점진적인 스카이라인 영역 결정 기법
(A Progressive Skyline Region Decision Method)

김 진 호 † 박 영 배 ††
(Jin-Ho Kim) (Young-Bae Park)

요 약 대부분의 스카이라인 질의에 대한 연구는 정적인 데이타에 관하여 이루어지고 있다. 하지만, 모바일 응용환경의 발전에 따라 이동객체에 대한 연속적인 스카이라인 질의에 대한 필요성이 증대되고 있다. 연속적인 스카이라인 질의를 처리하기 위하여 4단계 스카이라인 영역 결정 기법이 최근 제안되었지만, 이 기법은 스카이라인 영역 계산 비용이 크므로 대량의 데이타 객체에 대해서는 사용되기 힘든 문제점이 있다.

이 논문은 이러한 문제를 해결하기 위하여 먼저 4단계 영역 결정 기법에 대해서 이론적으로 분석하고, 이를 바탕으로 4 단계 영역 결정 기법을 위한 점진적인 스카이라인 영역 결정 기법을 제안한다. 제안하는 기법은 거리 기반 가지치기와 영역 결정 선분의 범위 축소 기법을 이용하여 기존 기법의 스카이라인 영역 결정 비용을 효율적으로 감소시킨다. 본 논문은 다양한 성능 시험을 통하여 제안된 기법의 효율성을 증명한다.

키워드 : 연속적인 스카이라인 질의, 점진적 스카이라인 영역 결정기법, 이동 객체, 거리기반 가지치기, 영역결정선분 범위 축소

Abstract Most of works for skyline queries have focused on static data objects. With the advance in mobile applications, however, the need of continuous skyline queries for moving objects has been increasing. To process continuous skyline queries, the 4-phased decision method of skyline regions has been proposed recently. However, it is not feasible for a large number of data because of the high cost of computing skyline regions.

To solve this problem, this paper first provides a theoretical analysis of the 4-phased decision method. Then we propose a progressive decision method of skyline regions for the 4-phased decision method, which consists of a distance-based pruning and an extent shrinking of region decision lines. The proposed method can efficiently reduce the cost of the decision of skyline region in the 4-phased decision method. This paper also presents the experimental results to show the effectiveness of the proposed method.

Key words : continuous skyline queries, progressive decision method of skyline regions, moving objects, distance-based pruning, extent shrinking of region decision lines

1. 서 론

스카이라인 질의(skyline query)는 전체 객체 집합에서 대상 객체의 여러 속성을 다른 객체가 지배하지 않는 관심 있는 객체 집합을 검색한다[1]. 스카이라인 질의는 대상 객체의 다중 속성을 고려해야 하는 여러 응용에서 중요한 연산이다. 이동 객체의 위치를 파악하여 다양한 정보를 제공하는 위치 기반 서비스(Location Based Service)의 응용에서는 이동 객체의 위치 변경에 따른 연속적인 스카이라인 질의 처리 기법이 필요하게 된다.

연속적인 스카이라인 질의(continuous skyline queries)는 대상 객체의 다중 속성과 이동 객체의 동적 속성을 모두 고려해야 한다. 예를 들어, 이동 사용자는 "현재 위치에서 가깝고 숙박료가 싸며, 해변과의 거리가 가까운 호텔을 검색하라"는 질의를 할 수 있다. 이 경우, 질의 발생 시점에 호텔과의 3 가지 속성을 고려하여 결과를 검색해야 한다. 그러나 이전의 질의 결과는 이동 사용자가 위치를 변경하면 유효하지 않으므로 연속적인 질의가 발생하게 된다.

† 학생회원 : 명지대학교 컴퓨터공학과
solbong@mju.ac.kr
†† 종신회원 : 명지대학교 컴퓨터공학과 교수
parkyb@mju.ac.kr
논문접수 : 2006년 7월 10일
심사완료 : 2006년 11월 14일

단일 동적 속성인 대상 객체와의 거리만을 고려한 이동 객체의 질의 처리 기법[3-8]들과 다중 정적 속성들을 기반으로 한 스카이라인 질의 처리 기법[1,9-11]들은 연속적인 스카이라인 질의를 처리할 수 없다.

이동 객체에 대한 연속적인 스카이라인 질의를 처리하는 방법을 처음으로 [2]에서 제안하였다. 이 연구에서는 스카이라인 영역(Skyline Region)의 개념과 4단계 영역 결정 기법을 제안한다. 스카이라인 영역이란 대상 객체가 정적 속성에 대한 지배 객체보다 이동 객체에 가까운 영역을 의미한다. 스카이라인 영역은 위치에 기반하기 때문에 이동 객체의 속도와 방향과는 무관하며, 효율적으로 이동 객체에 대한 스카이라인 질의를 처리할 수 있다. 그러나 이 연구에서 제안한 4단계 스카이라인 영역 결정 기법은 영역 결정 시간이 객체의 개수에 비례해서 현저히 증가하기 때문에 다수의 객체를 포함하는 응용 도메인들에는 적용하기 어렵다. 이러한 문제점은 스카이라인 영역이 지배 객체 집합의 부분 집합으로 이루어지는 특성을 고려하지 않았기 때문에 발생한다.

이 논문에서는 4단계 영역 결정 기법을 이론적으로 분석하고, 영역 결정에 불필요한 객체들을 제거할 수 있는 거리 기반 가지치기 기법과 영역 결정 선분의 범위 축소 기법을 제안한다. 제안한 기법들을 R*-트리와 INN(Incremental Nearest Neighbor) 알고리즘[13]에 적용함으로써 점진적으로 스카이라인 영역을 결정할 수 있으며 영역 결정 시간을 현저하게 감소시킬 수 있다. 제안한 기법의 성능 향상을 증명하기 위해 4단계 영역 결정 기법과의 비교 실험을 수행한다.

2. 관련 연구

2.1 이동 객체에 대한 연속적인 스카이라인 질의

이동 객체에 대한 스카이라인 질의 결과는 정적 속성에 대해서는 고정이지만 동적 속성에 대해서는 질의 위치에 따라 가변적이다. 즉 질의 결과 집합은 대상 객체의 정적 속성에 대한 스카이라인 객체들과 정적 속성에 대한 지배 객체보다 이동 객체와 가까운 객체들로 구성된다.

$$Skyline_{total} = Skyline_{static} + Skyline_{dynamic} \quad (1)$$

$$Skyline_{static} = \{p_i \mid \text{정적 속성에 대해 지배하는 객체가 없는 객체}\} \quad (2)$$

$$Skyline_{dynamic} = \{p_i \mid \exists p_j \prec p_i, dist(q,p_i) \prec dist(q,p_j)\} \quad (3)$$

예를 들어 호텔의 정적 속성 관계가 그림 1(a)와 같을 경우 "현재 위치(Q_1)에서 가깝고 숙박료가 싸고, 해변과의 거리가 가까운 호텔을 검색하라."는 질의의 결과는 그림 1(b)와 같이 정적 속성(숙박료, 해변과의 거리)에 대한 스카이라인 H_1과 H_4를 질의의 위치에 관계없이 포함하고, Q_1에서 가장 가까우며 정적 속성에 대한 지배 객체들 이동 객체와 가까운 H_3을 스카이라인 결과에 포함한다.

