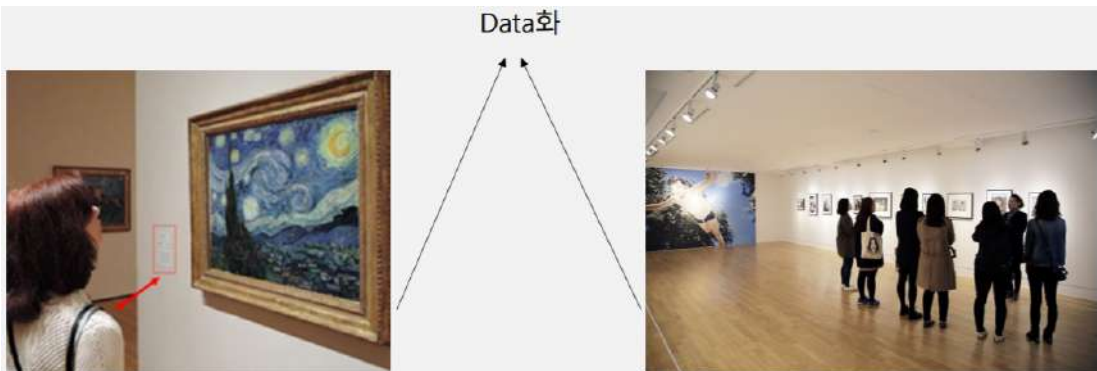




## 산학협력 프로젝트 결과보고서

과제명	영상분석을 통한 방문객 분석프로그램 개발		
협력기관명	바네컴퍼니	과제멘토	장영민
책임교수	고석주	소속	컴퓨터학부
참여인원	(총 06명) 기업체 01명, 참여교수 01명, 학부과정 04명		
수행기간	2020.09.02. ~ 20.12.07	유형	DIP
추진배경	<ul style="list-style-type: none"> <li>○ 미술품 관람객의 정보를 데이터화 하여 마케팅에 활용하기 위해 영상 처리 기술(OpenCV + Deep learning)을 사용하여 관람객의 정보를 데이터화 함.</li> <li>○ 가격 측정에 기준이 없는 미술품에 수요와 공급의 법칙을 적용시키기 위하여 프로젝트를 진행.</li> <li>○ 다른 분야에서 또한 활용하여 소비자의 정보를 데이터화 시켜 기업의 물품 마케팅에 소비자의 정보 데이터를 사용.</li> </ul>		
목표 및 내용	<ul style="list-style-type: none"> <li>○ 목표 : 미술품에 스마트 네임택을 설치하여 미술품에 대한 관람객의 반응 영상을 분석하고 Data화 시켜 미술품의 수요를 파악하여 가치를 정량화 하기위한 목적.</li> </ul> <div style="text-align: center;">  <p style="text-align: center;">Data화</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>스마트 네임택 :표정, 연령대, 성별</p> </div> <div style="text-align: center;">  <p>관람객 동선 확인</p> </div> </div> </div> <ul style="list-style-type: none"> <li>○ 개발 알고리즘 : <ul style="list-style-type: none"> <li>1) 표정 인식 알고리즘 <ul style="list-style-type: none"> <li>- MTCNN + BKNet을 이용한 표정 인식</li> </ul> </li> <li>2) 방문자 동선 추적 알고리즘 개발 <ul style="list-style-type: none"> <li>- Yolo V4를 통해 사람 탐지 후 frame별로 Dot을 찍어 동선 추적 및 방문객 빈도 확인</li> </ul> </li> <li>3) 사람 검지 알고리즘 개발 <ul style="list-style-type: none"> <li>- Yolo V4 + tensorflow로 사람 검지</li> </ul> </li> <li>4) 사람 수 계산 알고리즘 개발 <ul style="list-style-type: none"> <li>- dlib + openCV으로 해당 작품을 관람한 관람객 인원 측정</li> </ul> </li> <li>5) 방문자 나이대와 성별 파악 <ul style="list-style-type: none"> <li>- CNN(Convolutional Neural Network)기반 Multitask learning Model로 분석</li> </ul> </li> <li>6) 방문객 당 작품 별 체류시간 측정 <ul style="list-style-type: none"> <li>- dlib + openCV를 통해 관람객 구분 후 해당 작품 별 체류시간 계산</li> </ul> </li> </ul> </li> </ul>		

