TPR-트리에서 경계 사각형의 사장 공간을 줄이기 위한 효율적인 MBR 근사 기법

최석창[0] 송문배 강상원 황종선
고려대학교 컴퓨터학과
{choisc[0], mbsong, swkang, hwang}@disys.korea.ac.kr,

An Efficient MBR Approximation Scheme for Reducing Dead Space of Bounding Rectangles in The TPR-Tree

Seok-Chang Choi[0], Moon-Bae Song, Sang-Won Kang, Chong-Sun Hwang
Computer Science & Engineering, Korea University

요 약

이동 객체의 현재와 미래 위치 질의에 최적화된 색인 구조로써 TPR-트리가 있다. TPR-트리는 기존의 공간 색인 구조와 달리 이동 객체와 경계 사각형을 참조 위치와 속도 벡터를 매개 변수로 한 시간에 대한 선형 함수 형태로 모델링함으로써 갱신 비용을 줄이고 현재 및 가까운 미래 위치 정보의 예측을 가능하도록 한다. 하지만 TPR-트리는 시간의 경과에 따라 경계 사각형이 선형적으로 확장됨으로 인해 경계 사각형 내의 객체를 제외한 나머지 공간인 사장 공간과 경계 사각형들 간의 겹침 현상을 증가시켜 질의 성능이 떨어진다는 단점을 가진다. 본 논문에서는 질의 성능을 향상시키기 위하여 경계 사각형 내의 이동 객체들이 이동함에 따라 변경되는 최소 경계 사각형(MBR: Minimim Bounding Rectangle)을 베지어 곡선 함수를 이용하여 근사함으로써 사장 공간을 줄이는 적응 경계 사각형(ABR: Adaptive Bounding Rectangle) 기법을 제안한다.

1. 서 론

최근 GPS를 장착한 이동 단말기의 보급과 다양한 위치 기반 서비스 응용들의 등장으로 대량의 이동 객체들에 대한 위치 정보를 저장하고 관리하기 위한 시공간 데이터베이스 기술의 중요성이 증대되고 있다. 이러한 시스템에 대한 연구는 다루는 위치 정보가 과거 정보인지 현재 및 미래 정보인지에 따라 두 가지 범주로 분류될 수 있다. 본 논문은 현재 및 미래 정보 관리에 초점을 맞추고 있다.

TPR-트리[1]는 R*-트리[2]를 기반으로 하는 색인구조로써 각 이동 객체들의 위치 정보들은 참조 위치와 속도 벡터를 매개 변수로 하는 시간에 대한 선형 함수로 모델링된다. TPR-트리는 보존 경계 사각형(CBR: Conservative Bounding Rectangles)을 이용해서 객체들이 항상 경계 사각형의 내부에 있도록 보장한다. 하지만 시간이 경과함에 따라 보존 경계 사각형이 점차적으로 확장되어 사장 공간이 증가하게 된다. 그로 인하여 경계 사각형들 간의 겹침 현상이 증가하게 되고 질의 영역과 경계 사각형들이 겹치게 될 확률이 높아짐으로써 질의 처리 시 노드 접근 횟수가 증가하게 되어 질의 성능이 급격히 떨어지게 된다.

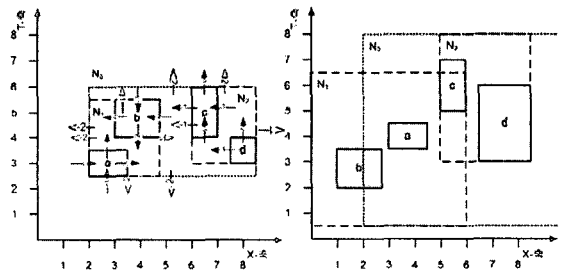
본 논문은 기존의 보존 경계 사각형이 시간에 대한 선형 함수 형태로 확장됨으로써 야기되는 질의 성능 저하를 개선하기 위한 새로운 경계 사각형 기법을 제안한다. 제안 기법은 베지어 곡선 2차 함수[3]를 이용해 최소 경계 사각형에 근사한 경계 사각형을 유지함으로써 사장 공간 및 겹침 현상을 줄인다.

