위치 검색 지도 서비스를 위한 k관심지역 검색 기법 k-Interest Places Search Algorithm for Location Search Map Service

조성환¹⁾ · 이경주²⁾ · 유기윤³⁾

Sunghwan Cho · Gyoungju Lee · Kiyun Yu

Abstract

GIS-based web map service is all the more accessible to the public. Among others, location query services are most frequently utilized, which are currently restricted to only one keyword search. Although there increases the demand for the service for querying multiple keywords corresponding to sequential activities(banking, having lunch, watching movie, and other activities) in various locations POI, such service is yet to be provided. The objective of the paper is to develop the k-IPS algorithm for quickly and accurately querying multiple POIs that internet users input and locating the search outcomes on a web map. The algorithm is developed by utilizing hierarchical tree structure of R*-tree indexing technique to produce overlapped geometric regions. By using recursive R*-tree index based spatial join process, the performance of the current spatial join operation was improved. The performance of the algorithm is tested by applying 2, 3, and 4 multiple POIs for spatial query selected from 159 keyword set. About 90% of the test outcomes are produced within 0.1 second. The algorithm proposed in this paper is expected to be utilized for providing a variety of location-based query services, of which demand increases to conveniently support for citizens' daily activities.

Keywords : POI, k-Nearest Neighbor, R*-tree, Spatial Join, MBR

초 록

최근 인터넷의 발달과 더불어 지리정보시스템(GIS, Geographic Information System)에 대한 인식이 저변 확대되면서 일반인들도 위치 검색 기능을 제공하는 웹GIS를 쉽게 이용할 수 있게 되었다. 현재 서비스되고 있는 모든 위치 검색 기능은 사용자가 하나의 검색어를 입력하고 그에 대한 결과를 보여주는 서비스에 한정되어 있다. 하지만 사용자의 검색 목적이 다양해짐에 따라, 여러 가지 행위를 동시에 할 수 있는 장소를 검색하는 서비스는 없었다. 예를 들어, 점심을 먹은 후, 은행에서 업무를 보고, 영화 한 편을 보고자 할 때 이러한 관심 지역(POI, Point of Interest)들이 모여 있는 장소를 필요로 할 수 있다. 따라서 본 논문에서는 사용자로부터 여러 장소를 입력받아 입력된 장소가 모여 있는 곳을 검색해주는 k-IPS 기법을 제안하고자 한다. 여기서 k는 다양한 행위를 할 수 있는 관심의 개수이다. 이 방법은 최소경계사각형(MBR, Minimum Bounding Rectangle)의 계층적 트리 구조인 R*-tree 색인 기법을 이용하여 공간을 분할하고, 기존 공간 Join 연산의 성능 개선을 위하여 R*-tree간의 겹치는 영역 추출하는 재귀적 공간 Join 연산을 구현하였다. k-IPS 기법의 성능 평가는 159개의 다양한 검색어 집합을 구성하여 k=2,3,4,6에 대한 검색 결과를 확인하였다. 실험 결과의 약 90%에 대해서 예상한대로 k개의 검색어 장소가 모여 있는 위치를 얻을 수 있었고, k=2,3,4의 처리 시간은 0.1초 이내의 응답을 얻을 수 있었다. k-IPS 서비스를 통하여 현대인의 순차적 생활 패턴에 맞춘 검색 서비스가 가능할 것으로 판단된다.

핵심어 : 관심지역, k-최근접이웃검색, R*트리, 공간조인, 최소경계사각형

1) Member, Seoul National University Engineering Research Institute (E-mail:hallem@snu.ac.kr)

2) Corresponding Author, Department of Urban Engineering, Korea National University of Transportation (E-mail:lgjracer@ut.ac.kr)

3) Member, Department of Civil & Environmental Engineering, Seoul National University (E-mail:kiyun@snu.ac.kr)

1. 서론

최근 인터넷의 발달과 더불어 GIS에 대한 인식이 저변 확대되면서 일반인들도 Google, Bing, Daum, Naver 등의 포털 사이트들에서 제공하는 웹GIS 서비스를 쉽게 이용할 수 있게 되었다. 이러한 웹GIS 서비스는 위치 검색기능을 통하여 관심지역(Point Of Interest: POI)의 위치를 화면상에 나타내고 이를 설명하는 텍스트 정보와 결합하는 방식으로 공간정보를 제공한다. 현재 위치 검색 서비스는 하나의 목적지를 검색하는 서비스가 대부분이다. 사용자가 다양한 활동목적을 가지고 공간적으로 인접해 있는 다수의 위치를 동시에 찾아서 정보를 제공해주는 검색 서비스는 아직 제공되지 않고 있다(Bae *et al.*, 2009a; Bae, 2010). 이에 대해 Bae *et al.*(2009a)은 이러한 서비스의 필요성을 강조하고 근접한 3개의 POI를 찾는 Close-3POI 알고리즘을 제안하였다.

