Journal of Korea Multimedia Society Vol. 23, No. 4, April 2020(pp. 517-524)
https://doi.org/10.9717/kmms.2020.23.4.517

가중치 그래프의 고유벡터 중심성에 따른 실시간 차량추적 알고리즘

김선형†, 김상욱††

Real-time Vehicle Tracking Algorithm According to Eigenvector Centrality of Weighted Graph

Seonhyeong Kim†, Sangwook Kim††

ABSTRACT

Recently, many researches have been conducted to automatically recognize license plates of vehicles and use the analyzed information to manage stolen vehicles and track the vehicle. However, such a system must eventually be investigated by people through direct monitoring. Therefore, in this paper, the system of tracking a vehicle is implemented by sharing the information analyzed by the vehicle image among cameras registered in the IoT environment to minimize the human intervention. The distance between cameras is indicated by the node and the weight value of the weighted-graph, and the eigenvector centrality is used to select the camera to search. It demonstrates efficiency by comparing the time between analyzing data using weighted graph searching algorithm and analyzing all data stored in databse. Finally, the path of the vehicle is indicated on the map using parsed json data.

Key words: Internet of Things, Weighted Graph, Path Generation

1. 서 론

최근 사물인터넷 환경에서 CCTV와 같은 공공분야의 카메라를 이용하여 카메라 주변의 환경을 모니터링하고 특정 객체를 추적하는 감시시스템에 대한 연구가 증가하고 있다[1,2]. 또한 이러한 감시시스템을 이용한 감시 애플리케이션의 수가 지속적으로 증가하고 있다[3]. 그러나 실시간으로 이동하는 객체를 감지하는 보안 시스템은 직접적인 모니터링이 필요하기 때문에 많은 시간과 인력이 요구되며, 단일기기를 이용할 시에 기기는 회전과 같은 단순 동작만을 수행하기 때문에 제한된 촬영범위를 가진다.

또한 카메라를 이용한 감시시스템을 통해 촬영된 영상을 분석하여 용의자의 인상착의 분석이나 차량번호판 인식 시스템에 대한 연구가 이루어지고 있다[4,5]. 그러나 용의자를 찾고 차량을 찾는 시스템은 사람의 개입이 필요하므로 실시간 추적이 어렵고, 많은 양의 영상과 이미지를 분석해야한다는 문제점이 있다. 이러한 방대한 데이터를 처리하기 위해 빅데이터 영상 처리를 이용하는 연구가 이루어지고 있다

※ Corresponding Author : Sangwook Kim, Address: (702-701) 80 Daehakro, Bukgu, Daegu, Korea, TEL : +82-53-940-8881, FAX : +82-53-950-6369, E-mail : kimsw@knu.ac.kr
Receipt date : Mar. 9, 2020, Revision date : Mar. 23, 2020.
Approval date : Apr. 1, 2020
† School of Computer Science and Engineering, Graduate School, Kyungpook National University
(E-mail : kimsh9510@gmail.com)
†† School of Computer Science and Engineering, Graduate School, Kyungpook National University
※ This research was supported by the BK21 Plus project of the Ministry of Education and the Korea Research Foundation (SW Human Resources Development Team for the realization of Smart Life at Kyungpook National University) (21A20131600005)

[6]. 따라서 빅데이터를 처리하는 연구뿐만 아니라 분석할 데이터의 양을 줄이는 연구가 필요하다.

사물인터넷 환경에서 IoT 기기의 네트워크를 그래프로 표현한 연구가 많이 이루어지고 있다[7]. 따라서 본 논문에서는 사물인터넷 환경에 등록된 카메라 간의 관련성을 가중치 그래프로 정의하고 나타내었다. 그래프에서 정의된 카메라 노드 간의 관계에 따라 분석한 정보를 서로 공유하고 탐색한다. 뿐만 아니라 차량의 이동반경을 예측하여 그래프의 가중치 값으로 나타내었고, 고유벡터 중심성을 계산하여 중요한 노드에 대한 탐색 결과에 따라 카메라의 영상을 분석하기 때문에 시간을 절약할 수 있다. 이를 통해 사용자들에게 차량의 이동경로를 지도에 출력하는 서비스를 제공함으로써 차량의 이동경로를 직관적으로 확인하고, 현재위치를 예측할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 소개하고 3장에서는 제안하는 가중치 그래프의 정의와 구조, 그리고 탐색 알고리즘을 설명한다. 또한 4장에서는 카메라 기기의 모니터링과 차량의 추적에 대한 현재 구현 내용을 기록하고 평가한다. 그리고 마지막으로 5장에서 정리한다.

