQ(query object Q)
P1 PriorityQueue Main_Q; // in order of distance from the query point
P2 PriorityQueue Return_Q; // in order of distance from the query point
P3 Queue Delay_Q; // No special ordering
P4 Numeric Qmax, Rmax;
P5 Main_Q.insert(0, root);
// Continue until all objects are returned
P6 while (.not. Main_Q.isempty()) .or. (.not. Return_Q.isempty())
                                  .or. (.not. Delay_Q.isempty())
P7    if (.not. Return_Q.isempty())
P8       if (.not. Main_Q.isempty())
               .or. d(Return_Q.top, Q) < MINDIST(Main_Q.top, Q)
P9          ReturnQPop();
P10         wait until next nearest neighbor is required;
P11      else
P12         MainQPop();
P13   else if (.not. Delay_Q.isempty())
P14      DelayQAllMove();
P15   else if (.not. Main_Q.isempty())
P16      MainQPop();
P17 endwhile;
P18 report ("no_more_object");
Procedure MainQPop()
M1  {
M2     top=Main_Q.top;
M3     Qmax=distance of the farthest object of the Return_Q;
M4     Rmax=0; /* distance of the object with the maximum distance
M5                among those selected to be put in the Return_Q */
M6     if top is leaf node
M7        for each child object of the top
M8           if the child object satisfies the condition
             // Check the filtering conditions
M9            (d(VP,Q)-MINMAXDIST(Q,R)≤d(VP,child object)
                                  ≤d(VP,Q)+MINMAXDIST(Q,R))
M10                            .and. (d(child object,Q)≥MINMAXDIST(Q,R))
M11              if d(VP, child object) > Rmax
M12                 { Rmax=d(VP, child object); }
             // if an object satisfies
                        the filtering conditions then sent to Return_Q
M13             put it into Return_Q;
M14          else
             // if an object does not satisfy
                        the filtering conditions then sent to Delay_Q
M15             put it into Delay_Q;
M16       for each object of the Return_Q
M17          if d(VP, object) > Min(Qmax, Rmax)
M18             put it into Delay_Q;
M19    else // top is intermediate node
M20       for each child of the top
M21          reinsert the child into Main_Q according to their MINDIST;
M22  }
Procedure ReturnQPop()
     { Update the Return_Q
          according to the distance after popping the top object; }
Procedure DelayQAllMove()
     { move all objects of Delay_Q
          to the Return_Q according to their distance; }
```

그림 10 VP 필터링을 이용한 추가객체요구 k-NNQ 알고리즘

4. 실험 및 평가

본 논문에서 제시한 알고리즘들의 효율성과 유용성을 보이기 위하여 알고리즘들을 구현하고 실험을 수행하였다. 영역질의의 경우에는 그림 2에서 설명한 알고리즘[1][2]과 비교하였으며, k-NNQ를 수행하는 추가 객체요구 질의 알고리즘은 그림3에서 제시한 알고리즘[9]과 비교하였다. 실험은 메모리 128Mbytes의 450MHz Pentium II PC에서 수행되었다. 인덱스 구조로는 R-트리 계열의 트리중 에서 R^*-트리를 사용하였다. 데이타의 차원 변화에 따른 알고리즘의 성능을 측정하기 위하여 3차원과 10차원의 인공 데이타를 각 50,000개씩 생성하였으며, 실제 데이타에서의 성능 측정을 위하여 50,000개의 자연 이미지를 가지고 인덱스를 구축하였다. R-tree를 비롯한 R-tree계열의 트리들은 10차원이내의 환경에서 좋은 성능을 나타내는 것으로 나타나 있으므로 본 연구에서도 10차원 내외의 데이타에 대한 실험을 수행하였다.