이러한 검색 서비스는 다양한 상황에서 필요할 수 있다. 예를 들면, “지하철역 이용이 편리하고 근처에 대형마트가 위치해 있는 호텔을 찾으시오.”와 같은 질의이다. 또 다른 예로는 초등학교 주변 유해 상업시설을 찾아내기 위해 “노래방, 바, 술집 근처에 위치하고 있는 초등학교를 찾으시오.” 혹은 일상적인 저녁 모임 장소를 선택하기 위해 “지하철 이용이 편리한 패밀리 레스토랑과 스타벅스가 모여 있는 장소를 찾으시오.” 등을 들 수 있다. 그러나 현재 제공되는 웹GIS 서비스의 경우 이러한 검색어 조합을 입력하여 검색하면 사용자가 예상하는 내용과는 다른 엉뚱한 결과가 제시되는 경우가 많다. 도시 공간상에서 한정된 시간에 많은 일을 처리해야 하는 현대 도시민의 입장에서 보면 다중 검색어 공간검색에 대한 서비스 수요는 점점 증가할 것으로 보인다. 따라서 사용자가 찾기를 원하는 다수의 위치들을 검색하여 정확한 결과를 산출할 수 있는 웹GIS 서비스가 필요한 시점이다.

다중 검색어 검색 알고리즘과 관련한 선행연구로서 Seon and Kim(2012)은 두 객체 집합에 대해 순환적 위치속성을 고려하여 가장 근접해 있는 k 개의 객체 쌍을 찾는 알고리즘을 제안하였다. 이 연구에서는 검색거리측도인 순환최소거리, 순환최대거리 그리고 순환최적거리를 제시하였고, R*-tree에서 k 개의 최대근접객체 쌍을 찾기 위해 방문할 필요가 없는 객체 쌍들을 방문대상에서 제외하는 알고리즘을 제안하였다. 유사한 맥락에서 Bae *et al.*(2009b)은 웹 환경에서 k -최근접 이웃 질의(k -NN)를 이용한 범위질의 시 질의점에서 가까이 있는 k 개의 위치 검색 방법을 제안하였고, 이에 대한 효율성 평가를 수행하였다. 이 연구에서는 웹상의 데이터를 대상으로 k -NN를 이용한 공간 범위 질의 기법을 수행할 때의 문제점을 지적

하고 이를 해결하기 위한 4가지의 구분된 기법을 제안하였다. 또한 Shin *et al.*(2000)은 두 공간 데이터 집합 사이의 데이터 쌍을 거리상으로 가까운 순으로 검색하는 공간 Join으로 적응식 다단계 거리 Join 기법을 제안하였다. 이 논문에서는 균등분포와 비균등 분포 가정 하에 한계 거리 값을 예측하는 실험하였고, CPU 비용과 I/O 비용에서 많은 향상이 있음을 보였다. Kim and Chang(2007)은 POI 기반 실체화 기법을 이용한 closest pairs 및 e-distance join 질의 알고리즘을 개발하였다. 하지만 전처리 과정을 통하여 대상 데이터 집합을 실체화 파일로 생성해야 하므로 데이터 집합의 변동이 발생하였을 경우 반복된 전처리를 수행해야 하는 문제점이 있다.

이 연구의 목적은 사용자로부터 k 개의 POI를 입력받아, 입력받은 관심지역이 모두 모여 있는 위치를 찾아낼 수 있는 검색 알고리즘을 개발하고 실증적 활용성을 검증하는 것이다. 이 알고리즘을 통하여 사용자가 입력한 $k(k>1)$ 개 이상의 POI 검색어들이 공간적으로 인접한 분포에 있는 장소들을 찾을 수 있도록 하고, 이에 소요되는 실행시간을 단축시키기 위해 재귀적 공간 Join 연산 알고리즘을 제안한다. 정리하면, 기존에 제안된 Close-3POI 알고리즘의 경우 3개의 POI만 검색할 수 있었던 반면에, k -IPS는 일반적인 k 개의 검색이 가능할 뿐 만 아니라 검색에 소요되는 시간 역시 단축했다는 점에서 알고리즘이 개선된 것으로 볼 수 있다. 이 알고리즘의 성능을 검증하기 위하여 159개의 테스트 데이터 집합에 대해 실험을 수행하였고, 그 결과를 Google Map에서 서울시의 일부 지역에 실증-적용하였다.

이 논문은 다음과 같이 구성된다. 2장에서는 공간 검색과 관련하여 기존 연구들에서 제시한 알고리즘을 검토한다. 3장에서는 기존의 공간 검색 알고리즘의 취약성을 보완하여 k -IPS 알고리즘을 제안한다. 이 알고리즘은 검색거리를 적응적으로 설정하여 반복 연산을 수행하는 Incremental Distance 방식을 적용하여 설계하였다. 4장에서 이 알고리즘을 실증자료에 적용하여 성능평가를 수행한다. 5장에서는 이 연구의 한계 및 이와 관련한 향후연구를 제안한다.

2. 관련 연구

공간색인 기법은 방대한 양의 공간 자료를 데이터 특성과 질의 조건을 고려하여 가장 적합한 공간 접근 방법(spatial access method)을 선택하는 과정이다. 공간색인은 크게 공간 기반(space-driven)의 색인과 데이터 기반(data-driven)의 색인으로 분류할 수 있다.

공간 기반의 색인에서는 2차원 공간을 다수의 수직선과

수평선에 의해 생성되는 격자로 관리한다. 이 때 색인은 실제 공간에 존재하는 객체들의 분포와 독립적으로 구축되며 Quad tree(Finkel *et al.*, 1974), k-d tree(Bentley, 1975), K-D-B tree(Robinson, 1981) 등이 이에 해당된다.

