남서울대학교 정보통신공학과 졸업작품 상세설명서

유동인구 통계 시스템 개발

Development of Floating Population Statistics System

Department of
Information and Communication Engineering

Namseoul University

Revision History

Revision	Description
1.00	조장: 김우주, 조원: 고유영, 윤용채, 이고은

목 차

1. 개발환경 구축

1.1 하드웨어 개발환경

□ 웹캠: 화상 카메라 (모델명 : 로지텍 C920 PRO)

ㅇ 출처:

http://www.ssg.com/item/itemView.ssg?itemId=1000027103640&siteNo=6001&salestrNo=6001&ckwhere=ssg_gshopsa&appPopYn=n&EKAMS=google.723.6063.21300.2032931.611081089&trackingDays=1&gclid=Cj0KCQjwkK_qBRD8ARIsAOteukDvhjsizzJjJIedC_AHql_0j8a32KCncvAe-qsLX0Z-eBY5jube4ckaAsFkEALw_wcB

<그림 1. Webcam>

□ 웹캠 삼각대: 삼각대

○ 출처:

http://www.11st.co.kr/product/SellerProductDetail.tmall?method=getSellerProductDetail&prdNo=2099179709&trTypeCd=21&trCtgrNo=585021&lCtgrNo=8286&mCtgrNo=1020764

<그림 2. 삼각대>

□ 데스크톱 PC: 데스크톱 PC 및 주변기기

○ Nvidia GPU - GTX 1050이상
○ Anconda - Python3.6과 CUDA Toolkit - V9.0, CuDNN - V7.05 탑재된 데스크톱 PC
○ 출처: 대학원 데스크톱 PC

<그림 3. 데스크톱 PC 및 주변기기>

1.2 소프트웨어 개발환경

□ 웹페이지

ㅇ 개발 도구 : Pycharm
- https://www.jetbrains.com/pycharm/

ㅇ 개발 언어 : Javascript, Python, Flask, CSS, HTML

□ 객체 인식

ㅇ 개발 도구 : Pycharm
- https://www.jetbrains.com/pycharm/

ㅇ 개발 언어 : Python

1.3 오픈 솔루션(회로도, 소스코드 등) 분석

□ 출처: https://github.com/ayooshkathuria/pytorch-yolo-v3

□ 소스코드 분석 : 주석처리

ㅇ 딥러닝(Deep Learning) 기반의 객체 인식

```
if __name__ == '__main__':
cfgfile = "cfg/yolov3.cfg"
weightsfile = "yolov3.weights"
num_classes = 80

start = 0
CUDA = torch.cuda.is_available()

num_classes = 80
bbox_attrs = 5 + num_classes #최적의 딥러닝(Deep Learning) 설정값

model = Darknet(cfgfile)
model.load_weights(weightsfile)

cap = cv2.VideoCapture(0)

assert cap.isOpened(), 'Cannot capture source' #webcam이 연결되지 않았을 시 문구

frames = 0
start = time.time()
while cap.isOpened():
ret, frame = cap.read()
if ret:
img, orig_im, dim = prep_image(frame, inp_dim)
if CUDA:
im_dim = im_dim.cuda()
img = img.cuda()
output = model(Variable(img), CUDA)
output = write_results(output, confidence, num_classes, nms = True, nms_conf =
nms_thesh)
if type(output) == int:
frames += 1
cv2.imshow("frame", orig_im)
key = cv2.waitKey(1)
if key & 0xFF == ord('q'):
break
continue

output[:,1:5] = torch.clamp(output[:,1:5], 0.0, float(inp_dim))/inp_dim
im_dim = im_dim.repeat(output.size(0), 1)
output[:,[1,3]] *= frame.shape[1]
output[:,[2,4]] *= frame.shape[0]
classes = load_classes('data/coco.names') #라벨을 coco.names 파일에서 라벨 data를 load한다.
colors = pkl.load(open("pallete", "rb"))

list(map(lambda x: write(x, orig_im), output)) # list를 출력한다.

cv2.imshow("frame", orig_im)
key = cv2.waitKey(1)
if key & 0xFF == ord('q'): #종료 key q
break
frames += 1
print("FPS of the video is {:5.2f}".format( frames / (time.time() - start))) #시간기록

else:
break
```