2. 관련연구

이동하거나 정지한 상태에서 차량 감지와 추적을 위해 단일 카메라를 사용하는 연구가 있다[8,9]. 차량에 탑재된 단일카메라에서 촬영된 영상에서 고성능 비전 알고리즘(WB)을 이용하여 다양한 환경 조건에서 차량을 인식한다. 또한 번호판 인식 기술을 이용하여 번호판 위치를 확인하고 번호를 식별하며 성능을 평가하는 연구가 있다[10,11]. 그러나 이러한 연구들은 차량의 감지와 단일 카메라의 이미지 분석을 통한 차량의 인식률에 초점을 두고 있고, 움직이는 차량의 인식과 차량의 구분에 연구초점을 두고 있다. 따라서 다중 카메라를 이용하여 카메라 간의 관계를 분석하거나 차량을 추적하지 않기 때문에 단일 카메라의 성능을 평가하는데 그친다.

또 다른 연구로 차량 도난이나 절도의 상황에서 차량을 찾고 모니터링하는 연구가 있다[12,13]. OCR 라이브러리를 이용하여 이미지에서 차량번호의 문자열을 추출하고 문자열을 이용하여 차량의 세부사항이나 경로를 파악한다. 그러나 본 연구는 자동화 시스템이 아니기 때문에 차량의 번호를 수동으로 점검해야하며, 동일한 번호판을 인식한 센서를 찾기 위해 'nearest sensor'를 탐색하는데 'nearest'의 기준이 모호하다. 그리고 직접 목적지를 센서의 이름으로 입력함으로써 최종 경로를 출력하기 때문에 사람의 개입이 필요하며 차량의 위치 변경만 인지하고 자동적으로 추가적인 탐색이 이루어지지 않아 도난된 차량의 빠른 추적에는 어려움이 있다.

또한 IoT환경에서의 사물인터넷 기기를 가중치 그래프에 적용한 연구가 있다[14,15]. 복잡한 네트워크를 그래프의 노드와 가중치로 단순하게 나타내었다. 또한 사물인터넷에서 생산되는 빅데이터를 실시간으로 탐색하고 쿼리하기 위한 뷰를 제공하기 위해 그래프를 이용하였고, 그래프와 노드 간의 연결성을 이용하여 커뮤니티를 생성한다. 그러나 본 연구는 받아온 빅데이터를 시각화하고 군집화 하는데 그치며 실시간으로 노드 간의 연결성 변화에 의해 새로운 군집을 생성하지는 않는다.

3. 가중치그래프를 이용한 차량의 이동경로 추적

3.1 가중치 그래프의 정의

제안하는 가중치 그래프 $G=(N,E,W)$는 노드의 집합 N과 간선의 집합 E, 그리고 간선의 가중치 W로 구성된다. 노드 N은 수식 (1)과 같이 정의하며, 시스템에 등록된 n개의 카메라를 나타낸다. 또한 간선 E는 수식 (2)와 같이 정의하며, 카메라와 카메라를 연결하고 방향성을 포함한다.

$$N=\{c_1, c_2, c_3, \cdots, c_n\} \tag{1}$$

$$E=\{e_{1,2}, e_{2,3}, e_{3,4}, \cdots, e_{n-1,n}\}\ (e_{n-1,n} = <e_{n-1}, e_n>) \tag{2}$$

카메라 간의 실제거리는 $getDeviceDistance$ 함수를 이용하여 계산한다. 위경도 소수점 좌표를 도(D), 분(M), 초(S) 좌표로 변환한 후 계산되며 아래 수식 (3)을 이용한다[16].

$$\begin{aligned} X &= cos(D_i)\times\left(\frac{2\pi R}{360}\right) \\ Y &= \frac{2\pi R}{360} \\ Z &= \frac{X}{60}, Z' = \frac{Y}{60} \end{aligned} \tag{3}$$

간선의 가중치 W의 값은 수식 (4)과 같이 정의한다[16]. 가중치 $W(e_{i,j})$는 간선 $e_{i,j}$의 길이를 의미하며 카메라 c_i와 c_j간의 실제 거리를 값으로 갖는다. 또한

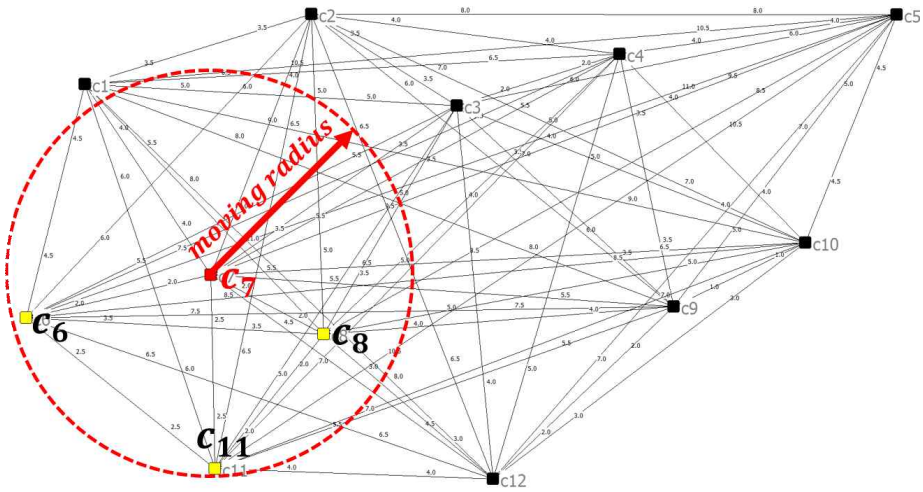

Fig. 1. Structure of Weighted Graph.