데이터 기반의 색인은 실제 객체들의 분포에 기초하여 공간을 분할하며 대부분 B+-tree(Comer, 1979)를 확장한 R-tree에 기반을 둔다. Fig. 1는 포인트 객체를 대상으로 R-tree 색인을 생성한 예이다. R-tree를 기반으로 확장한 색인 기법에는 R+-tree 및 R*-tree가 있고, 이들은 공간 Join 연산을 수행하는 데 가장 널리 사용되는 색인 기법이다.

R-tree는 노드 내 엔트리들 사이의 겹침(overlay)을 허용하기 때문에 공간 질의를 수행하기 위해서 방문해야 하는 경로(path)의 수가 증가한다는 단점을 가진다. 이러한 단점을 극복하고자 한 것이 R+-tree(Sellis *et al.*, 1987)로, 이는 노드들을 분할할 때 중간 노드들이 서로 겹치지 않도록 제한하는 특징을 갖는다. 따라서 R-tree에서는 특정 객체에 접근하기 위해 다수의 경로가 존재할 수 있었던 것에 반해, R+-tree에선 오직 하나의 경로만 이용하게 되어 검색 속도가 향상된다. 그러나 R+-tree는 노드 간 겹치는 영역은 없지만, 리프노드가 중복되는 엔트리를 가질 수 있는 문제점이 있다. 따라서 기존의 R-tree보다 많은 양의 저장 공간을 필요로 하고 트리의 높이가 더 높아질 수 있는 단점을 가진다. 한편, R*-tree(Beckmann *et al.*, 1990)는 삽입(insert) 알고리즘과 분할(split) 알고리즘을 보다 정교하게 개선하였고, 강제 재 삽입(forced reinsert) 알고리즘을 수행하는데, 이는 R-tree기반의 확장 색인 기법 중 가장 효율적인 공간색인 기법으로 알려져 있다(Brinkhoff *et al.*, 1993).

Fig. 1. Conceptual example of R-tree index

공간 데이터베이스(Spatial Databases)의 POI 공간 질의에 대한 연구는 크게 두 가지 주제로 요약된다. 첫째는 하나의 데이터 집합을 고려한 질의이고 다른 하나는 두 개 이상의 데이터 집합을 고려한 질의에 관한 연구이다(Kolahdouzan and Shahabi, 2004; Papadias *et al.*, 2003).

단일 POI 데이터 집합을 대상으로 검색하는 공간 질의에는 k 최근접 이웃 질의(k nearest neighbor query, kNN)와 범위 질의(range query)가 있다. kNN 질의란 사용자가 선택한 특정 위치인 질의점으로부터 공간상에서 가장 가까운 k개의 정적 객체를 검색하고 이를 반환하는 질의이다(Lee *et al.*, 2009; Papadias *et al.*, 2003). 이 방법은 먼저 루트 노드에 있는 엔트리들을 질의점과 최소거리에 따라 정렬한 후 힙을 구성한다. 그 다음 최소거리가 가장 작은 엔트리를 삭제하고, 이 엔트리가 가리키는 자식노드의 엔트리들을 최소거리에 따라 정렬하고 또다시 힙을 구성한다. 이러한 과정을 반복하여 질의점으로부터 가장 가까운 k개의 객체를 찾는다. 범위 질의는 주어진 영역 내에 존재하는 객체들을 찾는 질의이다. 유클리드 공간에서의 범위 질의는 공간상의 한 점으로부터 유클리드 거리 e 내에 존재하는 모든 객체들을 검색한다.

두 개 이상의 POI 데이터 집합을 대상으로 검색하는 공간 질의로는 최대근접 객체 쌍(Closest Pairs) 질의와 e-distance join 질의가 있다. 최대근접 객체 쌍 질의(Corral *et al.*, 2000)는 두 개의 데이터 집합을 대상으로 가까이 있는 객체 쌍을 검색하는 공간 Join 질의로, 두 공간 데이터 집합이 R-tree 구조로 저장되어 있다는 것을 기본적으로 가정한다. Closest Pairs 질의의 수행은 두 개의 R-tree의 각 루트 노드를 동시에 탐색함으로써 시작되며, 같은 레벨에 존재하는 비 리프 노드에 대하여 MBR들 간에 최소 거리를 갖는 노드 쌍을 찾아내게 된다. e-distance join 질의는 두 개의 공간 데이터 집합 S, T와 거리 값 e가 주어졌을 때, 거리가 서로 e 내에 존재하는 객체 쌍 (s, t) , $(s \in S, t \in T)$ 을 반환하는 질의이다. 유클리드 공간에서 e-distance join 질의는 R-tree를 이용한 공간 Join 연산을 수행하여 효율적으로 처리될 수 있다.

3. k-IPS 알고리즘

위치 검색 서비스에 소요되는 비용은 사용자의 검색어를 데이터베이스로부터 검색하는 단계, 위치 검색 알고리즘을 수행하는 단계, 결과를 화면에 출력하는 단계로 구성된다.

이 중 k-IPS에 의한 위치 검색 알고리즘을 수행하는 단계는 다음과 같은 세부 단계로 구성된다.