## 1. 과제 수행 배경

### 1) 미술품 가격을 매기는 부정확한 기준

- 2017년 4월, 국내 한 경매사의 미술품 경매에서 김환기 작가의 ‘고요(Tranquillity)’라는 작품이 한국 미술품 경매 최고가인 65억5000만원에 낙찰됐다.
- 현재 미술품의 가격을 측정하는 데 ‘호당가격제’, 보존상태, 작품이력, 희소성 등의 측정 기준들이 있다. 하지만 이 측정 기준들은 수긍할 수 없는 기준이다. 예를 들어, ‘호당가격제’의 경우 같은 작가의 작품이라도 완성도의 차이가 극명하게 남에도 불구하고 **획일적으로 가격이 매겨지는 경우가 있다.**
- 특히 그림 값의 결정은 한 작가의 작품이라 할지라도 시대마다 작품마다 **유동적이다**. 결국 미술품 가격은 사는 사람이 마음에 드는 작품을 손안에 넣을 수 있을 때까지 내는 것이 가격이라는 것이 정설이다. 즉 그림의 값은 **구매하는 사람이 결정한다**는 말이다.

### 2) 미술품의 부정적인 사용

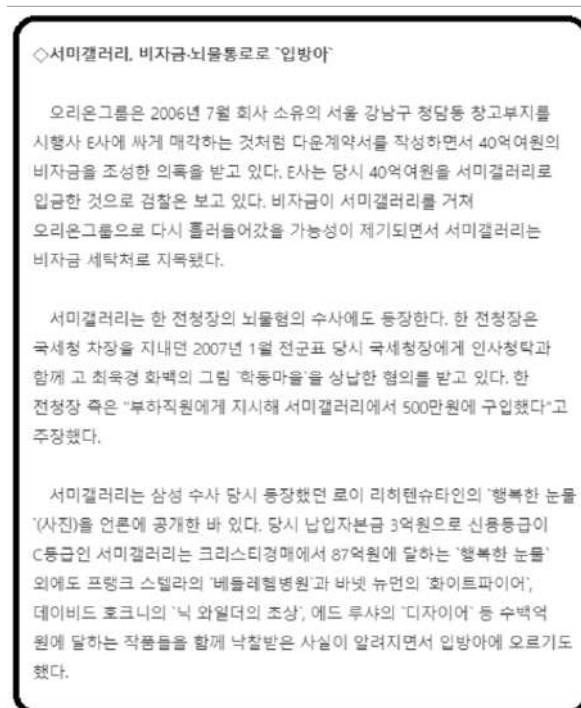


그림1-1. 미술품의 불법 면세

- 미술품은 특히 불법 면세나 돈세탁의 용도로 사용되는 경우가 많음
- 그림1-1의 경우 미술품이 면세임을 이용해 편법적으로 세금을 납부하지 않는 방법으로 미술품을 활용함



그림1-2. 미술품의 부적합한 용도(돈세탁)

- 그림1-2의 경우, 고가의 미술품을 사들여 되파는 방식으로, 미술품을 돈세탁으로 사용함
- 미술품의 부정적인 사용으로 인해 미술품 가격 책정에 합리적인 기준 필요성이 등장
- 미술품 가격 책정에 합리적인 기준 등장 시 미술품의 부정적인 사용 용도가 제거
- 일반인 또한 미술품의 가격 책정에 참여하여 미술품을 감상하고 소유 할 수 있음
- 고착화된 미술품 시장의 확장과 환기 시도 가능

## 2. 과제 목표

### 1) 미술품의 합리적인 가격 책정 기준 데이터 생성

- 관람객의 반응과 특정 행동을 데이터화 하여 제품의 소요를 구체화
- 면세와 편법 상속 및 돈세탁에 더 이상 미술품을 사용 할 수 없음
- 미술품 시장을 투명하게 관리하여 긍정적인 반응을 이끌어 낼 수 있음

