순환검색공간에서 K-최근접객체 쌍을 찾는 알고리즘에 관한 연구

Algorithm for Finding K-Nearest Object Pairs in Circular Search Spaces

선 휘 준* 김 홍 기**
Hwi Joon Seon Hong Ki Kim

요 약 최근의 검색시스템에서는 두 객체집합에 대하여 가장 근접해 있는 K개의 객체 쌍을 찾는 질의가 자주 발생한다. 이러한 K개의 최대근접 객체 쌍을 찾는 질의를 효율적으로 처리하기 위해서는 객체의 순환적 위치속성이 고려되어야 한다. 본 논문은 순환도메인을 갖는 검색공간에서 서로 간에 가장 근접해 있는 K개의 객체 쌍을 찾는 최적의 알고리즘을 제안하고 그 성능을 실험을 통하여 보인다. 제안한 알고리즘은 객체의 순환적 위치속성이 반영된 순환검색거리를 이용하여 K개의 최대근접객체 쌍을 찾는 비용을 최적화한다.

키워드 : 최대근접객체쌍, 순환검색거리, 순환위치속성, 순환도메인

Abstract The query of the K closest object pairs between two object sets frequently occurs at recently retrieval systems. The circular location property of objects should be considered for efficiently process queries finding such a K nearest object pair. In this paper, we propose the optimal algorithm finding the K object pairs which are closest to each other in a search space with a circular domain and show its performance by experiments. The proposed algorithm optimizes the cost of finding the K nearest object pairs by using the circular search distances which is much applied the circular location property.

Keywords : Nearest Object Pair, Circular Search Distance, Circular Location Property, Circular Domain

† 이 논문은 2011년 동신대학교 학술연구비에 의하여 연구되었음.

* 신경대학교 애니메이션학과 교수 hjseon@sgu.ac.kr
** 동신대학교 정보보안학과 교수 hkkim@dsu.ac.kr(교신저자)

1. 서 론

패턴 인식, 공간 데이터 분석, 이미지 처리, 지리정보 시스템 등과 같은 최근의 여러 응용분야에서는 더욱 복잡하고 많은 양의 공간 및 비공간적 성질을 갖는 객체를 취급하고 있다. 이러한 객체들을 관리하기 위해서는 많은 저장 공간과 처리시간이 요구되며, 적절한 색인구조와 질의처리 기법을 적용함으로써 공간 질의를 효율적으로 처리할 수 있다.

다양한 응용에서 두 객체집합에 대하여 공간상의 거리를 기준으로 하여 서로 간에 가장 근접해 있는 객체 쌍을 찾는 질의가 자주 발생한다[1, 2, 3, 11]. 예를 들면 서로 다른 종류의 지형정보를 갖고 있는 두 개의 객체집합들로부터 거리적으로 가장 근접해 있는 객체 쌍을 검색하는 응용이 있을 수 있다.

이러한 최대근접객체 쌍을 찾는 질의의 처리는 보조기억장치 접근 및 연산을 위한 많은 처리 시간이 요구된다. 보조기억장치의 접근 시간은 연산 시간에 비해 상대적으로 매우 크기 때문에 질의처리의 성능을 높이기 위해서는 보조기억장치의 접근 횟수를 줄여야 한다.

객체집합들이 R-트리 유형의 색인으로 구축되었을 때 최대근접객체 쌍을 찾는 질의 처리에 대한 대부분의 연구는 두개의 트리를 하향식으로 순회하면서 검색대상이 되는 노드 쌍을 우선순위 큐(priority queue)에 저장한 후, 검색대상이 되는 노드 쌍을 선택하기 위해 노드 간의 최소거리만을 사용하였다[4, 5, 6]. 그러나 노드들 간에 겹침 정도와 객체들의 순환적 위치속성을 고려하지 못했다. 최소거리만을 사용하여 양쪽노드의 모든 자식노드들에 대해 검색대상을 선정하는 것은 불필요한 노드의 접근이 발생하고 연산비용 및 보조기억장치의 접근 횟수가 증가하게 된다.

따라서 최대근접객체 쌍을 찾는 질의의 성능 향상을 위해서는 색인에서 불필요한 노드의 검사를 피하기 위한 검색거리 측도에 대한 연구가 반드시 필요하다. 또한 검색거리 측도에는 객체들의 순환적 위치속성이 반영되어야 한다.

본 논문에서는 R-트리 유형의 색인에서 찾고자 하는 객체 쌍의 수 K가 미리 정해진 상황에서 점진적으로 최대근접 객체 쌍을 찾는 최적의 알고리즘을 제안한다. 그리고 두 색인에서 불필요한 노드 쌍들의 검사를 피하기 위한 검색거리 측도인 순환최

소거리(circular minimum distance), 순환최대거리(circular maximum distance), 순환최적거리(circular optimal distance)를 정의한다. 또한 제시된 검색거리측도를 R*-트리[1]에 적용함으로써 기존의 방법들과 K개의 최대근접 객체 쌍을 찾는 질의 처리 비용을 비교 평가한다. 성능 평가는 최대근접객체 쌍을 찾는데 소요되는 처리시간과 보조기억장치 접근 횟수 등에 따라 분석된다.