단계1: 사용자가 입력한 k개의 검색어 결과의 각 위치의 중심에서 거리 d를 가지는 사각형 영역을 생성

단계2: 단계 1에서 생성된 서로 다른 k개의 결과 집합의 사각형 영역들을 대상으로 공간 Join 연산을 수행

단계3: 종료조건(연산 횟수, 최대 거리 D)에 부합하지 않는

경우 사각형 영역 확대를 위해 거리 d 를 증가 후 단계 1 수행

3.1 사각형 영역 생성

k-IPS 연산은 연산 횟수를 감소시키기 위해서 k개의 POI 점집합을 개수가 작은 순으로 $POI_1, POI_2, \dots, POI_k$ 집합이라 명명하고, $a \in POI_1$ 와 $b \in POI_2$ 에 대하여 사각형 영역으로 변환을 수행한다. 즉, a의 좌표를 (x_a, y_a) , b의 좌표를 (x_b, y_b) 라 할 때 a로부터 변환된 사각형 객체의 좌하단 좌표와 우상단 좌표는 각각 $(x_a - \frac{d}{2}, y_a - \frac{d}{2})$, $(x_a + \frac{d}{2}, y_a + \frac{d}{2})$ 이며, b로부터 변환된 사각형 객체의 좌하단 좌표와 우상단 좌표는 각각 $(x_b - \frac{d}{2}, y_b - \frac{d}{2})$, $(x_b + \frac{d}{2}, y_b + \frac{d}{2})$ 이다. 이렇게 생성된 사각형 객체들로 R*-tree를 생성하고, 각각의 R*-tree의 이름은 R*poi₁, R*poi₂로 한다.

계속해서 모든 $J_{POI_{k-2}, POI_{k-1}}$ 에 저장된 점 객체 a와 모든 $b \in POI_k$ 에 대하여 검색 거리 d 를 포함한 MBR 연산을 수행한다. a의 좌표를 (x_a, y_a) , b의 좌표를 (x_b, y_b) , a로부터 생성된 사각형 객체의 좌하단 좌표를 (X_1, Y_1) , 우상단 좌표를 (X_2, Y_2) 라 하면 MBR을 계산하는 연산은 Eqs. (1) and (2)과 같다.

$$X_1 = \min(x_{R_{POI_{k-2}, POI_{k-1}}}) - \frac{d}{2}, \quad Y_1 = \min(y_{R_{POI_{k-2}, POI_{k-1}}}) - \frac{d}{2} \quad (1)$$

$$X_2 = \max(x_{R_{POI_{k-2}, POI_{k-1}}}) + \frac{d}{2}, \quad Y_2 = \max(y_{R_{POI_{k-2}, POI_{k-1}}}) + \frac{d}{2} \quad (2)$$

3.2 재귀적 공간 Join 연산

k-IPS 알고리즘은 k개의 검색어 점집합 $D_1, D_2, D_3, \dots, D_k$ 를 대상으로 각 집합 간의 거리가 d 내에 존재하는 검색어($s_1, s_2, s_3, \dots, s_k$) (단, $s_1 \in D_{S1}, s_2 \in D_{S2}, s_3 \in D_{S3}, \dots, s_k \in D_{Sk}$) 집합을 찾는 알고리즘으로, 기존 알고리즘의 처리 속도를 개선하기 위하여 재귀적 공간 Join 연산을 수행한다. 이는 비교 대상 노드 쌍을 찾기 위해 탐색 횟수를 줄이는 방법으로 기존의 동일한 루틴의 반복과 정인 전수(brute force)기법이 아닌 재귀적(recursive) 공간 조인 기법을 구현하였다.

공간 Join 연산은 k 개의 점집합 $D_{S1}, D_{S2}, \dots, D_{Sk}$ 중 검색 결과의 개수가 적은 순서로 정렬 후, 두 개의 검색 결과 집합 POI_1, POI_2 을 대상으로 공간 Join 연산을 수행하는 과정에서 1차 연산과 1차 연산에서 얻은 결과 집합을 POI_3 과 공간 Join 연산을 수행하는 2차 연산으로 구성된다. 이러한 연산을 반복하면 k-1차 연산은 k-2차에서 얻은 공간 Join 결과와 POI_k 를 대상으로 공간 Join 연산을 수행하므로, 이러한 과정을 Eq. (3)과 같이 표현할 수 있다.

$$\begin{aligned} POI_1 &= D_{\min(N_{D_1}, N_{D_2}, \dots, N_{D_k})} \\ POI_2 &= D_{\min([N_{D_1}, N_{D_2}, \dots, N_{D_k}] - POI_1)} \\ POI_3 &= D_{\min([N_{D_1}, N_{D_2}, \dots, N_{D_k}] - [POI_1, POI_2])} \\ &\vdots \\ POI_{k-1} &= D_{\min([N_{D_1}, N_{D_2}, \dots, N_{D_k}] - [POI_1, POI_2, \dots, POI_{k-2}])} \\ POI_k &= D_{\min([N_{D_1}, N_{D_2}, \dots, N_{D_k}] - [POI_1, POI_2, \dots, POI_{k-1}])} \\ &= D_{\max([N_{D_1}, N_{D_2}, \dots, N_{D_k}])} \end{aligned} \quad (3)$$

공간 Join 연산은 4가지의 경우를 재귀적 탐색에 의하여 진행되는데, 재귀적 탐색을 수행하는 이유는 두 단말노드(LeafNode)의 MBR 간의 교차영역 탐색(IsOverlapping)의 수행 속도를 향상시키기 위해서이다. 즉, 입력 값인 두 R_A^* , R_B^* 의 트리 깊이가 다르기 때문에 비교 연산을 위하여 두 개의 입력 트리에 대해 LeafNode까지의 빠른 탐색이 수행되어야 한다.

