거리 관계 패턴을 기반한 k-최근접 질의 처리 기법

박용훈*, 서동민**, 복경수*, 유재수*
충북대학교 정보통신공학과*, 한국과학기술원 전산학과**
{yhpark, dmseo, ksbok}@netdb.cbnu.ac.kr, yjs@cbnu.ac.kr

A k-NN Query Processing Method Based on Distance Relation Pattern

Yong Hun Park*, Dong Min Seo**, Kyoung Soo Bok*, Jae Soo Yoo*
Dept. of Computer&Communication Engineering, Chungbuk National University*
Dept. of Computer Science, Korea Advanced Institute of Science and Technology**

요 약

최근 유클리드 공간 상에서 효율적인 연속 k-최근접(k-Nearest Neighbors) 질의 처리를 위해 그리드 구조 기반의 많은 색인 기법들이 연구되었다. 하지만 기존 기법들은 k-최근접 객체들을 연산하기 위해 불필요한 셀을 접근하여 연산 자원을 낭비하거나 근접한 셀을 알아내는데 너무 큰 연산 비용을 초래한다. 그래서 본 논문에서는 한 셀과 주변 셀과의 거리 관계 패턴을 이용하여 k-최근접 질의 처리시 적은 연산 비용과 적은 저장 공간을 사용하는 새로운 k-최근접 질의 처리 기법을 제안한다. 제안하는 기법은 k-최근접 질의 처리 시 거리 값을 기준으로 정렬된 거리 관계 패턴의 상대좌표를 순차적으로 적용하여 근접한 셀을 알아내기 때문에 O(n)의 셀 검색 비용이 요구된다. 또한 본 논문에서는 CPM[1]과 성능을 비교하여 제안하는 기법의 우수성을 입증한다.

1. 서론

휴대용 모바일 기기의 발전과 위치 인식 기술의 발전으로 인하여 위치 기반 서비스에 대한 관심이 급격하게 증가하고 있다. 이동 객체를 대상으로 하는 질의 처리 기법은 향상된 위치 기반 서비스를 제공하기 위해 기술적인 관점에서 매우 중요한 부분이다. 그 중에서도 연속 질의 처리 기법은 객체의 이동에 따라 질의 결과가 계속적으로 변하기 때문에 매우 중요한 연구 분야 중에 하나이다.

본 논문에서는 이동 객체를 대상으로 하는 연속 k-최근접 질의 처리를 위한 새로운 기법을 제안한다. k-최근접 질의는 임의 질의의 주체로부터 k개의 가장 근접한 객체들을 찾는 질의이다. k-최근접 질의는 많은 위치 기반 서비스에서 응용된다. 예를 들면, 콜택시 회사에서 택시를 원하는 고객으로부터 가장 근접한 택시를 알아내서 고객에게 제공할 수 있다. 또한 객체가 이동하는 환경에서는 시간이 지남에 따라 k-최근접 질의의 결과가 지속적으로 변하기 때문에 특정 시간 동안 결과를 모니터링 하는 연속 k-최근접 질의는 위치 기반 서비스에서 필수적이다.

[1]본 논문에서는 동일한 크기의 셀로 구성된 그리드 상에서 셀간의 거리 관계를 나타내는 거리 관계 패턴을 정의한다. 거리 관계 패턴은 O(n)의 비용으로 근접한 셀들을 접근하는데 도움을 준다. 그리고 거리 관계 패턴을 고려한 질의 색인구조와 k-최근접 질의 처리 기법을 제안한다. 제안하는 질의 처리 기법의 성능을 검증하기 위해 CPM[1] 기법과 비교를 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해서 기술한다. 3장에서는 거리 관계 패턴을 정의하고, 제안하는 질의 처리 기법과 질의 색인 구조를 기술한다. 4장에서는 성능 평가한 결과를 보여주고, 5장에서는 결론을 다룬다.

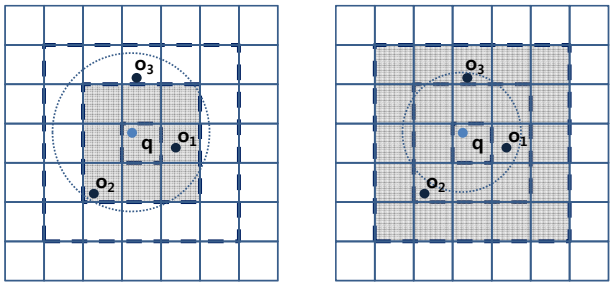
2. 관련 연구

지금까지 연구된 메인 메모리 기반의 질의 처리 기법은 크게 영역 질의 처리 기법[2][3]과 k-최근접 질의 처리 기법[1][4][5]으로 분류한다. 대표적인 k-최근접 질의 처리 기법으로는 질의 점을 둘러싸는 셀들을 한 겹씩 검사하는 YPK-CNN[4]와 질의 점을 중심으로 각 방향의 셀들을 그룹화하고 우선순위 큐를 이용한 CPM[1]이 제안되었다.