□ Open CV 및 Pytorch 기반의 Yolo V3 객체 인식 딥러닝 모델

<그림4. 객체 인식 결과>

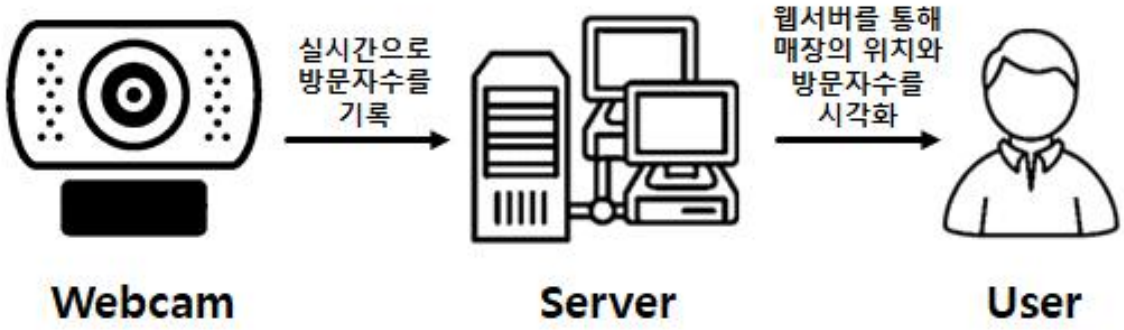
- Pytorch를 사용하여 빠른 객체 탐지 알고리즘 중 하나인 Yolo v3를 기반으로 객체 탐지기를 구현

- Yolo v3는 모든 셀에 대해 3개의 경계 상자를 예측한 후 경계 상자 안 객체를 분석

- coco database에 기반을 둬 분석된 객체에 라벨을 부여

2. 제안 시스템 및 핵심기술

2.1 서비스 모델



< 그림 5. 서비스 모델 >

□ 그림 8은 웹캠을 이용하여 실시간으로 매장의 방문자수를 기록하고 통계한 분석데이터를 제공함으로써 소상공인들이 효율적으로 매장을 운영할 수 있도록 지원하려는 목적으로 구현한 모델임

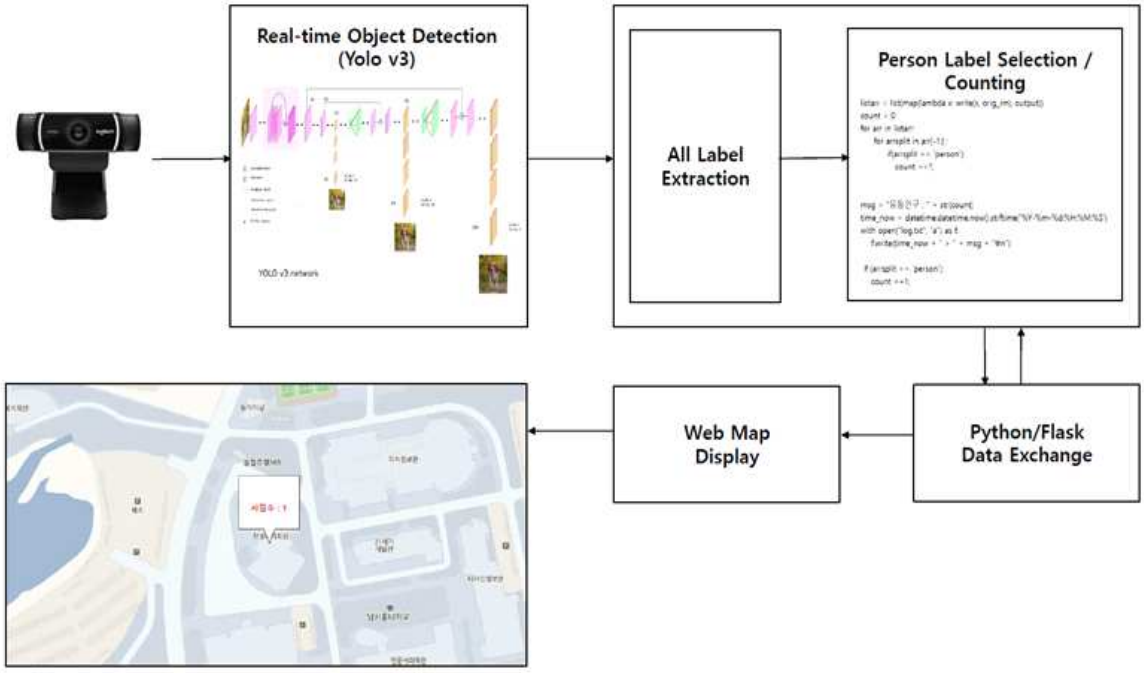
□ 객체(사람)를 개별적으로 인식하기 위해서 Pytorch 기반의 Yolo v3 딥러닝 모델을 사용하여 일반적인 객체 검출 알고리즘 보다 빠른 속도로 정확하게 객체를 인식함

