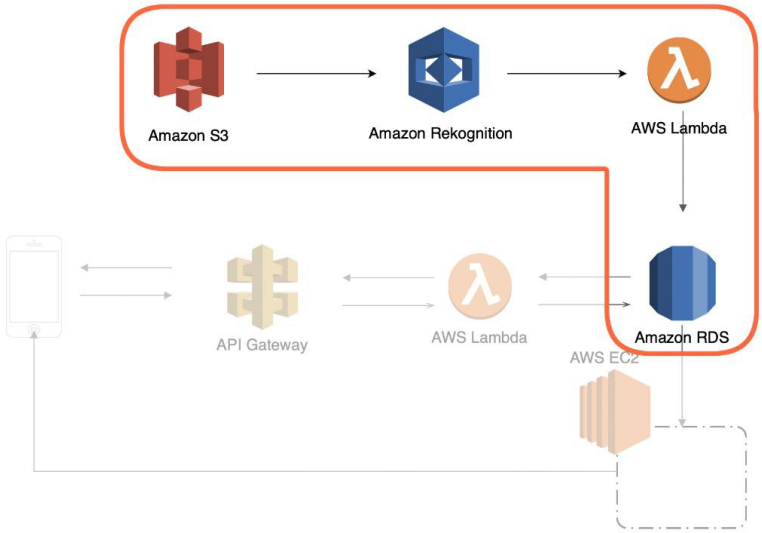
개인 결과보고서 – 2020125072 허유진

상세 개발 계획서



해당 프레임워크에 해당하는 부분의 개발을 진행하였다.

개발추진실적 중간 보고 내용

1. 영상 인식 프레임워크(라이브러리) 선정

   1) python openCV

haarcascades_frontface 분석모델로 진행할 시 전면에서 찍힌 사람은 80% 확률로 인식하였다. 하지만 불특정한 자세로 찍힌 사람들이 많은 본 프로젝트에서는 적합하지 않았다. haarcascades에서는 upper-body 모델이 가장 신뢰성이 높았으나 최저 10%정도의 정확도로 프로젝트에서 사용하기에는 어려웠다. 사진화질, 구도, 밝기, 인식속도에 따라 인식률은 높아졌지만 개발에 시간이 필요할 것으로 예상되었다.

```
haar_upper_body_cascade = cv2.CascadeClassifier('haarcascades\\haarcascade_upperbody.xml')
filename = "photo_2022-04-06_19-09-11.jpg"

frame = cv2.imread(filename)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # convert video to grayscale

upper_body = haar_upper_body_cascade.detectMultiScale(
    gray,
    scaleFactor = 1.1,
    minNeighbors = 1,
    minSize = (5, 5), # Min size for valid detection, changes according to video size or bod
    flags = cv2.CASCADE_SCALE_IMAGE
)
```

Figure 1 upperbody 분석 **Figure 2 frontface 분석**

2) AWS Rekognition

초기에는 높은 서비스 금액으로 고려하지 않았지만, 수집 사진에서 인원수 변동이 실시간으로 바뀌는 정도가 크지 않음을 고려해 선정하게 되었다. openCV에 비해 구현시간이 적고 정확도가 높았다. 초기 계획과는 달리 실시간이 아닌 5분에 한 번 사진을 찍고 한 시간 동안의 평균을 계산하였다.

2. 영상 수집 방법 선정

1) python 원격지 사진 촬영

자동으로 사진을 찍을 수 있는 코드를 작성하였다. 다른 코드와 연동이 쉽고 오류가 났을 때 빨리 원인을 찾을 수 있다는 장점이 있다. 추후 실시간 서비스 제공 시 쉽게 연동할 수 있다.

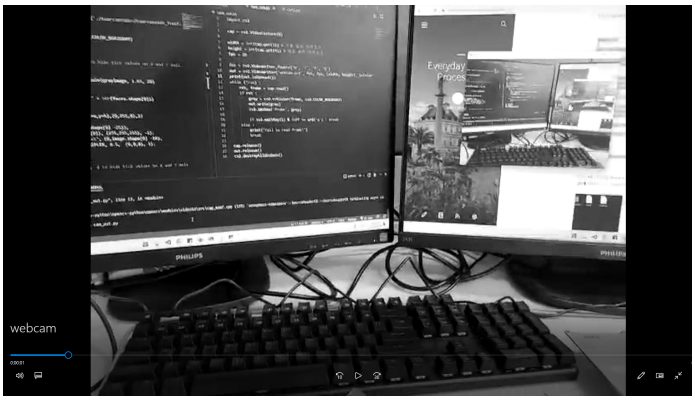


Figure 3 원격 촬영 영상 중 캡쳐

2) 상용 앱 사용

“오토클리커”라는 매크로 앱을 이용해 주기적으로 사진을 촬영하였다. 학교의 WIFI가 불안정하기에 원격에서 사진을 촬영하기 어려워 local에서 쉽게 운용할 수 있는 상용 앱을 사용하였다. 문제가 생겼을 때 오류를 찾기 어렵다는 문제가 있다.