### 2) 시제품의 소비자 반응을 파악하는데 사용 가능

- 미국의 b8ta회사가 좋은 예시(그림 2. 미국의 스타트업 'b8ta' 참고)
- 고객들의 반응을 분석하여 제품의 제조사에게 데이터를 제공
- 데이터를 제공받은 제조사는 데이터를 참조하여 시제품 보완 가능



그림 2. 미국의 스타트업 'b8ta'

### 3. 추진 방법 및 구성도

#### 3-1) 추진 방법

- 스마트 네임택은 작고 관람객에게 방해가 되지 않는 크기의 카메라를 사용



그림 3-1. 스마트 네임택(예상도)



그림3-2. 스마트 네임택(구현)

- 구현한 스마트네임택은 3d프린터로 제작
- 시즌마다 바뀌는 미술관에서 재활용하기 용이함
- 스마트 네임택은 라즈베리파이V4 + 카메라 모듈을 사용
- 미술품을 감상하는 미술관 관람객의 정보(나이, 성별 등)을 실시간으

로 서버로 전송하여 프로그램을 통해 관람객의 정보를 분석

### 3-2) 시스템 구성도

#### 3-2-1) 전체 구성도

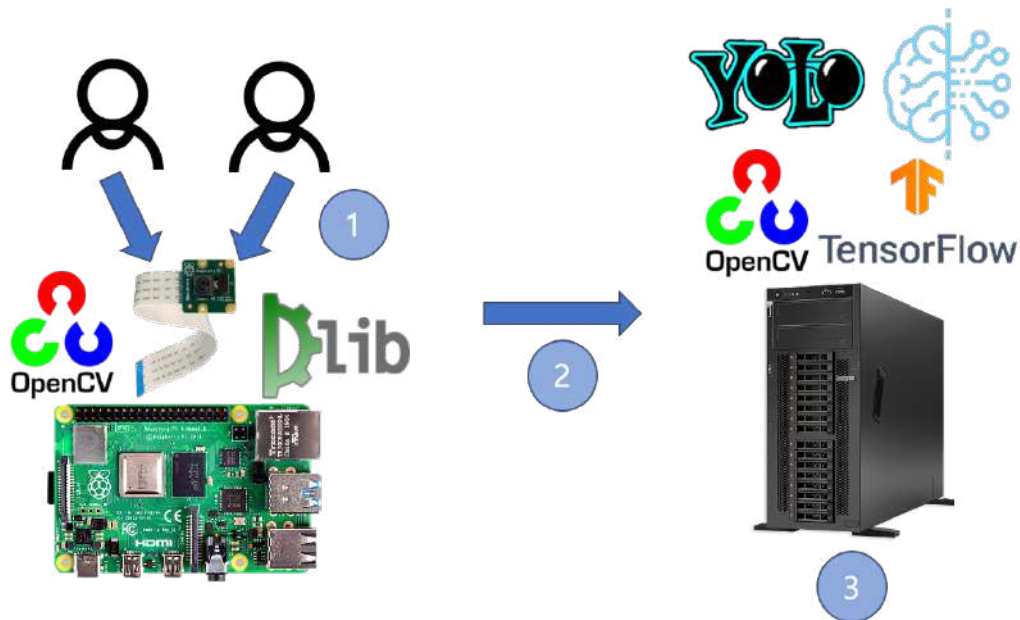


그림 3-3. 프로젝트 구성도

#### 3-2-2) 이미지 처리 개요도

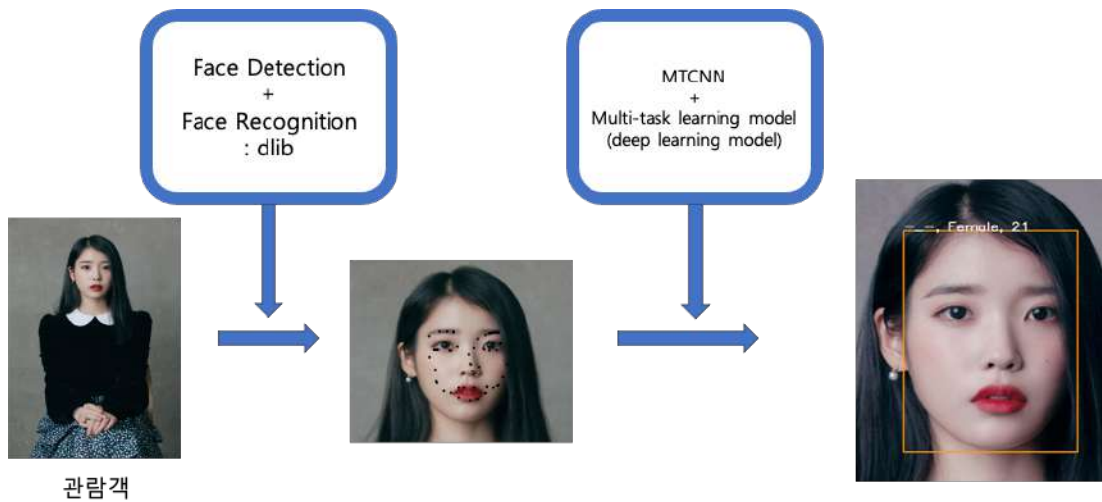


그림 3-4. 이미지 processing 처리 단계

- 그림 3-4는 관람객의 얼굴 인식을 한 후, 관람객의 정보(표정, 연령, 성별)를 추출하기까지의 일련의 과정을 보여줌



## 4. 과제 수행 결과

### 4-1) Raspberry Pi

- 스마트네임택이라는 환경을 구축하기 위해 여러 가지 사안이 제시되었다.