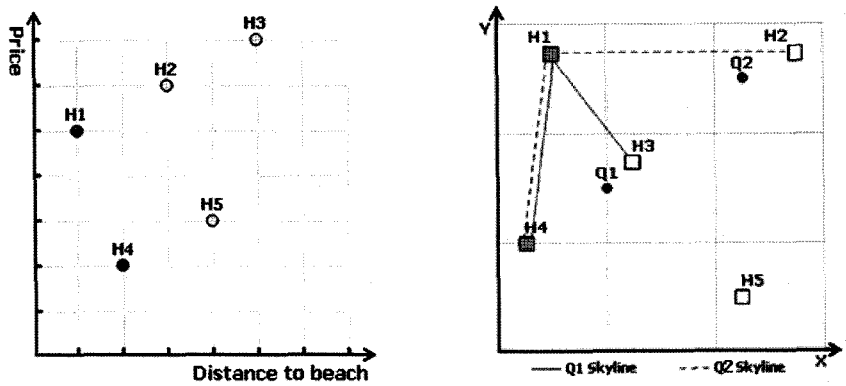
2.1.1 스카이라인 영역(SR: Skyline Region)

스카이라인 영역 SR_i이란 객체 p_i가 정적 속성에 대한 지배 객체 p_j들보다 이동 객체에 가까운 영역으로 p_i가 질의 결과에 포함될 수 있는 영역이다.

$$SR_i = \{q \mid dist(q,p_i) \langle dist(q,p_j), \forall p_j \prec p_i\} \quad (4)$$

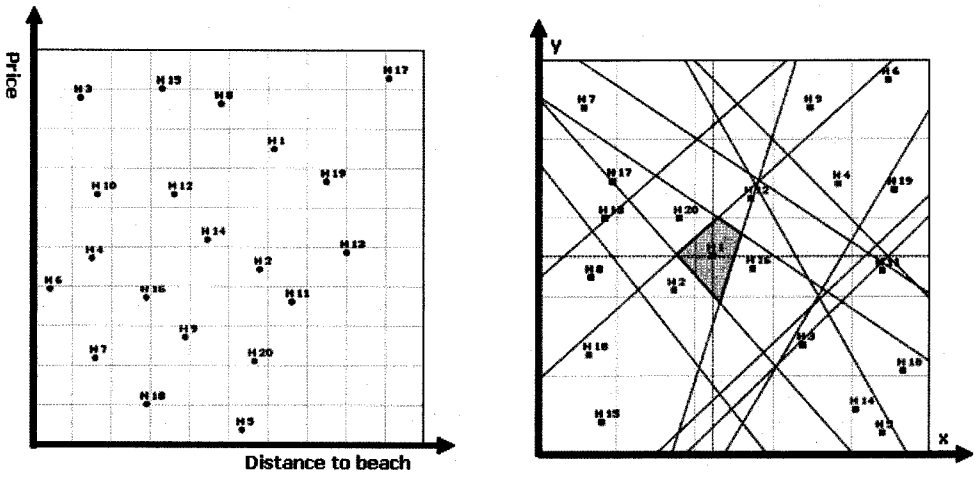
SR_i는 그림 2와 같이 객체 p_i의 위치 좌표와 정적 속성에 대한 지배 객체 p_j들의 위치 좌표가 생성하는 수직 이등분선들이 형성하는 영역으로 계산할 수 있다.

이동 객체가 위치를 변경할 때마다 대상 객체들과의 거리를 계산하여 스카이라인을 계산하는 것은 비효율적이다. 그러므로 정적 속성의 지배 관계에 따라 스카이라인 영역과 영역 간의 겹침 관계를 미리 계산하면, 이동 객체의 위치 좌표가 스카이라인 영역에 포함되는지 판



(a) 호텔의 정적 속성 관계 (b) 질의 위치

그림 1 이동 객체에 대한 스카이라인 질의



(a) 호텔의 정적 속성 관계 (b) H_1의 스카이라인 영역

그림 2 스카이라인 영역

별함으로써 질의 결과를 산출할 수 있다. 또한 스카이라인 영역을 이전 결과의 유효성 판단 기준으로 사용할 수 있으므로 연속적인 질의 처리가 가능하다.

그림 2(a)의 정적 속성 관계를 가진 객체들의 전체 스카이라인 영역들은 그림 3과 같다. Q_1과 Q_2의 질의 결과는 정적 속성에 의한 H_5, H_6, H_7, H_{18}을 공통으로 포함하며, 동적 속성에 대한 결과는 각 스카이라인 영역과 질의 위치의 관계에 따라 가변적으로 결정한다.

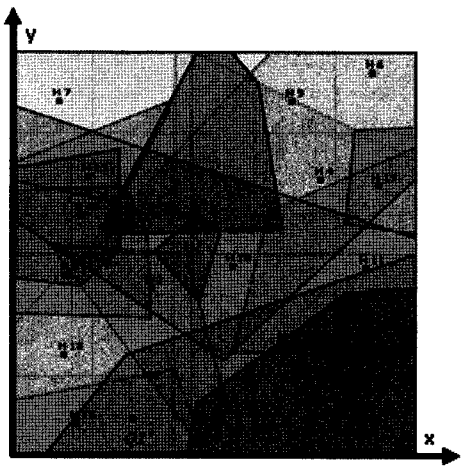


그림 3 전체 스카이라인 영역

2.1.2 스카이라인 영역 결정 기법

그림 2(b)와 같이 모든 지배 객체에 대한 수직이등분선 중 일부만이 스카이라인 영역을 형성한다. 이 경우, 불필요한 선분을 제거하여 스카이라인 영역을 형성하는 최소의 선분을 결정함으로써 질의 위치와 영역과의 포함 관계 연산에 대한 불필요한 비교 횟수를 줄이고 연속 질의에 의한 정확한 유효 영역을 계산할 수 있다. 스카이라인 영역을 형성하는 최소의 선분을 결정하기 위한 4단계는 다음과 같다.

1단계: 정적 속성에 대하여 p_i를 지배하는 객체들과의 수직이등분선 및 부등식 영역 계산

2단계: SR_i를 형성하는 수직이등분선들의 모든 교차점 계산

3단계: 연립부등식을 만족하는 교차점 계산(영역의 꼭지점)

4단계: 꼭지점 2개를 지나지 않는 선분 제거

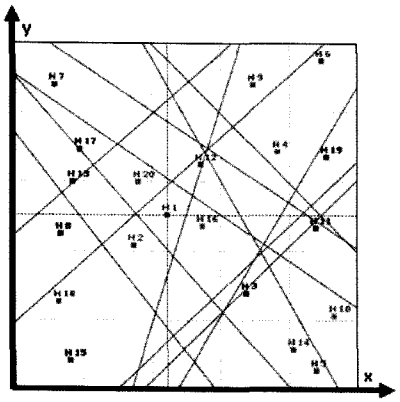
2.2 INN(Incremental Nearest Neighbor) 알고리즘

INN(Incremental Nearest Neighbor) 알고리즘은 R-트리의 최적 우선 순회(Best First Traversal) 기법으로 각 *MBR* 및 객체와의 최소거리(*MinDist*)를 기준으로 하는 우선 순위 큐(*priority queue*)를 이용하여 노드를 중복 방문하지 않고, 점진적으로 근접 객체를 검색하는 방법을 제안한다.

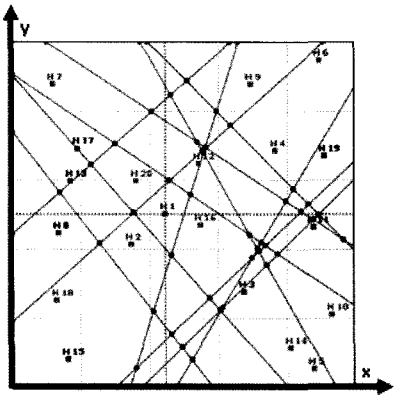
3. 점진적인 스카이라인 영역 결정 기법

3.1 4단계 스카이라인 영역 결정 기법의 이론적 분석

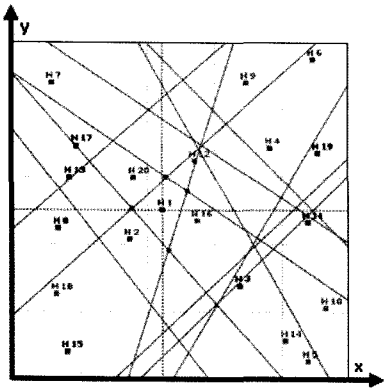
이 절에서는 4단계로 구성된 이전 연구의 스카이라인 영역 결정 기법의 이론적인 분석을 위해 영역 결정의 각 단계별 연산 시간을 결정하는 값들을 계산한다. d-차원 정적 속성에 대해 균등 분포하는 객체 집합에 대하여 정적 속성의 전체차원에 대한 평균 지배 객체의 개수는 식 (5)와 같다.