2. TPR-트리

TPR-트리는 R*-트리의 확장 구조로써 속도와 방향을 시간에 대한 함수의 매개 변수로 사용하여 이동 객체의 현재 및 미래 위치의 검색 방법을 제시하였다. TPR-트리는 이동 객체의 미래 위치 질의를 지원하기 위해서 경계 사각형의 형태로 위치 데이터를 모델링한다. 경계 사각형은 노드의 모든 엔트리를 포함하는 시간 함수 기반의 사각형을 말한다. 보존 경계 사각형은 경계 사각형의 각 축을 내부 이동 객체들의 이동 방향에 따라 가장 빠른 속도로 이동하는 이동 객체의 속도 값으로 확장한다[1].

그림 1은 시각 0일 때와 1일 때의 두 개의 노드 N_1, N_2의 보존 경계 사각형의 예를 보여준다. 경계 사각형의 각 축에 대한 상한과 하한은 내부에 위치한 객체의 최대 속도로 확장되므로 항상 모든 객체들을 포함하게 된다. 하지만 시간이 지남에 따라 사장 공간들을 포함하고 경계 사각형이 겹치는 부분이 증가됨을 볼 수 있다.

이러한 문제점을 해결하기 위해서 본 논문에서는 새로운 경계 사각형 기법을 도입한다. 기존의 보존 경계 사각형이 시간에 대한 선형 함수 형태로 점차적으로 확장되는 것에 반해서 본 논문에서 제안하는 적응 경계 사각형 기법은 곡선 함수를 이용하여 경계 사각형을 축소 또는 확장함으로써 사장 공간을 줄이고자 한다.



(a) 시각 0일 때의 최소 경계 사각형 (b) 시각 1일 때 확장된 경계 사각형

그림 1 최소 경계 사각형의 확장으로 인한 사장 공간과 겹침 현상의 증가

3. 적응 경계 사각형

본 절에서는 TPR-트리에서 시간에 따라 증가되는 경계 사각형의 사장 공간과 겹침 현상을 줄여 영역 질의 성능을 향상시키기 위한 적응 경계 사각형기법을 제안한다.

3.1 이동 객체 데이터 모형

[정의 1](이동객체) 이동 객체는 $(x(t), \bar{v}, t)$, 3 튜플로 표현한다. $x(t)$는 이동 객체의 위치이고 $\bar{v}$는 이동 객체의 이동 방향과 속도 벡터 값, 그리고 t는 $x(t)$와 $\bar{v}$가 측정된 시간을 나타낸다.

[정의 2](이동 객체의 위치) 이동 객체의 참조 위치와 속도 벡터를 매개변수로 하는 시간에 대한 함수는 $x(t) = x(t_{ref}) + \overline{v_x}(t - t_{ref})$로 표현한다. 이 때 t_{ref}는 최근 갱신 시간을 의미하는 상수이다.

[정의 3](경계 구간) 노드 내의 이동 객체들을 포함하는 폐쇄 경계 구간과 속도 벡터 경계 구간은 각각 $(x^{\vdash}, x^{\dashv})$, $(\overline{v_x^{\vdash}}, \overline{v_x^{\dashv}})$로 나타낸다.

[정의 4](트리 유효 시간 구간) 최근 갱신 시간으로부터 트리가 질의에 응답 가능한 미래시간 범위는 $[t_{ref}, t_{ref}+H]$시간 구간이다. 이 때 H는 트리 매개변수이다.

3.2 적응 경계 구간

본 절에서는 적응 경계 사각형 기법을 소개하기 위해 이동 객체들은 1차원 공간에서 일정한 속도와 방향을 가지고 이동

한다고 가정한다.

$[t_{ref}, t_{ref}+H]$ 시간 구간 내에서 경계 구간 내부에 존재하는 객체들의 이동에 따른 최소 경계 구간(Minimum Bounding Interval; MBI)(그림2 참조) 영역의 상한과 하한 궤적은 (t,x) 평면에서 연속된 선분의 집합으로 나타난다. 항상 최소 경계 구간을 유지하게 되면 사장 공간이 최소가 되어 질의 비용 측면에서 효율적이다. 하지만 잦은 갱신 연산으로 인한 과부하와 저장 공간 비용이 문제가 된다. 즉, 질의 성능과 갱신 비용 및 저장 공간 비용간의 균형 문제가 존재하게 된다[1,4].