- 가장 맨 처음 사안으로는 **OpenMV**를 사용하는 카메라 모듈 하나만으로도 얼굴인식이 되는 장치를 사용하기로 하였으나, 장치의 성능이 너무 떨어지고 마이크로파이썬이라는 생소한 프로그래밍 언어로 인해 자체 개발하여 사용하기로 결정하였다.
- 차후 사안으로, 스마트네임택을 구현하기 위해 **IoT서비스**의 대표 주자인 라즈베리파이에 카메라 모듈을 장착하기로 하였다.
- 라즈베리파이는 원가 절감의 이득을 노리기 위해 라즈베리파이3로 사용하였으나, 일반적인 OpenCV를 사용하는게 아닌 dlib을 통한 얼굴 인식 기능까지 구현해야 하기 때문에 dlib사용시 영상처리 과정에서 약 6초 정도의 딜레이가 발생하여 **라즈베리파이4**로 교체하였다.



제품		
제품 명	Raspberry Pi 4 B	Raspberry Pi 3 B+
CPU	1.5-GHz, Quad-Core Broadcom BCM2711B0 (Cortex A-72)	1.4-GHz, Quad Core Broadcom BCM2837B0 (Cortex A-53)
RAM	1GB/2GB/4GB LPDDR4-2400 SD RAM(depending on model)	1GB LPDDR2 SDRAM
GPU	500 MHz VideoCore VI	400 MHz VideoCore IV

표4-1-1. 라즈베리파이3와 라즈베리파이4 비교

- 라즈베리파이에 사용하는 카메라 모듈은 라즈베리파이에서 공식 지원

을 해주는 카메라 모듈 8MP V2를 사용하였다.



그림4-1-1. 8MP V2

- 카메라 모듈을 라즈베리파이의 CSI Port에 연결해 사용하고, 라즈베리파이 부팅 후, 아래의 과정에따라 환경 설정 실행을 실행해준다.