이미지를 다차원 데이타로 표현하기 위하여 본 연구에서는 이미지 타일 평균 RGB 방법[11]을 사용하였다. 이미지 타일 평균 RGB(ARIT : Average RGB by Image Tiling) 방법의 기본적인 개념은 전체 이미지를 동일 크기의 타일로 분할하고 각 타일에 대하여 평균 RGB 값을 구하여 일정한 순서에 의해 각 타일의 평균 RGB 값을 구한다. 구하여진 이미지 타일 평균 RGB 값을 이용하여 인덱스 트리를 구성함으로써 필터링 과정에서 이용한다. 분할된 각 타일에 대한 평균 RGB 값은 QBIC 시스템[12] 에서와 같이 고차원 컬러 히스토그램의 급격한 저차원 변환 과정으로 인한 정보 손실이 존재하지만, 필터링 수행 시에 비교하는 두 이미지의 동일한 위치의 타일에 대한 평균 RGB 값을 비교함으로써 저차원 변환에 대한 색상 정보 손실을 보완할 수 있다.

4.1 VP 필터링을 이용한 영역질의 성능

VP 필터링을 이용한 영역질의처리의 성능 평가를 위하여, 10개의 영역질의를 수행하였으며, 평균 질의응답시간을 측정하여 VP 필터링을 수행하지 않은 경우의 응답시간과 비교하였다. 10개의 영역질의는 질의점으로부터 5%범위에서 50%범위까지 점점 범위를 넓혀가면서 질의하였다. 인덱스는 3차원과 12차원으로 구성하여 비교하였다. 그림 11에서 알 수 있듯이 VP 필터링을 사

용한 경우의 질의 응답시간이 두 차원에서 모두 좋은 성능을 나타냄을 알 수 있다. 또한, 질의의 범위가 증가할수록 두 가지 방법의 응답시간의 차이가 점점 더 증가함을 알 수 있다.

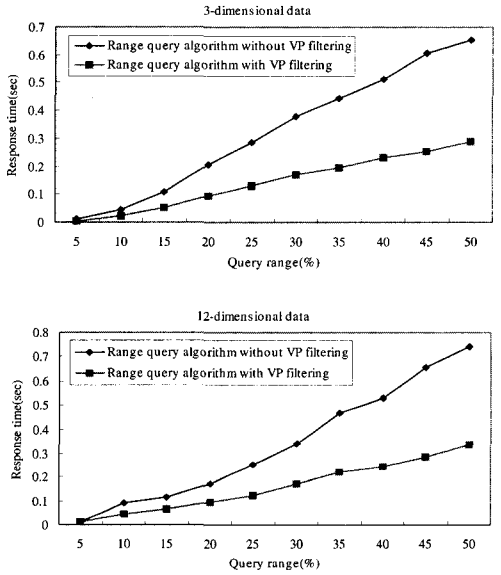

그림 11 영역질의 처리 응답시간 비교

4.2 VP 필터링을 이용한 추가 객체요구 k-NNQ 처리 성능

VP 필터링을 이용한 추가 객체요구 k-NNQ 알고리즘의 성능을 측정하기 위하여, 질의 객체로부터 가장 가까운 객체로부터 하나씩 다음 가까운 객체순으로 30개의 객체를 반환하여 그 결과가 반환되는 응답 시간을 측정하였다. 물론, 30개의 객체 이후의 객체 반환 시간에 있어서도 30개 이내의 반환시간과 비교하여 유사한 결과 추이를 보임을 알 수 있었다. 인위적으로 생성한 데이타에 대한 k-NNQ 실행 성능 결과가 그림 12에 나타나 있다. x축은 반환되는 객체의 수를 나타내며, 물론, 이 수치는 객체가 하나씩 반환될 때마다 축적 된 수치이다. y축은 마찬가지로 축적된 객체 반환 응답시간을 초단위로 나타낸다. 각 그래프들은 각각 다른 데이타 집합에 대하여 다른 질의를 수행한 것이므로 y축의 값의 범위가 그래프마다 다르다.

