산학연계SW프로젝트
중간보고서

프로젝트 제목 :
CCTV에서 딥러닝 기반 오브젝트 찾기

프로젝트 수행기간 : 2018.10.01 ~ 2019.05.31.

프로젝트 팀명 :

지도 교수 :

참여업체 명 :

2018. 11. 14

광운대학교
KwangWoon University

산학연계SW프로젝트 중간보고서

<table>
<tr><td>과 제 명</td><td colspan="4">여러개의 동영상에서 사진과 동일한 사람을 찾는 필터링</td></tr>
<tr><td>팀 명</td><td colspan="4"></td></tr>
<tr><td>수행기간</td><td colspan="4">2018년 10월 01일 ~ 2019년 05월 31일</td></tr>
<tr><td>과제비</td><td colspan="4"></td></tr>
<tr><td>지도교수</td><td>성 명</td><td>심동규 교수님</td><td>학 부</td><td>컴퓨터공학부</td></tr>
<tr><td rowspan="6">참여학생</td><td>성 명</td><td>학 부</td><td colspan="2">학 번</td><td>email</td></tr>
<tr><td></td><td>컴퓨터공학부</td><td colspan="2"></td><td></td></tr>
<tr><td></td><td>컴퓨터공학부</td><td colspan="2"></td><td></td></tr>
<tr><td></td><td>컴퓨터공학부</td><td colspan="2"></td><td></td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td></tr>
<tr><td rowspan="2">참여업체</td><td>회사명</td><td></td><td>담당자</td><td></td></tr>
<tr><td>연락처</td><td></td><td>email</td><td>..</td></tr>
<tr><td colspan="4">산학연계SW프로젝트의 결과물에 대한 동의서</td><td>동의여부</td></tr>
<tr><td colspan="2">프로젝트에 대한 지원</td><td colspan="2">본 프로젝트는 과학기술정보통신부 및 정보통신기술진흥센터의 SW중심대학사업의 지원으로 수행된 것임
(과제번호: 2017-0-00096)</td><td>■</td></tr>
<tr><td colspan="2">프로젝트 결과물의 활용</td><td colspan="2">본 프로젝트의 결과물은 프로젝트에 참여하고 지원한 광운대학교(학생들과 지도교수), 참여업체 그리고 정보통신기술진흥센터의 소유물이며 향후 활용과 소유에 관해서는 참여자 간의 협의에 의해 결정할 수 있음</td><td>■</td></tr>
</table>

『산학연계SW프로젝트』지원계획에 따라 중간보고서를 제출합니다.

2018년 11 월 14 일

팀 장 (인)
팀 원 (인)
팀 원 (인)
팀 원 (인)
팀 원 (인)

지도교수 (인)

광운대학교 소프트웨어융합대학 귀하

목 차

1. 과제의 개요 ····· 1
가. 배경 및 필요성 ····· 1
나. 목적 ····· 1
다. 설계 내용 ····· 1

2. 과제의 수행 ····· 2
가. 수행 방법 및 과정 ····· 2
나. 중간 결과 ····· 2
다. 향후 계획 ····· 2

3. 과제의 평가 ····· 3
가. 개선 방안 ····· 3
나. 기대 효과 ····· 3
다. 기타 보고사항 ····· 3

4. 별첨 ····· 4

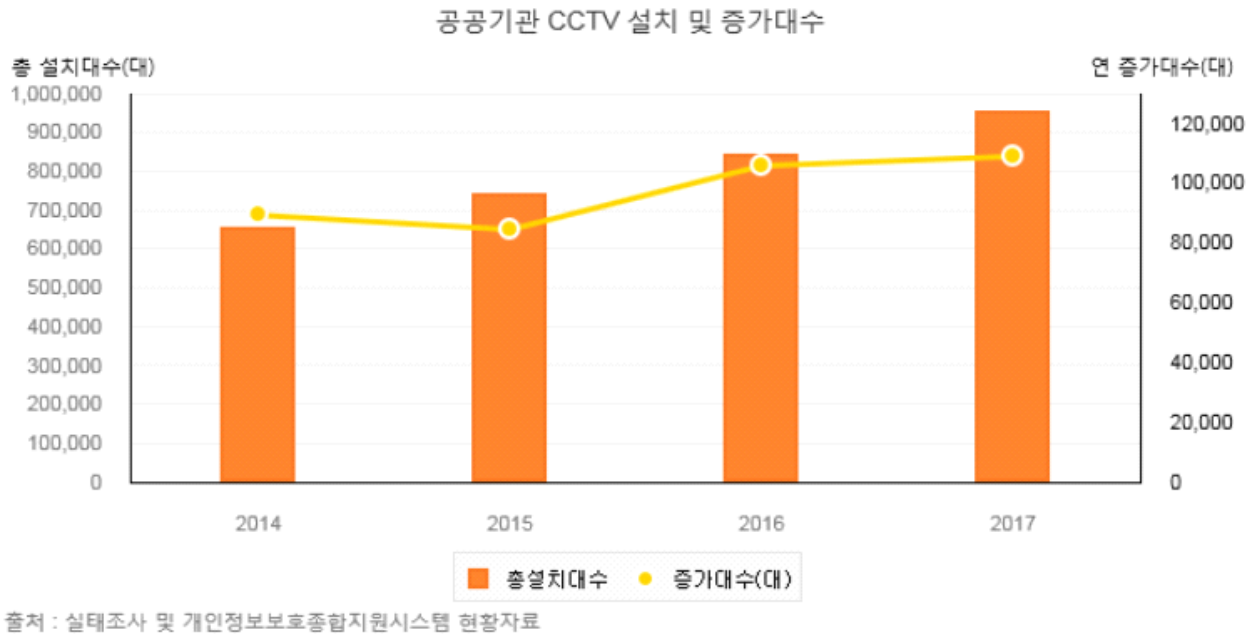
1. 과제의 개요

가. 배경 및 필요성

최근 여러 가지의 위험요인이 늘어나면서 각종 범죄들도 마찬가지로 증가하고 있는 추세임. 또한 범죄의 지능화가 진행되면서 검거에 어려움이 더해지고 있음. 이에 따라, CCTV나 블랙박스 등 영상매체의 분석을 통해 범죄자를 검거하는 수사방법은 정확한 수사방법으로 더욱 그 중요성이 증가할 것으로 예측됨. 현재까지 영상매체 분석 수사방법은 꾸준히 증가하고 있으며 앞으로도 더욱 효과를 발휘할 것으로 예측됨.

