The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 20, No. 1, pp.69-75, Feb. 29, 2020. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2020.20.1.69
JIIBC 2020-1-9

Open CV를 기반으로 한 주차 여유 공간 관리 시스템

Management System for Parking Free Space based on Open CV

남은주*, 안덕규*, 서유진*

Eun-Joo Nam*, Deouk-Kyi An*, You-Jin Seo*

요 약 본 논문에서는 번화가나 관광지 등 주차공간의 수요가 높은 지역에서 주차하면서 발생할 수 있는 불편함을 해결하기 위해 개발한 주차안내 서비스를 소개한다. 실제 주차장을 측정하고 자동차를 운전하면서 개발하는 데 어려움이 있어, 임시 주차장을 만들고 Arduino RC Car를 만들어서 실제 자동차를 대신하였다. Open CV를 기반으로 한 영상처리로 전체 주차장, 주차 가능 공간, 주차 완료된 공간을 확인하고 움직이는 자동차를 추적할 수 있도록 하였으며, 이 정보를 사용자가 애플리케이션을 통해 확인할 수 있도록 개발하였다. 사용자는 애플리케이션을 통해 원하는 주차공간을 예약할 수 있으며, 길 찾기 알고리즘을 도입하여 선택한 주차 칸까지의 최적 경로를 안내받을 수 있다.

Abstract This paper introduces the parking guide service developed to address the inconvenience of parking in areas where demand for parking spaces is high, such as busy streets and tourist attractions. Due to difficulties in measuring and developing the actual parking lot while driving the car, we created a temporary parking lot and created Arduino RC Car to replace the actual car. Video processing based on Open CV allows users to identify the entire parking lot, parking space, and completed parking space, and track moving cars, and this information has been developed to enable users to see through the application. The application allows the user to book the desired parking space and introduce a way-finding algorithm to guide them through the optimal path to the selected parking compartment.

Key Words : Open CV, Android, Arduino, Shortest-path algorithm

I. 서 론

국토교통부의 통계에 따르면 해마다 국내 자동차 등록 대수는 증가하고 있다. 그러나 번화가나 관광지 등 자동차 포화 지역에서의 주차공간 안내 서비스가 부족한 실태이다. 공용 주차장, 혹은 사설 주차장에 카메라를 설치하고 영상처리(Open CV)를 기반으로 주차 여유 공간 확인 프로그램을 구현하여 사용자들에게 효율적으로 주차할 수 있도록 도와주고자 한다. 카메라로 촬영한 주차공간을 영상처리(Open CV)를 통해 왜곡 보정을 시키고, 사용자 앱과 연동해서 실시간 차량 위치와 주차 현황을 보여준다. 이 프로그램으로 인해 주차장에 들어가기 전에 미리 여분 주차 칸의 수를 알 수 있고, 사용자가 원하는 주차공간을 예약하고, 원하는 주차 칸까지의 최적 경로를 알려주어 효율적으로 주차하기를 기대한다.

*학생회원, 한국산업기술대학교 컴퓨터공학과
접수일자: 2019년 9월 27일, 수정완료: 2020년 1월 12일
게재확정일자: 2020년 2월 7일

Received: 27 September, 2019 / Revised: 12 January, 2020 / Accepted: 7 February, 2020
*Corresponding Author: nejoo97@naver.com
Dept. of Computer Engineering, Korea Polytechnic University, Korea.

Ⅱ. 관련 연구

1. 다익스트라 알고리즘(Dijkstra Algorithm)

다익스트라 알고리즘은 하나의 정점에서 다른 모든 정점까지의 최단 경로를 구하는 알고리즘이다. 알고리즘의 기본 로직은 첫 정점을 기준으로 연결되어있는 정점들을 추가해가며 최단 거리를 갱신하는 것이다. 본 논문에서는 무작위 유향 그래프에서의 단일 소스 최단 경로 알고리즘 중 점근적으로 가장 빠른 알고리즘이라는 장점에 주목하여 활용하였다.