- ㅇ 객체 인식 기술은 의료분야에서 질병 식별과 산업 검사 및 로봇비전 등과 같은 기술로 활용할 수 있고, 특히 무인 자동차에서 가장 많이 활용되는 핵심 기술로 자동차가 신호를 인식하고 보행자와 장애물을 구별할 수 있도록 함

- ㅇ 방문자수에 따라 분위기(조명)와 환경(온도, 습도) 조절 서비스, 대기시간 알림 서비스에 활용될 수 있으며 매장 운영시간 종료 후 외부인 침입 감지 서비스 등으로 활용될 수 있음

- ㅇ 웹 서버를 통해 매장의 위치를 함께 제공함으로써 매장 주변의 상권 분석 서비스에도 활용될 수 있음

2.2 시스템 모델



< 그림 6. 시스템 모델 >

□ 시스템 구성

- 웹캠을 이용해 검출된 사람의 수를 flask 웹서버를 활용해 만든 웹 페이지로 모니터를 통해 확인 할 수 있음

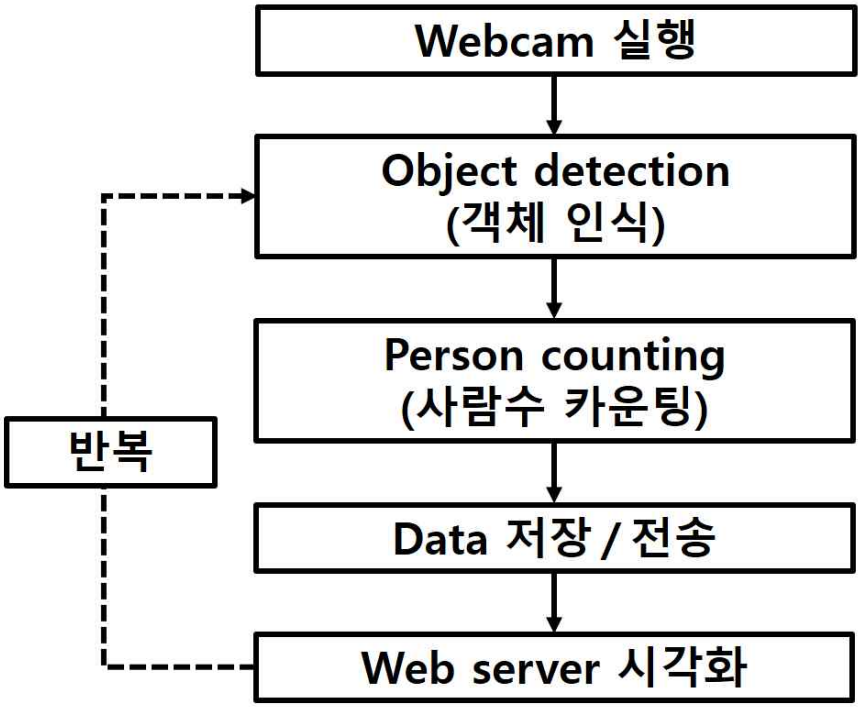
- 웹캠으로 찍은 이미지를 Pytorch 기반의 Yolo v3 딥러닝 모델을 사용하여 사람만을 객체로 검출함

- Yolo (You Only Look Once)는 이미지 내의 bounding box와 class probability를 single regression problem으로 간주하여, 이미지를 한 번 보는 것으로 object의 종류와 위치를 추측하는 알고리즘임

- Flask는 매우 가볍고 심플한 프레임워크(framework)로 사용자가 필요한 부분을 따로 추가하면서 자신의 필요에 맞게 프레임워크를 만들 수 있음 (자유도가 높음)

- Ajax는 웹에 존재하는 DHTML, CSS, XML, XMLHttpRequest등의 기술들을 합친 새로운 기술로 원하는 데이터를 페이지에 보여주기 위해 서버에 요청을 한 후 멈춰 있는 것이 아니라 부분적으로만 로딩을 통해 빠른 속도로 페이지 이동 없이 고속으로 화면을 전환 할 수 있음