거리로 나타낸 가중치의 값은 인접행렬인 2차원 배열에 저장된다.

$$W(e_{i,j}) = gDD(c_i^{loc}, c_j^{loc}) = weight[e_i][e_j]$$
$$= \sqrt{((D_i \times X)+(M_i \times (Z))+(S_i \times (\frac{(Z)}{60})))^2+((D_j \times Y)+(M_j \times (Z^{'}))+(S_j \times (\frac{(Z^{'})}{60})))^2} \quad (4)$$

3.2 가중치 그래프의 구조

제안하는 가중치 그래프의 구조는 Fig. 1.과 같다. 시스템에 등록된 n개의 카메라 기기를 나타내는 노드 N에 대해 생성할 수 있는 모든 경로를 간선 E로 표현하였다. 또한 카메라의 위경도 좌표 값을 이용하여 실제 거리를 계산한 값을 가중치 W로 나타낸다.

노드 $N(c_x)(x=7)$의 카메라에서 찾고자하는 차량이 탐색되었을 때, 카메라 c_x의 위치 c^{loc_x}의 값을 출력하고자 하는 경로($path$)에 추가한다. 또한 차량을 탐색한 시간 c^{t_x}과 현재시간($current\ time$)을 이용하여 차량의 이동반경($moving\ radius$)을 예측한다. 차량의 이동반경(d_{c_x})을 구하는 식은 다음 수식 (5)와 같다. 수식의 tm_{day},tm_{hour},$tm_{\sec}$는 각각 경과한 일, 시간, 분에 대한 변수이며 시간(h) 단위로 변환하여 속도($speed$)와의 곱으로 이동 반경을 계산하였다.

$$d_{c_x}(km) = (tm_{day} * 24 + tm_{hour} + tm_{\min} \times 1/60 + tm_{\sec} \times 1/3600) \times speed\,(km/h) \quad (5)$$

다음으로 수식 (6)의 조건에 따라 탐색하고자 하는 카메라 노드 N을 찾는다. 노드 $N(c_x)$를 시작노드로 하고 연결된 간선 $E(e_{x,k})$의 가중치의 값이 차량의 이동반경 보다 작은 노드 $N(c_k)(i=6,8,11)$에 대해 차량 탐색을 반복한다.

$$weight[c_x][c_k] < d_{c_x}(moving\ radius) \quad (6)$$

이동반경($moving\ radius$) 내에 있는 카메라에 대한 탐색순서는 중요한 노드를 기준으로 한다. 이때 중요한 노드는 고유벡터 중심성(Eigenvector Centrality, Ce)을 이용하여 결정한다. 이 때, 고유벡터 중심성이란 연결된 노드가 많은 노드가 중요한 노드가 되는 것이 아닌, 중요한 노드와 많이 연결된 노드를 중요하게 판단하는 방법이다. 따라서 고유벡터 중심성에 따라 중요한 노드 $N(c_7)$과 연결된 다른 노드들의 중심성을 반영하여 중요한 노드를 결정한다. 다음 Fig. 2는 찾고자 하는 차량을 탐색한 노드 $N(c_x)(x=7)$와 이동반경($moving\ radius$) 내에 있는 노드 $N(c_k)(i=6,8,11)$ 간의 간선 $E(e_{x,k})$ 만을 표현한 그래프이다.

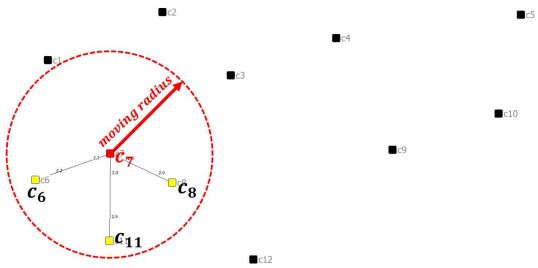

Fig. 2. Structure of Weighted Graph for Search.