2. 색 검출 알고리즘

본 연구에서는 촬영된 원본 사진의 RGB 값을 HSV로 변환한다. RGB 색 공간은 색을 빨강, 초록, 파랑의 조합으로 표현하는 것이다. 반면 HSV 색 공간은 우리가 보는 그대로의 색을 Hue 채널로, 색이 진하고 연한 정도를 Saturation 채널로, 그리고 밝기를 Value로 결정하기 때문에 RGB보다 HSV가 직관성이 더 좋다. 이 프로젝트에서 필요한 것은 색깔을 통해 주차장 전체를 검출해내는 것이므로 HSV가 적절하다고 판단하여 변환해준다.

3. 왜곡 보정 알고리즘

카메라로 전체 주차장을 찍으면 사다리꼴 형태이다. 이를 직사각형과 유사한 모양으로 바꾸기 위해 왜곡 보정 알고리즘이 필요하다. 이는 각 주차 칸을 라벨링 할 때 좀 더 정확하게 인식하게 하기 위함이다. 이 알고리즘의 원리는 다음 표 1과 그림 1~4와 같다.

표 1. 단계별 왜곡보정 단계

1단계	전체 주차장 사진(원본)을 찍는다.
2단계	원본을 흑백화(그레이스케일) 시킨다.
3단계	① 윗변의 왼쪽, 아랫변의 왼쪽을 찾는 경우
	해당 지점을 기준으로 위(+15)에서부터 아래(-15)까지 왼쪽에서부터 픽셀값을 탐색하면서 흰색 점이 나오는 지점의 좌표를 저장하여 기울기 변화를 통해 급격하게 상승/하강하는 좌표 저장
	② 윗변의 오른쪽, 아랫변의 오른쪽을 찾는 경우
	해당 지점을 기준으로 위(+15)에서부터 아래(-15)까지 오른쪽에서부터 픽셀값을 탐색하면서 흰색 점이 나오는 지점의 좌표를 저장하여 기울기 변화를 통해 급격하게 상승/하강하는 좌표 저장
4단계	저장된 각 모서리의 좌표를 원근 변환 행렬 계산식에 대입해서 변환 좌표를 계산한다.

그림 1. 원본사진
Fig. 1. Original Image Sample

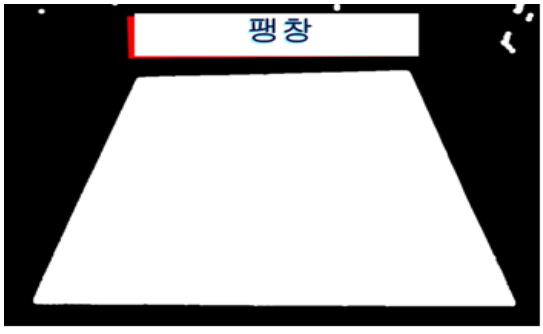


그림 2. 모폴로지

그림 3. 왜곡시킬 점 탐색
Fig. 3. Exploration of distorting points

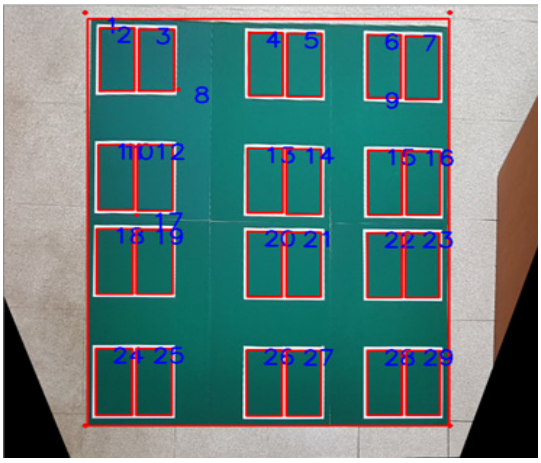


그림 4. 왜곡보정 이후 주차칸 탐색
Fig. 4. Exploration Parking Lot after Distortion Compensation

Ⅲ. 본 문

1. 개발환경

본 논문은 Window 10에서 개발되었으며 Visual Studio 2015, Open CV 버전 3.1.0, Android 버전 3.1.2, Arduino 등의 개발 툴을 사용하였다. 하드웨어로는 아두이노에서 Uno 3을 사용하였고 안드로이드용 단말기로는 삼성의 갤럭시 10, 갤럭시 S8+, 갤럭시 노트 8과 갤럭시 노트4를 사용하여 최대한 버전 간의 차이를 완화하려 하였다.