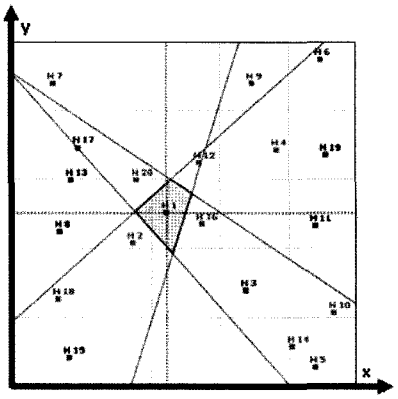
(a) 1단계: 수직이등분선 계산



(b) 2단계: 교차점 계산

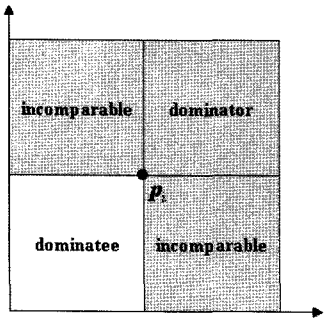


(c) 3단계: 꼭지점 계산

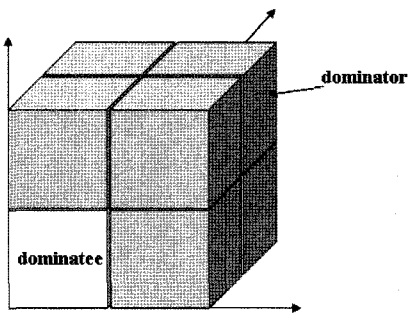


(d) 4단계: 선분 제거

그림 4 스카이라인 영역 결정 단계



(a) 2차원



(b) 3차원

그림 5 정적 속성의 차원과 객체 간의 지배 관계

$$Average(NS_i) = \frac{N}{2^d} \quad (5)$$

N은 전체 객체의 개수를 의미하고 NS_i는 객체 p_i ($i=1,2,3,\cdots,N$)의 지배 객체들의 개수를 나타낸다. 식 (5)는 영역 결정 1단계에서 소비되는 연산 시간을 결정하는 값이며, 균등 분포하는 경우, 차원이 높아질수록 전체 차원 지배 객체의 개수는 감소한다. 그러나 실제적으로 정적 속성이 다차원인 경우, 스카이라인 질의는 전체차원보다는 소수의 부분차원에 대해 발생한다. 그러므로 일반적인 응용에서는 지배 객체의 개수는 객체의 차원보다는 객체의 개수에 비례한다. 그림 5는 2차원 및 3차원 정적 속성을 가진 객체들의 지배 관계를 나타낸다.

2단계에서 지배 객체들이 형성하는 수직이등분선들의 교차점을 구하는 연산의 반복 횟수는 식 (6)과 같으며 교차점의 개수를 NCP_i라 하면 평균 교차점의 개수는 식 (7)과 같다.

$$Iteration(FindCP(NS_i)) = {}_{NS_i}C_2 + (NS_i \times 4) = NS_i \times \left(\frac{(NS_i - 1) + 8}{2}\right) \quad (6)$$

$$Average(NCP_i) = NS_i \times \left(\frac{(NS_i - 1) + 4}{2}\right) \quad (7)$$

식 (6)과 식 (7)과 같이 교차점은 지배 객체들과의 수직이등분선분들과 사각형의 공간 영역의 경계를 이루는 4 선분들에 의해 만들어진다. 교차점을 구하는 연산의 반복 횟수는 지배 객체의 개수에 제곱의 비례로 증가하고 4단계 중 가장 많은 연산 시간을 소비한다. 그림 6은 지배 객체의 개수와 반복 횟수의 관계를 나타낸다.

영역의 꼭지점을 구하는 3단계 연산의 반복 횟수는 교차점들과 수직이등분선분들의 부등식 판별에 의해 가변적이므로 반복 횟수의 평균을 계산한다. 3단계 연산의 평균 반복 횟수는 식 (8)과 같으며, 식 (9)와 같이 영역

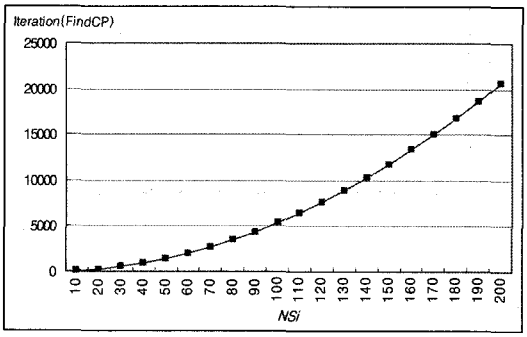


그림 6 교차점 연산 반복 횟수

의 꼭지점 2개를 지나지 않는 수직이등분선들을 제거하는 4단계 연산의 반복 횟수는 꼭지점의 개수 NV_i와 지배 객체의 개수 NS_i에 비례한다.

$$Average(Iteration(FindVertex(NCP_i))) = NCP_i \times \left(\frac{NS_i}{2}\right) \quad (8)$$

$$Iteration(RegionDecisionLine(NV_i)) = NV_i \times NS_i \quad (9)$$

이론적 분석 결과 4단계 스카이라인 영역 결정 기법에서 연산에 소비하는 시간은 지배 객체의 개수에 비례한다. 그러나 그림 4(b)와 같이 모든 지배 객체들이 스카이라인 영역을 형성하는데 영향을 미치지 않는다. 그러므로 연산에 소비하는 시간을 감소하기 위해서는 각 단계마다 모든 지배 객체를 연산의 대상으로 하는 것보다 스카이라인 영역을 형성하는데 불필요한 객체를 제거하여 연산을 최소화하는 기법이 필요하다.

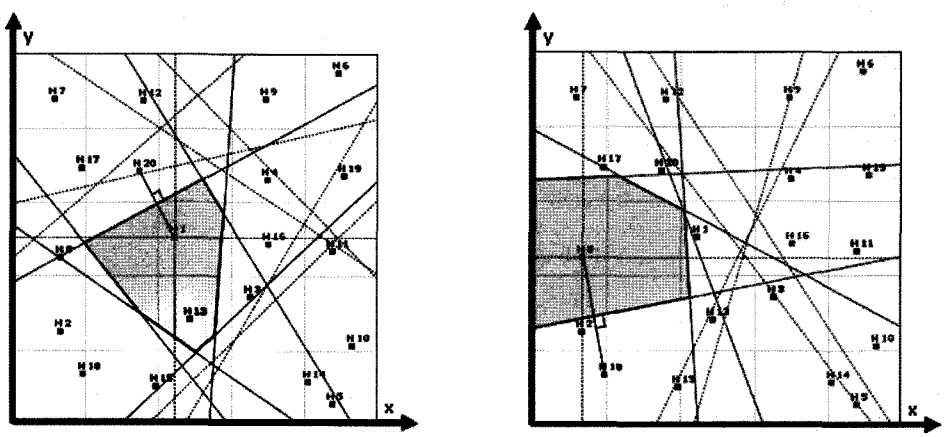
3.2 거리 기반 가지치기(Distance Based Pruning) 기법

그림 7은 스카이라인 영역과 최근접 지배 객체 간의 관계를 나타내며 스카이라인 영역 SR_i를 형성하는 영역 결정 선분 집합 S_{RDL}이라고 하면 [정리 1]과 같이 정의할 수 있다.

정리 1. 객체 p_i와 p_i의 최근접 지배 객체가 형성하는 수직이등분선은 반드시 S_{RDL}에 포함된다.

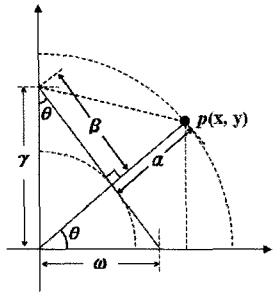
S_{RDL}의 첫 번째 영역 결정 선분을 [정리 1]에 의해 결정한 후 반복적으로 다음 근접 지배 객체의 수직이등분선 VBL_j와 현재 S_{RDL}의 선분 RDL_k들의 교차 조건을 비교하여 교차하는 경우, S_{RDL}에 VBL_j를 추가해간다. 그림 8의 수직이등분선의 특성 값 중에 x절편 값 ω나 y 절편 값 y을 이용하여 교차 여부를 판단할 수 있다.