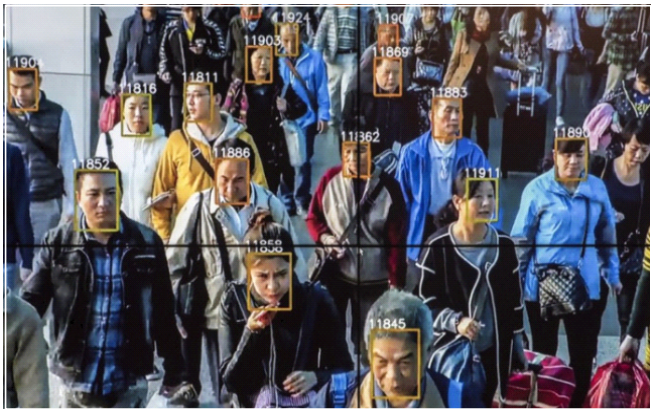
구분	범인검거 현황						
	소계	살인	강도	강간	절도	폭력	기타 형사범
14	1,627	1	14	22	737	347	506
15	10,157	2	16	242	1,445	5,250	3,376
16	20,430	3	7	354	1,817	12,358	5,891
17	28,004	0	7	483	2,285	17,388	7,841
18.8	789	0	7	91	620	3,443	2,759
합계	**61,007**	**6**	**51**	**1,192**	**6,904**	**38,786**	**20,373**

<그림 1. 2014년 이후 CCTV를 활용한 실시간 범인검거 현황 [자료=이재정의원실]>



<그림 2. 2014년 이후 CCTV설치 및 증가대수 현황>

영상매체를 통한 범죄 검거율이 증가하면서 CCTV설치 대수도 지속적으로 늘어나 시스템의 활용에 탄력이 더해지고 있음. 세계적으로도 얼굴인식 시장이 지속적으로 증가하면서 영상매체를 통해 범죄자를 검거하는 시스템 또한 개발 및 적용되고 있는 상황임. 중국의 경우 이미 해당 시스템을 경찰에 제공하여 CCTV뿐 아니라 웨어러블 디바이스에 적용하여 실시간으로 범죄 수사에 활용하고 있음.



<그림 3. 중국의 CCTV 안면인식 시스템>

<그림 4. 중국의 안경형 안면 인식장치>

나. 목적

현재 우리나라의 범죄수사에서 CCTV, 블랙박스 등 영상매체를 통한 범죄자 분석방식은 경찰 인력을 통해서 직접적으로 범죄자 식별을 진행하고 있어 효율적인 식별작업이 진행되고 있지 않고 있음. 영상을 일일이 확인하거나 여러 개의 영상을 여러 명의 인력이 수작업으로 확인하고 있는 실정임. 이에 따라, 범죄자 얼굴 인식 시스템에 대한 연구가 지속적으로 진행되고 있지만 여러 가지 한계점으로 인해 실무에 적용되고 있지 않음.

첫째로, 현재 얼굴 인식 알고리즘은 눈, 코, 입을 찾는 방식으로 이뤄지고 있기 때문에 눈, 코, 입 중 하나라도 없는 경우 얼굴 인식을 할 수가 없음. 따라서 마스크나 모자를 쓴 경우, 영상에서 얼굴이 측면인 경우, 얼굴인식이 제대로 되지 않음. 둘째, 대상이 너무 멀리 있어서 영상에서 얼굴이 작게 나타나는 경우, 조도가 낮아 영상이 어두운 경우, 얼굴 인식은 되나 얼굴 비교의 정확도가 떨어짐.

따라서 이 프로젝트에서는 이러한 한계점 중 영상의 얼굴이 측면인 경우와 영상이 어둡거나 대상이 멀리 있어 얼굴이 작은 경우의 얼굴 인식과 비교를 개선한 필터링을 SURF알고리즘을 이용하여 만들고자함.