2. 시스템 구성

본 연구의 기본적인 시스템 시나리오는 영상처리 시스템이 카메라로부터 받은 주차장 영상에서 데이터를 추출, 수집, 정제하여 웹 서버로 전달한다. 웹 서버는 애플리케이션이 원하는 정보를 제공하고 사용자의 요청과 편의에 맞추어 데이터를 저장·수정한다. 애플리케이션은 웹 서버로부터 받은 데이터를 UI에 표현하여 사용자가 보기 쉽게 나타내고, 빈 주차 칸의 유무, 길 찾기 등의 서비스를 제공한다. 아래 그림 5는 개략적인 시스템 구성을 나타낸다.

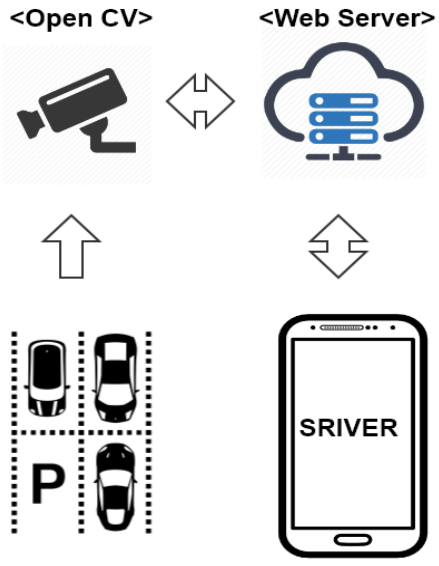


그림 5. 시스템 구성도
Fig. 5. System Diagram

3. 아두이노(Arduino)

가. 시연용 RC Car 제작

본격적인 개발에 앞서 시연 및 테스트용 RC Car를 제작하였다. 메인 보드로는 ATmega 328 기반의 UNO R3를 사용하였으며, RC Car 조종을 위한 블루투스 통신 모듈과 두 개의 모터 모듈로 이루어져 있다. 이후 위와 같은 구성의 자동차를 한 대 더 제작하였다. 아래는 RC Car를 구동하기 위해 개발 내용이다.

나. 조종을 위한 아두이노 및 안드로이드 개발

안드로이드에서는 RC Car와의 블루투스 통신을 통해 좌우 회전, 직·후진, 정지 신호를 전달한다. 아두이노에서는 안드로이드로부터 받은 신호에 따라 모터 모듈에 LOW, HIGH 신호를 주며 바퀴의 회전과 모터의 동작을 관리한다. 아래의 표 2는 RC Car의 좌회전에 관련된 코드를 일부 가져온 것이다.

표 2. RC Car 방향 관련 모터
Table 2. Motor about RC Car Direction

아두이노(Arduino)

void Left(){
digitalWrite(IN1,LOW); digitalWrite(IN4,LOW);
digitalWrite(IN2,HIGH); digitalWrite(IN3,HIGH);
digitalWrite(IN5,LOW); digitalWrite(IN8,LOW);
digitalWrite(IN6,HIGH); digitalWrite(IN7,HIGH);
}

4. Open CV(영상처리)

본 연구에서는 개발할 때 실제 주차장에서 자동차를 운전하면서 테스트하기 어려운 점이 있어서 하드보드지로 주차장 세트를 임시로 만들고 자동차는 Arduino RC Car로 대체해서 진행하였다.

이 프로젝트에서 Open CV가 필요한 2가지 상황은 다음과 표 3과 같다.