그림 9에서 (a)와 (b)는 그림 8의 좌표 원점을 객체 p_i의 위치 좌표로 변환한 것이며 ω와 y값을 일반화한 ω



(a) H_1's NN = H_{20} (b) H_8's NN = H_{18}

그림 7 스카이라인 영역과 최근접 지배 객체



(a) 수직이등분선

$$\tan\theta = \frac{y}{x}$$

$$\alpha = \frac{\sqrt{x^2+y^2}}{2}$$

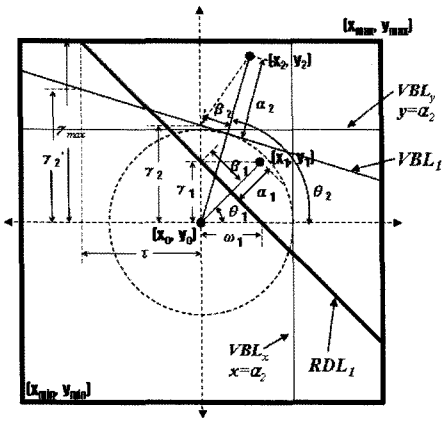
$$\beta = \frac{\alpha}{\tan\theta} = \frac{\alpha x}{y} = \frac{x\sqrt{x^2+y^2}}{2y}$$

$$\gamma = \frac{2\alpha^2}{y} \geq \alpha (\because 0 \leq y \leq 2\alpha) \Rightarrow \text{선분의 } y\text{절편}$$

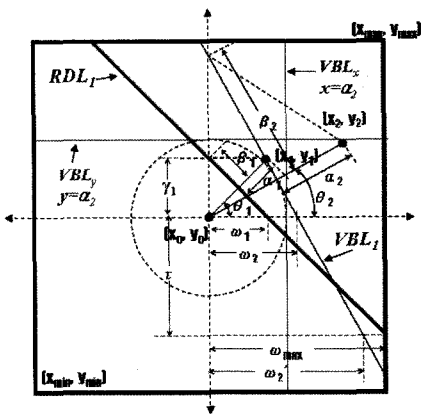
$$\omega = \frac{2\alpha^2}{x} \geq \alpha (\because 0 \leq x \leq 2\alpha) \Rightarrow \text{선분의 } x\text{절편}$$

(b) 특성 값

그림 8 수직이등분선의 특성



(a) $\theta_1 < \theta_2$

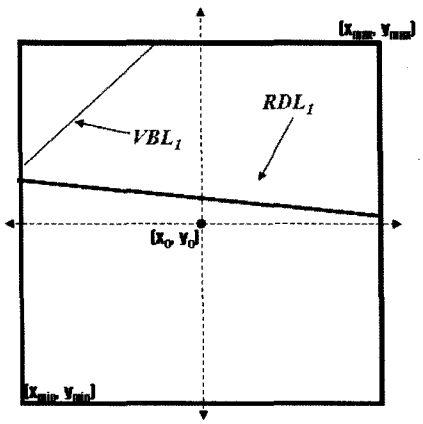


(b) $\theta_1 > \theta_2$

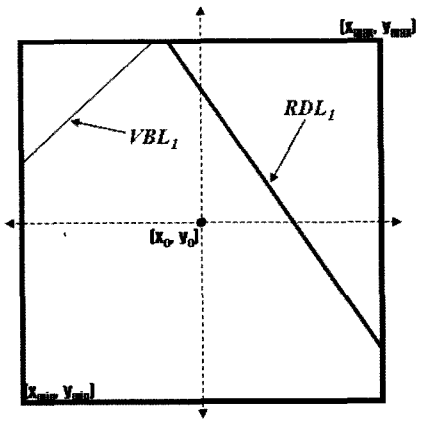
그림 9 원점 좌표 변환

$_{max}$와 γ_{max}값은 교차 여부를 판단하기 위해 사용하며 VBL_x와 VBL_y는 ω와 γ값이 최소인 수직이등분선으로 다음 근접 객체를 제거할 때 사용한다.

원점의 좌표를 변환하면 RDL과 교차하지 않지만 영역 결정 선분인 VBL의 예외적인 경우가 발생한다. 그림 10은 교차하지 않는 영역 결정 선분의 예를 보여준다. 그림에서 경우 1, 2의 VBL_1은 RDL_1에 교차하지 않지만 경우 2의 VBL_1은 영역 결정 선분이므로 S_{RDL}은

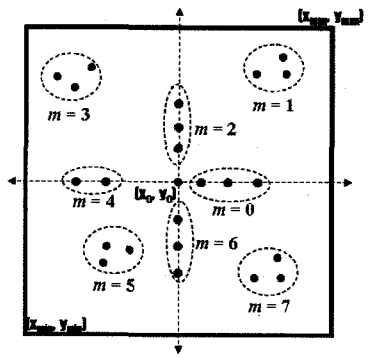


(a) 경우 1: $VBL_1 \notin S_{RDL}$



(b) 경우 2: $VBL_1 \in S_{RDL}$

그림 10 RDL과 교차하지 않는 선분



(a) 객체의 방향 벡터

$$m_p = \begin{cases} 0 & (x > x_0) \wedge (y = y_0) \\ 1 & (x > x_0) \wedge (y > y_0) \\ 2 & (x = x_0) \wedge (y > y_0) \\ 3 & (x < x_0) \wedge (y > y_0) \\ 4 & (x < x_0) \wedge (y = y_0) \\ 5 & (x < x_0) \wedge (y < y_0) \\ 6 & (x = x_0) \wedge (y < y_0) \\ 7 & (x > x_0) \wedge (y < y_0) \end{cases}$$

(b) 방향 벡터 판별식

그림 11 방향 벡터

VBL_1을 포함해야 한다. 예외적인 경우를 처리하기 위해서는 영역 결정 선분은 교차 조건뿐만 아니라 RDL의 형태에 따라 판별해야 한다.

그림 11과 같이 방향 벡터란 객체의 좌표 속성을 구별하기 위해 필요한 속성 값이다. 원점 좌표가 (x_0, y_0)이고, p의 좌표가 (x, y)일 때 방향 벡터 m은 그림 11(b)와 같이 계산할 수 있다.

RDL에 교차하지 않는 영역 결정 선분들은 방향 벡터 m과 전역 변수 SV_m 을 이용하여 판별할 수 있다. SV_m의 초기 값은 모두 *false*로 설정하고 S_{RDL}에 수직이등분선을 추가할 때 SV_m 의 값은 변경된다. 원점 좌표가 (x_0, y_0)이고, 방향 벡터 m의 $VBL(x_s, y_s, x_e, y_e)$를 S_{RDL}에 추가할 때 SV_m 은 식 (10)과 같은 값을 가진다. 교차하지 않는 영역 결정 선분은 m과 SV_m 값으로 정리 2와 같이 정의할 수 있다.

정리 2. RDL과 교차하지 않는 영역 결정 선분: S_{RDL}은 SV_m이 *false*이면 방향 벡터 m의 VBL을 RDL과 교차하지 않더라도 포함한다. 이 경우 VBL은 영역 결정 선분이며 m의 첫 번째 VBL이다.

$$m=0,\ \begin{cases} SV_1 = true & (y_s > y_0) \vee (y_e > y_0) \\ SV_0 = true & \\ SV_7 = true & (y_s < y_0) \vee (y_e < y_0) \end{cases}$$

$$m=1,\ \begin{cases} SV_7 = true & (x_e < x_{max}) \vee (x_e = x_{max} \wedge y_e = 0) \\ SV_0 = true & y_e \le y_0 \le y_s \\ SV_1 = true & \\ SV_2 = true & x_s \le x_0 \le x_e \\ SV_3 = true & (y_s < y_{max}) \vee (x_s = 0 \wedge y_s = y_{max}) \end{cases}$$

$$m=2,\ \begin{cases} SV_1 = true & (x_s > x_0) \vee (x_e > x_0) \\ SV_2 = true & \\ SV_3 = true & (y_s \le y_{max}) \vee (x_e = x_{max} \wedge y_e = y_{max}) \end{cases}$$

$$m=3,\ \begin{cases} SV_1 = true & (x_s \le x_0) \vee (x_e \le x_0) \\ SV_2 = true & x_s \le x_0 \le x_e \\ SV_3 = true & \\ SV_4 = true & y_s \le y_0 \le y_e \\ SV_5 = true & (x_s > 0) \vee (x_s = 0 \wedge y_s = 0) \end{cases}$$