다음의 경우에 대해 재귀적 호출 여부를 결정한다.

1. R_A^* , R_B^* 모두 SubNode인 경우 두 입력 트리의 자식 노드에 대한 재귀적 호출을 통하여 교차여부 체크
2. R_A^* 는 SubNode, R_B^* 는 LeafNode의 경우 R_A^* 의 자식노드에 대해 재귀적 호출을 통하여 교차여부 체크
3. R_A^* 는 LeafNode, R_B^* 는 SubNode의 경우 R_B^* 의 자식노드에 대해 재귀적 호출을 통하여 교차 여부 체크
4. R_A^* , R_B^* 모두 LeafNode인 경우 교차영역 두 입력 단말 노드 간의 교차 여부 체크하여 결과를 $R_{A,B}^*$ 에 삽입

Algorithm은 재귀적 공간 Join 연산과정을 보여준다. isLeaf 함수는 입력된 노드가 최하단 노드인지를 확인한다. isOverlapping 함수는 입력된 두 노드의 영역이 서로 겹치는 영역을 포함하고 있는지를 확인한다. R*-tree노드의 1행, 8행, 13행, 18행은 각각 위의 4가지 경우에 대한 조건문을 나타낸다. 이 중 1행, 8행, 13행의 경우는 하위 노드가 존재하는지를 판단하여 Join 연산자를 재귀적으로 호출한다. 18행은 위 4가지 경우 중 4번째 경우에 속하는 연산으로 A, B의 검색 결과 집합의 노드가 모두 LeafNode일 때 실행된다. 이 과정은 두 결과 집합의 비교 대상 노드의 포함 여부를 계산하여 한 쪽이 다른 한쪽을 완전하게 포함하는 경우 포함되는 노드의 MBR을 저장하게 되고, 만약 두 노드가 겹치는 경우에는 교차점을 계산하여 교차 영역의 MBR을 결과 트리에 저장하게 된다. 여기서 bound 함수는 입력 객체의 MBR을 반환한다.

Algorithm : Join of R*tree**Input :** R_A^*, R_B^* **Output :** $R_{A,B}^*$

```

1. IF  $\text{isLeaf}(R_A^*.node) \ \&\& \ \text{isLeaf}(R_B^*.node)$  do
2.   IF  $\text{isOverlapping}(R_A^*.node, R_B^*.node)$  do
3.     FOR EACH  $n_A$  in  $R_A^*.node$  do
4.       FOR EACH  $n_B$  in  $R_B^*.node$  do
5.          $R^*tree.Join(n_A, n_B)$ 
6.       Next
7.     Next
8.   IF  $\text{isLeaf}(R_A^*.node) \ \&\& \ \text{isLeaf}(R_B^*.node)$  do
9.     IF  $\text{isOverlapping}(R_A^*.node, R_B^*.node)$  do
10.      FOR EACH  $n_A$  in  $R_A^*.node$  do
11.         $R^*tree.Join(n_A, R_B^*.node)$ 
12.      Next
13.   IF  $\text{isLeaf}(R_A^*.node) \ \&\& \ \text{isLeaf}(R_B^*.node)$  do
14.     IF  $\text{isOverlapping}(R_A^*.node, R_B^*.node)$  do
15.       FOR EACH  $n_B$  in  $R_B^*.node$  do
16.          $R^*tree.Join(R_A^*.node, n_B)$ 
17.       Next
18.   IF  $\text{isLeaf}(R_A^*.node) \ \&\& \ \text{isLeaf}(R_B^*.node)$  do
19.     IF  $\text{isOverlapping}(R_A^*.node, R_B^*.node)$  do
20.       FOR EACH  $n_A$  in  $R_A^*.node$  do
21.         FOR EACH  $n_B$  in  $R_B^*.node$  do
22.           IF  $\text{isOverlapping}(n_A, n_B)$  do
23.              $r_{pnts} = \text{CalcIntersectPoint}(n_A, n_B)$ 
24.             IF  $r_{pnts} > 1$  do
25.                $R_{A,B}^*.insert(\text{bound}(r_{pnts}))$ 
26.             else IF  $\text{isInside}(n_A, n_B)$ 
27.                $R_{A,B}^*.insert(\text{bound}(n_B))$ 
28.             else IF  $\text{isInside}(n_B, n_A)$ 
29.                $R_{A,B}^*.insert(\text{bound}(n_A))$ 
30.           Next
31.         Next

```

k개의 검색어 집합에 대한 공간 Join 연산은 우선 R^*poi_1 , R^*poi_2 에 대하여 공간 Join 연산을 수행하여, 서로 거리 d 내에 존재하는 R^*poi_1 , R^*poi_2 의 점 객체 쌍 (a, b)를 구하고 이를 결과 집합 R_{POI_1, POI_2} 으로 저장한다. 이 과정을 반복하여 k-1차 공간 Join 연산에서는 k-2차 연산으로 도출된 $J_{POI_{k-2}, POI_{k-1}}$ 와 POI_k 에 대해 공간 Join 연산을 수행하여, 이로부터 거리 d 내에 존재하는 (a, b) (단 $a \in J_{POI_{k-2}, POI_{k-1}}$, $b \in POI_k$)의 결과 집합을 검색한다.