다. 설계 내용

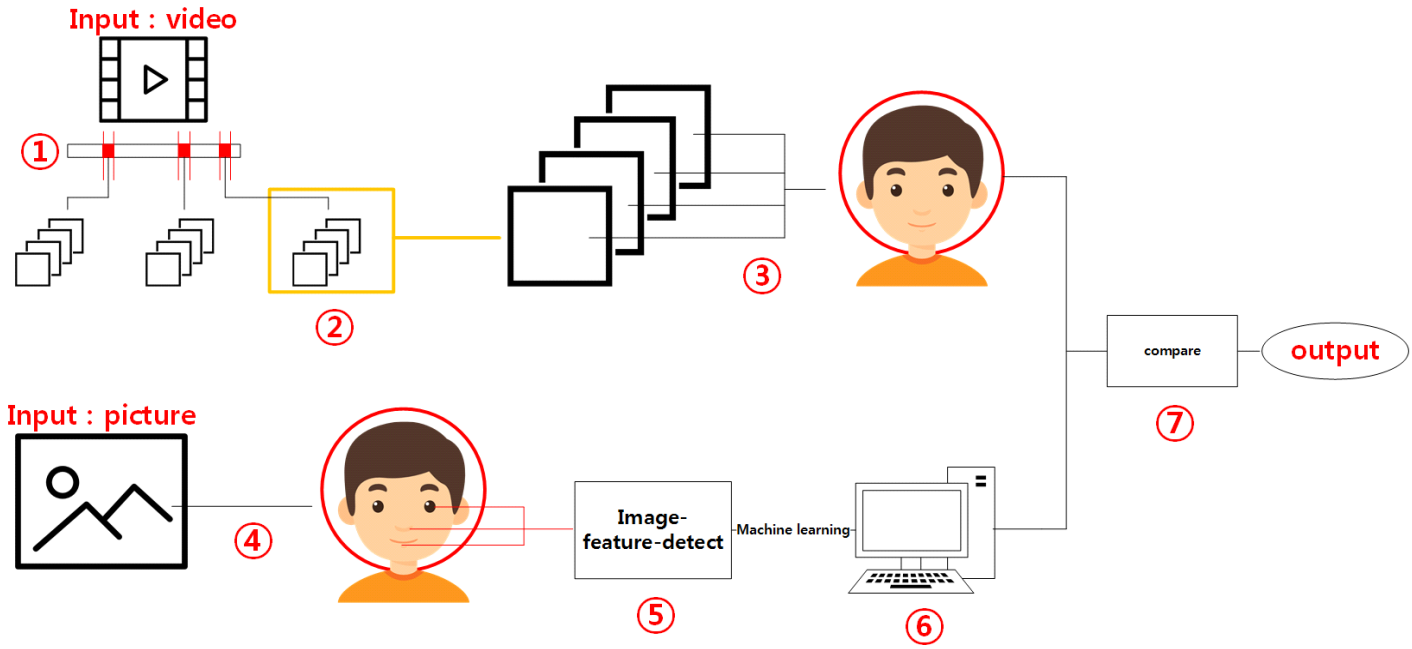
1) System flow

① CCTV 영상에서 event가 있는 부분만 잘라냄.
② event가 있는 구간을 frame으로 가져옴.
③ 각 프레임마다 얼굴을 인식함.
④ 범죄자의 사진을 얼굴 인식함.
⑤ 얼굴에서의 특징을 도출함.
⑥ 특징을 머신러닝함.
⑦ 특징을 영상에서 따온 얼굴들과 비교하고 일치율을 출력함.

2) Server

learning data set을 저장하는 DataBase server

프로그램의 정확성을 높이기 위해 수많은 양의 이미지(사람 얼굴 이미지)를 저장해야하므로 DataBase server를 구축함. DataBase server의 구축은 VMware를 통하여 리눅스 환경의 Virtual



Machine에 구축함. 이미지를 저장하고 처리하므로 용량과 속도가 중요하므로 적합한 DataBase server를 정함. 리눅스를 DataBase의 종류는 크게 Oracle, MS-SQL, MariaDB등이 있고, 본 프로젝트에서는 MariaDB를 사용함.

MariaDB를 선택한 이유는 크게 아래와 같음.

[1] SQL 언어를 지원함.

- SQL 언어를 지원하기 때문에 DataBase 조작에 용이함

[2] 오픈 소스 기반으로 개발됨.

- 오픈 소스 기반이므로 라이선스 정책에서 자유롭고, 무료로 배포된 라이브러리 사용 가능.

[3] 리눅스에서 가장 많이 사용하는 DataBase server.

- 비교적 윈도우보다 더 안전하고 덜 취약한 리눅스서버에서 많이 사용.

[4] 성능 면에서 가장 우수함.

- 다른 DataBase server에 비해 성능이 좋고, 저렴함.
- 'HammerOra'라는 DataBase server 측정 툴을 사용하여 Oracle, MS-SQL, MariaDB의 transaction(데이터베이스의 상태를 변화시키기 해서 수행하는 작업의 단위[1])를 측정한 결과는 아래와 같다.(transaction per minute이 클수록 성능이 좋음)

- Oracle : 10674 transaction per minute

- MS-SQL : 104766 transaction per minute

- MariaDB : 419958 transaction per minute

위의 결과를 보면 알 수 있듯이 MS-SQL이 Oracle 10배 더 많은 처리량을 나타냈고 MariaDB가 MS-SQL에 비해 더 많은 처리량을 나타냄. 또한 이 차이는 튜플의 수가 더 많아 질수록 극명하게 나타남.

본 프로젝트는 윈도우 환경이므로 리눅스에서 구축된 DataBase server에 접근이 가능해야함.

1) 위키피디아-transaction

DataBase의 client는 윈도우이며 Host는 리눅스로 설정.

Leaning data set은 정면 사람 얼굴 사진들의 data set과 측면 사람 얼굴 사진들의 data set으로 두 가지로 구성. 정면 얼굴과 측면 얼굴이 각각 learning이 됨. 이미지는 바이너리로 다루므로 저장 타입을 BLOB으로 함.

3) SURF

SURF(Speeded-Up- Robust Features) 알고리즘

SURF 알고리즘은 여러 개의 영상으로부터 스케일, 조명, 시점 등의 환경변화를 고려하여 환경변화에 불변하는 특징점을 찾는 알고리즘.

SURF 알고리즘을 사용하는 이유

- 어두운 환경에서 인식하는 작업이 빈번하기 때문에 환경변화에 불변하는 특징점을 찾는 것이 필수적.
- openCV에서 제공되는 기능이므로 사용에 있어서 편한 접근성이나 수정이 가능.
- 속도가 상당이 향상된 알고리즘으로 긴 동영상에서의 작업에서 크게 시간을 줄일 수 있음.

URF 알고리즘의 원리[5]

Scale space 상에서 Hessian 행렬의 행렬식(determinant)이 극대인 점들을 특징점으로 검출. SURF에서 사용한 특징점 추출 방법을 Fast Hessian이라 부름.

특징점 도출[6]

적분 영상(Integral Image)을 사용. 아래와 같은 수식을 이용하여 적분 영상 생성.

$$II_{\Sigma}(x,y)=\sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y}I(i,j)$$

Hessian 행렬[7]

다변수함수가 극값을 가질 때, 그것이 극대인지, 극소인지 판정할 때 사용. 실함수 $f(x_1,x_2,x_3,\ldots,x_n)$이 주어졌을 때, Hessian 행렬은 아래와 같이 주어짐.

$$H(f)=\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}\\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \vdots\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_n^2}\end{bmatrix}$$

[8]

Descriptor 생성[9]

특징점이 검출되면 특징점 중심으로 특징점이 검출된 스케일 정보 s를 이용하여 반지름 6s의 원 안의 픽셀들에 대하여 x 방향과 y 방향으로 Haar Wavelet Response인 dx와 dy를 구함. 윈도우 5도 간격으로 회전하면서, 60도 윈도우 내의 벡터의 크기를 더해 가장 큰 크기를 가지는 방향을 특징점의 주 방향으로 결정. 주 방향이 결정되면, descriptor 생성. descriptor는 특징점 주변의 일정한 영역 내에 이웃하고 있는 픽셀의 밝기 변화를 나타냄. 각 픽셀을 haar wavelet filter를 사용하여 각

$$\sum dx, \sum |dx|, \sum dy, \sum |dy|$$

4개를 구함.