본 논문은 메인 메모리 기반의 색인 기법을 제안하기 때문에 YPK-CNN과 CPM에 대해서 자세히 설명한다. YPK-CNN은 영역 질의 기법을 k-최근접 질의 처리에 적용한 기법으로 k-최근접 객체들을 찾기 위해 셀 영역을 확장하면서 질의 처리를 수행한다. 주어진 질의 주체를 포함하는 셀을 탐색하고, 만약 그 셀에 k개의 객체가 존재하지 않으면, 그 셀을 둘러싸는 주변의 셀들을 탐색한다. 검색 영역 확장을 통한 탐색은 k-최근접 객체를 보장하는 k개의 객체를 찾을 때까지 수행된다. 그림 1은 YPK-CNN을 통해 질의 점 q로부터 2-최근접 객체를 구하는 예제로 어두운 셀은 o_1과 o_3를 찾기 위해 탐색한 셀들을 나타낸다. 그림 1의 (a)에서 o_1과 o_2가 검색되지만, 질의 점 q를 중심으로 k번째

이 논문은 2006년도 정부(과학기술부)의 재원으로 한국과학재단의 지원(No. R01-2006-000-1080900)과 2007년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원(KRF-2007-314-D00221)을 받아 연구되었음

객체 o_2까지의 거리 반경에 포함되는 셀들에 대한 탐색이 수행되지 않았기 때문에 2-최근접 객체임을 보장할 수 없다. 그래서 그림 1의 (b)와 같이 영역 확장을 수행하여 o_2보다 질의 점에 가까운 o_3을 찾아낸다. 이때 검색 영역은 질의 점 q를 중심으로 o_3까지의 거리를 반경으로 만들어진 영역을 모두 포함하기 때문에 o_1, o_3는 2-최근접 객체임을 보장한다.

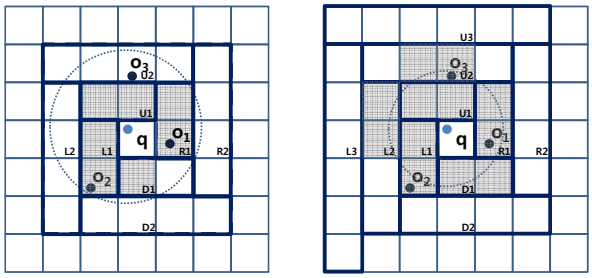


(a) 2개의 객체 탐색 (b) 2-NN을 보장하는 객체 탐색

그림 1. YPK-CNN을 이용한 2-최근접 질의 처리

CPM 기법은 가지치기(branch and bound) 알고리즘의 best-first 기법을 트리 구조가 아닌 그리드 구조에 적용한 기법이다. 이 기법은 우선 순위 힙을 이용하여 항상 질의 주체로부터 가장 근접한 셀을 순서대로 찾는 것을 보장한다. 질의에 사용되는 메모리 사용량을 줄이기 위해 그리드 영역의 모든 셀을 힙에 삽입하지 않고, 질의 주체를 중심으로 한 주변 셀들을 가상으로 그룹화 하여 삽입한다. 그룹은 트리 구조의 색인관점에서 중간 노드와 같다. 힙에는 질의 주체를 포함하는 셀과 그룹들이 질의 주체와의 거리를 기준으로 정렬된다.

그림 2는 CPM을 통해 2-최근접 질의를 처리하는 과정을 보여준다. 질의 처리시 질의 점 q를 포함하는 셀을 힙에 삽입하고, 그 주변 셀들을 포함하는 그룹 *U1, D1, R1, L1*을 힙에 삽입하고 객체 탐색을 수행한다. 객체를 탐색하기 위해 힙에서 가장 가까운 셀이나 그룹을 꺼낸다. 만약 꺼낸 것이 그룹이면 그 그룹이 포함하는 셀과 그 그룹 바깥쪽의 다음 그룹을 힙에 삽입한다. 만약 꺼낸 것이 셀이면 객체를 검색하고 만약 k개의 객체가 존재하지 않으면 위의 과정을 반복한다. 그림 2(a)는 o_1, o_2가 발견 될 때까지 검색한 셀들로 힙에서 꺼낸 셀들의 영역은 어둡게, 힙에 들어있는 셀들과 그룹들은 밝은 색으로 표현되었다. 하지만 검색된 2개의 객체는 질의 점 q로 부터 2-최근접 객체임을 보장하지 못하므로 추가적인 셀을 검색해야 한다. 힙을 이용한 질의 처리는 질의 점 q와 힙에서 꺼낸 셀 또는 그룹과의 거리가 검색된 객체 중 k번째 객체의 거리보다 클 때까지 계속된다. 힙을 이용한 질의 처리를 계속한다. 그림 2(b)는 2-최근접 객체인 o_1, o_3을 알아내기까지 힙에서 꺼낸 셀들을 어둡게 표현하였고, 힙에 아직 들어있는 그룹과 셀들은 밝은 색으로 표현하였다.