그래프에서 알 수 있듯이 응답시간의 증가가 계단모양을 나타냄을 볼 수 있다. 이는 다음 최근접객체의 거리와 상관 관계가 있다. 즉, 만약 다음 최근접객체가 많이 떨어져 있다면, 그 객체를 찾기 위하여 보다 많은 노드들을 검색하여야 한다. 그러므로, 급작스런 응답시간의 증가를 야기할 수 있다. 반면에, 다음 최근접객체가 가까운 곳에 위치한다면, 적은 수의 노드를 검색으로 찾을 수 있거나, 현재 최근접객체와 동일 노드 내에 위치할 가능성이 있으므로, 응답시간의 증가가 없을 가능성이 있다.

그래프에서 볼 수 있듯이, VP 필터링 기법을 사용하여 질의를 처리한 경우가 사용하지 않은 경우에 비하여 항상 더 좋은 성능을 나타냄을 알 수 있다. 또한, 응답시간의 차이가 데이타의 차원이 증가함에 따라 더욱 커짐을 알 수 있다. 여러 데이타의 차원에 걸쳐서 평균적으로 약 12-19%의 성능 향상을 보여 준다.

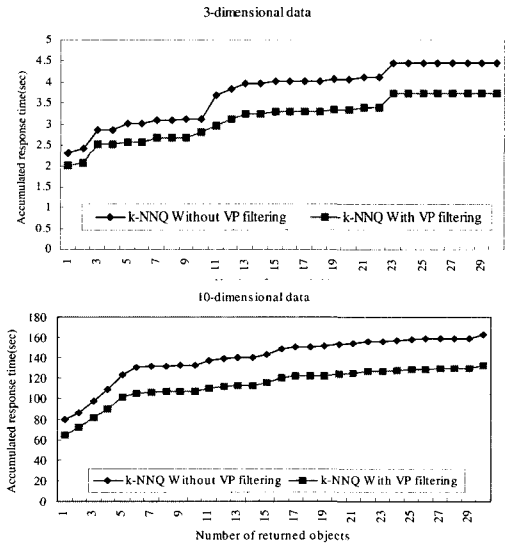

그림 12 인공데이타에서의 추가객체요구 k-NNQ 응답시간 성능비교

그림 13에서는 제안된 추가 객체요구 k-NNQ 처리 알고리즘의 필터링 능력을 분석하고 질의 응답시간과의 상관성을 분석하기 위하여 Return_Q로 보내지는 객체의 개수를 Delay_Q에 보내지는 객체의 개수와 비교하였다. 만약에 필터링 메카니즘이 없다면 모든 객체는 하나의 큐에 존재하게 되며, 모든 객체의 거리계산 또한 수행되게 된다. 그러므로, 필터링 메카니즘을 통하여 객체의 거리계산을 가능한 한 늦춤으로 인하여 전체적인 질의 응답시간을 빠르게 하는 것이다. 따라서, Delay_Q로 보내지는 객체의 개수가 상대적으로 많을수록 많은

이득이 발생할 수 있다. 3차원과 10차원에서 k-NNQ로 반환되는 첫 번째 객체에서부터 30번째 객체에 이르기까지의 각각의 경우에 Delay_Q와 Retrun_Q로 반환되는 객체의 개수를 비교하여 보여준다. 그림 12의 3차원과 10차원 그래프와 그림 13을 비교해보면, 두 점간의 증가경사도가 필터링을 적용한 경우의 그래프가 완만한 곡선을 보임을 알 수 있다. 특히, Delay_Q로 보내지는 객체의 수가 많은 경우에는 더욱 그러한 경향을 보인다. 예를 들어, 그림 12에서 3차원 데이타에 대한 그래프를 보면, x축의 9지점과 11지점 사이를 비교해보면 알 수 있다. 이러한 성능의 향상은 k-NNQ에서 차원이 증가할수록 함께 증가함을 알 수 있다.