3) PCA

PCA(Principal Components Analysis)

이미지를 사용하는 만큼 고차원의 데이터를 가지고 학습을 하기 때문에 많은 시간과 오히려 역효과를 야기할 수 있음. 따라서 학습하기 전에 고차원의 데이터를 저차원으로 환원시키는 PCA를 사용.

PCA의 원리

서로 연관 가능성이 있는 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간(주성분)의 표본으로 변환하기 위해 직교 변환을 사용. 주성분의 차원수는 원래 표본의 차원수보다 작거나 같음. 주성분 분석은 데이터를 한개의 축으로 사상시켰을 때 그 분산이 가장 커지는 축을 첫 번째 주성분, 두 번째로 커지는 축을 두 번째 주성분으로 놓이도록 새로운 좌표계로 데이터를 선형 변환. 이 변환은 첫째 주 성분이 가장 큰 분산을 가지고, 이후의 주성분들은 이전의 주성분들과 직교한다는 제약 아래에 가장 큰 분산을 갖고 있다는 식으로 정의되어 있음. 중요한 성분들은 공분산 행렬(covariance matrix)의 고유벡터(eigenvector)이기 때문에 직교하게 됨.

PCA의 과정

1) Feature Extraction

기존 변수를 선형결합(Linear Combination)을 통하여 새로운 변수로 창출. 변수가 p개, 관측치가 n개 있는 데이터 X($p \times n$)로 새로운 변수 z를 아래와 같이 만드는 과정.

$$\begin{aligned}
\overrightarrow{z_1} &= \alpha_{11}\overrightarrow{x_1} + \alpha_{12}\overrightarrow{x_2} + \ldots + \alpha_{1p}\overrightarrow{x_p} = \overrightarrow{\alpha_1}^T X \\
\overrightarrow{z_2} &= \alpha_{21}\overrightarrow{x_1} + \alpha_{22}\overrightarrow{x_2} + \ldots + \alpha_{2p}\overrightarrow{x_p} = \overrightarrow{\alpha_2}^T X \\
&\cdots \\
\overrightarrow{z_p} &= \alpha_{p1}\overrightarrow{x_1} + \alpha_{p2}\overrightarrow{x_2} + \ldots + \alpha_{pp}\overrightarrow{x_p} = \overrightarrow{\alpha_p}^T X
\end{aligned}$$

출처: ratsgo's blog for textmining - 주성분 분석

위와 같이 변수 추출로 새롭게 만들어진 행렬Z는 아래와 같음.

2) 데이터 세트를 제로 평균으로 조정(기존 값에서 평균 빼기)

$$Z = \begin{bmatrix} \overrightarrow{z_1} \\ \overrightarrow{z_2} \\ \cdots \\ \overrightarrow{z_p} \end{bmatrix} = \begin{bmatrix} \overrightarrow{\alpha_1}^T X \\ \overrightarrow{\alpha_2}^T X \\ \cdots \\ \overrightarrow{\alpha_p}^T X \end{bmatrix} = \begin{bmatrix} \overrightarrow{\alpha_1}^T \\ \overrightarrow{\alpha_2}^T \\ \cdots \\ \overrightarrow{\alpha_p}^T \end{bmatrix} X = A^T X$$

3) 공분산 행렬(Covariance matrix) 만들기

원데이터 행렬 X의 분산을 최대한 보존해야 하므로 Z의 분산이 최대화가 되어야 함. Covariance matrix는 아래와 같은 식으로 만들 수 있음.

$$\frac{cov(X,Y) = \sum_{i=1}^{n}(X_i - \overline{X})(Y - \overline{Y})}{n-1}$$

4) 공분산 행렬의 고유 벡터 및 고유 값 얻기

Eigenvector decomposition을 통하여 공분산 행렬의 고유 벡터 및 고유 값을 구함. 공분산 행렬의 고유 벡터는 주성분이 됨. 공분산 행렬은 서로 다른 고유벡터끼리는 서로 직교(orthogonal)하는 성질이 있음. 원데이터를 공분산 행렬의 고유벡터로 사영하기 전에는 변수 간 연관성이 있었더라도 PCA 변환에 의하여 좌표축이 바뀐 데이터들은 서로 무상관(uncorrelated)이게 됨.

5) 구성 요소를 선택하고 특징 벡터를 형성

6) 특징 벡터를 이용한 변환

특징 벡터를 이용한 변환은 아래와 같이 변환한다.

$$FinalData = RowFeatureVector \times RowDataAdjust$$

2. 과제의 수행

가. 수행 방법 및 과정

역할 분담

성명	역할	공동
김태원	-1,3주차 내용 스터디 및 발표 -영상처리 관련 OpenCV 담당	중간보고서 작성 전체 매커니즘 설계
장다빈	-2,4주차 내용 스터디 및 발표 -DB서버 관련 담당	
이강풍	-5,6주차 내용 스터디 및 발표 -Python tensorflow 관련 담당 -회의록 작성 담당	

수행 일정

대분류	소분류	2018년			2019년	
		10	11	12	1	2
딥러닝	딥러닝 이론					
	Python					
	Tensorflow					
	Matlab					
	Feature-Face comparing					
영상 처리	OpenCV					
	Cutting Algorithm					
	Face Detect					
서버	SQL					
	DB 설계					
	서버 구축					

수행 과정

* Server

1. MariaDB 사용

MariaDB 실행 및 database 생성 :

learning data set table 생성(data type : BLOB)

```
MariaDB [learning_data_set]> CREATE TABLE images (picid int NOT NULL AUTO_INCREMENT,
    -> description VARCHAR(40),
    -> image BLOB,
    -> PRIMARY KEY (picid));
Query OK, 0 rows affected (0.14 sec)
```