(a) 2개의 객체 탐색 (b) 2-NN을 보장하는 객체 탐색

그림 2. CPM을 이용한 2-최근접 질의 처리

YPK-CNN은 k-최근접 질의 처리시 불필요한 셀을 많이 방문하여 처리 비용이 낭비된다. 그래서 불필요한 셀 탐색을 막기 위해 CPM은 질의 점으로부터 가장 근접한 셀을 순서대로 탐색하는 기법을 제안하였다. 하지만 CPM은 연속 질의 처리를 위해 방문 리스트와 사용했던 힙을 질의 마다 유지하여 저장 공간의 낭비를 초래한다. 저장 공간의 낭비는 질의들이 k-최근접 객체들을 구하기 위해 평균적으로 탐색하는 셀의 개수에 비례하여 증가한다.

3. 제안하는 k-최근접 질의 처리 기법

본 논문에서 제안하는 색인 기법은 동일한 크기를 가지는 셀간의 거리 관계 패턴을 이용하여 질의 처리시 O(n)의 비용으로 근접한 셀들을 알아낸다. 그리고 한 점과 셀과의 거리 측정은 CPM과 동일한 방식을 이용한다. 표 1은 본 논문에서 사용하는 기호들을 나타낸다.

표 1. 기호 설명

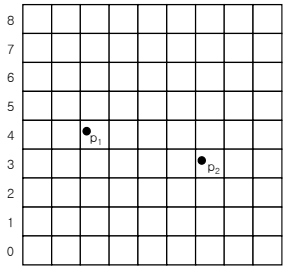
기호	설명
PC(p)	점 p가 포함되는 셀 내에서의 상대 좌표
CLA(p)	점 p로부터 셀들을 근접한 순으로 절대좌표로 정렬하여 표현한 리스트
CLR(p)	점 p로부터 셀들을 근접한 순으로 상대좌표로 정렬하여 표현한 리스트
N_{clr}	한 셀 내에서 CLR의 기준이 되는 점들의 개수
N_{sort}	한 CLR에서 유지하는 상대 좌표의 개수
k-dist	질의 결과 셋에서 k번째 객체와 질의 점과의 거리
CP(v)	영역 v의 중심점
dist()	인자로 입력된 두 점 또는 점과 셀과의 거리

3.1. 거리 관계 패턴

거리 관계 패턴은 한 셀 내의 임의의 점으로부터 다른 셀들을 거리에 대한 오름차순으로 정렬하고 그 셀들을 상대 좌표로 나타낸 셀 좌표 리스트이다. 거리 관계 패턴은 동일한 크기의 셀들이 동일한 간격으로 구성된 형태의 색인 구조에서 k-최근접 질의 처리를 위해 사용될 수 있다.

정의 1. 그리드 상의 위치가 다른 두 점 p_n, p_m 에 대해서 만약 $PC(p_n) = PC(p_m)$ 이면, $CLA(p_n) \neq CLA(p_m)$이지만 $CLR(p_n) = CLR(p_m)$이다.

정의 1은 다른 두 점이 셀 내에서의 상대 좌표가 동일 하다면 주변 셀과의 상대적인 거리 관계도 동일하다는 것을 보장한다. 그림 3은 정의 1의 예를 보여준다. 그림 3의 (a)는 다른 셀에 존재하지만 셀 안에서의 위치는 동일한 임의의 두 점 p_1과 p_2를 나타내고, 그림 3의 (b)는 각 두 점의 *CLA*와 *CLR*을 나타낸다. 두 점이 존재하는 셀이 다르기 때문에 $CLA(p_1)$과 $CLA(p_2)$는 다르다. 하지만 $CLR(p_1)$과 $CLR(p_2)$는 동일한 것을 확인할 수 있다.



(a) 임의의 두 점 위치 표현

Closest Cell list of p_1

$CLA(p_1)$	$CLR(p_1)$
C(2,4)	(0,0)
C(1,4)	(+1,0)
C(2,5)	(0,+1)
C(1,5)	(−1,+1)
C(2,3)	(0,−1)
...	...

Closest Cell list of p_2

$CLA(p_2)$	$CLR(p_2)$
C(6,3)	(0,0)
C(5,3)	(+1,0)
C(6,4)	(0,+1)
C(5,4)	(−1,+1)
C(6,2)	(0,−1)
...	...