앞의 과정에서 생성된 MBR들로 R*-tree를 생성하고, R*-tree의 이름은 각각 $R^*j_{k-2,k-1}$, R^*poi_k 으로 한다. $R^*j_{k-2,k-1}$, R^*poi_k 에 대하여 공간 Join 연산을 수행하여, 서로 중첩하는 $J_{POI_{k-2}, POI_{k-1}}$ 와 POI_k 의 사각형 객체 쌍 (a, b)를 구한다. 이로부터 거리 d 내에 존재하는 ($J_{POI_{k-2}, POI_{k-1}}$, POI_k)를 계산하여 최종 분석 결과인 ($POI_1, POI_2, \dots, POI_k$)의 Join 연산 결과 집합을 생성한다.

3.3 검색 거리 증가

모든 검색어에 대한 연산은 항상 최초 연산에 의해 만족할 만한 결과를 얻는 것은 아니다. 본 연구에서는 사용자가 걸어 갈 수 있는 거리 500m를 최초 검색 거리로 입력하였지만, 검색 결과가 없을 경우 검색거리 d를 100m씩 증가시키면서 반복 연산을 수행하였고, 그에 따른 종료 조건과의 부합여부를 판단하였다.

아래와 같은 경우에 검색 거리 d를 증가시키며 반복 연산을 수행하였다.

1. 모든 검색어에 대한 공간 Join 연산 결과가 없고, 각 차수의 공간 Join 연산 $J_{POI_{k-2}, POI_{k-1}}$ 의 결과가 5개 미만인 경우
2. 동일한 검색어 집합에 대해 검색거리 d를 증가시키며 수행한 연산 횟수가 10회 미만인 경우

4. 실험 및 결과

4.1 데이터 및 적용

이 연구에서 제안한 k-IPS 검색 기법은 윈도우즈 XP 플랫폼에서 성능 평가를 수행하였다. 하드웨어는 2.4GHz 쿼드코어 프로세서, 4GB 메모리의 컴퓨터를 사용하였다. DBMS로는 ORACLE 10g를 사용하였고, 알고리즘은 C++을 이용하여 구현하였다. 실험 결과는 KML(Keyhole Markup Language)을 이용하여 생성하고, 이를 Google Map에 표현하였다.

성능 실험을 위하여 k=2, k=3, k=4에 대해 각각 159개의 검색어 집합을 생성하고 아래와 같은 내용의 검색을 수행하였다.

검색어 집합1 : 일상적인 생활에서 발생할 수 있는 검색을 수행한 결과. 전철역 주변의 패밀리레스토랑에서 식사를 하고, 스타벅스나 카페에서 커피를 마실 수 있는 장소 검색

검색어 집합2 : POI의 범주에 대한 검색으로 식사를 하고

Table 1. Performance of keywords searching algorithm

	POI datasets				k=2	k=3	k=4
	POI_1	POI_2	POI_3	POI_4	$[POI_1, POI_2]$	$[POI_1, POI_2, POI_3]$	$[POI_1, \dots, POI_4]$
1	starbucks	Outback	cafe	subway Station	0.00315	0.011118	0.038552
2	restaurant	karaoke	pub	cafe	0.04038	0.069257	0.102723
...	...	...	...	...	...	...	...
159	elementary school	pub	karaoke	bar	0.001351	0.002796	0.00553

커피를 마신 후, 맥주 및 노래방 장소 검색
 검색어 집합3 : 초등학교에 유해가 되는 장소를 검색하는 것으로 술집, 바, 노래방이 인접해 있는 초등학교 검색

Table 1은 k-IPS($k \in \{2, 3, 4\}$) 검색 알고리즘을 적용한 검색어 집합 예시이다. Table 1에서 예시한 바와 같이 각각의 검색어 집합을 4개의 POI로 구성하였고, $k \in \{2, 3, 4\}$ 의 각 연산에 대해 $(a, b) \in (POI_1, POI_2)$, $(a, b, c) \in (POI_1, POI_2, POI_3)$, $(a, b, c, d) \in (POI_1, POI_2, POI_3, POI_4)$ 의 실험을 수행하였고, 추가적으로 재귀적 공간 Join 연산의 성능 테스트를 위해 $k=6$ 일 때의 처리 속도를 측정하였다. $k=6$ 의 실험 데이터 집합은 $k=4$ 의 데이터 집합에 두 개의 검색어를 추가하여 구성하였다.

4.2 평가 및 분석

검색어 집합 1의 예에서는 아웃백에서 저녁을 먹은 후, 가까운 스타벅스에서 커피를 마시고, 전철을 이용하여 집에 오기까지 최소한의 지리적 이동을 위한 장소들을 검색하고자 하였다. 검색어 집합1에 대한 결과는 Fig. 2와 같다. 아웃백 및 스타벅스는 가맹점을 가지는 단일 명칭이므로 여러 개의 위치 정보를 가진다. 전철에 대한 검색은 POI 테이블의 레코드를 확인하여 '출구'를 검색어로 사용하였다. Fig. 3는 이에 대한 상세한 위치 분포를 보여준다. 검색어 집합 1의 활동을 만족시켜줄 수 있는 후보 지역은 서울 시청역(a), 여의도역(b), 교대역(c), 천호역(d), 수원역(e), 안양역(f) 주변인 것으로 나타났다.