이미지 파일 넣기

```
MariaDB [learning_data_set]> INSERT INTO images VALUES(NULL,'1',
    -> 'C:₩Users₩장다빈₩Documents₩카카오톡 받은 파일₩KakaoTalk_20180403_015458292.jpg');
Query OK, 1 row affected (0.01 sec)
```

```
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or ₩g.
Your MariaDB connection id is 2
Server version: 10.0.37-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]> create database learning_data_set;
Query OK, 1 row affected (0.01 sec)

MariaDB [(none)]> create user 'team_zzagle2'@'localhost'identified by 'input_password';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| learning_data_set  |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

파일 확인

```
MariaDB [learning_data_set]> SELECT * FROM images i;
+-------+-------------+-----------------------------------------------------------------------------------+
| picid | description | image                                                                             |
+-------+-------------+-----------------------------------------------------------------------------------+
|     1 | 1           | C:Users장다빈Documents카카오톡 받은 파일KakaoTalk_20180403_015458292.jpg |
|     2 | 1           | NULL                                                                              |
+-------+-------------+-----------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

데이터가 들어간 위치 찾기

```
MariaDB [learning_data_set]> show variables like 'datadir';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| datadir       | C:₩Program Files₩MariaDB 10.0₩data₩ |
+---------------+-----------------------------------------+
1 row in set (0.01 sec)
```

* System

1. Video Cutting

: CCTV 영상의 시간은 매우 길기 때문에 전체 영상에 대하여 얼굴인식을 수행하면 time complexity 가 매우 높아짐. 따라서 이 과정에서는 사전에 필요가 없다고 생각되는 부분을 미리 제외하여 전체 수행시간을 낮춤.

사용언어는 파이썬이며 프레임 라이브러리인 OpenCV와 딥러닝 라이브러리 tensorflow를 사용함. 현재 구현한 코드는 해당영상이 아무런 움직임 없는 경우만을 제외하도록 하였음. 차후 다른 예외처리를 추가함으로써 complexity를 높일 생각이다. 밑에 사진은 구현한 코드이다.

```
files = glob.glob('./images/*')
for f in files:
    os.remove(f)
print("Extracting frames from the video.....")
print("Please Wait...₩n₩n")
def video2frames(path, output=None, skip=1, mirror=False):
    video_object = cv2.VideoCapture(path)

    # setup the output folder
    if output is None:
        output = path[:-4]
    else:
        if not output.endswith('/') and not output.endswith('₩₩'):
            output += '/'
        output += 'py_image'

    index = 0
    last_mirrored = True
    while True:
        success, frame = video_object.read()
        if success:
            if index % skip == 0:
                if mirror and last_mirrored:
                    frame = _mirror_image(frame)
                last_mirrored = not last_mirrored

                cv2.imwrite(output + "_" + str(datetime.now()) + ".jpg", frame)  # assumes that the extension is three letters long
        else:
            break

        index += 1
```

위 코드는 Video Cutting System중에 extract.py 모듈이며 반복문을 통해 해당 영상을 프레임으로 나누어 image폴더 안에 저장함.

```
def run_classifier():
        flist = []
        list1 = glob.glob("./images/*.jpg")
        list1.sort()
        print("Printing the time of Interesting Events.....₩n₩n")
        temp = str(inference.run_inference_on_image())
        for i in range(len(list1) - 1):
                inference.imagePath = list1[i]
                temp2 = str(inference.run_inference_on_image2())
                inference.imagePath = list1[i+1]
                temp = str(inference.run_inference_on_image2())
                if temp2 != temp:
                        print("Time : " + str(datetime.timedelta(seconds=(i*3))))
                        flist.extend([i*3])
                else:
                        print("." ,)
        d = np.array(flist)
        d.sort()

        diff = [y - x for x, y in zip(*[iter(d)] * 2)]
        avg = sum(diff) / len(diff)

        m = [[d[0]]]

        for x in d[1:]:
            if x - m[-1][0] < avg:
                m[-1].append(x)
            else:
                m.append([x])
```

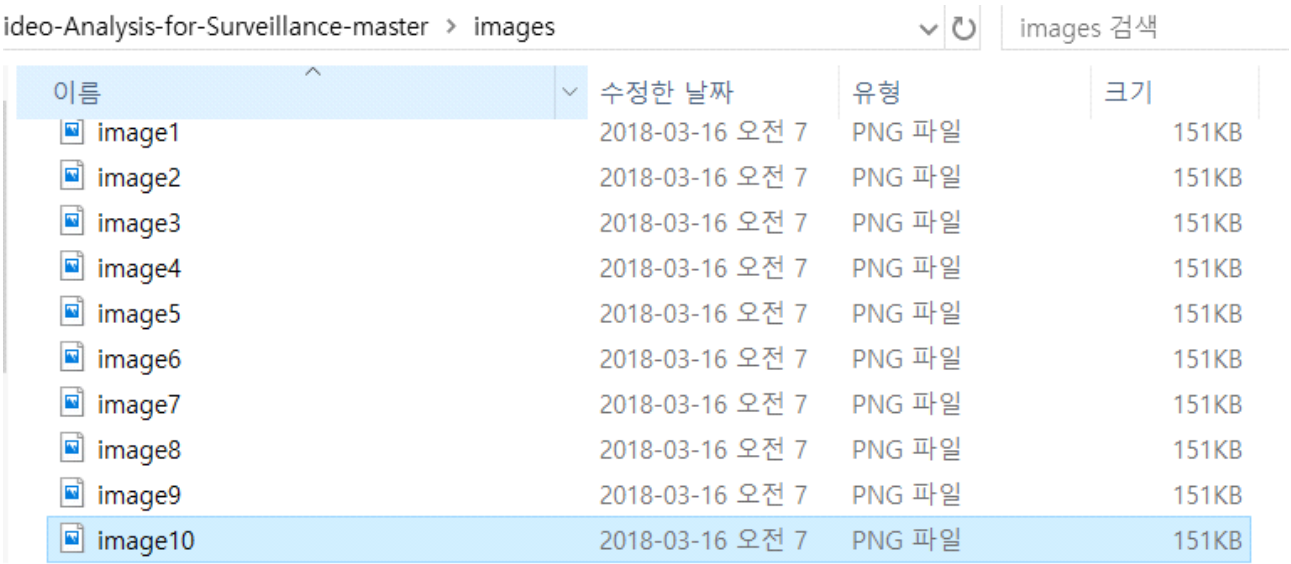
다음으로 classify.py 모듈에서 image 폴더안에 저장된 프레임들을 반복문을 통해 앞뒤 프레임이 같은 경우를 제외하고 나머지 프레임들의 시간을 출력한다.

```
:~/Video-Analysis-for-Surveillance/videos$ cd ..
:~/Video-Analysis-for-Surveillance$ python extract.py ./videos -o ./images/ --skip 75
```

```
Extracting frames from the video.....
Please Wait...


[h264 @ 0x3190ee0] error while decoding MB 50 44, bytestream -28
100% Extracted
Begin Analysis.....
Please Wait....


Printing the time of Interesting Events.....
```




```
.
.
Time  :  0:01:39
Time  :  0:01:42
.
.
Time  :  0:01:51
Time  :  0:01:54
Time  :  0:01:57
Time  :  0:02:00
Time  :  0:02:03
Time  :  0:02:06
```