(b) 각각 두 점에 대한 CLA와 CLR표현

그림 3. CLA와 CLR의 차이

*CLR*은 *CLA*와는 다르게 셀 내에서의 상대좌표만 동일하다면 모든 셀에서 동일하게 만들어진다. 이러한 특성을 이용하여 셀 내의 모든 점에 대해서 *CLR*을 유지한다면 별도의 연산 없이 근접한 셀들을 차례로 접근할 수 있다. 하지만 셀 내의 모든 점에 대해서 *CLR*을 유지한다는 것은 거의 불가능 하다. 공간에서 점의 개수는 무수히 많기 때문이다. 그래서 셀 내에서 대표적으로 의미 있는 점들을 추출하여 그 점들을 기준으로 *CLR*을 유지한다. 이 *CLR*들의 집합을 본 논문에서는 거리 관계 패턴으로 정의한다.

정의 2. N_{clr}은 거리 관계 패턴이 가지는 CLR개수이고 값은 n^2를 가진다. 셀 영역을 가상으로 분할하고, 분할된 작은 셀을 *VDC(Virtual Divided Cell)*라고 한다. 한 *VDC v*의 중심점은 *CP(v)*로 표현하고, *CLR(CP(v))*는 그 *v* 영역을 대표하는 *CLR*이다.

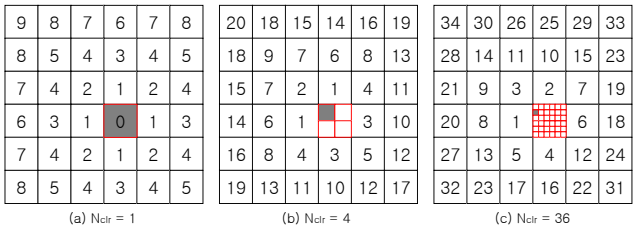
한 *CLR(CP(v))*가 나타내는 *CLR clr*은 배열구조를 가지며 *clr*[n]은 점 *CP(v)*로 부터 n번째 근접한 셀의 상대 좌표를 가진다.

정의 3. 주어진 *VDC v*에 대해서 *CLR*은 다음과 같은 속성을 갖는다.

dist(cp, clr[i]) <= dist(cp, clr[i+1])

단, cp = CP(v), clr = CLR(cp)

그림 4는 N_{clr}의 값에 따라 셀 내의 분할된 영역과 또한 만들어진 *CLR*의 정확성을 보여준다. 그림 4(a)는 N_{clr}이 1이기 때문에 셀의 중심점이 그 *CLR*의 기준점이 되고 그 영역을 대표한다. 그림 4(b)와 4(c)는 N_{clr}이 각각 4와 36이기 때문에 영역을 4개와 36개로 분할한다. 그리고 분할된 영역의 중심점이 각 *CLR*의 기준점이 된다. 분할된 영역별로 *CLR*을 만들고 그 영역에 포함되는 점을 기준으로 근접한 셀들을 찾을 때는 그 *CLR*을 따른다.



(a) N_{clr} = 1 (b) N_{clr} = 4 (c) N_{clr} = 36

그림 4. N_{clr}에 따른 거리 관계 패턴

거리 관계 패턴의 최대 오차는 N_{clr}에 의해 분할된 한 영역 *VDC v*내의 모든 점들과 중심점인 *CP(v)*로부터 다른 셀들까지의 거리 측정 시 발생할 수 있는 최대 거리 차이이다.

정의 4. 그리드 내의 셀들의 집합을 $C=\{c_1, c_2, \ldots, c_n\}$로 나타내고, 주어진 한 영역 *VDC v* 내의 점들의 집합을 $P_v=\{p_1, p_2, \cdots, p_n\}$로 나타낸다. 최대 오차 *E*는 아래의 수식과 같다

$$E(C, v, P_v) = \max\{|\, dist(CP(v), c_i) - dist(p_i, c_i)\,|\}$$
$$단, c_i \in C, p_i \in P_v$$

정의 5. 2차원 평면에서 세 개의 점 p_1, p_2 그리고 p_3가 있을 때, $|dist(p_1, p_3) - dist(p_2, p_3)| <= dist(p_1, p_2)$은 보장된다. 만약 사각형 B가 있을 때, $|dist(p_1, B) - dist(p_2, B)| <= dist(p_1, p_2)$이다.

정의 5는 한 *VDC v* 에 존재하는 두 점 p_i와 p_j가 있을 때, $E <= max(dist(p_i, p_j))$를 보장한다. 그래서 거리 관계 패턴의 최대 오차는 *v*에 포함된 점들 중 *CP(v)*로부터 가장 멀리 떨어진 점까지의 거리이다. α가 한 셀의 측면 길이라고 할 때, 최대 오차 *E*는 아래의 수식과 같다.

$$E = \alpha\sqrt{2} / (2 \times \sqrt{n_{clr}})$$

그림 5는 N_{clr}이 4일 때, 거리 관계 패턴의 최대 오차 E를 보여준다.