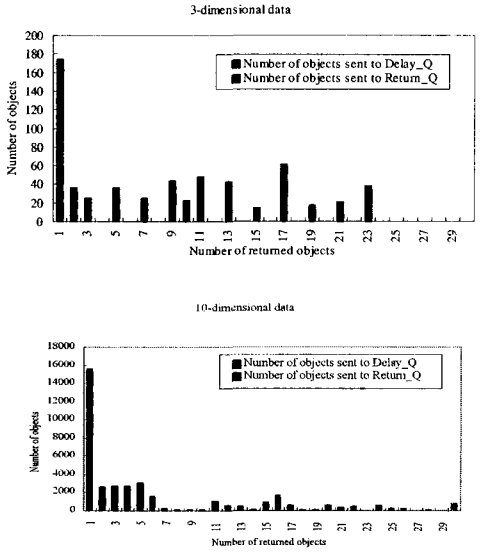

그림 13 인공데이타에서의 추가객체요구 k-NNQ의 큐 객체수 성능비교

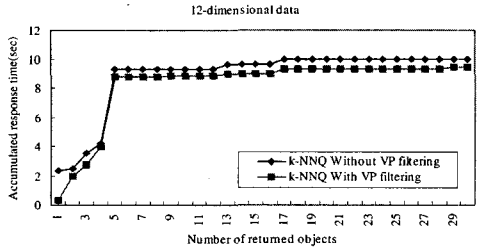

그림 14 실제 이미지 데이타의 추가객체요구 k-NNQ의 응답시간 성능비교

그림 14은 실제 이미지 데이타에 대하여 k-NNQ를 수행한 경우의 질의 응답시간의 변화를 보여준다. 결과의 경향은 인공 데이타의 경우와 유사한 경향을 보였다. 또한, 인공 데이타와 마찬가지로 데이타의 차원이 증가함에 따라 VP 필터링을 적용한 경우가 적용하지 않은 경우보다 월등한 성능 향상을 보임을 알 수 있다. 12차원 데이타에서의 성능 증가는 약 9%정도인 것으로 나타났다.

5. 결 론

본 논문에서는 멀티미디어 데이타를 인덱싱하고 질의처리를 수행하기 위한 R-트리 계열의 다차원 인덱싱 구조에서의 효율적인 질의처리 방법을 소개하였다. MVP-트리의 필터링 아이디어를 R-트리 계열의 인덱싱 구조에 적용한 VP-필터링을 사용하였다. VP-필터링을 위하여 전체 데이타 도메인에 대하여 하나의 VP를 설정하고, 모든 객체에 대한 VP와의 절대거리를 가지고 트리에 삽입되도록 하였다. 이 VP와의 거리는 영역질의, 추가객체요구 NNQ를 수행할 때, 필터링 조건에서 사용되었다. VP거리를 이용한 필터링 조건에 만족하지 않는 객체는 해답에서 제외되므로, 다차원의 거리계산 시간을 절약하도록 알고리즘을 전개하였다. VP 필터링을 적용하기 위해서는 트리의 구성 시에 삽입 객체와 VP간의 거리값을 계산하여 함께 저장한다. 그러나, 새로운 객체가 삽입되거나, 삭제되더라도 VP와의 거리는 절대적인 값을 지니므로 추가적인 거리계산은 필요하지 않는다.

실험에서는 기존의 VP 필터링을 적용하지 않은 경우와 VP 필터링을 적용한 경우의 질의 처리 시간을 측정하여 상당한 성능 증가를 얻었음을 보여주었다. 영역질의와 추가 객체요구기법을 적용한 NNQ 모두의 경우에 항상 VP 필터링을 적용한 경우에 짧은 응답시간을 보임을 알 수 있었다. 또한 이러한 응답시간의 단축이 VP 필터링과 매우 긴밀하게 연관되어 있음을 보였다.