$$m=4,\ \begin{cases} SV_3 = true & (y_s > y_0) \vee (y_e > y_0) \\ SV_4 = true & \\ SV_5 = true & (y_s < y_0) \vee (y_e < y_0) \end{cases}$$

$$m=5,\ \begin{cases} SV_3 = true & (x_s > 0) \vee (x_s = 0 \wedge y_s = y_{max}) \\ SV_4 = true & y_s \le x_0 \le y_e \\ SV_5 = true & \\ SV_6 = true & x_s \le x_0 \le x_e \\ SV_7 = true & (y_e > 0) \vee (x_e = x_{max} \wedge y_e = 0) \end{cases}$$

$$m=6,\ \begin{cases} SV_5 = true & (x_s < x_0) \vee (x_e < x_0) \\ SV_6 = true & \\ SV_7 = true & (x_s > x_0) \vee (x_e > x_0) \end{cases}$$

$$m=7,\ \begin{cases} SV_5 = true & (y_s > 0) \vee (x_s = 0 \wedge y_s = 0) \\ SV_6 = true & x_s \le x_0 \le x_e \\ SV_7 = true & \\ SV_0 = true & y_e \le y_0 \le y_s \\ SV_1 = true & (x_e < x_{max}) \vee (x_e = x_{max} \wedge y_e = y_{max}) \end{cases} \quad (10)$$

수직이등분선의 특성에 의해 그림 12와 같이 공간 좌표에서 원점으로부터 동일 거리 $2a$에 있는 점 p_1, p_2, p_3의 수직이등분선 VBL_1, VBL_2, VBL_3에 대한 ω는 p_2일 때 최소이고 γ는 p_3일 때 최소이므로 정리 3의 거리 기반 가지치기 조건을 만족한다.

정리 3. 거리 기반 가지치기 조건: $2a_j$ 거리에 있는 지배 객체 p_j의 수직이등분선 VBL_j이 다음 조건을 만족하면 S_{RDL}은 p_j보다 먼 거리($\ge 2a_j$)에 있는 지배 객체들의 수직이등분선을 포함하지 않는다:

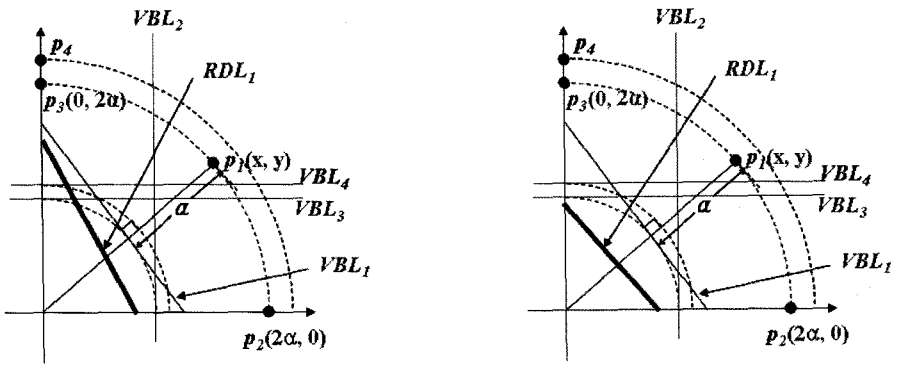
$$\alpha_j = \alpha(p_j)$$

$$\omega_{\max} = \max_{RDL_k \in S_{RDL}} (\omega(RDL_k)),\ \gamma_{\max} = \max_{RDL_k \in S_{RDL}} (\gamma(RDL_k))$$

Pruning Condition :

$$\{\omega_{\max} < \omega(VBL_j)\} \wedge \{\gamma_{\max} < \gamma(VBL_j)\} \wedge \{\omega_{\max} < \alpha_j\} \wedge \{\gamma_{\max} < \alpha_j\} \quad (11)$$

증명. S_{RDL}의 ω와 γ의 최대 값을 각각 ω_{max}와 γ_{max}라 하자. $2a_j$ 거리에 있는 객체의 VBL_j의 최소 a_j와 γ_j는 a_j 이므로 VBL_j가 S_{RDL}에 포함되기 위해서는 a_j가 ω_{max}보다 적거나 a_j가 γ_{max}보다 적어야 한다. 그러므로 ω_{max}, γ_{max}가 a_j가 적다면 $2a_j$ 보다 먼 거리에 있는 객체들의



(a) 경우 1: p_4 제거 불가능 (b) 경우 2: p_4 제거 가능

그림 12 거리 기반 가지치기 기법

VBL_j의 최소 ω와 γ값들은 a_j 보다 큰 값을 가지므로 S_{RDL}에 포함될 수 없다.

그림 12는 거리 기반 가지치기의 예를 보여준다. 경우 1, 2 모두 영역 결정 선분 RDL_1은 VBL_1과 교차하지 않고 SV_m 값도 *true*이므로 S_{RDL}은 VBL_1을 포함하지 않는다. 그러나 그림 12(a)에서는 동일 거리에 있는 p_3의 VBL_3는 교차하므로 p_4를 제거할 수 없다. 그림 12(b)에서는 VBL_2, VBL_3 모두 교차하지 않고 식 (11)의 거리 기반 가지치기 조건을 만족하여 p_4를 제거할 수 있으므로 불필요한 연산을 줄일 수 있다.

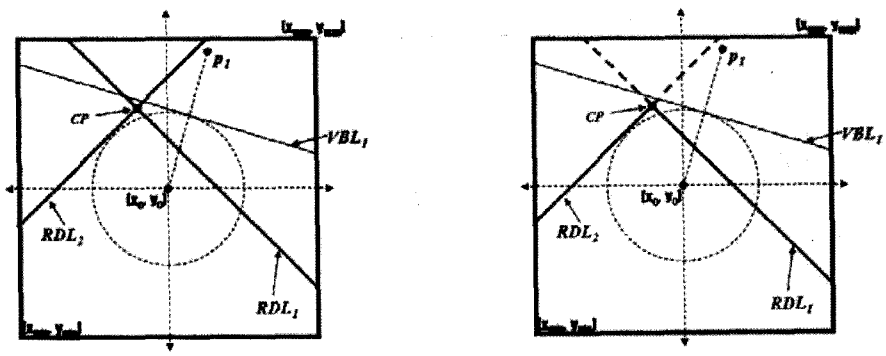
3.3 범위 축소(Extent Shrinking) 기법

영역 결정 선분 RDL_k의 범위를 전체 데이타 공간의 경계선과의 교점으로 정의하면 S_{RDL}은 그림 13(a)와 같이 스카이라인 영역을 형성하는데 불필요한 선분 VBL_1을 포함하지만 13(b)와 같이 영역 결정 선분의 범위를 다른 영역 결정 선분과의 교차점으로 축소하면 VBL_1을 포함하지 않는다. 그림 13(a)와 같이 선분의 범위를 정의하면 최종 S_{RDL}을 결정하기 위해서는 연산의 마지막에 [2]에서 제안한 4단계 영역 결정 기법과 같이 스카이라인 영역의 교차점 및 꼭지점 등을 계산하여 불필요한 선분을 제거하는 과정이 필요하지만, 그림 13(b)와 같이 범위를 축소하면 그런 과정은 불필요하게 된다.

3.4 점진적인 스카이라인 영역 결정 기법(A Progressive Skyline Region Decision Method)

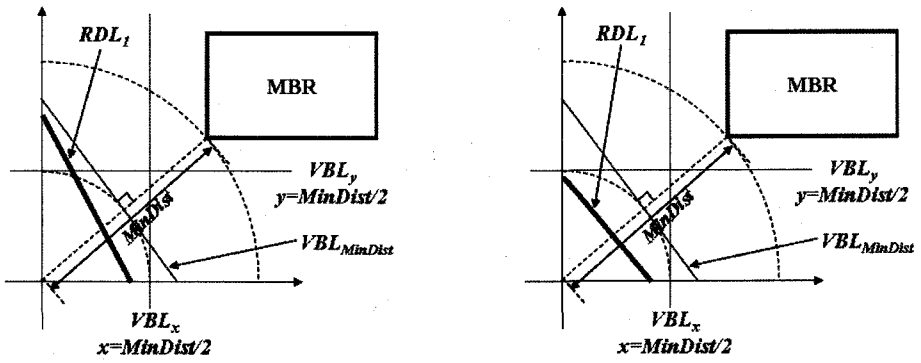
객체들의 공간 좌표를 R*-트리로 색인하고 INN알고리즘에서 제안한 거리(*MinDist*) 기반의 우선 순위 큐(priority queue)를 이용하면, 모든 지배 객체와의 거리를 계산하여 정렬하지 않아도 거리 순으로 객체를 검색할 수 있기 때문에 불필요한 객체를 점진적으로 제거할 수 있다. 그림 14는 거리 기반 가지치기 조건을 R*-트리 *MBR*에 적용한 예제이며, 그림 14(b)와 같이 *MinDist*을 이용하여 *MBR*에 속한 객체들을 제외할 수 있음을 보여준다.