기존의 TPR-tree에서는 보존 경계 사각형을 이용하여 경계 사각형의 내부에 항상 객체들이 포함되는 것을 보장한다. 하지만 경계 사각형이 시간에 따라 선형적으로 확장되고 사장 공간과 겹침 현상이 급격히 증가하여 질의 성능이 크게 저하된다는 문제점을 가지고 있다. 따라서 $[t_{ref}, t_{ref}+H]$ 시간 구간에서 최소 경계 구간의 궤적에 근사하는 비선형 함수 표현이 필요하다. 저장 공간 비용 면에서 단일 함수로 표현되는 곡선 함수가 적합하다. 본 논문에서 제안하는 적응 경계 사각형 기법은 최소 경계 구간의 궤적 상의 정점들을 보간하는 베지어 곡선 함수[3]를 이용하여 최소 경계 구간을 곡선 궤적으로 근사한다.

베지어 곡선 공식은 조절점의 개수에 비례하여 다항식의 차수도 높아지게 된다. 차수가 높아지게 되면 노드에 저장해야할 데이터가 증가하므로 트리의 팬아웃의 증가로 트리의 높이가 높아져서 질의 성능이 떨어지게 된다[4]. 따라서 본 기법에서는 최소 차수의 다항식으로 표현되는 곡선을 함수를 구하기 위해 조절점의 개수를 3개로 제한하여 2차 다항식으로 표현되는 베지어 곡선 함수[3]를 사용한다.

베지어 곡선 공식은 매개변수 $0 \le u \le 1$에 대한 함수 형태이다. u가 0일 때 첫 번째 조절점에, u가 1일 때 마지막 조절점에 도달한다. 노드에 저장되는 함수는 시간에 대한 함수 형태가 되어야 하므로 t에 대한 함수 형태로 변환되어야 한다.

u와 t가 선형관계를 가지게 되려면, 즉, $u = t/H$가 항상 성립하려면 조절점의 t좌표 값이 각각 $t_{ref}, t_{ref}+H/2, t_{ref}+H$이어야 한다. 곡선 함수를 얻기 위해 먼저 $[t_{ref}, t_{ref}+H]$ 시간 구간의 최소 경계 구간의 상한과 하한 궤적 상의 정점들을 계산하고 t좌표 값이 t_{ref}와 $t_{ref}+H$인 정점들을 끝 점으로 하고 궤적 상에 t좌표 값이 $t_{ref}+H/2$ 인 점을 조절점으로 하는 베지어 곡선 함수를 구한다. $t_{ref}+H$시간 이후에 경계 구간은 다시 계산되어야 한다.

그림 2는 기존의 최소 경계 구간과 보존 경계 구간, 그리고 본 논문에서 제안하는 적응 경계 구간의 차이점을 보여준다. 적응 경계 구간은 최소 경계 구간의 궤적에 근사한 곡선 궤적을 가지므로 선형적으로 확장되는 보존 경계 사각형의 사장 공간을 크게 줄인다.

다음 절에서는 베지어 곡선 함수를 이용한 경계 사각형 표현에 대하여 설명한다.

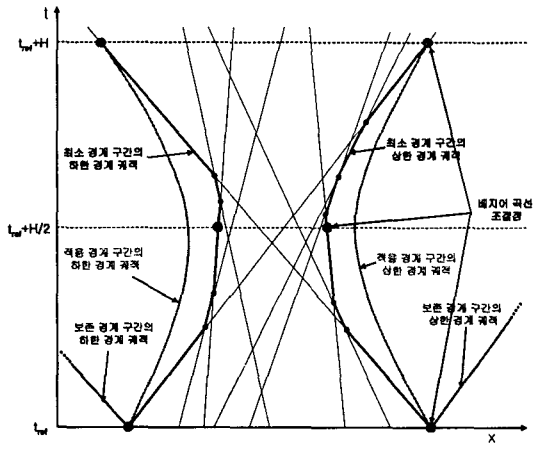


그림 2 보존 경계 사각형과 적응 경계 사각형의 비교

3.3 베지어 곡선 함수의 적용

베지어 곡선 함수는 $N+1$개의 조절점 $p_k(k=0,1,\ldots,N)$에 의해 얻어지는 곡선을 구하기 위한 연속 함수이고 $u=0$이면 첫 번째 조절점 ($k=0$)에, $u=1$이면 마지막 조절점($k=N$)에 도달한다. 베지어 곡선 공식은 다음과 같다[3].