Table 1. Eigenvector Centrality

Node	*Eigenvector Centrality*
c_1	0
c_2	0
c_3	0
c_4	0
c_5	0
c_6	2.494
c_7	4.404
c_8	2.267
c_9	0
c_{10}	0
c_{11}	2.834
c_{12}	0

다음 그래프의 고유벡터 중심성을 계산한 결과는 Table. 1과 같다. 차량을 가장 먼저 탐색한 노드 $N(c_x)(x=7)$의 고유벡터의 값은 4.404로 가장 크다. 또한 나머지 고유벡터의 값은 $c_{11}(2.834)$,$c_6(2.494)$, 그리고 $c_8(2.267)$ 순서로 큰 값을 가지므로 다음 탐색할 카메라는 c_{11}, c_6, c_8 의 순서로 탐색한다.

이동반경 내에 있는 노드들 중 고유벡터 중심성이 큰 노드부터 탐색하여, 이전에 지나온 경로를 찾는데 초점을 두는 것이 아니라 최신의 경로를 찾는데 초점을 둔다. 고유벡터 중심성과 가중치의 중요도에 따라 탐색 우선순위를 정의하는 것은 수식 (7)과 같다. 따라서 수식에 따라 정의한 $sequence$ 배열에 가중치의 크기가 큰 값부터 내림차순으로 저장한다.

$$sequence[\,] = MAX\_DESC(weight[c_x][c_k]) \tag{7}$$

3.3 가중치 그래프 탐색 알고리즘

가중치 그래프를 이용한 차량 탐색 알고리즘은 아래 Fig. 3과 같다.

Input : 추적하고자 하는 차량의 번호(num_x)를 입력으로 한다.

Output : 추적하고자 하는 차량의 이동경로 $path$이다.

(1) 시스템에 등록된 카메라의 집합 C에 대해 카메라의 고유번호, 촬영시간, 기기의 종류, 기기의 위도좌표, 기기의 경도좌표, 촬영된 영상의 저장경로, 분석한 차량번호, 그리고 카메라의 연결 여부를 데이터베이스에 실시간으로 저장한다.

(2) 카메라 집합의 영상 CV 중 x번째 카메라의 영상 cv_x가 카메라 연결의 오류 없이 분할, 저장되었을 때 분할된 이미지를 img_x에 저장한다. img_x를 분석하여 원하는 차량의 번호판 문자 (num_x)를 인식하였을 때, 차량의 이동경로 $path$에 카메라 c_x의 위치정보(c_x^{loc})를 저장한다.

(3) 번호판이 인식된 이미지가 촬영된 시간(c_x^t)와 현재시간($current\ time$)의 비교를 통해 경과한 시간(t_{c_x})을 계산하고, 차량의 평균 속도를 이용하여 이동반경(d_{c_x})을 예측한다.

(4) 이동반경(d_{c_x}) 이내에 있는 카메라 c_k에 대해 카메라 c_x와의 거리를 계산하여 $distance[\,]$에 저장한다. 또한 이동반경 내부에 있는 카메라 c_k에 대해 거리 값이 큰 순서대로 $sequence[\,]$에 저장한다.

(5) 가장 큰 거리 값 $sequence[0]$을 갖는 카메라 c_l에 대해 실시간으로 촬영된 이미지를 img_l에 저장한다. img_l을 분석한 문자열 값이 찾고자 하는 차량의 번호판 문자(vm_x)와 동일할 때, 차량의 이동경로 $path$에 카메라 c_l의 위치정보(c^{loc_l})를 저장한다. 이때 img_l 분석은 카메라 c_x가 vm_x를 인식한 시간(c^{t_x})의 이후 시

Operation of real-time tracking system

Input	num_x *is the plate number of vehicle.*
Output	*path is the path of the tracking vehicle*

```
1  addCamInfo()
2  if takeVideo(cv_x) = true then
3      ALOOP:
4      for each c_x in C
5          img_x ← captured image(cv_x)
6          if string(img_x = num_x) then
7              path[ ] ← pop(c_x^loc)
8              t_{c_x} = current time − c_x^t
9              d_{c_x} = t_{c_x} × speed
10             for each c_k in C
11                 distance[e_{x,k}] ← gDD(c_x^loc, c_k^loc)
12                 if distance[e_{x,k}] < d_{c_x} then
13                     sequence[ ] ← max(distance[e_{x,k}], DESC)
14                     BLOOP:
15                     for each c_l in C
16                         img_l ← captured image(cv_{sequence[0]})
17                         if string(img_l = num_x) then
18                             path[ ] ← pop(c_l^loc)
19                         else
20                             break BLOOP:
21                             return path
22                         end for
23             end for
24         else
25             break ALOOP:
26             return path
27     end for
28 return path
```

Fig. 3. An Algorithm of real—time tracking system.