INN 알고리즘에 의해 R*-트리를 *MinDist* 순으로 탐색할 때 방향 벡터 m에 대한 거리 기반 가지치기 조건을 만족하면 동일 m 값을 가진 우선 순위 큐의 엔트리를 제거하여 방향 벡터 m에 대해서는 더 이상 진행하지 않는다. 그러므로 우선 순위 큐의 각 엔트리는 방향



(a) 선분의 범위: 영역 경계선 (b) 선분의 범위: 교차점

그림 13 범위 축소 기법



(a) 경우 1: *MBR* 제거 불가능 (b) 경우 2: *MBR* 제거 가능

그림 14 *MBR*과 거리 기반 가지치기 기법

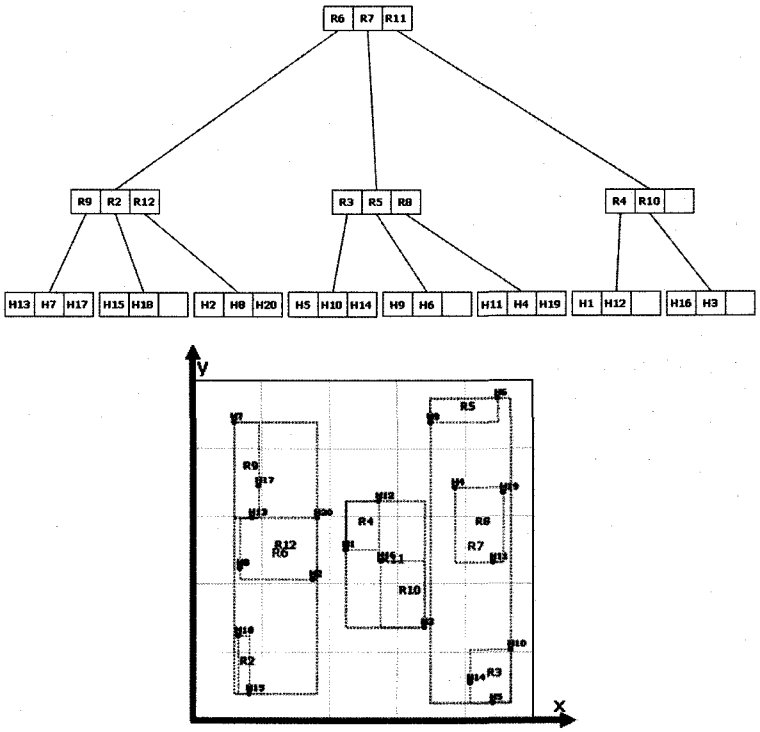


그림 15 R*-트리

벡터 m 값을 유지하여야 한다. *MBR*의 방향 벡터 m은 식 (12)와 같이 계산한다. m=-1은 *MBR*이 x나 y축에 걸쳐 있는 경우이다.

$$m_{MBR}=\begin{cases}1 & (x_l>x_0)\wedge(x_h>x_0)\wedge(y_l>y_0)\wedge(y_h>y_0)\\ 3 & (x_l<x_0)\wedge(x_h<x_0)\wedge(y_l>y_0)\wedge(y_h>y_0)\\ 5 & (x_l<x_0)\wedge(x_h<x_0)\wedge(y_l<y_0)\wedge(y_h<y_0)\\ 7 & (x_l>x_0)\wedge(x_h>x_0)\wedge(y_l<y_0)\wedge(y_h<y_0)\\ -1 & otherwise\end{cases} \quad (12)$$

R*-트리와 INN 알고리즘을 이용한 점진적인 스카이라인 영역 결정 기법을 설명하기 위해 그림 2의 정적 속성과 그림 15의 R*-트리를 예제로 사용한다. 표 1은 H_1을 기준으로 하는 각 객체와 노드의 특성 값들을 나타낸다.

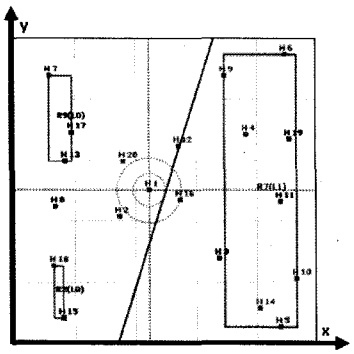
H_1의 스카이라인 영역 SR_1을 형성하는 선분 집합 S_{RDL}을 결정하기 위해 먼저 H_1의 최근접 지배 객체를 검색하고, S_{RDL}을 초기화 한다. 표 2는 H_1의 최근접 지배 객체인 H_{10}의 검색 단계별 우선 순위 큐의 내용이다. 그림 16은 최근접 객체 검색 후 S_{RDL}의 초기화 상

표 1 예제 데이타

Object	*MinDist*	*m*	*dominator*	*Node*	*MinDist*	*m*
H_2	655	5	true	R_2	1887	5
H_3	1609	7	false	R_3	2332	7
H_4	1842	1	true	R_4	0	-1
H_5	3111	7	true	R_5	2250	1
H_6	3155	1	true	R_6	425	-1
H_7	2487	3	true	R_7	1243	-1
H_8	1574	5	false	R_8	1600	-1
H_9	2250	1	true	R_9	1359	3
H_{10}	2828	7	true	R_{10}	537	7
H_{11}	2170	7	false	R_{11}	0	-1
H_{12}	873	1	true	R_{12}	425	-1
H_{13}	1455	3	false			
H_{14}	2661	7	true			
H_{15}	2531	5	false			
H_{16}	537	7	true			
H_{17}	1582	3	false			
H_{18}	2011	5	true			
H_{19}	2459	1	false			
$0H_{20}$	637	3	True			

표 2 최근접 객체 검색 단계별 큐의 내용

Action	Queue content	Description
access root	<$R11$, 0, -1> <R_6, 425, -1> <R_7, 1243, -1>	
Expand R_{11}	<R_4, 0, -1> <R_6, 425, -1> <R_{10}, 537, 7> <R_7, 1243, -1>	
Expand R_4	<R_6, 425, -1> <R_{10}, 537, 7> <H_{12}, 873, 1> <R_7, 1243, -1>	H_1: Same object
Expand R_6	<R_{12}, 425, -1> <R_{10}, 537, 7> <H_{12}, 873, 1> <R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	
Expand R_{12}	<R_{10}, 537, 7> <H_{20}, 637, 3> <H_2, 655, -1> <H_{12}, 873, 1> <R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	H_8: non-dominator
Expand R_{10}	<H_{16}, 537, 7> <H_{20}, 637, 3> <H_2, 655, -1> <H_{12}, 873, 1> <R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	H_3: non-dominator
Object H_{16}	<H_{20}, 637, 3> <H_2, 655, -1> <H_{12}, 873, 1> <R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	H_{16}: NN dominator $VBL(H_{16}) \in S_{RDL}$



(a) 노드의 MBR

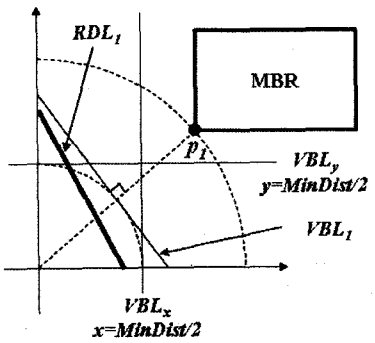
SV_m	value	SV_m	Value
SV_0	true	SV_4	false
SV_1	true	SV_5	false
SV_2	false	SV_6	true
SV_3	false	SV_7	true

(b) SV_m

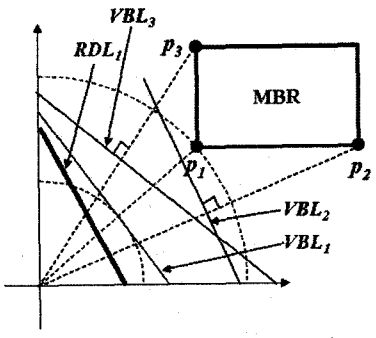
그림 16 SRDL 초기화

태와 현재 우선 순위 큐에 들어 있는 노드들의 MBR을 나타낸다.