2.3 핵심 솔루션



< 그림 7. 플로우 차트 >

□ 개발한 유동인구 통계 시스템의 상세설명은 다음과 같음

- ㅇ 웹캠을 이용해 실시간으로 객체를 검출
- ㅇ 검출된 사람을 person이라는 라벨을 붙여 리스트에 카운팅(Counting) 시킨 후 결과 값을 텍스트 파일 형태로 시간과 함께 저장
- ㅇ Python에서 편리하게 웹 서버를 구축할 수 있는 웹 프레임워크(Web Framework) 중 하나인 Flask 웹서버와 연동시켜 저장된 Python 결과값을 송수신
- ㅇ Flask 웹서버에 네이버 지도 API를 이용하여 현재 있는 장소를 지도에 보여줌
- ㅇ Java Script를 이용하여 웹브라우저와 웹서버가 내부적으로 데이터 통신하는 Ajax를 통해 지도에 카운팅한 결과값이 실시간으로 표기됨

2.3 핵심 솔루션

```
listarr = list(map(lambda x: write(x, orig_im), output)) #리스트 정렬
count = 0
for arr in listarr: # 정렬된 리스트를 나눔
for arrsplit in arr[-1:] : #리스트의 마지막을 확인
if(arrsplit == 'person'): #리스트의 마지막이 person이라면
count +=1; #+1 카운팅을 한다.

msg = "유동인구 : " + str(count)
time_now = datetime.datetime.now().strftime('%Y-%m-%d:%H:%M:%S')
with open("log.txt", "a") as f:
f.write(time_now + " > " + msg + "\n") # 결과값을 .txt형태로 시간과 함께 저장


threading.Thread(target=func).start() #Flask와 연동하여 카운팅 값 송수신
app.run("0.0.0.0", 5001) #Flask를 통한 html 주소

-html 소스
var mapOptions = {
center: new naver.maps.LatLng(36.909990, 127.143497),
zoom: 13
}; #필요한 장소의 좌표설정 및 지도 옵션

var map = new naver.maps.Map('map', mapOptions); #네이버 지도 API사용

$(window).on("load", function() {
setInterval(function() {
$.get( "/val", function( data ) {
console.log(data);
var center = map.getCenter();
var infowindow = new naver.maps.InfoWindow();
infowindow.setContent('<div style="padding:10px;"><h5
style="margin-bottom:5px;margin-top:5px;color:#f00;">' + data + '</h5></div>');
infowindow.open(map, center);
}); #python데이터를 송신하여 Html에 실시간
}, 1000); #1초당 데이터 송수신

});
```

□ 핵심 개발

○ 탐지된 여러 객체 중 특정 객체(사람)만을 카운팅(Counting)

- 기존에 학습된 여러 객체를 탐지하는 것을 기반으로 사용자가 원하는 특정 객체(사람)의 숫자를 기록할 수 있게 변경

○ 웹서비스

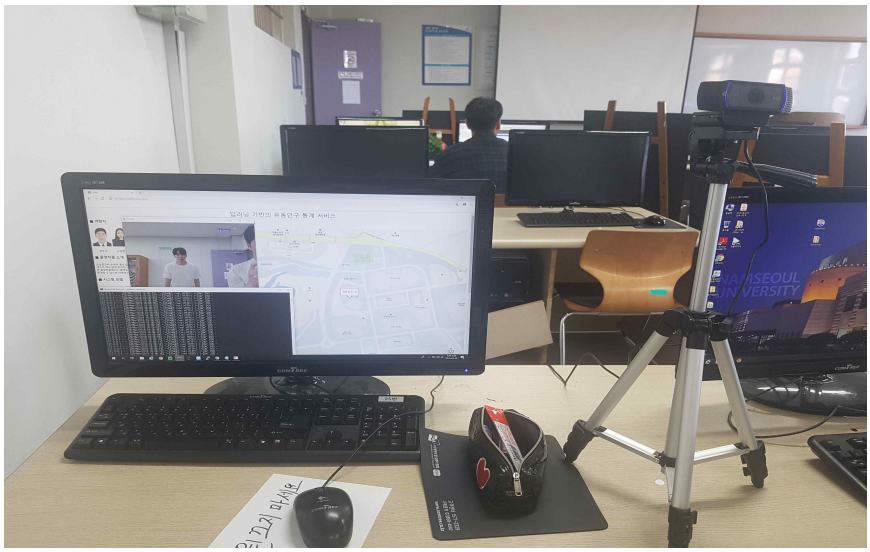
- Flask와 연동하여 카운팅(Counting) 된 결과를 웹페이지(지도) 상에 표기할 수 있게 확장