cam_id	time	cam_type	cam_lat	cam_lng	pic_link	car_num	cam_connect
1	2020-02-19 16:51:49	web_cam1	35.889147	128.610314	Z:/HDD1/cam1/img_000000.jpg	NULL	on
2	2020-02-19 16:51:49	web_cam2	35.894074	128.606904	Z:/HDD1/cam2/img_000000.jpg	NULL	on
3	2020-02-19 16:51:49	web_cam3	35.902965	128.618665	Z:/HDD1/cam3/img_000000.jpg	NULL	on
1	2020-02-19 16:51:52	web_cam1	35.889147	128.610314	Z:/HDD1/cam1/img_000001.jpg	NULL	on
2	2020-02-19 16:51:52	web_cam2	35.894074	128.606904	Z:/HDD1/cam2/img_000001.jpg	NULL	on
3	2020-02-19 16:51:52	web_cam3	35.902965	128.618665	Z:/HDD1/cam3/img_000001.jpg	NULL	on
1	2020-02-19 16:51:55	web_cam1	35.889147	128.610314	Z:/HDD1/cam1/img_000002.jpg	NULL	on
2	2020-02-19 16:51:55	web_cam2	35.894074	128.606904	Z:/HDD1/cam2/img_000002.jpg	AS60	on
3	2020-02-19 16:51:55	web_cam3	35.902965	128.618665	Z:/HDD1/cam3/img_000002.jpg	AR5LC2QE	on
1	2020-02-19 16:51:57	web_cam1	35.889147	128.610314	Z:/HDD1/cam1/img_000003.jpg	NULL	on
2	2020-02-19 16:51:57	web_cam2	35.894074	128.606904	Z:/HDD1/cam2/img_000003.jpg	AA5602PE	on
3	2020-02-19 16:51:57	web_cam3	35.902965	128.618665	Z:/HDD1/cam3/img_000003.jpg	IIYI	on
1	2020-02-19 16:52:00	web_cam1	35.889147	128.610314	Z:/HDD1/cam1/img_000004.jpg	NULL	on

Fig. 4. Database Monitoring.

간에 대해서만 탐색한다.

(6) 이미지 분석을 통해 vm_x를 탐색하지 못한 경우 탐색을 종료하고 최종 이동경로($path$)를 출력한다.

4. 차량 번호판 인식 구현과 평가

4.1 카메라 데이터 모니터링

본 시스템은 등록된 카메라에 대해 수집한 정보를 데이터베이스에 실시간으로 저장한다. Fig. 4은 데이터베이스에 저장된 데이터를 모니터링 하는 화면이다. 데이터베이스 테이블의 구조는 카메라 각각을 구분할 카메라 고유번호(cam_id), 각 카메라에서 영상을 분할하여 이미지를 저장한 시간($time$), 카메라 기기의 종류(cam_type), 카메라의 위도 좌표(cam_lat), 카메라의 경도 좌표(cam_lng), 분할 저장된 이미지의 경로(pic_link), 분석한 차량의 번호(car_num), 그리고 카메라의 연결 여부($cam\_connect$)로 구성되어있다.

카메라의 연결이 끊어졌을 때 다음과 같이 카메라의 연결 여부($cam\_connect$)가 "on"과 "off"로 데이터베이스에 저장됨에 따라, 실시간 카메라의 상태를 모니터링 할 수 있다. 그리고 데이터베이스의 과부하를 막기 위해 차량의 번호(car_num)에는 기본적으로 "$NULL$" 값을 저장한다. 찾고자 하는 차량이 생겼을 때 특정 카메라에 대한 이미지 경로(pic_link)만을 이용하여 차량의 번호(car_num)를 분석하며 탐색된 문자열이 있을 경우에 해당 문자열을 저장한다.

4.2 탐색할 이미지의 선택과 차량 이동경로 출력

시스템에 등록된 카메라는 실시간으로 촬영한 영상을 이미지로 분할하여 서버에 저장한다. 실험 환경은 Fig. 5와 같다. 위치가 고정된 카메라 3대(cam_1, cam_2,cam_3)를 이용하여 실험을 진행하였다. 카메라의 위치는 위경도 좌표 값을 이용하여 임의로 지정하였다. 또한 차량 번호판의 알파벳과 숫자는 KNN(K Nearest Neighbors) 알고리즘을 이용하여 학습시킨 후 인식하였다.

다음 Fig. 6는 탐색하고자 하는 카메라의 선택과 이미지 경로(pic_link)를 통한 차량의 번호(car_num) 분석을 모니터링 하는 그림이다. (a)에서 찾고자하는 차량의 번호판 번호 "$AA5602PE$"를 입력하면, 데이터베이스에 저장된 모든 카메라의 이미지 경로(pic_link)에 대해 탐색을 시작한다. 이미지 경로(pic_link)가 "$Z:/HDD1/cam2/img\_000003.jpg$"일 때 번호 "$AA5602PE$"를 인식하면, 탐색을 멈추고 이미지가 촬영된 시간(c^{t_2})과 현재시간의 비교를 통해 이동반경($moving\ radius$)을 계산한다.