논문의 구성은 다음과 같다. 관련 연구인 2장에서는 기존의 방법들이 갖고 있는 문제점들을 알아보고, 3장에서는 최대근접 객체 쌍을 찾는 처리 비용을 최적화하기 위한 검색거리 측도인 순환최소거리, 순환최대거리 그리고 순환최적거리를 정의한다. 그리고 정의된 순환검색거리 측도를 적용한 K개의 최대근접 객체 쌍을 찾는 알고리즘을 제안한다. 4장에서는 실험을 통하여 제안된 알고리즘의 처리성능을 기존의 방법들과 비교 분석하고, 끝으로 5장에서는 결론을 내린다.

2. 관련연구

일반적으로 최대근접 객체 쌍을 찾는 방법은 두 개의 객체집합에 속하는 모든 객체들로 이루어진 순서쌍의 모든 원소를 상호간의 거리 순으로 정렬하여 상위 K개를 질의결과로 산출한다. 그러나 이 방법은 질의 결과를 산출하는데 따른 수행시간이 순서쌍을 이루는 객체의 수에 결정되는 단점을 가지고 있다.

최대근접 객체 쌍을 찾는 알고리즘의 실행시간 최적화를 위해 깊이우선(depth-first) 검색방법과 넓이우선(breadth-first) 검색방법이 연구되었다[7, 8]. 깊이우선 검색방법은 색인에서 리프노드에 더 가까이 위치한 노드 쌍에 높은 우선순위를 부여하여 다음 검색대상이 되도록 하였다. 그러나 이 방법은 노드 쌍들 간에 최대 근접성을 정확히 반영하지 못했으며 노들 간에 겹침도 고려하지 않았다. 넓이우선 검색방법은 면적이 더 큰 노드를 가지는 노드 쌍에 더 높은 우선순위를 부여하여 검색대상을 선택하였으나 깊이우선 검색방법과 동일한 문제점을 가지고 있다.

객체 쌍의 개수 K가 미리 정해져 있는 경우 K에 따른 최소거리 값을 거리 큐(distance queue)에 저장하는 방법이 제안되었다[9]. 이 방법에서는 알고리즘의 시작단계에서 현재까지의 계산된 객체 쌍의 거리 중 가장 작은 K개를 유지하고 그 중 가장 큰 거리 값을 한계 값 Dmax라고 정한다. 차후 생성되는 노드 쌍들 중에서 거리가 Dmax의 값보다 더 큰 값을 가지는 노드들은 검색대상에 제외된다. 그러나 이 방법은 두 색인에서 겹쳐져 있는 노드 쌍들이 많이 발생할수록 최소거리 값이 0인 노드 쌍들이 많이 존재하게 됨으로써 방문되는 노드 쌍의 개수가 증가하게 된다.

이러한 문제점들을 해결하기 위해 최대 거리 우선 방법과 겹침 비율 우선 방법이 제안되었다[9].

최대 거리 우선 방법에서는 각 노드 쌍을 이루는 두 노드 간에 최대 거리를 계산하여 그 값이 작은 노드 쌍을 기준으로 검색대상을 결정한다. 노드 쌍의 최대 거리가 더 작다는 것은 거리 값이 더 작은 범위 안에서 자식 노드 쌍들이 생성되기 때문에 색인에서 방문되는 노드의 수를 줄일 수 있다. 그러나 최대 거리 우선 방법은 노드의 겹침 정도를 정확히 반영하지 못하는 단점을 가지고 있다(그림 1). 그림 1에서 노드 쌍 (a)의 최대거리가 노드 쌍 (b)의 최대거리보다 작기 때문에 노드 쌍 (a)가 다음 방문대상이 된다. 그러나 노드 쌍 (b)의 겹침 정도가 커서 더 작은 거리 값을 가지는 노드 쌍들이 더 많기 때문에 노드 쌍 (b)가 방문대상으로 선택되어야 한다.

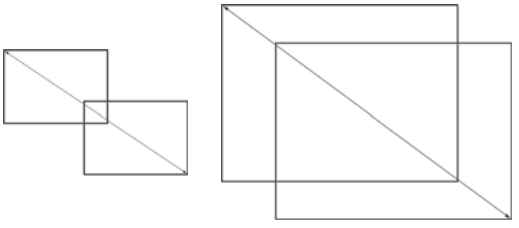
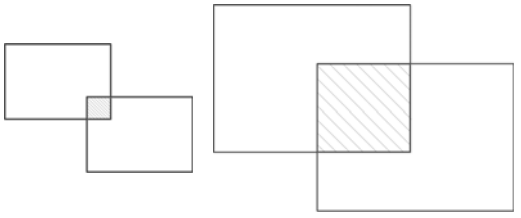


그림 1. 최대 거리 우선 방법

겹침 비율 우선 방법에서는 두 노드의 겹침 정도가 더 큰 노드 쌍을 기준으로 검색될 노드들을 선정한다. 노드 쌍의 겹침 비율은 두 노드의 겹치는 영역의 면적을 두 노드의 면적의 합으로 나누어 계산한다. 두 노드의 겹침 비율이 높을수록 거리 값이 작은 노드 쌍을 많이 생성할 수 있게 되어 색인에서 방문되는 노드의 수를 줄일 수 있다 그러나 겹침 비율 우선 방법은 두 노드 간의 거리를 정확히

반영하지 못하는 문제를 가지고 있다(그림 2). 그림 2에서 노드 쌍 (b)의 겹침 정도가 노드 쌍 (a)의 겹침 정도보다 커서 노드 쌍 (b)가 다음 방문대상이 된다. 그러나 노드 쌍들의 거리 값을 비교해 볼 때 (a)가 평균적으로 (b)보다 더 작은 거리 값을 유지하기 때문에 노드 쌍 (a)가 방문대상으로 선택되어야 한다.