○ 서비스 기록 및 시각화

- 웹페이지에 실시간으로 표기된 결과값을 txt 파일 형태로 기록이 가능하며, 유동인구 통계 시스템으로 확장

3. 구현 및 실험 결과

3.1 실험결과

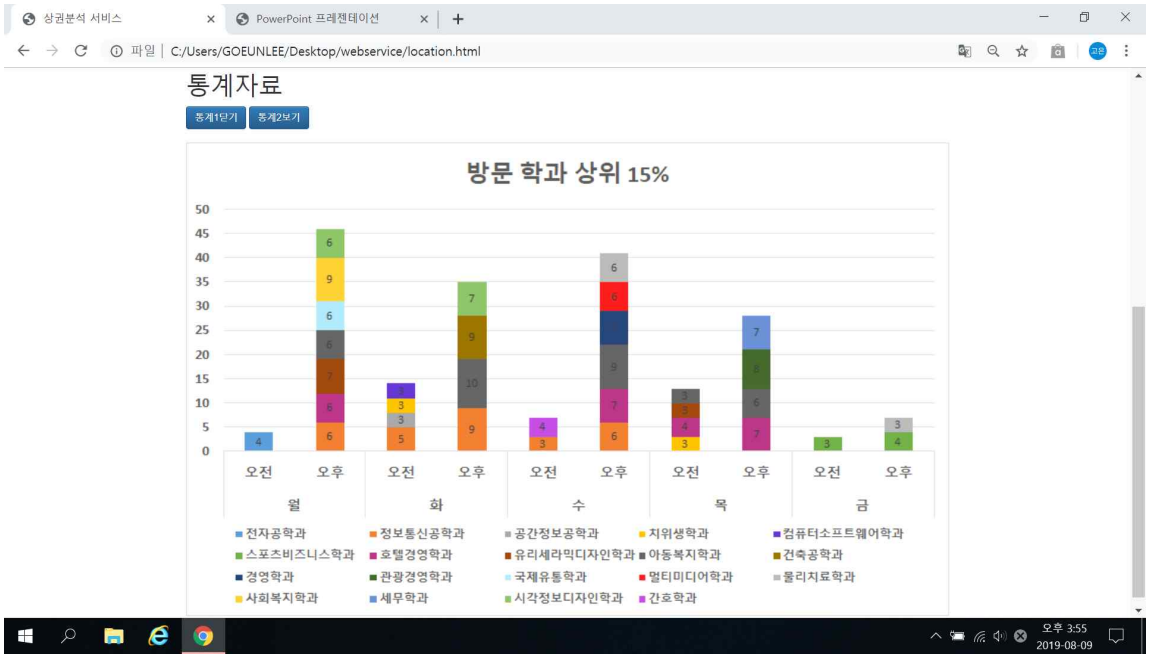


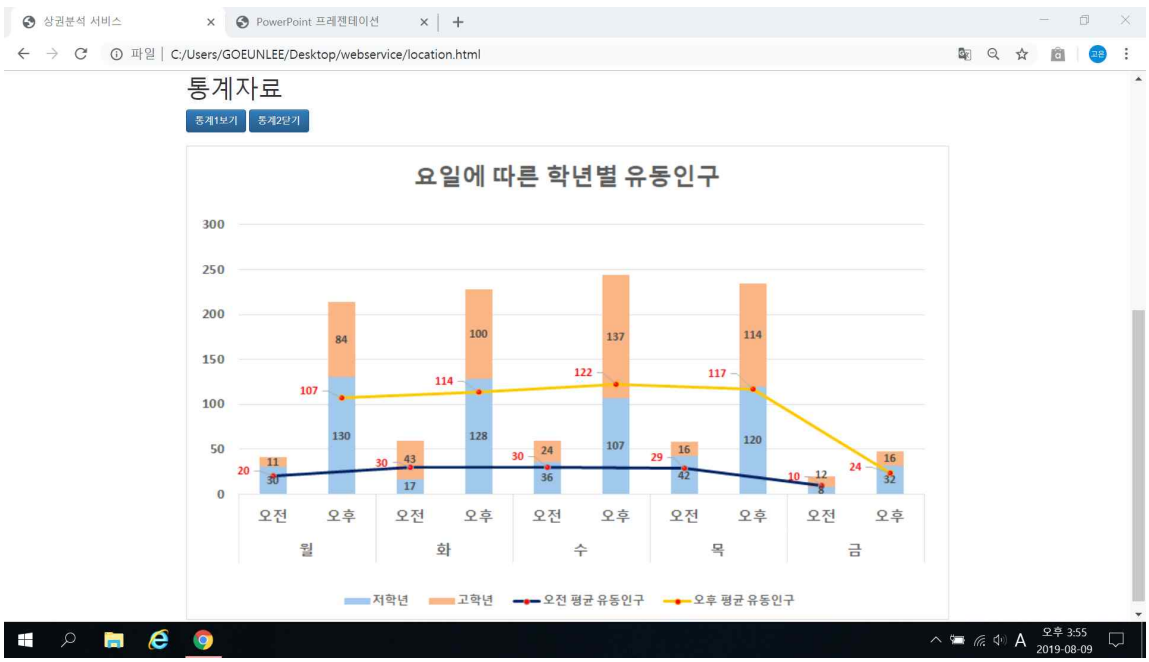
< 그림 8. 테스트 환경 >

□ 유동인구 통계 시스템 구현결과