위 사진은 출력결과 사진임. images 폴더안에 프레임 사진들이 저장되었고 콘솔창에 event가 일어난 부분의 시간을 출력하고 있음.

2. Object Detection

: 각 각의 프레임들에 대해 오브젝트를 추출하는 과정임. 이는 YOLO알고리즘을 사용할 예정임. YOLO 알고리즘은 처리속도가 매우 빠르지만 정확도가 낮다는 단점이 있음. 하지만 이 프로젝트에서는 오브젝트 알고리즘 후 얼굴인식을 한번 더 진행하기 때문에 정확도가 낮다는 단점을 보안하는 것이 가능함.

*YOLO(You Only Look Once) 알고리즘

: YOLO 알고리즘은 이미지 내의 bounding box와 class probability를 single regression problem으로 간주하여, 이미지를 한 번 보는 것으로 object의 종류와 위치를 추측함.

-YOLO의 처리과정

1. Input image를 S X S grid로 나눈다.

2. 각각의 grid cell은 B개의 bounding box와 각 bounding box에 대한 confidence score를 갖 는다. (만약 cell에 object가 존재하지 않는다면 confidence score는 0이 된다.)

Confidence Score: Pr(Object)∗IOUtruthpred

3. 각각의 grid cell은 C개의 conditional class probability를 갖는다.

Conditional Class Probability: Pr(Classi|Object)

4. 각각의 bounding box는 x, y, w, h, confidence로 구성된다.

(x,y): Bounding box의 중심점을 의미하며, grid cell의 범위에 대한 상대값이 입력된다.

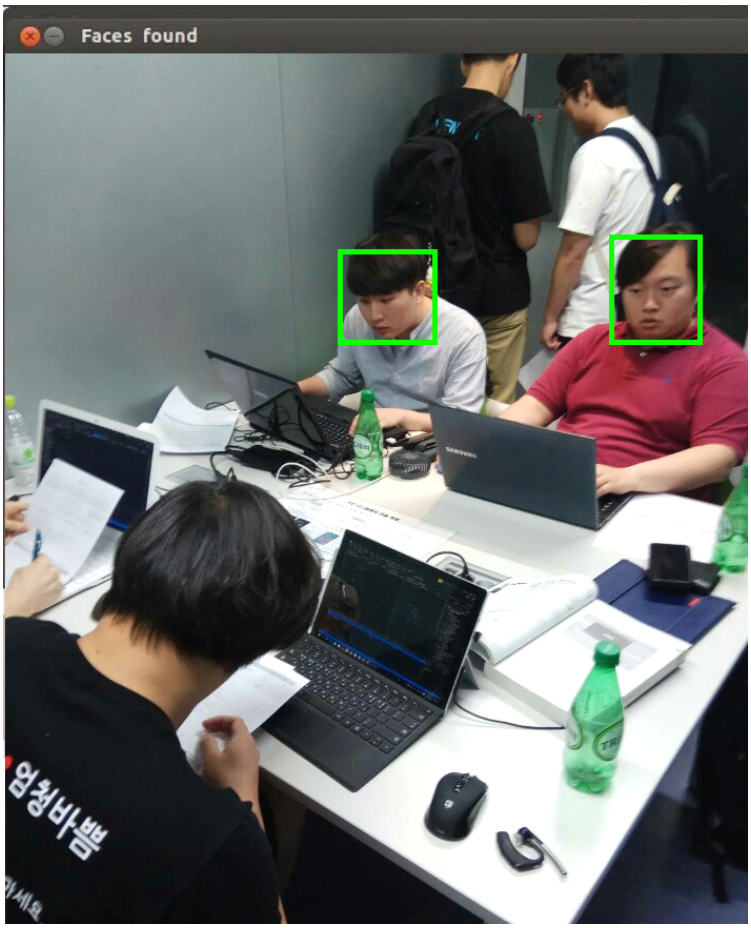
(w,h): 전체 이미지의 width, height에 대한 상대값이 입력된다.

*Test time에는 conditional class probability와 bounding box의 confidence score를 곱하여 class-specific confidence score를 얻는다.

ClassSpecificConfidenceScore=ConditionalClassProbability∗ConfidenceScore=Pr(Classi|Object)∗Pr(Object)∗IOUtruthpred=Pr(Classi)∗IOUtruthpred

3. Face Detection

: 영상의 프레임과 범죄자의 사진으로부터 얼굴부분을 찾는 과정임. 이 부분은 기존의 알고리즘을 사용하되 추가적으로 발전시켜 한계를 극복하는 것이 목표임. Github에서 가장 인지도있는 얼굴인식 코드인 'ageitgey/face_recognition' 를 사용하며 여러 한계 중 '멀리 있어 작은 얼굴에 대한 인식'과 '조명이 어두운 경우의 인식'에 대해 추가적인 알고리즘을 구현할 예정임.