$$B(u) = \sum_{k=0}^{N} p_k \frac{N!}{k!(N-k)!} u^k (1-u)^{N-k} \text{ for } 0 \le u \le 1 \qquad (1)$$

조절점이 3개일 경우는 공식이 다음과 같이 정리된다.

$$B_t(u) = p_{t0}(1-u)^2 + 2p_{t1}u(1-u) + p_{t2}u^2 \qquad (2)$$

$$B_x(u) = p_{x0}(1-u)^2 + 2p_{x1}u(1-u) + p_{x2}u^2 \qquad (3)$$

수식(2)와 (3)을 t에 대한 함수 형태로 다음과 같이 정리할 수 있다.

$$B_x\left(\frac{t}{H}\right) = \frac{(p_{x0}-2p_{x1}+p_{x2})}{H^2} t^2 - \frac{(2p_{x0}-2p_{x1})}{H} t + p_{x0} \qquad (4)$$

각 노드들은 수식(4)의 계수들을 저장한다.

3.4 적응 경계 사각형의 계산 알고리즘

먼저 $[t_{ref}, t_{ref}+H]$ 시간 구간에서의 최소 경계 구간의 궤적 상의 정점들을 계산한다(그림 3 findVertices). 아래에 기술한 알고리즘은 경계 구간의 하한 궤적 상의 정점을 찾는 알고리즘이다. 상한 궤적 상의 정점을 찾는 과정은 아래의 과정과 대칭적이다. 찾아진 정점들 중에서 t좌표 값이 각각 $t_{ref}, t_{ref}+H$인 정점을 추출하고 $t_{ref}+H/2$인 점을 계산한다(그림 4 SelectVertices). 마지막으로 위의 과정을 통해 찾아진 상한, 하한의 각각 3개씩의 점을 조절점으로 하는 베지어 곡선 함수를 구한다(그림 5 CalculateCurve).

각각의 알고리즘은 다음과 같다.

Algorithm FindVertices

Input. F_{object}, A set of linear functions of objects
Output. $P^{\vdash}$, A set of vertices on the polyline of the trajectory of MBI lower bound

1. $min = \arg\min_{0 \le i \le |F_{object}|-1} f_i(t_{ref})$ // $f_i(t_{ref})$: t_{ref}시각의 하한경계함수
2. $f^{\vdash} \leftarrow f_{min}$
3. add $(t_{ref}, x(t_{ref}))$ to $P^{\vdash}$ // $P^{\vdash}$: 하한 경계 궤적 상의 정점들의 집합
4. *while* $\exists f \in F_{object}, f.v < f^{\vdash}.v$
5. clear IP // IP 다른 함수들과 만나는 점들의 집합
6. begin
7. for $i \leftarrow 0$ *to* $|F_{object}|-1$ do
8. if $f_i.v < f^{\vdash}.v$ *then*
9. $t \leftarrow \frac{f_i.a - f^{\vdash}.a}{f^{\vdash}.v - f_i.v}$, $x \leftarrow f^{\vdash}.v\left(\frac{f_i.a - f^{\vdash}.a}{f^{\vdash}.v - f_i.v}\right) + f^{\vdash}.a$
10. add (t, x, i) to IP
11. end-of-if
12. end-of-for
13. $low = \arg\min_{0 \le i \le |IP|-1} ip_i.t$
14. if $ip_{low}.t > H$ *then* break
15. add $(ip_{low}.t, ip_{low}.x)$ to $P^{\vdash}$
16. $f^{\vdash} \leftarrow f_{ip_{low}.i}$
17. end-of-while
18. add $((t_{ref}+H, x(t_{ref}+H))$ *to* $P^{\vdash}$
19. SelectVertices($P^{\vdash}$)
20. return $P^{\vdash}$