검색어 집합 2는 검색어 결과 집합 중 크기가 작은 술집과 노래방에 대한 연산이 우선적으로 수행되었고, 다음으로 카페, 식당 순서로 연산이 수행되었다. $k=4$ 에 대한 실험은 카페와 식당의 개수는 어느 POI의 개수 보다 많기 때문에 $k=4$ 에서 연산 횟수가 증가로 인해 응답 시간이 증가한 것을 알 수 있다.

검색어 집합 3은 집합의 크기가 가장 큰 초등학교가 $k=2, 3$,

4 모든 연산에 사용되었다. POI_2, POI_3, POI_4 의 집합 크기는 POI_1 의 집합 크기보다 상대적으로 작기 때문에 k 에 따른 응답 시간은 전반적으로 집합의 크기가 가장 큰 POI_1 에 의해 영향을 받은 것으로 판단된다.

Fig. 2. Results for applying search algorithm

Fig. 3. zoomed-in screen shots of results in Fig. 2

Table 2. Success rate in keywords search

Keyword datasets	k=2	k=3	k=4	k=6
A,B	159	137	129	120
A,B,C	-	119	96	69
A,B,C,D	-	-	78	51
A,B,C,D,E	-	-	-	31
A,B,C,D,E,F	-	-	-	26
Success rate	100%	75%	49%	16%

Table 3. Searching time Comparison between all keywords and successfully retrieved keywords

Category		k=2 (sec)	k=3 (sec)	k=4 (sec)	k=6 (sec)
All keywords	Total processing time	6.1391	7.3548	23.1839	102.2022
	Average	0.0386	0.0463	0.1458	0.6428
	Standard deviation	0.0801	0.1021	0.3991	3.7027
Successfully retrieved keywords	Total processing time	6.1391	5.5081	7.0669	59.2353
	Average	0.0386	0.0463	0.0906	3.0940
	Standard deviation	0.0801	0.0940	0.1835	8.5313

결과 그래프 Fig. 4를 보면, 예상했던 것과 같이 k=2의 검색 속도는 대체로 k=3,4,6의 경우보다 빠른 것으로 나타났다. 159 개의 실험 검색어 집합에 대해 실험을 수행한 결과 모든 4개 미만의 k개에 대해 약 90%의 실험 대상은 0.1초 이내의 검색 결과를 얻을 수 있었다. 나머지 10% 정도는 검색 시간의 초과 또는 적응적 검색 거리의 증가 회수를 초과하였기 때문에 결과를 얻지 못하였다. 이들 검색어 중에는 '빌라'가 포함된 POI가 상당수 있었으며, 그 데이터 수가 22,724개에 달하여 Join 연산 시 제한 시간을 초과한 것으로 판단된다. k=6의 실험에서 몇몇 사례의 경우 k=2,3,4의 연산 속도보다 빠르게 나타났다. 이는 공간 Join 연산 시 검색어의 결과 집합 크기가 작은 순서대로 수행하기 때문인 것으로 풀이된다. 즉, 1차 연산에 사용된 두 결과 집합의 크기가 k=2,3,4의 실험 때보다 더 작아졌기 때문에 제약조건에 부합되어 종료되었다.

Table 2에서 보면, 검색어의 개수가 증가할수록 위치 검색 결과의 성공률이 낮아지는 것을 알 수 있다. k=2인 경우에는 모든 테스트 데이터 집합에 대한 검색 결과를 얻은 반면 k=3,4,6의 경우 각각 75%, 49%, 16%의 검색 결과를 얻을 수 있었다. 이를 통하여 검색어의 개수가 증가할수록 해당 장소가 한 곳에 모여 있는 위치의 검색 성공률이 낮아지는 것을 알 수 있었다. 또한 Table 3의 결과와 같이 실험에 사용한 모든 검색어 집합의 처리속도와 성공적으로 최종 결과를 얻은 검색어 집합의 처리속도를 살펴본 결과, k=4에서는 전체 대비 성공한 검색어의 처리시간이 약 30%를 차지하지만, k=6에서는 약 58%를 차지하는 것으로 나타났다. 이는 최종 결과를 얻지 못한 검색어 데이터 집합의 공간 Join 연산이 몇 단계 Join 과정에서 중단되었는가에 따라 처리 속도가 달라지기 때문인 것으로 풀이된다.

Fig. 4. searching time graph for k=2,3,4,6

5. 결론

이 연구에서는 다양한 활동을 위한 관심 있는 장소가 인접한 위치를 찾기 위하여 k-IPS 검색 알고리즘을 개발하였고 실험을 통하여 웹에서의 활용 가능성을 검증하고자 하였다. 즉, 인접한 위치 연산을 위하여 R*-tree 공간 색인을 사용하였고, 서로 다른 검색어의 질의 결과 데이터 집합에 대한 위치적 공간 탐색을 위해 공간 Join 기능을 개발하였다. k개의 검색어 데이터 집합을 얻기 위해 k=2, 3, 4에 대한 실험을 진행하였

고, 실험 중 약 90%가 0.1초 이내에 연산 결과를 얻을 수 있었다. 이 연구에서 제안한 k-IPS 알고리즘은 다음과 같은 면에서 기존의 타 검색 알고리즘과 차별성을 지닌다. 첫째, 기존 Close-3POI에서 제안한 서로 다른 깊이에서의 R*-tree기반의 공간 Join 알고리즘을 재귀적 처리 기법을 통해 개선하여 검색 속도를 향상시켰다. 둘째, 기존의 연구에서 3개의 POI만을 검색할 수 있었던 것을 일반화하여 k개의 POI를 검색할 수 있도록 하였다.

k-IPS 서비스를 통하여 식사를 하고, 커피를 마신 후, 영화를 보고, 전철을 타는 등의 현대인의 순차적 생활 패턴 맞춤형 검색 서비스가 가능할 것으로 판단된다. 또한 초등학교 주변의 다양한 유흥시설을 검색하는 등의 공간분석에도 활용할 수 있을 것으로 예상된다.