(b)에서는 2번 카메라(cam_2)를 기준으로 이동반경($moving\ radius$) 내에 있는 카메라 중 거리가 가장 큰 카메라(cam_3)에 대해 탐색을 다시 시작한다. 이 때, 탐색은 Query의 조건문을 이용하여, 카메라 고유번호(cam_id)가 "3"이고 이미지를 저장한 시간($time$)이

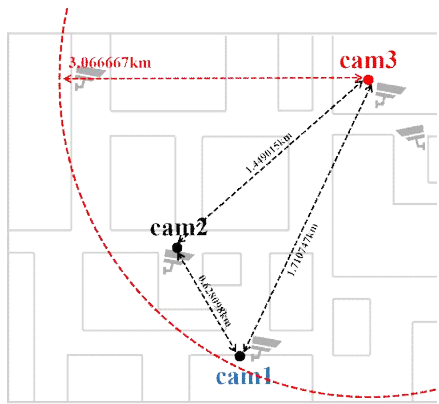

Fig. 5. Experiment Environment.

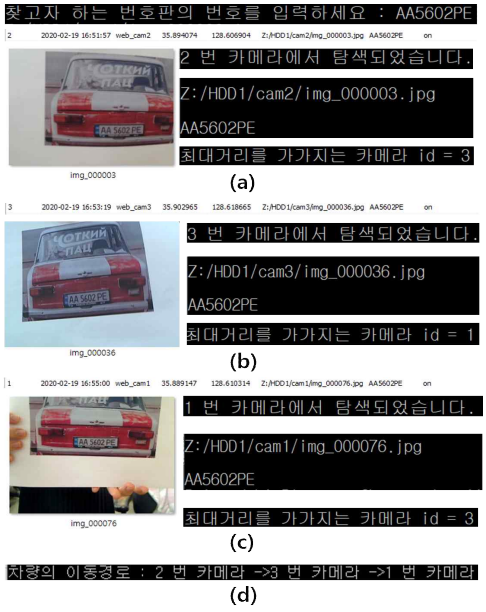

Fig. 6. Vehicle number recognition. (a)cam_2, (b)cam_3, and (c)cam_1.

"$>= c^{t_2}$" 인 조건을 만족하는 이미지 경로(pic_link)에 대해서 탐색을 한다. 따라서 차량이 c^{t_2} 시간 이후의 이동한 경로에 대해 탐색을 할 수 있다.

그리고 (c)에서는 3번 카메라(cam_3)를 기준으로 이동반경($moving\ radius$)을 구하고, 이동반경 내에 있는 카메라 중 거리가 가장 큰 카메라(cam_1)에 대해 탐색을 반복한다. 이 때, 탐색은 Query 조건문을 이용하여 카메라 고유번호(cam_id)가 "1"이고 이미지를 저장한 시간($time$)이 "$>= c_3^t$" 인 조건을 만족하는 이미지 경로(pic_link)에 대해서 탐색을 한다. 그리고 미리 지정한 경로의 개수(3)에 따라 최종경로를 출력하고 탐색을 멈춘다.

카메라를 탐색하여 지정해둔 경로의 개수(3)만큼 차량을 찾았거나 더 이상 차량을 탐색할 수 없을 때, 탐색을 멈추고 최종 경로를 반환한다. 차량을 탐색한 이미지의 경로(pic_link)를 이용하여 카메라의 고유번호(cam_id)를 찾고 최종 경로($path$)에 추가하여 (d)와 같이 차량의 이동경로를 카메라 고유번호(cam_id)를 이용하여 출력하였다.

4.3 차량 이동 경로 지도 구현

사용자가 직관적으로 차량의 이동경로를 확인하고 차량을 찾을 수 있도록 안드로이드 Google Maps API를 기반으로 지도를 구현했다. 차량번호를 인식한 데이터베이스를 탐색하여 json 형식으로 파싱한 데이터를 StringBuffer로 받아온 뒤 JsonArray의 key값(cam_lat, cam_lng)의 value 값을 이용하였다. 실험에 이용한 카메라의 위경도 좌표 값은 임의로 부여하였으며, 다음 Fig. 7의 (a), (b), (c)의 순서대로 차량이 탐색된 위경도 좌표의 위치를 Maps SDK for Android를 이용하여 마커(marker)로 나타내었다. 각 마커의 인포윈도우(InfoWindow)에는 차량을 탐색한 순서, 카메라의 고유번호(cam_id), 그리고 차량을 탐색한 시간($time$)을 표기하였다. 또한 차량의 이동을 마커를 연결한 Polyline 으로 간략하게 나타내었다.