표 3. Open CV 호출 조건
Table 3. Open CV Call Condtion

①	초기 주차장 데이터를 웹 서버로 전송하는 경우
②	주차장 데이터를 실시간으로 웹 서버로 전송하는 경우

가. 색 검출

원본 사진(그림 1 참조)에서 초록색 주차장 영역 중 한 지점을 클릭해서 HSV 값으로 변환한 후 그 값과 오차 범위 내에 있는 모든 픽셀을 한 번에 그레이스케일(gray-scale)시키고 모폴로지 침식과 팽창을 적용한다. 그러면 그림 2(2page 참조)와 같은 결과가 나오고 가장 큰 흰색 부분을 주차장이라고 판단한다.

나. 왜곡 보정(Distortion)

본 연구에서는 왜곡 보정을 2번 적용한다. 내용은 표 4와 같다.

표 4. 오차를 줄이기 위한 2차 왜곡보정
Table 4. Secondary Distortion Compensation to reduce errors

1차 왜곡 보정	2차 왜곡 보정
·주차장 아닌 부분 제거 ·사다리꼴을 직사각형과 유사하게 변형 ·그림 1 → 그림 4	·주차 칸 위치의 오차를 줄이기 위해 모서리 부분의 주차 칸 위치로 왜곡 보정

1차 왜곡 보정에서는 사다리꼴 윗변과 아랫변의 길이가 비슷해지도록 윗변을 늘렸다고 볼 수 있다. 이 상태로 사용자 앱에 주차장 데이터를 전송하게 되면 같은 층, 같은 줄의 주차 칸의 위치의 오차가 커지므로 정확한 결과를 얻을 수 없게 된다. 따라서 2차 왜곡 보정을 적용해서 주차 칸들의 오차율을 줄이는 작업을 수행해야 한다.

다. 라벨링(Labelling)과 데이터 전송

라벨링은 주차장 왜곡 보정이 수행된 후에 실행한다. 각 주차 칸들을 빨간색 사각형으로 라벨링(Labelling)하여 각 주차 칸의 좌표 및 크기를 웹 서버로 전송한다. 이때 0번 인덱스에는 제일 큰 라벨링(주차장 세트)의 크기가 전송된다.

라. Tracking 차량 추적

차량 추적에서는 카메라로 촬영을 하면서 전 frame과 현재 frame을 비교해서 그레이 스케일(gray-scale)을 하는 Threshold를 이용해 움직임이 있으면 라벨링을 한다.

움직이고 있는 자동차들의 라벨링 영역과 Open CV에 저장된 차량 객체와의 Mapping 작업을 완료 후, Mapping 작업으로 인한 delay(시간 차이)를 이용해, 전 frame에 현재 frame을 저장한다. 이때 발생할 수 있는 차량 상태는 아래 표 5와 같다.

표 5. RC Car의 추적
Table 5. Tracking of RC Car

Open CV의 차량 좌표와 Threshold의 차량 좌표를 비교	
①	자동차 객체 좌표가 전 frame에서 주차장 밖에 있고 현재 frame에서는 안에 있는 경우
②	자동차 객체 좌표가 전 frame에서 주차장 안에 있고 현재 frame에서는 밖에 있는 경우
③	전 frame과 현재 frame 둘 다 좌표가 주차장 안에 있는 경우
④	전 frame과 현재 frame 둘 다 좌표가 주차장 밖에 있는 경우

표 5에 나온 경우의 수는 사용자의 Flag 값을 이용해서 어떤 경우인지 판단한다. 사용자들이 앱을 사용할 때, 회원가입만 하고 로그인은 하지 않았거나 차량은 들어와 있는데 로그아웃을 한 경우 등 다양한 상황이 존재할 수 있다. 이러한 상황이 발생해도 이 시스템에서는 어떠한 차량의 상태도 놓치면 안 되므로, Flag 값을 사용해서 추적해야 하는 차량의 상태를 데이터베이스에 저장해 놓는다. 표 5에서 ①은 자동차가 새로 들어오는 상황이다. 주차장에 새로 들어온 차량이 있다면 자동차 객체를 생성

하고 Flag 값 2를 주어 차량을 계속 추적한다. 표 5의 ②는 차량이 주차장을 빠져나가는 상황이고, 차량이 완전히 나가면 Flag 값을 0으로 바꿔준다. 표 5의 ③은 차량이 주차장 내에서 이동 중인 상태이고, ④는 차량이 아예 들어오지 않았거나 이미 빠져나간 후인 경우이다. 위와 같은 시스템으로 차량을 실시간으로 추적한다.