이 연구에서 제시한 알고리즘은 검색어 개수 k가 증가할수록 그리고 입력된 k개의 POI 검색 결과 집합이 클수록 공간 Join 횟수가 증가하여 응답 속도에 영향을 미치는 한계를 가지고 있다. 제안한 알고리즘은 웹 환경에서의 서비스를 위하여 제약사항을 설정하여 이를 초과하는 경우 연산을 중단하도록 하였지만 향후에는 검색어의 대체 장소를 반영한 향상된 알고리즘 개발이 필요하다.

감사의 글

본 연구는 국토교통부 도시건축 연구개발사업의 연구비지원(11첨단도시G10)에 의해 수행되었습니다.

References

- Bae, Y. (2010), *Design, Application, and Analysis of Close-3POI Algorithm for Location Search Map Service based on Web GIS*. MD. Dissertation, Seoul National University, Seoul, Korea. (in Korean with English abstract)
- Bae, Y., Cho, S., and Yu, K. (2009a), Design of the web map service searching the location of Close-3POI, *2009 GIS joint fall conference*, The Korean Society for Geospatial Information System, pp. 201-202. (in Korean with English abstract)
- Bae, W., Alkobaisi, S., Kim, S., Narayanappa, S., and Shahabi, C. (2009b), Web data retrieval: solving spatial range queries using k-nearest neighbor searches. *Geoinformatica*, Vol. 13, No. 4, pp. 483-514.
- Beckmann, N., Kriegel, H. P., Schneider, R., and Seeger, B. (1990), *The R*-tree: an efficient and robust access method for points and rectangles*, ACM, Vol. 19, No. 2, pp. 322-331.
- Bentley, J. L. (1975), Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, Vol. 18, No. 9, pp. 509-517.
- Bing map(2013), Microsoft, <http://www.bing.com/maps>(last date accessed: 5 June 2013)
- Brinkhoff, T., Kriegel, H. P., and Seeger, B. (1993), Efficient processing of spatial joins using R-trees, pp. 237-246.
- Comer, D. (1979), The Ubiquitous B-tree. *ACM Computing Surveys*, Vol. 11, NO. 2, pp. 121-138.
- Corral, A., Manolopoulos, Y., Theodoridis, Y., and Vassilakopoulos, M. (2000), Closest Pair Queries in Spatial Databases. *ACM SIGMOD*, Vol. 29, No 2, pp. 189-200
- Daum Map (2013), *Daum Communication*, <http://local.daum.net>(last date accessed: 3 June 2013)
- Finkel, R. A. and Bentley, J. L. (1974), Quad Trees - A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, Vol. 4, No. 1, pp. 1-9.
- Google Maps (2013), *Google Inc.*, <http://maps.google.co.kr>(last date accessed: 5 June 2013)
- Kim, Y. and Chang, J. (2007), Closest Pairs and e-distance Join Query Processing Algorithms using a POI-based Materialization Technique in Spatial Network Databases. *Journal of Korea Spatial Information System Society*, Vol.9, No.3, pp.67-80. (in Korean with English abstract)
- Kolahdouzan, M. and Shahabi, C. (2004), Voronoi-based k nearest neighbor search for spatial network databases. *In Proceedings of the Thirtieth international conference on Very large data bases, VLDB Endowment*, Vol. 30, pp. 840-851.
- Lee, S., Kim, S., Lee, J., and Yoo, J. (2009), Approximate Indexing in Road Network Databases, *2009 ACM Int'l Symp. on Applied Computing*, ACM, pp. 1568-1572.
- Naver Map (2013), NHN, <http://map.naver.com>(last date accessed: 3 June 2013)
- Papadias, D., Zhang, J., Mamoulis, N., and Tao, Y. (2003), Query Processing in Spatial Network Databases, *In Proc. Int'l Conf. on Very Large Data Bases, VLDB*, Vol. 29, pp. 802-813.

- Robinson, J. T. (1981), The K-D-B Tree A Search Structure for Large Multidimensional Dynamic Indexes, *Proc. of 4CM-SIGMOD Conf.*, pp. 10-18.
- Sellis, T., Roussopoulos, N., and Faloutsos, C. (1987), The r+-tree: A dynamic index for multi-dimensional objects.
- Seon, H. and Kim, H. (2012), Algorithm for Finding K-Nearest Object Pairs in Circular Search Spaces. *Journal of Korea Spatial Information System Society*, Vol. 20, No. 2, pp. 165-172. (in Korean with English abstract)
- Shin, H., Moon, B., and Lee, S. (2000), Adaptive multi-stage distance join processing. In *ACM SIGMOD Record, ACM*, Vol. 29, No. 2, pp. 343-354.
- Yahoo Map (2013), *Yahoo! Inc.*, URL: <http://maps.yahoo.com>(last date accessed: 5 June 2013)

(Received 2013. 05. 29, Revised 2013. 06. 12, Accepted 2013. 08. 12)