(a) (b)

그림 2. 겹침 비율 우선 방법

대부분의 최대근접 객체 쌍을 찾는 알고리즘은 노드 쌍들 간에 거리, 면적 그리고 겹침 정도를 기준으로 하여 검색되는 노드들의 수를 최소화했다. 그러나 R-트리 유형의 색인에서 찾고자 하는 객체 쌍의 수 K가 미리 정해진 경우에 최대근접 객체 쌍을 찾는 알고리즘의 최적화 방안에 관한 연구는 거의 이루어지지 않았다. 또한 객체들의 순환적 위치 속성과 두 색인에서 노드들 간의 거리 및 겹침 정도를 정확히 반영할 수 있는 검색거리에 관한 연구도 이루어지지 않았다.

3. 순환검색거리 측도를 이용한 K-최근접 객체 쌍을 찾는 알고리즘

본 장에서는 K개의 최대근접객체 쌍을 찾는 비용을 최적화하기 위한 검색거리 측도로 순환최소거리와 순환최적거리를 알아보고 두 노드 쌍 간의 가능한 거리 중 최대거리인 순환최대거리를 정의한다. 그리고 정의된 순환검색거리 측도를 적용한 K개의 최대근접 객체 쌍을 찾는 알고리즘을 제안한다.

3.1 순환검색거리

[12]에서는 최대근접질의의 처리비용을 최소화하기 위한 검색거리 측도로 순환최소거리 CMINDIST와 순환최적거리 COMINDIST를 정의하였다.

본 논문에서는 두 노드 쌍 M과 M'사이의 가장 가까운 거리인 순환최소거리(Circular MINimum DISTance: CMINDIST)를 다음과 같이 수정한다.

【정의 1】 $D_i = \{d_1, d_2, \ldots, d_m\}$ (여기에서 $d_j < d_{j+1}$, $j = 1, 2, \cdots, m-1$)를 N차원 검색공간을 구성하는 i번째 도메인이라 할 때, N차원 검색공간에서 두 노드 쌍 M와 M'간의 순환최소거리 $CMINDIST$ 는 다음과 같다.

$$CMINDIST(M, M') = \sum_{i=1}^{N} d_i^2(M, M')$$

만약 D_i가 순환도메인이면

$$d_i(M, M') = \min(|M_i - M_i'|, \quad |m_i - d_1| + |m_i' - d_m|)$$

그렇지 않으면

$$d_i(M, M') = |M_i - M_i'|$$

여기에서

$$M_i = \begin{cases} M_{Li}, & M_{Li} > M_{Ui}' \\ M_{Ui}, & M_{Ui} < M_{Li}' \\ 0, & otherwise \end{cases}$$

$$M_i' = \begin{cases} M_{Li}', & M_{Ui} < M_{Li}' \\ M_{Ui}', & M_{Li} > M_{Ui}' \\ 0, & otherwise \end{cases}$$

$$m_i = \begin{cases} M_{Li}, & M_{Li} > M_{Ui}' \\ M_{Li}, & M_{Ui} < M_{Li}' \\ 0, & otherwise \end{cases}$$

$$m_i' = \begin{cases} M_{Ui}', & M_{Ui} < M_{Li}' \\ M_{Ui}', & M_{Li} > M_{Ui}' \\ 0, & otherwise \end{cases}$$

$d_i(M, M')$: D_i에서 M와 M'사이의 거리

M_{Li}, M_{Ui} : 도메인 D_i에서 노드 M 범위의 시작과 끝

M_{Li}', M_{Ui}' : 도메인 D_i에서 노드 M' 범위의 시작과 끝

정의된 CMINDIST는 순환도메인이 포함된 N차원 검색공간에서 노드 M에 포함되어 있는 부검색공간들 중에서 노드 M'에 가장 근접하고 있는 객체 또는 부검색공간을 결정하기 위한 최소의 거리이다. 순환최적거리는 노드 M에서 노드 M'의 임의의 한 변을 포함할 수 있는 거리들 중 최소거리로 계산된다. 또한 순환도메인이 존재한다면 COMINDIST (Circular Optimized MINimum value of all DISTances)는 질의결과의 산출시 객체들의 순환적 위치속성을 정확히 반영한 거리측도이다.