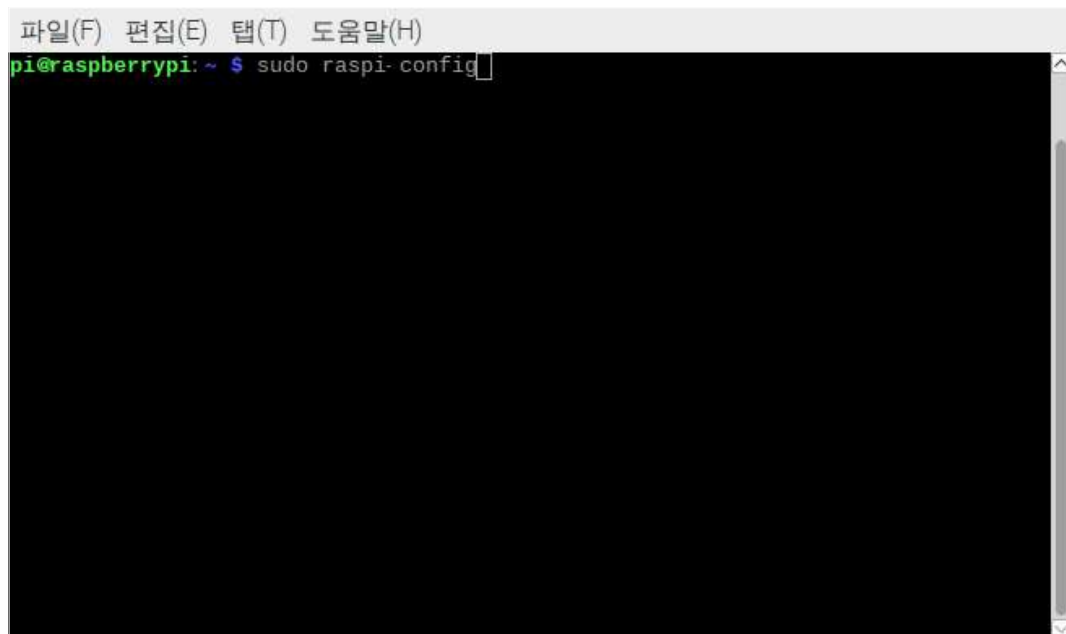


그림4-1-2. 환경설정 단계

- 라즈베리파이는 기본적으로 Camera 설정이 꺼져 있기 때문에 Enable로 바꾸어 주기 위해 라즈베리파이의 CMD창에서 관리자 권한으로 `raspi-config` 커맨드를 입력해준다.





그림4-1-3. 환경설정 단계

- 그 후 **Camera Tab**으로 이동해 Camera를 Enable로 설정해준다.
- 위의 과정을 통해 라즈베리파이에서 카메라 모듈을 사용할 준비를 마치게 되면, 라즈베리파이에서 성공적으로 카메라모듈을 인식하게 된다. 이후, 라즈베리파이는 파이썬에서 **OpenCV**를 통해 카메라 모듈의 Data Stream을 불러오고 서버와의 통신을 실시한다.

## 사용코드

```
1  import cv2, dlib, sys
2  import numpy as np
3  import socket
4  import argparse
5  import threading
6  import sys
7  import time
8  import os
9  from os.path import exists
10
11
12  port = 10000
13  host = "192.168.43.104"
14
15  def getFileSize(Filename):
16      filesize = os.path.getsize(Filename)
17      print(filesize)
18      return str(filesize)
19
20  scaler = 0.7
```

그림 4-1-4. Client.py중 일부

- 라즈베리파이에서 실행하는 Client.py의 사용 라이브러리이다.
- 주된 라이브러리는 OpenCV와 dlib을 사용한다.
- OpenCV를 통해 Camera의 DataStream을 읽어 들어와 영상처리를 가능하게 한다.
- 포트는 10000 port를 사용하여 통신하였다.
- host에 Server의 ip주소를 입력하였다.
- **GetFileSize(Filename)** 함수를 통해 내가 보낼 이미지 File의 Size를 서버로 전송하게 되고, 서버는 File의 Size만큼의 Data를 라즈베리파이에서 받아서 이를 모아 이미지로 저장하게 된다.

```

20 scaler = 0.7
21
22 detector = dlib.get_frontal_face_detector()
23 predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
24
25 # load video
26 cap = cv2.VideoCapture(0)
27
28 face_roi = []
29 face_sizes = []
30 img_count = 0
31
32
33 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
34 client_socket.connect((host,port))
35 print("connect sucess!")