그림 3 최소 경계 구간 궤적 상의 정점들을 찾는 과정

Algorithm SelectVertices
Input. $P^{\vdash}$, A set of vertices on a polyline of a trajectory of MBI lower bound
Output. $V^{\vdash}$, A set of three selected vertices from $P^{\vdash}$
1. if $|P^{\vdash}|=2$ then
2. add $\forall p \in P$ and $(t_{ref}+H/2,\ x(t_{ref}+H/2))$
3. else $|P^{\vdash}|\geq 3$ then
4. add $(t_{ref},\ x(t_{ref}))$, $(t_{ref}+H,\ x(t_{ref}+H)) \in P^{\vdash}$ to $V^{\vdash}$
5. calculate $(t_{ref}+H/2,\ x(t_{ref}+H/2))$
6. add $(t_{ref}+H/2,\ x(t_{ref}+H/2))$ to $V^{\vdash}$
7. return $V^{\vdash}$

그림 4 3개의 조절점을 추출하는 과정

Algorithm CalculateCurve
Input. A set $V^{\vdash}$ of three selected vertices from $P^{\vdash}$
Output. Coefficients of Function Polynomial for Bezier Curve
1. substitute $\forall p_k^{\vdash}(k=0,1,2)$ for

$$B^{\vdash}\left(\frac{t}{H}\right)=\frac{(p_0^{\vdash}-2p_1^{\vdash}+p_2^{\vdash})}{H^2}t^2-\frac{(2p_0^{\vdash}-2p_1^{\vdash})}{H}t+p_0^{\vdash}$$

2. return values of $\frac{(p_0^{\vdash}-2p_1^{\vdash}+p_2^{\vdash})}{H^2}$, $\frac{(2p_0^{\vdash}-2p_1^{\vdash})}{H}$, $p_0^{\vdash}$

그림 5 베지어 곡선 함수를 구하는 과정

4. 비교 및 분석

본 절에서는 4개의 1차원 객체에 대하여 보존 경계 구간을 사용하였을 때와 적응 경계 사각형을 사용하였을 때의 경계 구간 크기를 비교한다.
임의의 이동 객체들의 위치는 다음의 선형함수로 표현된다.
$o_1: x=t+3,\ o_2: x=6,\ o_3: x=8,\ o_4: x=-t+12$
$t_{ref}=0$이고 트리 매개변수 $H=10$이라고 가정한다.

● 보존 경계 구간 함수
보존경계 사각형의 상한과 하한의 선형함수는 각 방향으로 가장 빠른 속도를 가진 객체의 함수를 $t_{ref}=0$일 때 x좌표 값이 가장 작은 객체와 큰 객체의 위치로 평행 이동한 함수이다.
$f^{\vdash}: \quad x=-t+3$
$f^{\dashv}: \quad x=t+12$

● 적응 경계 구간의 함수
위의 이동 객체들의 선형함수들을 입력으로 받아 3.4절의 FindVertices 알고리즘은 최소 경계 구간 하한과 상한 궤적 상의 정점들을 계산한다. 그 결과는 다음과 같다.
$P^{\vdash}=\{(0, 3), (3, 6), (6, 6), (12, 0)\}$
$P^{\dashv}=\{(0, 12), (3, 9), (6, 9), (12, 15)\}$

위의 점들을 입력으로 받아 SelectVertices 알고리즘은 이 점들 중 x좌표 값이 t_{ref}, $t_{ref}+H/2$, $t_{ref}+H$인 점들을 추출한다. 그 결과는 다음과 같다.
$V^{\vdash}=\{(0, 3), (6, 6), (12, 0)\}$
$V^{\dashv}=\{(0, 12), (6, 9), (12, 15)\}$

CalculateCurve 알고리즘은 위 점을 조절점으로 하여 베지어 곡선 함수를 구한다. 함수는 다음과 같다.
$B_x^{\vdash}(t/10)=-0.09t^2+0.6t+3$
$B_x^{\dashv}(t/10)=0.09t^2-0.6t+12$

[0, 10] 시간 구간 동안 매 단위 시간 마다 보존 경계 사각형과 적응 경계 사각형의 크기를 계산하였다. 계산 메트릭과 결과 데이터는 표 1과 표2에 나타내었다.