【정의 2】 $D_i = \{d_1, d_2, ..., d_m\}$ (여기에서 $d_j < d_{j+1}$, $j = 1,2,\cdots,m-1$)를 N차원 검색공간을 구성하는 i번째 도메인이라 할 때, N차원 검색공간에서 두 노드 쌍 M와 M'간의 순환최적거리 $COMINDIST$는 다음과 같다.

$$COMINDIST(M, M') = \min\{d_i(M,M'), d_k(M,M'), d(M,M')\}$$

만약 D_i가 순환도메인이면

$$d_i(M,M') = \min_{1 \leqq i \leqq N} \left\{ \begin{array}{l} (|m_i - d_1| + |m_i' - d_m| + \sum_{\substack{i \neq k \\ 1 \leqq k \leqq N}} |Mr_k - Mr_k'|), \\ (|m_i - d_1| + |M_i' - d_m| + \sum_{\substack{i \neq k \\ 1 \leqq k \leqq N}} |Mr_k - mr_k'|) \end{array} \right\}$$

만약 D_k가 순환도메인이면

$$d_k(M,M') = \min_{1 \leqq i \leqq N} \left\{ \begin{array}{l} (|mr_i - mr_i'| + \sum_{\substack{i \neq k \\ 1 \leqq k \leqq N}} |M_k - d_1| + |M_k' - d_m|), \\ (|mr_i - Mr_i'| + \sum_{\substack{i \neq k \\ 1 \leqq k \leqq N}} |M_k - d_1| + |m_k' - d_m|) \end{array} \right\}$$

그렇지 않으면

$$d(M,M') = \min_{1 \leqq i \leqq N} \left\{ \begin{array}{l} (|mr_i - mr_i'| + \sum_{\substack{i \neq k \\ 1 \leqq k \leqq N}} |Mr_k - Mr_k'|), \\ (|mr_i - Mr_i'| + \sum_{\substack{i \neq k \\ 1 \leqq k \leqq N}} |Mr_k - mr_k'|) \end{array} \right\}$$

여기에서

$$mr_i = \begin{cases} M_{Li}, & M_{Li} \geqq M_{Ui}' \\ M_{Ui}, & M_{Ui} \leqq M_{Li}' \\ M_i, & otherwise \end{cases}$$

$$M_i = \begin{cases} M_{Li}, & \frac{(M_{Li}+M_{Ui})}{2} \leqq \frac{(M_{Li}'+M_{Ui}')}{2} \\ M_{Ui}, & otherwise \end{cases}$$

$$M_k = \begin{cases} M_{Lk}, & \frac{(M_{Lk}+M_{Uk})}{2} \geqq \frac{(M_{Lk}'+M_{Uk}')}{2} \\ M_{Uk}, & otherwise \end{cases}$$

$$m_i = \begin{cases} M_{Li}, & M_{Li} \geqq M_{Ui}' \\ M_{Li}, & M_{Ui} \leqq M_{Li}' \\ M_{Ui}, & otherwise \end{cases}$$

$$Mr_k = \begin{cases} M_{Lk}, & \frac{(M_{Lk}+M_{Uk})}{2} \geqq \frac{(M_{Lk}'+M_{Uk}')}{2} \\ M_{Uk}, & otherwise \end{cases}$$

$$mr_i' = \begin{cases} M_{Li}', & \frac{(M_{Li}+M_{Ui})}{2} \leqq \frac{(M_{Li}'+M_{Ui}')}{2} \\ M_{Ui}', & otherwise \end{cases}$$

$$mr_k' = \begin{cases} M_{Lk}', & \frac{(M_{Lk}+M_{Uk})}{2} \leqq \frac{(M_{Lk}'+M_{Uk}')}{2} \\ M_{Uk}', & otherwise \end{cases}$$

$$m_i' = \begin{cases} M_{Ui}', & \frac{(M_{Li}+M_{Ui})}{2} \leqq \frac{(M_{Li}'+M_{Ui}')}{2} \\ M_{Ui}', & otherwise \end{cases}$$

$$m_k' = \begin{cases} M_{Uk}', & \frac{(M_{Lk}+M_{Uk})}{2} \leqq \frac{(M_{Lk}'+M_{Uk}')}{2} \\ M_{Uk}', & otherwise \end{cases}$$

$$Mr_i' = \begin{cases} M_{Li}', & \frac{(M_{Li}+M_{Ui})}{2} \geqq \frac{(M_{Li}'+M_{Ui}')}{2} \\ M_{Ui}', & otherwise \end{cases}$$

$$Mr_k' = \begin{cases} M_{Lk}', & \frac{(M_{Lk}+M_{Uk})}{2} \geqq \frac{(M_{Lk}'+M_{Uk}')}{2} \\ M_{Uk}', & otherwise \end{cases}$$

$$M_i' = \begin{cases} M_{Li}', & \frac{(M_{Li}+M_{Ui})}{2} \geqq \frac{(M_{Li}'+M_{Ui}')}{2} \\ M_{Li}', & otherwise \end{cases}$$

$$M_k' = \begin{cases} M_{Lk}', & \frac{(M_{Lk}+M_{Uk})}{2} \geqq \frac{(M_{Lk}'+M_{Uk}')}{2} \\ M_{Lk}', & otherwise \end{cases}$$

$d_i(M,M')$: D_i에서 M와 M'사이의 거리

M_{Li}, M_{Ui} : 도메인 D_i에서 노드 M 범위의 시작과 끝

M_{Li}', M_{Ui}' : 도메인 D_i에서 노드 M' 범위의 시작과 끝

정의된 COMINDIST는 두 노드 쌍 M과 M'간에 겹치지 않는 경우만을 고려하였으나, N차원 검색공

간에서 M과 M'이 겹쳐있는 경우로 확장하여 정의될 수 있다.