제안된 VP 필터링 기법은 기존의 R-트리 계열의 인덱싱 구조를 이용하여 개발된 다차원 데이타 검색시스템에 그대로 적용가능하며 상당한 성능의 향상을 가져다줄 수 있으며, 지리정보 시스템의 데이타에서도 적용할 수 있다.

제안된 필터링 기법의 기본적인 개념은 다차원 객체간의 거리계산 시간을 줄이는 것으로서, R-트리 계열의 인덱싱 구조뿐만 아니라 새로이 발표되는 인덱싱 구조에 대한 필터링에 대한 연구가 요구된다.

참 고 문 헌

[1] Antonin Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," Proceedings of the ACM SIGMOD Conference, pages 47-57, 1984.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," Proceedings of the ACM SIGMOD Conference, pages 322-331, 1990.

[3] Stefan Berchtold, Daniel A. Keim, and Hans Peter Kriegel, "The X-Tree: An Index Structure for High-Dimensional Data," Proceedings of the VLDB Conference, pages 28-39, 1996.

[4] King-Ip Lin, H. V. Jagadish, and Christos Faloutsos, "The TV-tree - An Index Structure for High-Dimensional Data," VLDB Journal, Vol. 3(4), pages 517-542, 1994.

[5] Tolga Bozkaya and Meral Ozsoyoglu, "Distance-Based Indexing for High-Dimensional Metric Spaces," Proceedings of the ACM SIGMOD Conference, pages 357-368, 1997.

[6] Jeffrey K. Uhlmann, "Satisfying General Proximity/ Similarity Queries with Metric Trees," Information Processing Letters, Vol. 40, pages 175-179, 1991.

[7] Thomas Seidl and Hans-Peter Kriegel, "Optimal Multi-Step k-Nearest Neighbor Search," Proceedings of the ACM SIGMOD Conference, pages 154-165, 1998.

[8] Gisli R. Hjaltason and Hanat Samet, "Ranking in Spatial Databases," Proceedings of the 4th International Symposium on Large Spatial Databases, Lecture Notes in Computer Science 951, Springer-Verlag, pages 83-95, 1995.

[9] Peter N. Yianilos, "Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces," ACM-SIAM Symposium on Discrete Algorithms, pages 311-321, 1993.

[10] Nick Roussopoulos, Stephen Kelley, and Frederick Vincent, "Nearest Neighbor Queries," Proceedings of the ACM SIGMOD Conference, pages 71-79, 1995.

[11] Byung-Gon Kim, Jung-Woon Han, Jaeho Lee, and Hae-Chull Lim, "Feature Extraction and Query Processing Technique in Image Database Applications : Design and Evaluation," Proceedings of the ICACT2000, 2000

[12] C. Faloutsos, R. Barber, M. Flickener, J. Hafner, et al., "Efficient and effective Querying by Image content," Journal of Intelligent Information Systems, Vol3, pages 231-262, 1994

김 병 곤

1990년 홍익대학교 전자계산학과 이학사. 1992년 홍익대학교 전자계산학과 이학석사. 2001년 홍익대학교 전자계산학과 이학박사. 2001년 ~ 현재 부천대학 사무자동화과 전임강사. 관심분야는 멀티미디어 데이타베이스, 웹 데이타베이스

이 재 호

1987년 홍익대학교 전자계산학과 이학사. 1989년 홍익대학교 전자계산학과 이학석사. 1996년 홍익대학교 전자계산학과 이학박사. 1996년 ~ 현재 인천교육대학 컴퓨터교육과 조교수. 관심분야는 컴퓨터교육, 분산 데이타베이스, 웹 데이타베이스

임 해 철

1976년 서울대학교 계산통계학과 이학사. 1978년 한국과학기술원 전산학과 이학석사. 1988년 서울대학교 컴퓨터공학과 공학박사. 1989년 ~ 1990년 University of Florida Post doc. 1981년 ~ 현재 홍익대학교 컴퓨터공학과 교수. 관심분야는 멀티미디어, 분산, 객체지향 데이타베이스