표 1 계산 메트릭

u_{CBR}	보존 경계 구간의 상한 좌표값
u_{ABR}	적응 경계 구간의 상한 좌표값
l_{CBR}	보존 경계 구간의 하한 좌표값
l_{ABR}	적응 경계 구간의 하한 좌표값
Int_{CBR}	보존 경계 구간 크기($u_{CBR}-l_{CBR}$)
Int_{ABR}	적응 경계 구간 크기($u_{ABR}-l_{ABR}$)
Ratio	$Int_{ABR}/Int_{CBR} \times 100(\%)$

표 2 결과 데이터

t	0	1	2	3	4	5	6	7	8	9	10
u_{CBR}	12	13	14	15	16	17	18	19	20	21	22
u_{ABR}	12	11.49	11.16	11.01	11.04	11.25	11.64	12.21	12.96	13.89	15
l_{CBR}	3	2	1	0	-1	-2	-3	-4	-5	-6	-7
l_{ABR}	3	3.51	3.84	3.99	3.96	3.75	3.36	2.79	2.04	1.11	0
Int_{CBR}	9	11	13	15	17	19	21	23	25	27	29
Int_{ABR}	9	7.98	7.32	7.02	7.08	7.5	8.28	9.42	10.92	12.78	15
Ratio	100	72.54	56.30	46.80	41.65	39.47	39.42	40.96	43.68	47.3	51.72

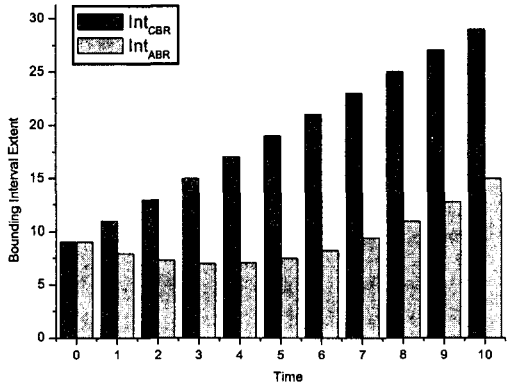


그림 6 보존 경계 구간과 적응 경계 구간의 크기 비교

보존 경계 구간은 그림 6과 같이 시간의 경과에 따라 선형적으로 증가하므로 이에 비례하여 사장 공간이 증가된다. 반면에 적응 경계 구간은 상대적으로 구간의 증가 정도가 적음을 알 수 있다. 이는 보존 경계 구간에 비해 적응 경계 구간이 사장 공간을 크게 줄여준다는 것을 의미한다.

5. 결론 및 향후 연구

TPR-트리에서 보존 경계 사각형을 이용하여 이동 객체와 경계 사각형들을 모델링했을 때 시간에 따라 사장 공간이 증가하여 경계 사각형들 간의 겹침 현상이 증가된다. 따라서 질의 성능이 점차적으로 저하된다. 본 논문에서는 이러한 문제점을 해결하기 위해 경계 사각형의 사장 공간을 줄여서 질의 성능을 향상시키기 위한 적응 경계 사각형 기법을 제안하였다.

보존 경계 사각형은 시간에 따라 선형 함수 형태로 확장되어 사장 공간이 점차적으로 증가되는 반면에 본 논문에서 제안하는 적응 경계 사각형은 2차 다항식의 베지어 곡선 함수 형태로 경계 사각형을 모델링하여 최소 경계 사각형을 근사함으로써 사장 공간을 크게 줄인다. 향후 연구 과제로는 최소 경계 사각형에 더 근사한 함수에 대한 연구가 필요하다.

6. 참고 문헌

[1] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez., "Indexing the Positions of Continuously Moving Objects". *In Proc. of SIGMOD*, 2000.
[2] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangle." *In Proc. of SIGMOD*, 1990.
[3] A.R., Forrest, "Interactive Interpolation and Approximation by Bezier Polynomials." *Computer Journal* Volume 15, Issue 11. 1972.
[4] C. M. Procopiuc, P. K. Agarwal, and S. Har-Peled., "STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects". *In Proc. of ALENEX*, 2002.