두 개의 노드 쌍 간에 가장 가까운 최대근접객체를 검색하는 동안 CMINDIST와 COMINDIST의 차이가 아주 크거나 작은 경우가 발생할 수 있다. 이를 위해 두 노드를 나타내는 최소경계사각형의 모든 꼭지점들 중에서 가장 먼 점까지의 거리인 순환최대거리(Circular MAXimum DISTance value of all distances: CMAXDIST)를 사용한다.

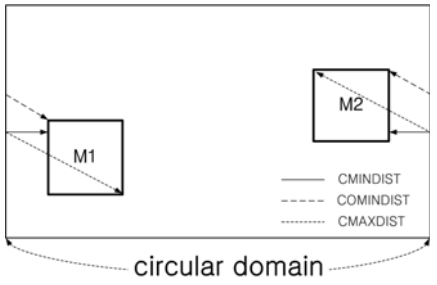


그림 3. 순환검색거리 측도의 예

【정의 3】 $D_i = \{d_1, d_2, ..., d_m\}$ (여기에서 $d_j < d_{j+1}$, $j = 1,2,\cdots,m-1$)를 N차원 검색공간을 구성하는 i번째 도메인이라 할 때, N차원 검색공간에서 두 노드 쌍 M과 M'간의 순환최대거리 $CMAXDIST$ 는 다음과 같다.

$$CMAXDIST(M, M') = \sum_{i=1}^{N} d_i^2(M, M')$$

만약 D_i가 순환도메인이면

$$d_i(M, M') = \min(|Mr_i - Mr_i'|, \quad |mr_i - d_1| + |mr_i' - d_m|)$$

그렇지 않으면

$$d_i(M, M') = |Mr_i - Mr_i'|$$

여기에서

$$Mr_i = \begin{cases} M_{Li}, & \frac{(M_{Li} + M_{Ui})}{2} \leqq \frac{(M_{Li}' + M_{Ui}')}{2} \\ M_{Ui}, & otherwise \end{cases}$$

$$Mr_i' = \begin{cases} M_{Li}', & \text{if } \frac{(M_{Li} + M_{Ui})}{2} \geqq \frac{(M_{Li}' + M_{Ui}')}{2} \\ M_{Ui}', & otherwise \end{cases}$$

$$mr_i = \begin{cases} M_{Ui}, & M_{Li} > M_{Ui}' \\ M_{Ui}, & M_{Ui} < M_{Li}' \\ 0, & otherwise \end{cases}$$

$$mr_i' = \begin{cases} M_{Li}', & M_{Ui} < M_{Li}' \\ M_{Li}', & M_{Li} > M_{Ui}' \\ 0, & otherwise \end{cases}$$

$d_i(M, M')$: D_i에서 M와 M'사이의 거리

M_{Li}, M_{Ui} : 도메인 D_i에서 노드 M 범위의 시작과 끝

M_{Li}', M_{Ui}' : 도메인 D_i에서 노드 M' 범위의 시작과 끝

정의된 CMAXDIST는 순환도메인이 포함된 N차원 검색공간에서 노드 M에 포함되어 있는 부검색공간들 중에서 노드 M'에 가장 근접하고 있는 객체 또는 부검색공간을 결정하기 위한 최대거리이다.

그림 3은 x 축이 순환적 순서범위를 갖는 도메인으로 구성된 검색공간에서 두 노드 쌍 간에 순환검색거리측도인 CMINDIST, COMINDIST, CMAXDIST를 나타낸 것이다.

3.2 K-최근접객체 쌍을 찾는 알고리즘

제안된 알고리즘에서는 두 개의 R*-트리를 검색하는 동안 방문할 필요가 없는 노드 쌍들을 방문대상에서 제외하기 위해 다음과 같은 전략들을 사용한다. 여기에서 M_P, M_Q는 두 개의 R*-트리 P, Q에서 비교대상이 되는 노드 또는 객체를 나타내는 최소경계사각형이다.

S1 : 모든 노드 쌍들에 대해 K번째로 작은 CMAXDIST(M_{Pk},M_{Qk}) 거리를 갖는 노드 쌍보다 큰 CMINDIST(M_{Pi},M_{Qj}) 거리를 갖는 노드 쌍들은 다음 검색대상에서 제외한다.

S2 : 모드 노드 쌍들에 대해 K번째로 작은 COMINDIST(M_{Pk},M_{Qk}) 거리를 갖는 노드 쌍보다 큰 CMINDIST(M_{Pi},M_{Qj}) 거리를 갖는 노드 쌍들은 다음 검색대상에서 제외한다.