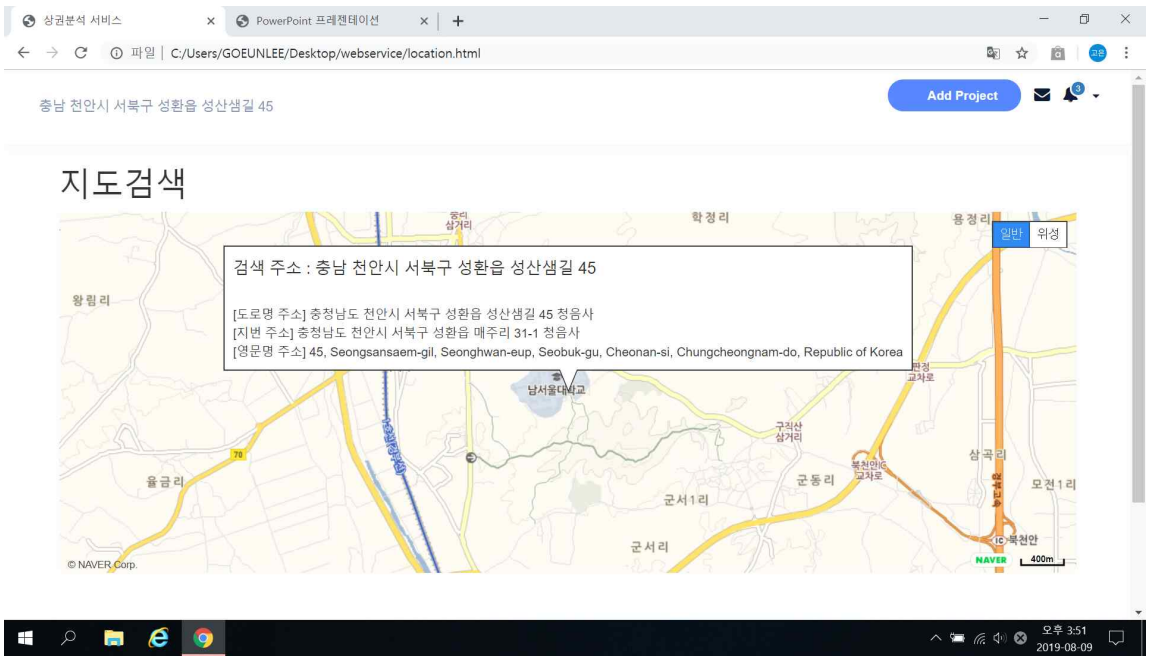
- ㅇ 그림 8와 같이 제안한 서비스를 구현하기 위하여 모니터에 웹캠(Webcam)을 장착함
- ㅇ 그림 8와 같이 실시간 방문자 카운팅(Counting) 한 결과를 Flask를 통해 Html과 연동하여 방문자를 실시간으로 시각화
- ㅇ 실시간 방문자 카운팅(Counting) 결과를 텍스트 형태로 시간과 실시간 결과를 저장