최근접 지배 객체를 결정하고 S_{RDL} 및 SV_m을 초기화한 후 다음 근접 객체부터는 S_{RDL}의 선분들과의 교

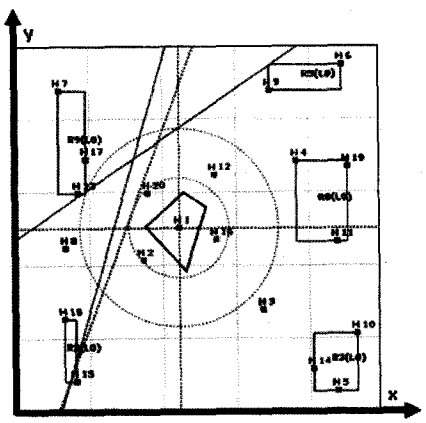


(a) 1 point 교차 비교: *MBR* 제거 불가능

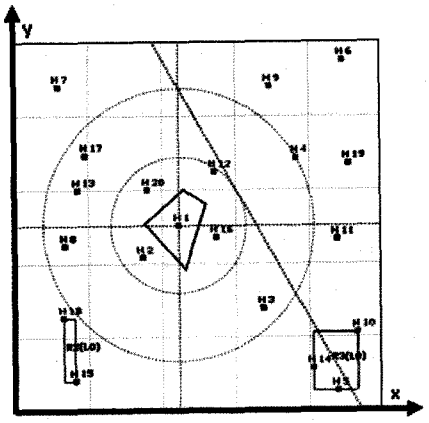


(b) 3 point 교차 비교: *MBR* 제거 가능

그림 17 *RDL*과 *MBR*의 교차



(a) *MBR*에 의한 가지치기



(b) 객체에 의한 가지치기

그림 18 예제 데이타: 거리 기반 가지치기

차 여부뿐만 아니라 SV_m의 값을 고려하여 영역 결정 선분을 결정해야 한다. 방향 벡터 m에서 정리 3과 그림 14의 조건을 만족하면 큐에서 해당 m 값을 가진 모든 엔트리를 삭제한다. *MBR*의 m=-1이면 교차 여부를 판단하지 않고 하위 노드로 진행한다.

그림 17은 *MBR*의 교차 여부를 판단하는 더욱 효율적인 방법을 예를 보여준다. 그림 17(a)에서는 RDL_1과의 교차 여부를 판단하기 위해 p_1의 VBL_1만을 비교한다. 이 경우 VBL_x가 RDL_1과 교차하기 때문에 큐에서 삭제하지 못하고 하위 노드로 계속 진행해야 한다. 그러나 그림 17(b)와 같이 p_1, p_2, p_3에서의 교차 여부를 비교하면 *MBR*을 큐에서 제거할 수 있기 때문에 더 이상 하위 노드로 진행하지 않아도 된다.

MBR 교차에 의한 제거와 거리 기반 가지치기에 의한 제거는 유사하지만 서로 다른 개념이다. 거리 기반 가치치기와는 달리 *MBR* 교차에 의한 제거는 해당 *MBR*만 삭제한다. 그림 18에서는 거리 기반 가지치기의 예를 나타낸 것이고, 표 3은 표 2의 과정에 연속해서 교차 조건 및 SV_m 값에 따른 예제 데이타의 S_{RDL}을 결정하는 과정이다.

4. 실험 및 성능 평가

제안한 기법의 성능을 평가하기 위해 4단계 영역 결정 기법과 비교하여 실험을 수행하였다. 지배 객체 개수에 따른 단일 스카이라인 영역 결정과 대상 객체 개수에 따른 전체 스카이라인 영역 결정 성능에 대한 실험을 수행하였다. 실험 환경은 하이퍼-스레딩을 지원하는 Pentium-IV 2.6GHz 프로세서와 512MB의 메인 메모리, 80GB의 하드 디스크를 가진 Windows XP 운영 체제의 PC에서 자바언어(JSDK 1.5)를 이용하여 구현하였다.

실험에 사용한 데이타는 2차원 정적 속성과 공간 좌표를 가진 객체들이며 정적 속성 및 공간 좌표를 모두 균등하게 생성하였고 각 실험에 적합한 객체를 무작위로 선택하여 성능을 측정하였다. 지배 객체 개수나 대상

표 3 SRDL 결정 과정

Action	Queue content	Description
Object H_{20}	<H_2, 655, -1> <H_{12}, 873, 1> <R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	$VBL(H_{20}) \in S_{RDL}$
Object H_2	<H_{12}, 873, 1> <R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	$VBL(H_2) \in S_{RDL}$
Object H_{12}	<R_7, 1243, -1> <R_9, 1359, 3> <R_2, 1887, 5>	$VBL(H_{12}) \in S_{RDL}$
Expand R_7	<R_9, 1359, 3> <R_8, 1600, -1> <R_2, 1887, 5> <R_5, 2250, 1> <R_3, 2332, 7>	
Expand R_9	<R_8, 1600, -1> <R_2, 1887, 5> <R_5, 2250, 1> <R_3, 2332, 7>	*pruning m=3*
Expand R_8	<H_4, 1842, 1> <R_2, 1887, 5> <R_5, 2250, 1> <R_3, 2332, 7>	H_{11}: non-dominator H_{19}: non-dominator
Object H_4	<R_2, 1887, 5> <R3, 2332, 7>	$VBL(H_4) \notin S_{RDL}$ *pruning m=1*
Expand R_2	<R_3, 2332, 7>	*pruning m=5*
Expand R_3		*pruning m=7*

객체 개수의 변화에 따른 제안 기법의 성능 평가는 실제 데이타가 아닌 모의 데이타를 통한 실험만으로도 충분하다고 판단되어 모의 데이타만을 사용한 실험을 수행하였다.

4.1 지배 객체 개수에 따른 단일 스카이라인 영역 결정 성능 비교

지배 객체의 개수를 10에서 100까지 변화하면서 단일 스카이라인 영역 결정 시간을 μs 단위로 측정하였다. 그림 19는 4단계 기법의 단일 영역 결정 시 각 단계별 결정 시간을 나타내며, 그래프의 y축은 지배 객체 개수에 따라 단계별로 현저하게 차이가 나기 때문에 *log* 눈금을 사용하여 표시하였다. 그림에서 나타내는 바와 같이 지배 객체의 개수가 증가할수록 2단계(교차점 계산) 연산 시간은 영역 결정 시간에 많은 영향을 미친다.

그림 20은 지배 객체 개수에 따른 단일 영역 결정 성능의 비교 결과를 나타낸 것이다. 4단계 기법은 지배 객체가 2배 증가했을 때 결정 시간이 약 5~14배 이상 증가하지만 제안한 기법은 단지 1~2배 정도만 증가하였고, 지배 객체가 90개 증가했을 때 4단계 기법의 결정시간이 약 1,640배 증가한 반면 제안한 기법은 약 4배 정도만 증가하였다.

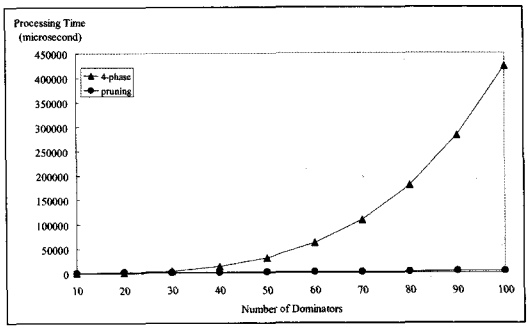


그림 20 지배 객체 개수에 따른 단일 영역 결정 성능 비교

그림 21은 지배 객체 개수에 따른 단일 영역 결정 시간의 변화를 측정하기 위해 지배 객체의 개수를 100에서 1000까지 100씩 증가하면서 실험한 결과이며 지배 객체가 900 개 증가했지만 단일 영역 결정 시간은 약 2배 정도만 증가했다. 이 실험에서는 4단계 기법은 현저하게 결정 시간이 증가하기 때문에 제외하였다.