표 6. 추적 Flag
Table 6. Tracking Flag

Flag	상태
2	주차장에 새로 들어온 차량이 있고, 계속 추적하는 상태
1	차량 객체를 생성하지 않은 상태
0	차량이 주차장을 나간 상태

5. 사용자용 안드로이드 애플리케이션

본 애플리케이션은 웹 서버로부터 받은 데이터를 기반으로 사용자에게 주차안내 서비스를 제공하기 위해 제작되었고 'SRIVER'라고 명명하였다.

가. 스레드(Thread)를 통한 데이터 동기화

스레드(Thread)를 통해 일정 시간 주기적으로 비동기 작업 (Asynchronous Task)를 호출하여 웹 서버의 데이터와 'SRIVER'의 데이터를 동기화 및 처리한다. 이를 통해 애플리케이션이 사용자에게 제공하는 실시간 서비스(Realtime Service)를 전반적으로 관리한다.

나. 데이터 정렬(Sort) 및 보정

초기 'SRIVER'이 실행될 때 웹 서버에서 받은 주차 칸 데이터들은 무작위로 라벨링 되어있기 때문에 x 좌표와 y 좌표 기준으로 정렬한다. y 좌표를 기준으로 먼저 정렬을 진행하고 n 칸과 n+1칸의 시작 y 좌표의 차이가 평균 주차 칸 높이의 50%보다 작다면 같은 층이라고 판단한다. 이후 주차 칸 높이 평균으로 모든 주차 칸들의 높이를 보정 한다. 아래 그림 6과 같다.

같은 방법으로 k 층을 선택하여 시작 x 좌표로 정렬한 후, 주차 칸들의 최종 정렬 결과를 ParkingPoint ArrayList에 저장한다. 그리고 각 층의 x, y 좌표를 해당 층, 줄의 평균값으로 보정 한다.

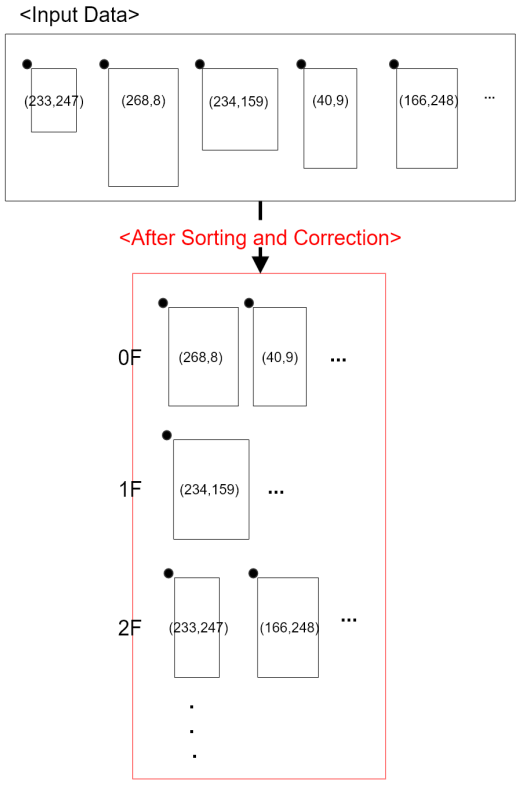


그림 6. App 출력 좌표 보정
Fig. 6. Correction of App Output Coordinate

다. 교차로 탐색(Search)

위에서 구현한 ParkingPoint ArrayList의 k 층에서 n 칸과 n+1칸의 시작 x 좌표의 차이가 평균 주차 칸 넓이의 1.5배 이상이라면 길(Road)이 있다고 판단한다. y 축에도 동일하게 적용한다. 이후 rposX와 rposY ArrayList에 주차 칸을 1, 길(Road)을 0으로 저장한다. 아래 표 7은 의사코드이다.