정의 6. 주어진 *VDC v* 내의 점들의 집합을 $P_v=\{p_1, p_2, \cdots, p_n\}$로 나타낸다. *cp = CP(v), clr = CLR(cp)*일 때,

만약 $dist(p_i, clr[k]) + dist(p_i, cp) < dist(cp, clr[k])$ 이면, $dist(p_i, c_j) < dist(cp, clr[k + \beta])$ 이다. 단, $\beta \geq 0$ 이다.

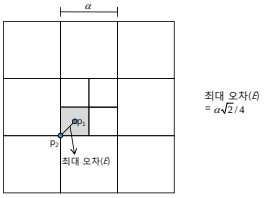


그림 5. 최대 오차 *E* 계산

거리 관계 패턴을 기반으로 영역 *VDC v*에 존재하는 임의 점 p_i로부터 근접한 셀들을 순차적으로 접근할 때, p_i와 *CP(v)*의 거리 차이로 인해 정확성을 보장하지 못할 수도 있다. 거리 오차는 $dist(p_i, CP(v))$보다 작으므로 원하는 검색 반경을 D_{want}라고 할 때, 실제 검색 반경 D_{search}는 아래의 수식과 같다

$$D_{search} = D_{want} + dist(p_i, CP(v))$$

거리 관계 패턴은 두 개의 인자를 이용하여 생성된다. 하나는 정렬 할 셀의 개수 N_{sort}이고, 나머지 하나는 CLR의 기준이 되는 점들의 개수 N_{clr}이다. N_{sort}와 N_{clr}은 거리 관계 패턴이 차지하는 저장 공간에 영향을 미친다. 또한 N_{clr}은 거리 관계 패턴의 정확성에 영향을 미친다.

정의 7. 거리 관계 패턴의 저장 공간(S_{pat})은 N_{sort}와 N_{clr}에 비례한다. 저장 비용은 아래의 수식을 따른다.

$$S_{clr} = S_{crd} \times N_{sort}$$
$$S_{pat} = S_{clr} \times N_{clr}$$

S_{crd} : 좌표를 표현하기 위한 저장 공간 크기
S_{clr} : 한 *CLR*이 차지하는 저장 공간 크기

거리 관계 패턴을 이용하면 O(n)의 비용으로 근접한 셀들을 순서대로 접근할 수 있다. 그리고 거리 관계 패턴의 오차를 줄이기 위해 N_{clr}을 크게 지정할 수 있지만 N_{clr}의 증가는 저장 공간의 소모를 초래하기 때문에 적절한 값을 결정해야 한다. N_{clr}에 따른 저장 공간의 소모와 질의 성능 평가는 4장에서 보여준다.

3.2. k-최근접 질의 처리 기법

본 논문에서 제안하는 패턴 기반의 k-최근접 질의 처리 기법은 3.1장에서 정의한 거리 관계 패턴을 이용한다. 거리 관계 패턴은 *CLR*의 집합으로서 해시 함수를 통해 *CLR*을 할당한다. 할당 받은 *CLR*을 이용하여 k-최근접 질의 처리를 수행한다.

질의 처리는 3가지 단계로 수행된다. 첫 번째로 N_{sort}와 N_{clr}을 인자로 하여 거리 관계 패턴을 생성한다. 거리 관계 패턴은 그리드를 구성하는 셀의 모양과 배치를 기반으로 해서 만들어지기 때문에 한번 생성하면 영구적으로 사용 가능하다. 또한 거리 관계 패턴은 *CLR*들을 관리하고 좌표를 해시 키로 사용한다.

두 번째로 질의를 생성하고 *CLR*을 할당 받는다. k-최근접 질의는 질의 점, 검색할 객체의 개수, 결과 객체, 할당 받은 *CLR* 정보를 갖는다. 거리 관계 패턴으로부터 질의 점을 키로 적절한 *CLR*을 할당 받는다.

세 번째로 할당 받은 *CLR*을 이용하여 k-최근접 질의를 처리한다. *CLR*의 상대 좌표를 순서대로 참고하여 k개의 객체가 검색될 때까지 근접한 셀을 순서대로 탐색한다. 우선 질의 점이 존재하는 셀을 탐색하고, 발견된 객체가 k개 미만이면, *CLR*의 첫 번째 상대 좌표로 다음 셀을 알아내고 탐색한다. 발견된 객체가 k개 미만이면, 다시 *CLR*의 두 번째 상대 좌표로 다음 셀을 알아내고 탐색한다. 이러한 방식으로 k객체를 찾을 때까지 셀을 탐색한다. 이때 질의 점에서 k번째 객체까지의 거리를 *k-dist*라고 한다. 검색된 k개의 객체가 k-객체임을 보장하는지 확인하기 위해 추가적으로 *k-dist*보다 질의 점에 가까운 모든 셀들의 탐색을 순서대로 진행한다. 만약 추가적으로 발견된 객체가 k-dist보다 질의 점에 더 가까우면, 결과 셋과 *k-dist*를 갱신한다. 남은 셀들이 *k-dist*보다 질의 점에 가까운 것이 없으면 처리를 중단한다.