S3 : 노드 쌍 (M_{Pi},M_{Qj})의 거리가 K번째로 작은 COMINDIST (M_{Pk},M_{Qk}) 거리보다 크다면 노드 쌍 (M_{Pi},M_{Qj})은 최대근접객체 쌍에서 제외한다.

S4 : 현재까지 발견된 K번째 최대근접객체 쌍 (M_{Pk},M_{Qk})의 거리보다 더 큰 CMINDIST (M_{Pi},M_{Qj}) 거리를 갖는 노드 쌍들은 다음 검색대상에서 제외한다.

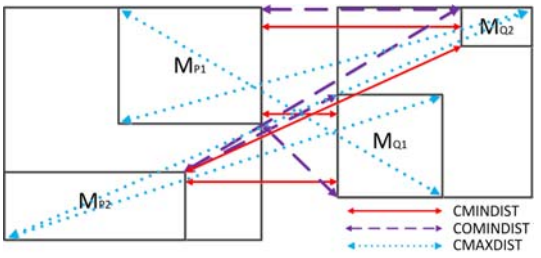


그림 4. 두 노드 쌍 간에 순환검색거리의 예

그림 4는 K=1인 경우에 대하여 두 R*-트리의 노드에 포함되어 있는 노드 쌍들 간의 순환검색거리들을 나타낸 것이다. 앞에서 제시한 가지치기 전략을 적용하기 위해 대상이 되는 모든 노드 쌍 간에 CMINDIST, COMINDIST 그리고 CMAXDIST를 계산한다. 그리고 CMINDIST의 값이 작은 순서대로 순서화한 후 방문할 필요가 없는 노드들을 제거한다. 그림에서는 노드 쌍 간에 CMINDIST의 값이 가장 작은 노드 쌍(M_{P1},M_{Q1})의 COMINDIST의 값보다 CMINDIST값이 더 큰 노드들은 방문대상에서 제거된다. 즉, CMINDIST(M_{P2},M_{Q2})>COMINDIST(M_{P1},M_{Q1}), CMINDIST(M_{P1},M_{Q2})>COMINDIST(M_{P1},M_{Q1}), CMINDIST(M_{P2},M_{Q1})>COMINDIST(M_{P1},M_{Q1})이다. 따라서 다음 방문대상이 되는 노드는 (M_{P1},M_{Q1})이 된다.

제안하는 K-최근접객체 쌍을 찾는 알고리즘에서는 서로 가장 가까운 노드 쌍을 찾기 위한 검색연산이 동일한 높이를 갖는 두 개의 R*-트리의 루트 노드부터 시작해서 하위레벨의 노드까지 재귀적인 방법으로 이루어진다(그림 5). 여기에서 두 R*-트리의 노드를 $NODE_P$, $NODE_Q$라 한다.

CP1. $NODE_P$와 $NODE_Q$의 엔트리가 중간노드이면 두 노드 쌍의 각 엔트리 간에 CMINDIST 값을 계산한 후 오름차순으로 순서화 한다. CMINDIST에 의해 순서화된 두 노드 쌍에 대해 K번째로 작은 CMINDIST와 COMINDIST, CMAXDIST의 값과 비교 연산 후 검색할 필요가 없는 노드 쌍들을 제거한다(가지치기 전략 S1,S2를 적용한다.). 방문대상이 되는 노드 쌍들에 대해서만 재귀적으로 반복 수행한다.

CP2. $NODE_P$와 $NODE_Q$의 엔트리가 리프노드이면 버켓에 있는 객체 쌍들 간에 거리(dist)를 계산한 후 K번째로 작은 dist 값 보다 큰 CMINDIST를 갖는 노드 쌍들을 제거한다(가지치기 전략 S3,S4를 적용한다.).

그림 5. 순환검색거리를 이용한 K-최근접객체 쌍을 찾는 알고리즘

4. 실험을 통한 성능 평가

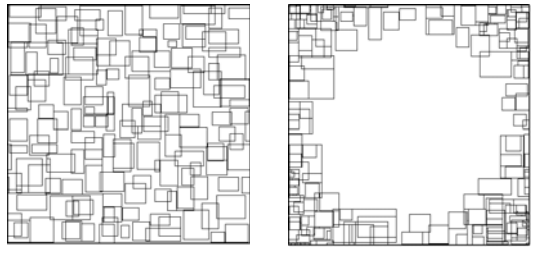
이 장에서는 순환검색거리의 성능을 K개의 최대근접객체 쌍을 찾는데 소요되는 디스크 접근 횟수와 질의 처리 시간에 따라 평가한다. 실험에서는 최대거리 우선방법(MAXDIST)과 겹침비율 우선방법(OVERLAP) 그리고 본 논문에서 제안된 순환검색거리(CDIST) 방법을 K-최근접객체 쌍을 찾는 알고리즘에 적용한 후 그 성능을 비교 평가한다.

실험에서는 이차원 검색공간에서 중복되지 않은 20,000개의 사각형 객체들을 사용하였으며, 이를 두 개의 R*-트리로 구축하였다. 사각형 객체들의 면적은 전체 검색공간의 0.1%, 0.001%인 경우를 고려하였으며, 이는 색인에 삽입되는 대부분의 객체가 어느 정도 겹치는 경우(0.1%)와 거의 겹치지 않는 경우(0.001%)에 대해서 순환검색거리의 특성을 보이기 위해서이다.