표 7. App 교차로 탐색 코드
Table 7. App Crossroad Exploration Code

```
difference = |n.startX - (n+1).startX|
if difference 〉 averageWidth * 1.5
    rposX.add(0)
else
    rposX.add(1)
```

두 posX, posY ArrayList에서 '101'이 연속되는 구간을 찾고 구간이 중복되는 지점을 찾아 교차로 지점(CrossPoint)을 지정하고 교차점을 저장하는 CrossPoint ArrayList 를 구현한다. 아래 그림 7과 같다.

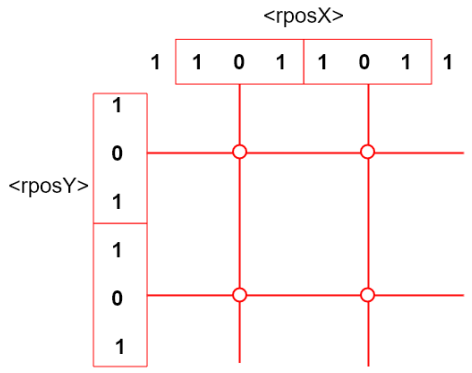


그림 7. 교차로 탐색 알고리즘
Fig. 7. Crossroad Exploration Algorithm

라. 길 찾기 서비스 구현

최단 경로를 탐색하기 위해 총 노드의 구성과 개수는 아래의 표 8과 같다.

표 8. 길 찾기 알고리즘
Table 8. Tracing the route Algorithm

시작지점 + 모든 교차점(CrossPoint)(n개)+ 도착점 = n+2개

시작지점이 인덱스(Index) 0이 되며 도착지점이 인덱스 n+1이 되고, 시작지점과 도착지점의 위치에 따라 아래의 표 9로 나뉜다.

표 9. 길 찾기 경우의 수

①	시작지점과 도착지점이 교차점 사이에 있는 경우
	가장 가까운 두 교차점에 연결을 해제하고 해당 지점과 두 교차점을 연결
②	시작지점과 도착지점이 교차로를 벗어난 경우
	가장 가까이에 있는 교차점을 탐색

이후 다익스트라 알고리즘을 이용해 시작지점(0) → 도착지점(n+1) 노드까지의 패스(Path), 즉 꼭 지나야 하는 교차점들을 구하고 그 점들을 경로라고 인식하여 드로우(Draw) 함수에서 좌표로 변환해주고 라인(Line)을 표시해서 사용자에게 최종적인 길 찾기 서비스를 제공한다.

6. 구현 결과

영상처리 부분에서는 주차장 정보를 받아오며 빈 주차칸과 비어있지 않은 주차 칸을 판별하고 움직이는 차량을 추적(Tracking)할 수 있도록 하였다. 안드로이드 부분에서는 기본적인 로그인, 회원가입과 자차의 위치를 확인하며 원하는 목적지까지 길 찾기를 해주는 서비스까지 구현되었다. 한 주차 칸에서 5초 이상 움직임이 없는 경우, 자동으로 주차 인식이 되도록 구현하였다. 아래 그림 8, 9, 10은 실제 구현된 프로그램과 애플리케이션의 모습이다.