그림 5는 *CLR*을 이용하여 3-최근접 질의를 처리하는 과정을 보여준다. 그림 5(a)는 3개의 객체를 찾기 위해서 *CLR*을 순서대로 적용하여 셀을 탐색하는 것을 보여준다. 그림 5(b)는 3-최근접 객체임을 보장하는 3개의 객체를 찾는 과정을 보여준다. 어두운 부분은 질의에 의해서 탐색된 셀들을 나타낸다.

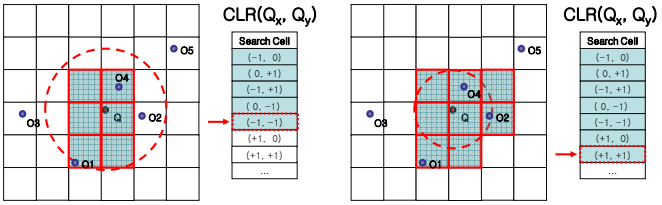


그림 6. 거리 관계 패턴을 이용한 3-최근접 질의 처리

하지만 *CLR*에는 오차가 있기 때문에 추가적인 셀 탐색이 필요하다. *CLR*상의 다음 순서에 탐색할 셀들이 이전 순서에 탐색한 셀들보다 질의 점에 근접할 수 있다. 이것은 *CLR*이 담당하는 영역에 존재하는 모든 점에 대해서 근접한 셀의 순서를 만족 시키지 않기 때문이다. 오차를 극복하기 위해 *search-dist*를 이용하여 셀을 탐색한다. *EQ(q)*는 질의 *q*에 대해서 *k-dist*의 오차 값을 나타낼 때, *search-dist*는 아래의 수식과 같다.

$$search\text{-}dist = k\text{-}dist + EQ(q)$$

그림 7은 *E*와 *EQ(q)*의 차이 그리고 *search-dist*와 *k-dist*의 차이를 보여준다. k-최근접 객체를 보장하기 위해 *CLR*에서 제공하는 순서로 셀 탐색 시, *search-*

*dist*와 탐색하는 셀과의 거리를 비교하면 거리 관계 패턴의 오차가 고려된다. 하지만 실제 셀을 탐색할 때는 질의 점으로부터의 거리가 *k-dist*보다 작은 셀만을 탐색한다.

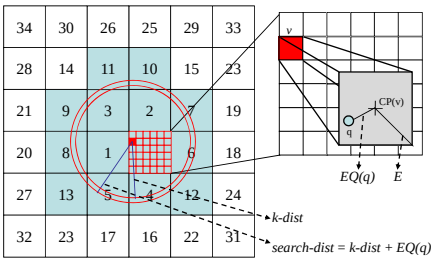


그림 7. search-dist를 고려한 검색 영역

그림 8은 k-최근접 질의를 연산하는 알고리즘이다. 질의가 내려지면 질의 점을 포함하는 *VDC*영역의 *CLR*을 할당하고(줄03), 질의 점을 포함하는 셀의 좌표를 구한다(줄04). α는 한 셀 측면 길이 이다. 그 셀의 좌표에 *CLR*상의 위치를 나타내는 p_{clr}에 따라 *CLR*을 순서대로 적용하여 k개의 객체를 찾아 *result_set*에 삽입한다. *CLR*을 순서대로 적용하여 k개의 객체를 찾아 질의 결과 셋에 삽입한다(줄06~12). 그리고 *k-dist*와 *search-dist*를 계산한다(줄13). k개의 객체가 k-최근접 객체임을 보장하는지 확인하기 위해 *CLR*의 순서에 따라 질의의 *search-dist*보다 *CLR*의 기준점으로부터의 거리가 작은 셀들을 대상으로 *k-dist*보다 질의 점에 근접한 셀이 있으면 탐색한다(줄14~16). 만약 *k-dist*보다 근접한 객체가 발견되면 *result_set*, *k-dist* 그리고 *search-dist*를 모두 갱신한다. *CLR*의 순서에 따라 셀을 검색 하는 동안, CLR의 중심점으로부터 셀과의 거리가 *search-dist*보다 크게 되면 질의 처리를 중단하고 *result_set*을 출력한다.