위 사진은 'ageitgey/face_recognition' 오픈소스를 통해 얼굴인식한 모습임. 아직 알고리즘에 대한 이해는 하지못했으며 실습만 해보았음.

4. Face feature Detection

: 범죄자 사진에서 얼굴부분을 찾았다면 해당 얼굴 부분에서 특징을 도출하는 과정임. 수행과정은 먼저 Matlab을 통해 범죄자의 이미지를 intensity image(강도 이미지)로 바꿈. Intensity image란 data matrix로서 이미지의 강도를 숫자로서 표현한 데이터이다.

다음으로 intesity image를 PST(Phase Stretch Transform)을 이용해 계산함. PST란 신호 및 이미지 처리에 대한 계산방식 중 하나로서 감지 및 분류에 유용함. 이 시스템에서는 PST가 intensity image의 숫자들 중에 급격하게 변화하는 부분을 찾는데에 사용됨. PST function의 경우 UCLA의 Jalali lab에서 개발한 함수를 오픈소스로 사용할 예정이다.

아직 코드로서 구현한 부분은 없으며 추가적인 알고리즘 공부도 더 필요함.

5. Feature Deep Learning

: 이 과정에서는 도출한 특징을 Deep Learning하는 단계임. 아직 이에 대해서 메커니즘과 알고리즘을 정확히 세우지는 못했지만 관련하여 MNIST(손으로 쓴 글씨들의 DB)를 머신러닝하는 것에 대해서 배우고 실습 코딩을 해봤음.

*MNIST- 일치하는 숫자 찾기

각 이미지는 28X28X1 pixel인 train data를 보고 학습하는 코드 작성.
input image가 28X28X1이므로 크기가 784이며, GradientDescentOptimaizer에 cost가 가장 낮도록 학습.
결과는 아래와 같음.

```
Epoch: 0001 cost = 2.868104637
Epoch: 0002 cost = 1.134684615
Epoch: 0003 cost = 0.908220728
Epoch: 0004 cost = 0.794199896
Epoch: 0005 cost = 0.721815854
Epoch: 0006 cost = 0.670184430
Epoch: 0007 cost = 0.630576546
Epoch: 0008 cost = 0.598888191
Epoch: 0009 cost = 0.573027079
Epoch: 0010 cost = 0.550497213
Epoch: 0011 cost = 0.532001859
Epoch: 0012 cost = 0.515517795
Epoch: 0013 cost = 0.501175288
Epoch: 0014 cost = 0.488425370
Epoch: 0015 cost = 0.476968593
Learning finished
Accuracy:  0.888
```

정확도가 거의 89%까지 나옴.
다음은 학습한 걸 토대로 직접 test하는 코드 작성.

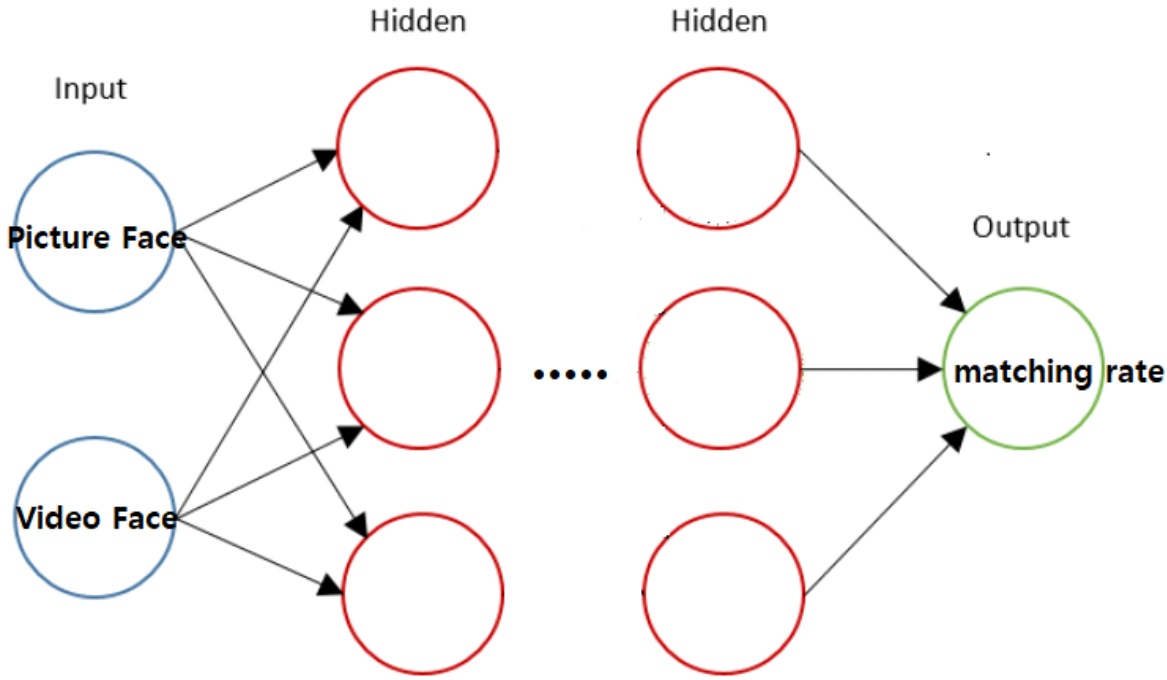
결과는 아래와 같음.

```
Label:  [3]
Prediction: [3]
```

6. Feature-Face Comparing

: 도출한 특징을 머신러닝 한 것을 토대로 영상의 얼굴들을 비교하는 과정임. 아직 앞에 과정인 특징 도출과 머신러닝을 완료하지 못했기 때문에 별다른 진행사항은 없음. 관련하여 시중에 있느 오픈소스들의 알고리즘에 대해 공부한 결과, Neral Network 라이브러리인 Keras를 사용하여 CNN(Convolution Neral Network) 형태로 설계할 예정. 입력값은 사진의 얼굴, 영상의 얼굴이며 출력

값은 2개의 일치율이다. Hidden Layer 에 대해서는 아직 설계하지 못함.