4.4 평가

본 시스템은 사물인터넷 플랫폼에 등록된 카메라

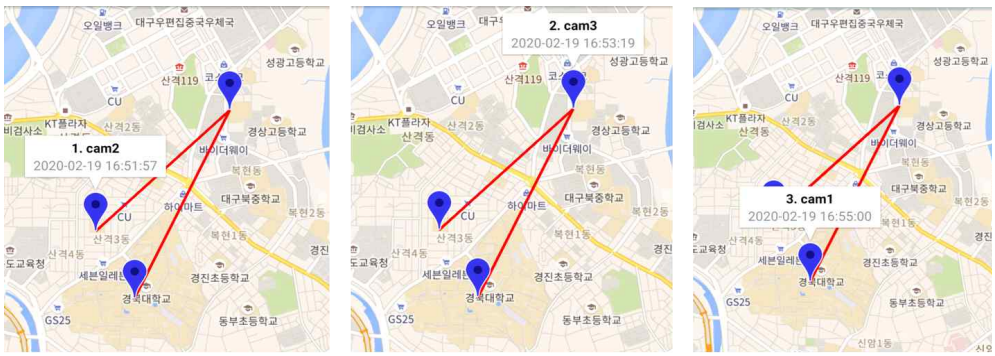

Fig. 7. Map API Implementation. (a)cam_2, (b)cam_3, and (c)cam_1.

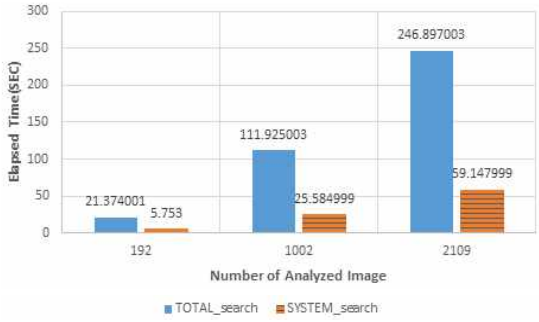

Fig. 8. Elapsed time Comparison.

기기를 사용하였으며, 제안하는 시스템은 특정 차량을 찾기 위해 데이터베이스에 저장된 카메라 기기의 정보를 이용한다. 또한 저장된 모든 정보를 이용하지 않고, 차량의 이동반경을 예측하고 탐색함으로써 차량을 탐색하는 시간을 단축시키는데 효율적임을 평가하기 위해 비교 실험을 하였다. Fig. 8은 데이터베이스에 저장된 모든 이미지를 탐색하였을 때(TOTAL_search)와 제안한 시스템을 이용해 탐색하였을 때(SYSTEM_serach)의 경과시간(ELAPSED TIME)을 비교하는 그림이다.

본 실험은 시스템에 등록된 카메라 3대를 이용하여 진행하였으며, 3분 간 촬영한 이미지(카메라 당 1분 간격) 192개와 15분 간 촬영한 이미지(카메라 당 5분 간격) 1002개, 그리고 30분 간 촬영한 이미지(카메라 당 10분 간격) 2109개를 이용하여 비교 실험을 하였다. SYSTEM_search를 이용한 탐색방법이 TOTAL_search 탐색방법보다 3분 간 촬영한 이미지는 약 3.8배, 15분 간 촬영한 이미지는 약 4.4배, 그리고 30분 간 촬영한 이미지는 약 4.2배 빠른 것으로 계산되었다. 따라서 실험 결과에 의해 이미지의 개수가 증가해도 SYSTEM_search를 이용한 탐색방법이 더 빠른 것을 확인할 수 있다.

5. 결 론

본 논문에서는 시스템에 등록된 카메라 기기를 그래프를 이용하여 정의하고, 기기의 관계를 가중치 값을 이용하여 나타내었다. 또한 가중치를 이용한 고유벡터 중심성 값을 통해 탐색하고자 하는 카메라를 선택하여 탐색함으로써 탐색의 효율을 높이고자 하였다. 또한 등록된 카메라의 기기정보와 영상정보를 실시간으로 데이터베이스에 저장한다. 이 때, 시스템의 부하를 줄이기 위해 카메라에서 분석한 차량의 번호는 실시간으로 분석하지 않는다. 찾고자하는 차량의 번호를 입력하면 탐색할 카메라를 선정하고, 선정된 카메라의 이미지 데이터만 분석하며, 분석할 데이터의 순서와 분석할 시간을 지정함으로써 분석할 데이터의 양을 줄일 수 있다. 또한 차량을 탐색한 카메라의 위치정보를 추출함으로써 차량의 이동경로를 예측하고, 지도상에 표시하여 모니터링 하는 사용자의 직관적인 이해를 돕는다.