3.3. 연속 k-최근접 질의 처리기법

거리 관계 패턴을 이용한 k-최근접 질의 처리 기법을 점진적인(incremental) 연속 질의에 적용하기 위해 객체 색인과 더불어 질의 색인이 필요하다. 그림 9은 그리드 색인에서 유지하는 자료구조를 나타낸다. 각 셀마다 *oid_list*와 *qid_list*를 유지한다. *oid_list*는 그 셀의 영역에 있는 객체들의 식별자 *oid*를 유지하고, *qid_list*는 그 셀의 영역과 질의 점으로부터 *k-dist*반경의 원에 겹치는 질의들의 식별자 qid를 유지한다. query table에는 질의 식별자 *qid*, 검색하고자 하는 객체 개수 k, k번째 객체까지의 거리 *k-dist* 그리고 질의 결과 *result_set*를 유지한다.

객체는 자신의 위치를 주기적으로 서버로 보고하고, 서버는 그 시간 간격으로 질의의 결과를 갱신한다. 질의마다 *k-dist* 안으로 들어온 객체의 리스트와 개수 그리고 밖으로 나간 객체의 수를 유지한다. 만약 들어온 객체가 많으면 들어온 객체의 리스트와 *result_set*을 비교하여 *result_set*과 *k-dist*를 갱신한다. 만약 나간 객체들이 많거나 질의가 이동하면, k-최근접 질의 처리를 처음부터 수행한다.

```
kNN Computation (G, H, q, k)
// Input = G : 색인, H: 거리관계패턴, q: 질의, k:필요 객체 개수
// Output = result_set : k-최근접 결과 집합
01  k-dist = 0; result_set = NULL; clr = NULL; p_clr = 0;
02  search_dist = ∞;result_count = 0;
03  clr = HCLR(q.x, q.y);
04  x = q.x / α;   y = q.y / α;
05  C_cur = G(x, y);
06  while (k > result_count)
07    C_cur = G(x + clr [p_clr].x,   y + clr [p_clr].y);
08    For each object o ∈ OL(C_cur)
09      Insert o into result_set;
10      result_count++;
11    p_clr++;
12  end;
13  Update k-dist and search-dist;
14  while (search_dist > clr[p_clr].dist)
15    if (k-dist < dist(q, C_cur))
16      p_clr++; continue;
17    C_cur = G(x + clr [p_clr].x,   y + clr[p_clr].y);
18    For each object o ∈ OL(C_cur)
19      If (k-dist > dist(o, q))
20        Remove o^kth from result_set; Insert o into result_set;
21        Update k-dist and search-dist;
22    p_clr++;
23  end;
24  return result_set;
```

그림 8. k-최근접 질의 처리 알고리즘

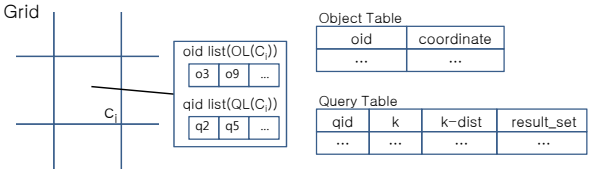


그림 9. 연속 k-최근접 질의 처리를 위한 자료구조

4. 성능 평가

본 논문에서 제안하는 거리 관계 패턴을 기반한 k-최근접 질의 처리 기법의 특성을 보이기 위해 CPM 기법과의 성능 비교를 수행한다. 성능 평가는 객체의 개수, 질의의 개수, 객체의 이동, 질의의 이동에 따른 질의 처리 속도를 평가하고 또한 저장 공간의 효율성에 대해 평가한다. 연속 질의 평가를 위해 객체의 속도, 이동 비율, 질의의 속도 그리고 이동 비율에 대해서 평가를 수행했지만 지면의 제한으로 인하여 기술을

생략하였다. 구현 및 수행은 IBM xSeries 336, CPU 3.2GHz, RAM 2Gbyte에서 이루어졌다. 제안하는 기법은 PB-kNN으로 표기한다. 표 2는 성능 평가에서 사용하는 인자들을 나타낸다.

표 2. 성능 평가 인자

구분	기본값	설정 범위
N_{clr}	4^2	$1^2 \sim 10^2$
그리드 크기	512^2	$64^2 \sim 1024^2$
객체 개수	300K	100K ~ 1000K
질의 개수	5K	1K ~ 10K
k 값	32	1 ~ 256