하나의 도메인의 범위가 [0,1)이라할 때, 실험에서 사용된 객체의 분포는 다음과 같다(그림 6 참조).

- 균일분포(uniform distribution) : 객체들의 중심점이 중복되지 않고 균일하게 분포
- 모서리분포(corner distribution) : 객체들의 중심점이 평균 0.5인 이차 함수 분포

균일분포는 객체의 분포가 이상적인 상태에서의 K-최근접객체 쌍을 찾는 알고리즘의 처리성능을 규명하기 위한 것이다. 그리고 모서리 분포는 객체들이 검색공간의 가장자리에 집중되어 발생하고 순환적인 성질이 많이 반영된 상태에서 순환검색거리의 특성을 규명하기 위한 것이다. 디스크에 저장된 R*-트리의 노드와 버켓은 동일한 크기를 가지며,

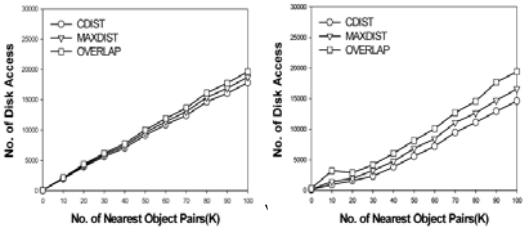


(a) 균일분포 (b) 모서리분포

그림 6. 실험에서 사용된 객체의 분포 (객체크기=0.1%)

모든 노드와 버켓의 접근시간은 동일하다고 가정하였다. 실험에서는 100개의 최대근접객체 쌍을 찾는데 필요한 디스크 접근 횟수와 질의 처리 시간에 따른 성능을 알아보았다. 성능평가에서는 객체들의 순환적 위치속성이 고려되어야 하는 분포인 모서리 분포를 중심으로 기술한다.

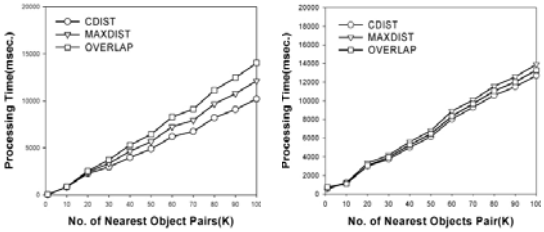
그림 7은 객체의 크기가 0.01%, 0.1%일 때, 최대근접객체 쌍의 수에 따른 디스크 접근 횟수를 나타낸 것이다. 그림에서는 K값이 증가함에 따라 디스크 접근 횟수가 선형적인 증가를 보이며, 순환검색거리를 이용한 방법이 가장 좋은 성능을 보였다(객체의 크기가 0.01%일 때 전체 평균 디스크 접근 횟수가 CDIST는 9064, MAXDIST는 9513 그리고 OVERLAP은 9962이다. 객체의 크기가 0.1%일 때는 전체 평균 디스크 접근 횟수가 CDIST는 9569, MAXDIST는 10710 그리고 OVERLAP은 11075이다.). 이는 두 색인에서 검색대상이 되는 노드 쌍의 선택 시 순환검색거리를 이용한 방법이 다른 방법보다 방문대상에서 제외되는 노드들이 더 많기 때문에 결과적으로 디스크 접근 횟수가 더 적어지게 된다. 그리고 균일분포에서는 실험대상이 되는 모든 방법들이 K값의 증가에 따라 디스크 접근 횟수가 선형적인 증가를 보였고 성능이 유사함을 알 수 있었다.



(a) 객체크기=0.01% (b) 객체크기=0.1%

그림 7. 객체크기에 따른 디스크접근횟수

그림 8은 객체의 크기가 0.001%, 0.1%일 때, 최대근접객체 쌍의 수 K를 찾는데 소요되는 누적된 처리시간을 나타낸 것이다. 그림에서는 객체들의 크기가 작은 경우 순환검색거리를 이용한 방법이 실험대상이 되는 다른 방법들과의 성능 차이가 더 커짐을 보여준다(객체의 크기가 0.01%일 때 전체 평균 처리시간이 CDIST는 5039msec, MAXDIST는 5881msec 그리고 OVERLAP은 6723msec이다. 객체의 크기가 0.1%일 때 전체 평균 처리시간이 CDIST는 6522msec, MAXDIST는 6807msec 그리고 OVERLAP은 7155msec이다.). 이러한 이유는 두 색인에서 방문대상이 되는 노드 쌍들의 선정 시 순환검색거리를 이용한 방법이 다른 방법들보다 선택되는 노드들이 더 적기 때문이다. 그리고 객체들의 순환적 위치속성이 거의 적용되지 않은 균일분포의 경우 실험대상이 되는 모든 방법들이 객체크기에 관계없이 처리시간은 비슷한 경향을 보였다.