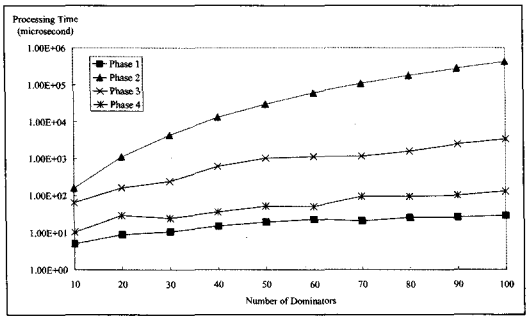


그림 19 각 단계별 연산 시간: 4단계 영역 결정 기법

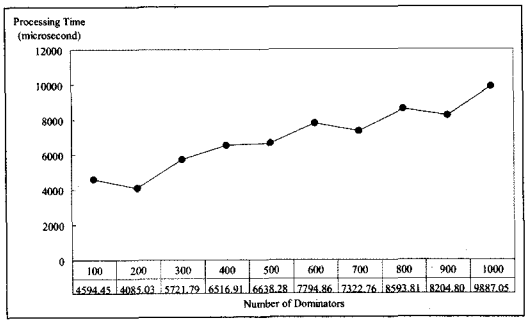


그림 21 지배 객체 개수에 따른 성능 변화

4.2 대상 객체 개수에 따른 전체 스카이라인 영역 결정 성능 비교

대상 객체의 개수를 40에서 200까지 20씩 증가하면서 전체 스카이라인 영역 결정 시간을 ms 단위로 측정하였다. 그림 22는 대상 객체 개수에 따른 전체 스카이라인 영역 결정 성능 비교 결과를 나타낸 것이다. 4단계 기법은 대상 객체가 2배 증가하면 결정 시간이 약 20~27배 증가했지만 제안한 기법은 약 3배 정도만 증가하였고, 대상 객체가 160개 증가했을 때 4단계 기법은 결정 시간이 약 1,500배 증가를 보였지만 제안한 기법은 약 14배 정도만 증가하였다.

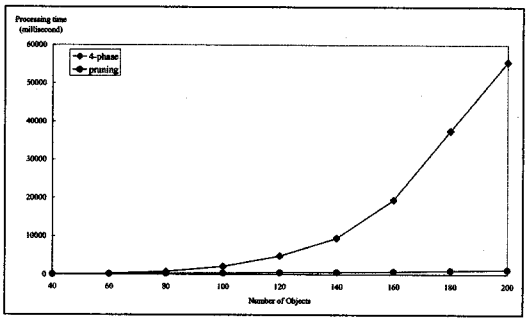


그림 22 대상 객체 개수에 따른 전체 영역 결정 성능 비교

그림 23은 대상 객체 개수에 따른 전체 영역 결정 시간의 변화를 측정하기 위해 대상 객체의 개수를 100에서 1000까지 100씩 증가하면서 실험한 결과이며 대상 객체가 900 개 증가했을 때 전체 영역 결정 시간은 약 34배 정도 증가했다. 이 실험에서도 그림 21과 같은 이유로 4단계 결정 기법은 제외하였다.

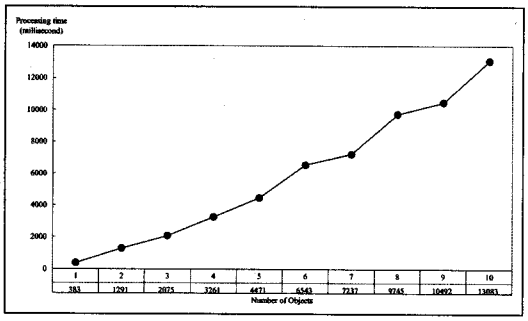


그림 23 대상 객체 개수에 따른 성능 변화

5. 결 론

4단계 스카이라인 영역 결정 기법은 스카이라인 영역이 지배 객체 집합의 부분 집합으로 이루어지는 특성을 고려하지 않았기 때문에 대상 객체의 개수에 비례하여 영역 결정 시간이 현저하게 증가하는 문제점이 있었다.

이 논문에서는 4단계 영역 결정 기법을 이론적으로 분석하였고, 영역 결정에 불필요한 객체들을 제거할 수 있는 거리 기반 가지치기 기법과 영역 결정 선분의 범위 축소 기법을 제안하였다. 제안한 기법은 영역 결정 시간을 현저하게 감소시켜 4단계 영역 결정 기법보다 훨씬 뛰어난 성능을 보임을 실험을 통해 증명하였다.

제안한 기법은 4단계 영역 결정 기법과의 성능 비교 결과, 단일 스카이라인 영역 결정 시간은 지배 객체 개수에 따라 평균 91.7%, 전체 스카이라인 영역 결정 시간은 대상 객체 개수에 따라 평균 80% 감소시켰다. 대상 객체가 200개인 경우, 제안한 기법의 전체 스카이라인 영역 결정 성능은 4단계 기법에 비해 약 43배 차이를 보였으며, 500개의 이하의 대상 객체를 가진 경우 5초 이내에 전체 스카이라인 영역을 결정할 수 있어 온라인에도 적용할 수 있음을 보였다.

참 고 문 헌

[1] Borzsonyi, S, Kossmann, D., Stocker, K. "The Skyline Operator," In ICDE, pp.421-430, 2001.
[2] 나경석, 박영배 "연속적인 스카이라인 질의를 위한 효율적인 영역 결정기법", 명지대학교 석사학위논문, 2005.
[3] Y. Theodoridis, J. R.O. Silva, and Mario A. Nascimento, On the Generation of Spatiotemporal Datasets, In Proceedings of the 6th Int'l Symposium on Large Spatial Database(SSD), 1999.
[4] Zheng B., Lee, D. "Semantic Caching in Location-Dependent Query Processing," SSTD, pp.97-116, 2001.
[5] Song, Z., Roussopoulos, N. K-Nearest Neighbor Search for Moving Query Point. SSTD, 2001.
[6] Benetis, R., Jensen, C., Karciauskas, G., Saltenis, S. Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects. IDEAS, 2002.
[7] Tao, Y., Papadias, D., Shen, Q. Continuous Nearest Neighbor Search. VLDB, 2002.
[8] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based spatial queries. In SIGMOD, pp.443-454, 2003.
[9] Tan, K., Eng, P. Ooi, B. "Efficient Progressive Skyline Computation," In VLDB, pp.301-310, 2001.
[10] D. Kossmann, F. Ramsak, S. Rost, "Shooting Stars in the Sky: an Online Algorithm for Skyline Queries," In VLDB, pp.275-286, 2002.
[11] Papadias, D., Tao, Y., Fu, G., Seeger, B. "An Optimal and Progressive Algorithm for Skyline Queries," In SIGMOD, pp.443-454, 2003.
[12] Yufei Tao, Xiaokui Xiao, Jian Pei, "SUBSKY: Efficient Computation of Skylines in Subspaces,"

In ICDE, p. 65, 2006.
[13] G. R. Hjaltson, H. Samet, "Distance Browsing in Spatial Databases," ACM Transaction on Database Systems, 24(2), pp.265-318, 1999.

김 진 호
1994년 군산대학교 전자공학과(공학사) 2000년 명지대학교 대학원 컴퓨터공학과(공학석사). 2003년 명지대학교 대학원 컴퓨터공학과(박사수료). 2003년~현재 명지전문대학 컴퓨터정보과 겸임교수. 관심분야는 Mobile DB, Spatial DB, Data Stream Management

박 영 배
1993년 서울대학교 대학원 컴퓨터공학과(공학박사). 1990년~1992년 명지대학교 전자계산소장. 1997년~2001년 명지대학교 산업대학원장. 1981년~현재 명지대학교 컴퓨터공학과 교수. 관심분야는 Mobile DB, Spatial DB, 한국어 정보처리, Large Fingerprint DB