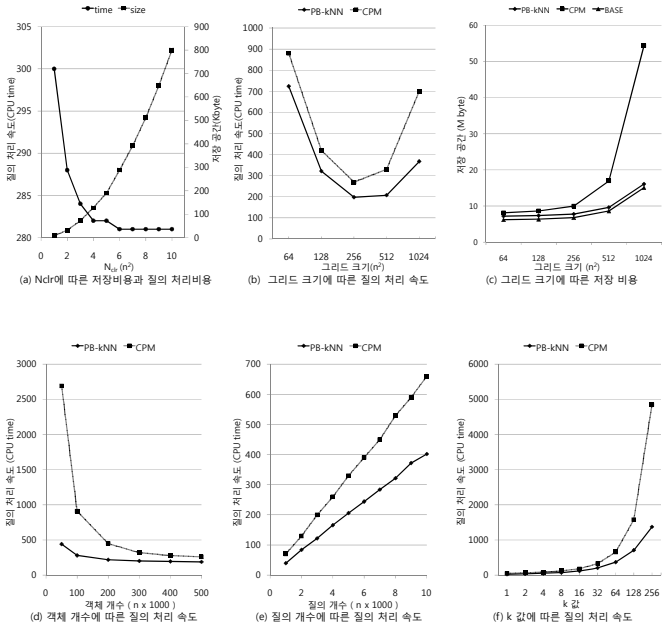


그림 10. 성능 비교

k-최근접 질의 처리시 거리 관계 패턴이 필요로 하는 저장 공간의 크기와 질의 처리 성능의 관계를 조사하였다. 이것은 거리 관계 패턴이 N_{clr}의 값에 따라 최대 오차가 E가 결정 되기 때문이다. 그림 10(a)는 N_{clr}의 증가에 따른 질의 처리 비용과 거리 관계 패턴이 차지하는 저장 공간의 관계를 보여준다. N_{clr}의 값이 4^2부터 질의 처리 비용은 거의 변화가 없지만 거리 관계 패턴의 크기는 여전히 증가하는 것을 볼 수 있다. (b)와 (c)는 그리드 크기에 따른 질의 처리 속도와 저장 저장 비용을 CPM 기법과 비교하여 나타낸다. BASE는 질의 테이블과 거리 관계 패턴을 제외한 기본 저장 비용이다. CPM의 질의 테이블 저장 비용에 비해 PB-kNN의 질의 테이블과 거리 관계 패턴의 저장공간은 매우 작다. (d)는 객체 개수, (e)는 질의 개수 그리고 (f)는 k값에 따른 질의 처리 속도를 비교한다. PB-kNN과 CPM은 가장 근접한 셀을 접근하여 k-최근접 질의를 처리하는 기본적인 방식이 동일하기 때문에 비슷한 성능 패턴을 가진다. 하지만 패턴 기반 질의 처리 기법은 사전에 연산된 패턴을 이용하여 셀 탐색 시 단지 O(n) 비용을 이용하기 때문에 CPM 기법보다 질의 처리 비용이 적다. 그리고 CPM기법이 빠른 질의 처리를 위해 각 질의 들이 유지하는 자료구조에 비해, 패턴 기반 질의 처리 기법은 모든 질의가 공유하여 사용하는 거리 관계 패턴을 기반으로 하기 때문에 질의 처리에 따른 저장 비용이 매우 적다. PB-kNN은 모든 부분에서 CPM보다 좋은 성능을 보이며 생략된 연속 질의 처리 성능 또한 모든 부분에서 우수하게 나타났다.

5. 결론

본 논문은 효율적인 연속 k-최근접 질의 처리를 위해 그리드 환경에서의 거리 관계 패턴을 제안하였다. 또한 제안한 거리 관계 패턴을 기반으로 하는 k-최근접 질의 처리 기법을 제안하였다. 동일한 모양과 크기를 갖는 셀들이 동일한 간격으로 배치 되어 있을 경우 제안하는 거리 관계 패턴은 O(n)의 비용으로 근접한 셀들을 순서대로 탐색해 질의 처리 비용을 감소시킨다. 또한, 저장 비용을 감소시켜 CPM의 문제를 해결했다. 마지막으로, 본 논문은 대표적인 k-최근접 질의 처리 기법인 CPM과의 성능 비교를 통해, 저장 비용과 처리 비용 측면에서 제안하는 k-최근접 질의 처리 기법의 성능이 모두 우수함을 확인하였다.

참고문헌

[1] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias, "Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring", Proceedings of the ACM Conference on Management of Data, pp. 634-645, 2005.

[2] D. V. Kalashnikov, S. Prabhakar, and S. E. Hambrusch, "Main Memory Evaluation of Monitoring Queries over Moving Objects", Distributed and Parallel Databases, Vol.15, No.2, pp.117-135, 2004.

[3] K. L. Wu, S. K. Chen and P. S. Yu, "Incremental Processing of Continual Range Queries over Moving Objects", IEEE Transactions on Knowledge and Data Engineering, vol.18, no.11, pp.1560-1575, 2006.

[4] X. Yu, K. Pu, and N. Koudas, "Monitoring k-nearest Neighbor Queries over Moving Objects", Proc. International Conference on Data Engineering, pp.631-642, 2005.

[5]K. Mouratidis, and D. Papadias, "Continuous Nearest Neighbor Queries over Sliding Windows", IEEE Transactions on Knowledge and Data Engineering, vol.19, no.6, pp.789-803, 2007.