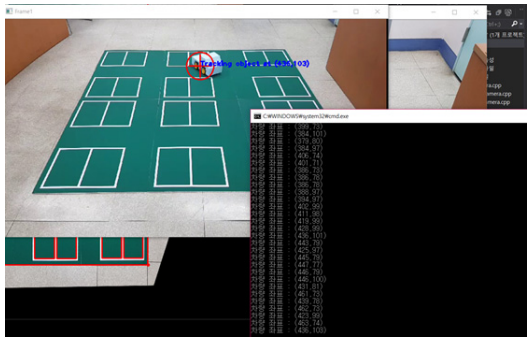


그림 8. 추적영상
Fig. 8. Tracking Picture

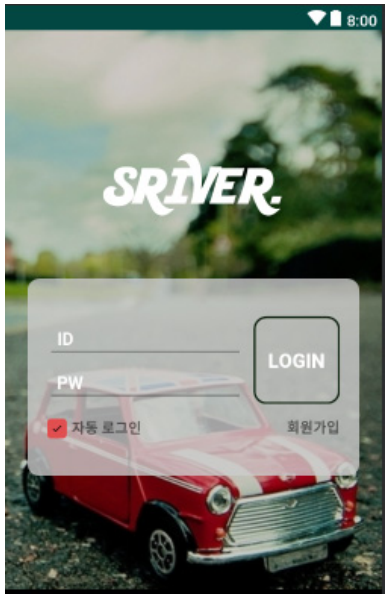


그림 9. App 초기화면
Fig. 9. Start-up Screen of App

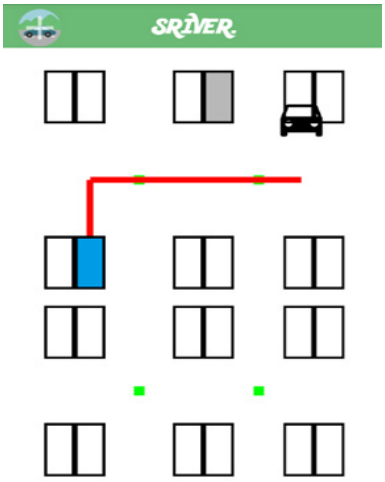


그림 10. App 실행화면
Fig. 10. Execution Result Screen

IV. 결 론

본 논문에서는 자동차의 이용 대수가 점점 증대하는 추세에 맞추어 더 경제적이고 상용화되기 쉬운 매체를 통해 주차안내 서비스를 개발하였다. 수십 대, 크게는 수백 대의 센서가 필요한 기존의 주차안내 서비스의 번잡스러움과 경제적 비효율성에 주목하여 대중적이고 용이성 있는 카메라를 이용하였고 영상처리를 통해 이미지에서 데이터를 얻어 애플리케이션으로 사용자에게 가시화하였다.

본 시스템은 반복적인 테스트를 통해 예측 가능한 대략적인 오차와 오류들을 영상처리, 애플리케이션 드로우 단계에서 수정 및 예외처리 하였다. 이 외에도 많은 테스트를 통해 더 세부적이고 다양한 환경에서의 오류처리 프로세스가 구축될 것으로 기대된다.

References

[1] Ho-sik Park and Cheol-soo Bae, An Efficient Vehicle Parking Detection Method Using Gray Scale Images. KICS, vol.36, no.10, pp.629-634, 2011

[2] Gary Bradski and Adrian Kaehler, Learning OpenCV (Computer Vision with the OpenCV Library), O'Reilly Media, 2008

[3] Ji-Hoon Jo and Sang-Gu Lee, PTZ Camera Based Real-time Object Tracking System Using OpenCV, KKITS, vol.7, no.1, pp.59-65, 2012

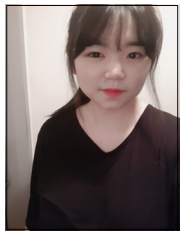
[4] Gray R and Bradski, Computer Vision Face Tracking For use in a perceptrual User Interface, Microcomputer Reseach Lab, 2002

[5] Choi Jinhae, An Examination of an Efficient UI of Smartphone Home Screen Structure, J Ergon Soc Korea, vol.36, no.5, pp. 437-446, 2017

[6] Dijkstra's algorithm https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

저 자 소 개

남 은 주(준회원)



- 한국산업기술대학교 4학년
- 본 연구팀장
- 영상처리 담당

안 덕 규(준회원)

- 한국산업기술대학교 4학년
- 본 연구팀원
- 안드로이드, 아두이노, 영상처리 담당

서 유 진(준회원)

- 한국산업기술대학교 4학년
- 본 연구팀원
- 안드로이드, 아두이노 담당

※ 본 연구는 한국산업기술대학교 졸업작품으로 수행된 연구결과임.