3. Rekognition - DB 연계

- 사진 데이터 처리 프로세스 자동화

사진 수집 -> 수집사진 S3 업로드 -> Rekogniton 사진 분석 -> lambda- DB연계 -> 분석결과 DB 전송

위의 프레임워크와 프로세스를 이용하여 Rekogntion에서 사진을 분석한 결과를 DB로 연결해 저장하였다.

```python
import pymysql


def connect_RDS():
    print("start connect_RDS function")
    host = "*****"
    port = *****
    username = "*****"
    database = "*****"
    password = "*****"

    conn = pymysql.connect(
        host=host,
        user=username,
        passwd=password,
        db=database,
        port=port
    )
    return conn

def send_RDS(conn, person_count, imageDateTime,coordinate):
    print("start send_RDS function")
    cur = conn.cursor()
    cur.execute("INSERT INTO Jullulim (person_count, Time, coordinate) VALUES (%s,%s,%s);",(person_count, imageDateTime, coordinate))
    conn.commit()
```

Figure 4 RDS 연결 함수

- DB 전송 내용 : 인식 사람수, 촬영일자, 인식좌표

변수명	변수형	설명
Person_count	int	인식 사람수
imageDateTime	datetime	사진촬영날짜+시간
coordinate	string	인식 좌표 (디버깅용)

Gonggalbbang.Jullulim: 182 행 (총) (대략적)

KEY	person_count	Time	coordinate
0	0	2022-05-02 09:43:00	[]
1	-1	2022-05-03 15:32:00	[[45, 0, 97, 53]]
2	3	2022-05-03 15:35:00	[[59, 38, 37, 40], [59, 20, 5, 7], [72, 28, 15, 25]]
4	2	2022-05-03 15:37:00	[[60, 34, 4, 7], [83, 50, 21, 15]]
5	2	2022-05-03 15:42:00	[[60, 34, 4, 7], [84, 35, 25, 15]]
6	2	2022-05-03 15:47:00	[[60, 34, 4, 7], [84, 36, 26, 14]]
7	2	2022-05-03 15:52:00	[[78, 40, 34, 20], [60, 34, 4, 7]]
8	2	2022-05-03 15:57:00	[[60, 34, 4, 7], [87, 39, 25, 12]]
9	2	2022-05-03 16:02:00	[[72, 50, 20, 26], [60, 34, 4, 7]]
10	2	2022-05-03 16:07:00	[[81, 61, 19, 17], [60, 34, 4, 7]]

Figure 5 DB 전송결과 (heidisql 캡쳐)

문제점 및 해결 내용

1. 사진 수집 시 지속적인 카메라 꺼짐 문제

1) 파이썬 자동 촬영 코드

대체 방안으로 상용 매크로앱이 아닌 파이썬 자동 촬영코드 작성하였다. 라즈베리파이에 파이썬 파일을 업로드 후 공유기로 단말기와 라즈베리파이를 연결하여 촬영 진행하는 방안을 마련해두었다.

2) 원인 파악을 위한 장시간 지속 감시

수집장소에서 단말기 수거 후 장시간 앱을 돌리며 문제상황을 재연하였다. 그 결과, 기본 카메라가 일정 시

간이 지나면 자동으로 종료되는 것을 확인하였다. 이를 통해 매크로 앱의 문제가 아닌 기본 카메라의 문제임을 확인할 수 있었다. 기본 카메라가 아닌 타 카메라 앱으로 촬영진행을 하여 정상작동을 확인하였다.

2. DB 연결 - Rekognition 함수 Integration 과정 중 오류

1) json 파일 (db 접속 정보) 불러오기 오류

DB 접속 정보 등 민감한 정보들은 json으로 저장 후 관리자가 필요할 때 가져와 사용할 수 있도록 코드를 작성하였다. Lamda에서 json파일을 호출하는 중 오류가 발생하여 lamda에는 직접 변수로 할당하고 git에는 해당 부분을 제외 후 업로드하였다.

2) import pymysql 오류

Lamda 에서 파이썬 라이브러리를 불러오는 중 오류가 발생하였다. Pymysql 라이브러리를 zip로 압축하여 함께 업로드하여 해결하였다.

3. Coordination 값 저장 관련 Issue

Rekognition에서 사진 분석 결과 중 좌표 값의 DB 저장에 대한 issue가 있었다. 디버깅을 위해서는 필요한 정보이지만 좌표에 대한 숫자가 많고 사람의 수에 따라 값의 개수가 변하기 때문에 논의가 필요하였다. 최종적으로 서비스 이용자가 필요한 값을 아니라고 판단하여 관리자만 디버깅 시 읽을 수 있게 하나의 string으로 저장하기로 결정하였다.

4. Opencv

초기에 혼잡도 계산을 위한 사람 수 인식에 대해 openCV 라이브러리를 활용하였다. 사진의 화질, 구도, 밝기, 해석model을 바꾸어 가며 인식률을 높이려 노력하였다. 정확도가 낮아 AWS의 Rekognition을 사용하기로 결정하였다.

계획 대비 효과 자체 평가

OpenCV 라이브러리를 이용해 python 내에서 개발을 진행하고 싶었으나 역량이 부족해 정확도를 높일 수 없던 점이 아쉽다. Opencv가 아닌 rekognition을 사용하면서 실시간으로 분석하여 데이터를 저장하기 어려웠지만 프로젝트의 목적인 혼잡도를 계산에는 적합한 프레임워크를 찾아 사용했다고 생각한다. 현재 사진 수집과 사진 업로드 사이의 과정은 수동으로 진행하고 있는데 좋은 네트워크 환경이 갖추어지면 모든 과정을 자동화하여 실시간 분석까지 확대할 수 있는 서비스라고 생각한다.

개인 기여 내용 및 배운 점

기여 내용

사진 인식 프레임워크 선정

Aws rekognition을 이용한 데이터 분석 및 데이터 추출

Recognition-lamda-DB 연결

배운 점

컴퓨터 비전에 많이 사용하는 opencv와 aws-rekognition에 대해서 배울 수 있었다. 어떤 과정으로 컴퓨터가 사진을 인식하는지 배울 수 있었으며 직접 활용해서 결과를 도출할 수 있었다. AWS의 구조와 사용 방법에 대해서 배우고 각 프레임워크를 연결하는 방법에 대해서 고민하고 설계해볼 수 있었다.