나. 중간 결과

: 지금까지의 진행사항은 서버와 관련해서는 DB를 구축하여 이미지를 저장하는 것이 가능하다. 이후 서버 구축이 필요한 상황임.

시스템과 관련해서는 영상의 필요없는 프레임을 잘라내는 시스템을 아무런 움직임이 없는 경우에 한해서 간단하게 구현함. 하지만 Complexity를 더 줄이기 위해서는 추가적인 예외처리가 필요함. 다음으로 얼굴인식 시스템은 오픈소스를 사용해보고 출력을 확인해봄. '어두운 경우', '크기가 작은 경우'에 대한 향상을 위해서 추가적인 알고리즘 개선이 필요함.

다. 향후 계획

분류	12월				1월				2월				3월			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
영상 처리 스터디(openCV 등)																

DataBase 서버 구현																
딥러닝 모델 훈련																
실험 및 오류 수정																

3. 과제의 평가

가. 개선 방안

축적된 영상을 분석하므로 길이가 긴 영상을 프로그램에서 다루게 됨. 이때, 상용화와 성능 면에서 강점을 갖기 위하여 시간을 줄이는 것이 우선. 시간을 줄이기 위해서는 불필요한 부분을 자르는 작업을 수행해야 함. 아직까지의 프로그램 진행정도는 움직임이 없는 영상을 줄이는 것 까지만 설계하도록 진행함. 그러나 그럼에도 불필요한 부분이 매우 많은 것을 확인할 수 있었음. 따라서 추가로 다른 방법들을 구현, 적용한다면 프로그램의 실행으로 불필요한 시간을 더욱 줄일 수 있을 것이라고 예상.

얼굴 인식을 하는 부분에서는 대부분 어두운 상황에서 인식을 진행해야 하는 경우가 많음. 이에 따라 어두운 환경에서 얼굴을 더욱 정확히 인식할 수 있도록 하여 프로그램의 신뢰성을 향상함. 또한, CCTV는 위치상으로 주로 높은 곳에 설치되어 있는 경우가 대다수임. 그래서 화면에 나타나는 인물들을 식별할 수 있는 얼굴들은 형태가 비교적 작을 것임. 따라서 작은 얼굴을 인식하는 부분도 개선하여 적용.

이러한 개선방안들을 개선하기 위하여 불변하는 특징점을 찾는 SURF 알고리즘을 적용할 예정. DataBase server에서 learning data set에 사진을 그대로 저장하기에는 많은 크기가 필요하며 불필요한 size가 생길 가능성이 있음. 이를 개선하기 위하여 웹서버를 하나 구현하여 url을 DataBase에 저장하는 방법을 생각하고 있음.

나. 기대 효과

머신러닝 방식으로 데이터의 인식과 식별에 대한 프로그램의 동작이 실행되기 때문에 빅데이터를 활용하면 더욱 정확한 데이터의 인식/ 식별이 가능할 것으로 예상됨.

서버를 구축하여 프로그램이 동작

CCTV, 블랙박스 등 영상매체를 통해 특정 인물을 추적하는 일이 인력을 통해 이루어지지 않고 프로그램으로 실행되면서 이에 따라 정확성과 신속성이 증가할 것으로 예측됨.

또한 이에 따라 기존의 시스템에 투입되는 불필요한 인력을 줄이면서 더욱 효율적인 시스템운영이 가능할 것임.

이 뿐만 아니라 아래와 같은 다양한 분야에서 해당 기술을 활용할 수 있을 것임.

[a. 실종자 수색: 실종자를 수색하는 과정에서 현재 경찰인력을 이용해서 실종자를 추적하고 있음. 그런데 이때, 광범위한 영상데이터를 분석하여 실종자를 수색한다면 더욱더 높은 실종자 추적률을 보일 것으로 예상됨.]

[b. 노무관리: 프로그램으로 통하여 노동자의 업무실태를 관리/ 감독. 얼굴인식 프로그램을 통해서 노무에 대한 관리를 진행함으로 좀 더 정확히 노동자들의 업무실태를 관리 할 수 있을 것이고 이에 따라 노동력은 더욱 향상될 것임.]

[c. 임의 단체의 회원관리: 등록된 이미지가 인식되는 회원을 식별하여 시설관리. 기존의 보안카드 발급, 회수, 관리/지문인식 등 생체인식 절차의 간소화할 수 있을 것임.]

[d. 보안강화: 기존의 보안카드, 지문인식 등의 보안 시스템과 해당 인식기술을 같이 실행하여 보안의 안전성을 강화. 나아가 금융, 전자상거래, 개인정보보호 등 더욱 안전성이 요구되는 분야에 대한 프로그램의 적용으로 좀 더 안전하게 데이터를 관리.]

[e. 소외계층 감소: 디지털 금융서비스를 사용하는 이용자들에게 신분식별 촉진을 강조. 그런데 신분 확인이 안되어 기본적으로 금융서비스를 받지 못하는 사람이 전세계 약 15억명으로 추산됨. 이들에게 얼굴인식 서비스를 제공해 신분확인을 진행하여 금융서비스 제공.]

다. 기타 보고사항

***참조**

[1]위키피디아-transaction
[2]http://opendatabase.tistory.com/entry/ORACLE-vs-MSSQL-vs-MariaDB
[3],[4]https://docs.google.com/presentation/d/1cVwqMpERToATs1JGYps0F3MLARP8OAlw6ZIe-lpPHYs/edit#slide=id.g1b2fb7758e_0_269
[5]http://blog.naver.com/PostView.nhn?blogId=msnayana&logNo=80121151044&redirect=Dlog&widgetTypeCall=true
[6],[9]https://m.blog.naver.com/PostView.nhn?blogId=laonple&logNo=220913997108&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F
[7],[8]http://rex0725.tistory.com/9
[9] PST(Phase Stretch Transform)
: https://en.wikipedia.org/wiki/Phase_stretch_transform
[10]intensity image
: https://edoras.sdsu.edu/doc/matlab/toolbox/images/intro6.html