3.2 구현결과





< 그림 9. 실험결과-1 >



< 그림 10. 실험결과-2 >

□ 유동인구 통계 시스템 구현 결과

○ 그림 9와 그림 10은 사용자들이 모니터로 확인 할 수 있는 웹페이지임

○ 그림 10과 같이 검색창에 웹캠(Webcam)을 설치한 매장의 주소를 검색하면 현재 매장의 위치를 지도상에 보여줌

○ 그림 9과 같이 매장의 주소를 검색하면 유동인구 집계 그래프가 나오는 것을 확인함

○ 그림 9은 남서울 대학교 브레댄코에서 데모한 결과를 보여줌

4. 결론

□ 효율적으로 매장을 운영할 수 있도록 지원하는 딥러닝(Deep Learning) 기반의 유동인구 통계 서비스를 개발함

□ 웹캠(Webcam)으로 매장의 방문자 수를 기록하고 매장이 위치해 있는 지도에 실시간으로 방문자의 수가 계수됨을 확인함

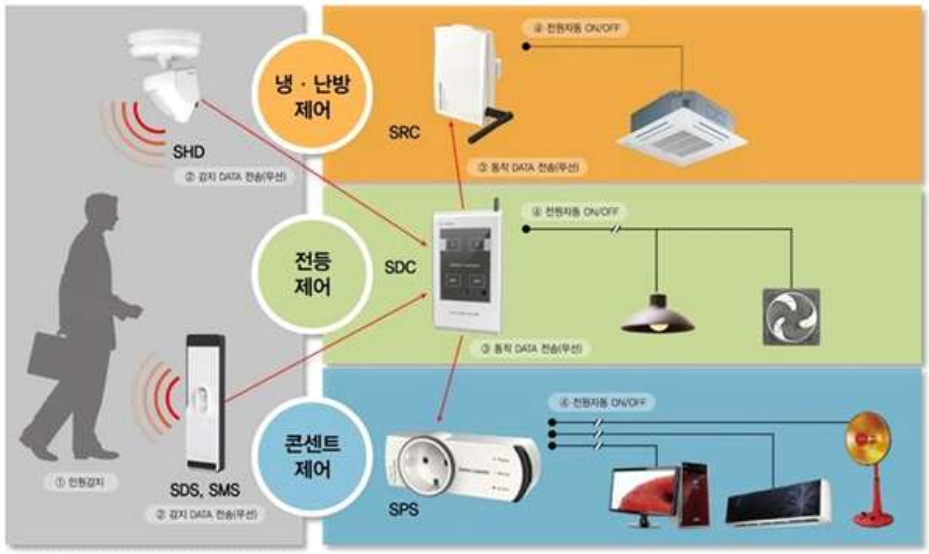
□ 방문자의 수를 기록하고 수집하여 통계 분석 그래프를 나타냄

□ 유동인구의 수는 알 수 있지만 성별이나 연령대에 관한 측면을 고려하지 않았음

□ 매장의 효율적인 운영뿐만 아니라 방문자 수에 따라 분위기(조명, 음악) 와 환경(온도, 습도) 조절 서비스, 대기시간 알림 서비스, 영업 종료 후의 침입감지 서비스 등으로 활용될 수 있음 [< 그림 11. 활용 방안 > 참고]

□ 웹 서버를 통해 매장의 위치를 함께 제공함으로써 매장 주변의 상권 분석 서비스에도 활용될 수 있으며 조금 더 확장하여 무인 스토어 운영서비스에도 활용될 수 있을 것으로 예상됨

□ 객체 인식 기술은 자동차 번호판 인식, 생체 인식 등 여러 분야에서 활용될 수 있고, 특히나 신호, 보행자, 장애물 인식에 있어서 무인 자동차에서 가장 많이 활용되고 있는 기술임



< 그림 11. 활용 방안 >
[이미지 출처: http://ecoway4u.com/html/?page_id=37252]

5. 참고문헌

1) Online Source :

[1] https://opentutorials.org/course/3084

[2] https://bootsnipp.com/snippets/1KA79

2) Yolo [Internet]. Available : https://curt-park.github.io/2017-03-26/yolo/

3) Flask [Internet]. Available : https://ko.wikipedia.org/wiki/플라스크_(웹_프레임워크)

4) Ajax [Internet]. Available : https://jayzzz.tistory.com/67