(a) 객체크기=0.01% (b) 객체크기=0.1%

그림 8. 객체크기에 따른 처리시간

5. 결 론

대용량의 멀티미디어 데이터베이스 시스템에서 K개의 최대근접객체 쌍을 찾는 알고리즘의 처리는 많은 디스크 접근과 질의처리시간을 요구한다. 또한 데이터 차원이 증가함에 따라 검색비용이 크게 증가할 수 있다. 따라서 최대근접객체 쌍을 찾는 처리비용을 최적화하기 위해서는 색인에서 방문되는 노드수를 최소화할 수 있는 측도가 필요하다.

본 논문에서는 객체들의 순환적 위치속성을 고려한 검색거리측도인 순환최소거리, 순환최대거리 그리고 순환최적거리를 제시하였다. 그리고 두 개의 R^*-트리에서 K개의 최대근접객체 쌍을 찾는 알고리즘을 제안하고 실험을 통하여 그 성능을 평가하였다.

실험에서는 K개의 최대근접객체 쌍을 찾는데 소요되는 처리시간과 디스크접근회수에 따라 기존의 방법들과 그 성능을 비교 평가하였다. 실험 데이터로는 균일분포와 모서리분포를 이루는 사각형 객체들을 이용하였다. 실험결과에 의하면, 순환검색거리를 이용한 K-최근접객체 쌍을 찾는 질의의 처리는 객체의 분포형태, 객체의 크기에 관계없이 항상 낮은 디스크 접근 횟수를 보였다. 특히 객체들의 순환

적 위치속성이 많이 반영되는 모서리 분포인 경우 객체의 크기가 클수록 순환검색거리를 이용한 방법이 기본의 방법들 보다 디스크 접근 횟수가 더욱 적어진다. 또한 검색해야할 최대근접객체 쌍의 수가 많아질수록 질의 처리 시간이 상대적으로 더 적어짐을 알 수 있었다.

참 고 문 헌

[1] N.Beckmann, H.Kriegel, R.Schneider and B. Seeger,1990, “The R*-tree:an Efficient and Robust Access Method for Points and Rectangles”, Proc. ACM SIGMOD Int. Conf. on Management of Data, pp.322-331.

[2] T.Brinkhoff, H.P.Krigel, B.Seeger, 1993, “Efficient Processing of Spatial Joins using R-Trees”, Proc. ACM SIGMOD Conf., pp.237-246.

[3] A.Coral, Y.Manolopoulos, Y.Theodoridis, 2000, “Closest Pair Queries in Spatial Databases”, Proc. ACM SIGMOD Conf., pp.189-200.

[4] A.Coral, M.Vassilakopoulos, 2001, “The Impact of Buffering for the Closest Pairs Queries using R-trees”, Proc. 5th ADBIS Conf., pp.41-54.

[5] A.Coral, Y.Manolopoulos, Y.Theodoridis, M.Vassilakopoulos, 2004, “Algorithms for Processing K- Closest Pair Queries in Spatial Databases”, Proc. Data & Knowledge Engineering 49,pp.67-104.

[6] G.R.Hjaltason, H.Samet, 1998, “Incremental Distance Join Algorithm for Spatial Databases”, Proc. of ACM SIGMOD.

[7] Y.W.Huang, N.Jing, E.A.Rundensteiner, 1997, “Spatial Joins using R-trees: Breadth-First Traversal with Global Optimization”, Proc. 23rd VLDB Conf., pp.396-405.

[8] S.A.Nene, S.K.Nayar, 1996, “Closest Point Search in High Dimensions”, In Proc. Intl. Conf. on Computer Vision and Pattern Recognition, pp.859-865.

[9] D.Papadias, N.Mamoulis, Y.Theodoridis, 1999, “Processing and Optimization of Multiway Spatial Joins using R-trees”, Proc. 18th ACM PODS Conf., pp.44-55.

[10] H.J.Seon, et al., 2005, “The Spatial Indexing Method for Supporting a Circular Location Property of Object” In Proc. AIRS, pp. 147-159.

[11] 김영창 외 1, “분산 그리드 기법을 위한 연속 k-최근접질의처리 알고리즘”, 한국공간정보시스템학회논문지, 11권, 3호, 2009.

[12] 선휘준, 2009, “객체의 순환적 위치속성을 고려한 최대근접질의 처리 방법”, 한국공간정보학회 논문지, 11권, 4호, pp. 85-88.

[13] 선휘준, 김원호, 2010, “순환검색거리를 이용하는 최대근접질의 처리의 성능분석”, 한국컴퓨터정보학회 논문지, 15권, 1호, pp. 83-90.

논문접수 : 2011.11.25
수 정 일 : 2012.03.29
심사완료 : 2012.04.13

선 휘 준
1990년 전남대학교 대학원 이학석사
1998년 전남대학교 대학원 이학박사
1997년~2005년 서남대학교 컴퓨터정보 통신학과 교수
2006년~현재 신경대학교 애니메이션학과 교수
관심분야는 공간자료구조, 컴퓨터그래픽스, 디지털콘텐츠

김 홍 기
1984년 전남대학교 이학사
1986년 전남대학교 대학원 이학석사
1996년 전남대학교 대학원 이학박사
1991년~현재 동신대학교 정보보안학과 교수
관심분야는 공간데이터베이스, 정보보안