제안하는 시스템을 통해 재난, 도난과 같은 여러 상황에서 공공 카메라(CCTV)에 촬영된 영상을 기반으로 주변 상황을 판단하고, 사용자에게 맞춤형 서비스를 제공함으로써 빠른 시간 내에 차량을 찾는데 도움을 줄 수 있다. 본 연구는 서버와 카메라 간 데이터 공유를 통해 차량을 추적하기 때문에 사용자의 개입이 필요 없고, 이동 반경 내 제한된 카메라에 대해 정보를 공유하기 때문에 분석을 최소화 한다. 추후에 이동성 있는 개인용 카메라(블랙박스 등)를 이용해 차량의 탐색에 이용할 수 있다.

REFERENCE

[1] M. K. Hossen, and S. H. Tuli, “A surveillance system based on motion detection and motion estimation using optical flow,” *IEEE 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 646-651, 2016.

[2] Y. Seungho, and K. Jaemin, “Realtime Vehicle Tracking and Region Detection in Indoor Parking Lot for Intelligent Parking Control,” *Journal of Korea Multimedia Society*, Vol. 19, No. 2, pp. 418-427, 2016.

[3] D. K. Yadav, L. Sharma, and S. K. Bharti, “Moving object detection in real-time visual surveillance using background subtraction technique,” *IEEE 14th International Conference on Hybrid Intelligent Systems*, pp. 79-84, 2014.

[4] Z. Chen, T. Ellis, and S. A. Velastin, “Vehicle detection, tracking and classification in urban traffic,” *15th International IEEE Conference on Intelligent Transportation Systems*, pp.

951-956, 2012.

[5] Pokale, A. Tushar, D. T. Ingole, and M. D. Ingole, "A review on real time integrated CCTV system using face detection for vehicle seat vacancy identification with image processing technique," *International Journal on Recent and Innovation Trends in Computing and Communication*, Vol. 6, No. 7, pp. 121-123, 2018.

[6] S. Yang, C. Choi, and H. Choi, "Design and Implementation of Vehicle Route Tracking System using Hadoop-Based Bigdata Image Processing," *Journal of Digital Contents Society*, Vol. 14, No. 4, pp. 447-454, 2013.

[7] A. Paul, "Graph based M2M optimization in an IoT environment," *In Proceedings of the 2013 Research in Adaptive and Convergent Systems*, pp. 45-46, 2013.

[8] C. Caraffi, T. Vojíř, J. Trefný, J. Šochman, and J. Matas, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," *15th international IEEE conference on intelligent transportation systems*, pp. 975-982, 2012.

[9] H. Lee, D. Kim, D. Kim, and S. Y. Bang, "Real-time automatic vehicle management system using vehicle tracking and car plate number identification," *IEEE International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No. 03TH8698)*, Vol. 2, pp. II-353, 2003.

[10] S. L. Chang, L. S. Chen, Y. C. Chung, and S. W. Chen, "Automatic license plate recognition," *IEEE transactions on intelligent transportation systems*, Vol. 5, No. 1, pp. 42-53, 2004.

[11] M. Betke, E. Haritaoglu, and L. S. Davis, "Real-time multiple vehicle detection and tracking from a moving vehicle," *Machine vision and applications*, Vol. 12(2), pp. 69-83, 2000.

[12] D. M. Babu, K. Manvitha, M. S. Narendra, A. Swathi, and K. P. Varma, "Vehicle tracking using number plate recognition system," *International Journal of Computer Science and Information Technologies*, Vol. 6, No. 2, pp. 1473-1476, 2015.

[13] Y. Y. Chen, J. R. Chen, L. H. Chang, and S. C. Hung, *U.S. Patent No. 9,761,135. Washington, DC: U.S. Patent and Trademark Office*, 2017.

[14] S. Misra, R. Barthwal, and M. S. Obaidat, "Community detection in an integrated Internet of Things and social network architecture," *In 2012 IEEE Global Communications Conference*, pp. 1647-1652), 2012.

[15] D. Le-Phuoc, H. N. M. Quoc, H. N. Quoc, T. T. Nhat, and M. Hauswirth, "The Graph of Things: A step towards the Live Knowledge Graph of connected things," *Journal of Web Semantics*, Vol. 37, pp. 25-35, 2016.

[16] Geographical_distance, https://en.wikipedia.org/wiki/Geographical_distance.

김 선 형

2018년 8월 경북대학교에서 학사학위 취득하였으며, 2018년 9월부터 현재까지 경북대학교 컴퓨터학부에서 석사과정 중임. 관심분야는 모바일 컴퓨팅, 사물 인터넷, 빅데이터 등 임.

김 상 욱

1979년 2월 경북대학교에서 전자계산기공학으로 학사학위 취득, 1981년 2월 서울대학교에서 컴퓨터과학으로 석사학위취득, 1989년 2월 서울대학교에서 컴퓨터과학으로 박사학위를 취득하였다. 1982년부터 경북대학교 IT대학 컴퓨터학부 교수로 재직 중임. 관심분야는 모바일 미디어, 소셜 미디어, 인간과 컴퓨터의 상호작용, 사물 인터넷 등 임.