```

#### 그림 4-1-5. Client.py중 일부

- cap은 cv2.VideoCapture(0)을 통해 라즈베리파이 카메라모듈의 DataStream을 조작 할 수 있다.
- 또한 Server와의 Data 통신을 위해 Socket통신을 실시한다.
- Socket의 ip주소와 port를 지정하여 Socket통신을 열어준다. 기본적으로 Client입장에서는 1:1 connect TCP 통신으로 연결한다.
- Socket을 통해 Server에 cap에서 불러들여온 DataStream을 전송해 준다.

```

38 while True:
39     ret, img = cap.read()
40     if not ret:
41         break
42
43     img = cv2.resize(img, (int(img.shape[1] * scaler), int(img.shape[0] * scaler)))
44     ori = img.copy()
45
46     # detect faces
47     faces = detector(img)
48     try:
49         face = faces[0]
50         print(face)
51         dlib_shape = predictor(img, face)
52         shape_2d = np.array([[p.x, p.y] for p in dlib_shape.parts()])
53
54         top_left = np.min(shape_2d, axis=0)
55         bottom_right = np.max(shape_2d, axis=0)
56
57         face_size = max(bottom_right - top_left)
58
59         center_x, center_y = np.mean(shape_2d, axis=0).astype(np.int)
60
61         img_count+=1
62         imgname = './img/'+str(img_count)+'.jpg'
63         cv2.imwrite(imgname, img)
64         fd = open(imgname, 'rb')
65         client_socket.send(getFileSize(imgname).encode())
66         client_socket.recv(32)
67         b = fd.read()
68         client_socket.send(b)
69
70         img = cv2.rectangle(img, pt1=(face.left(), face.top()), pt2=(face.right(), face.bottom()), color=(255, 255, 255),
71                             thickness=2, lineType=cv2.LINE_AA)
72
73         for s in shape_2d:
74             cv2.circle(img, center=tuple(s), radius=1, color=(255, 255, 255), thickness=2, lineType=cv2.LINE_AA)
75
76         cv2.circle(img, center=tuple(top_left), radius=1, color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)
77         cv2.circle(img, center=tuple(bottom_right), radius=1, color=(255, 0, 0), thickness=2, lineType=cv2.LINE_AA)
78
79         cv2.circle(img, center=tuple((center_x, center_y)), radius=1, color=(0, 0, 255), thickness=2, lineType=cv2.LINE_AA)
80
81     except IndexError:
82         pass
83     cv2.imshow('img', img)
84     # cv2.imshow('result', result)
85     cv2.waitKey(1)
86
87 client_socket.close()

```

#### 그림 4-1-6. Client.py중 일부

- 라즈베리파이는 작동중, 서버로 끊임없이 비디오 화면을 캡처해서 보내게된다.

- 일반적으로 비디오 스트림을 계속 보낼 수 있겠지만, dlib을 이용하여 만약 사람의 얼굴이 인식이 될 경우만 이미지를 Server로 보내게 되어 불필요한 이미지 생성을 막는다. dlib의 기능 설명은 추후 4-3) Detection에서 진행하도록 하겠다.

- dlib을 통해 얼굴을 인식하게 된다면 사진의 크기를 Server에서 가공하기 위한 크기로 조절해주고, 이를 img 폴더 아래에 순차적으로 저장하게 된다. 이미지 파일은 텍스트 파일에 비해 크기가 큰 편이기 때문에 Server에서 성공적으로 받기 위해선 먼저 file의 크기를 얻어오고, file의 크기를 Server로 보내고 난 뒤, Server에서 file의 크기만큼 받아오게 한다.

## 4-2) Server

```
1  import socket
2  import argparse
3  import threading
4  import time
5  import os
6  import shutil
7
8  host = ''
9  port = 10000
10 user_list = []
11 notice_flag = 0
12
13 def msg_func(msg):
14     print(msg)
15     for con in user_list.values():
16         try:
17             con.send(msg.encode('utf-8'))
18         except:
19             print("연결이 비 정상적으로 종료된 소켓 발견")
20             exit()
```

그림 4-2-1. Server.py 중 일부

- Server는 자신의 ip주소로 연결을 요청하는 Client를 연결해서 Data만 받아주면 되기 때문에 따로 OpenCV를 사용하지 않는다.
- User\_list를 통해 Client들의 Socket을 관리한다.

```
while 1:
    client_socket, addr = server_socket.accept()

    user_list.append(addr)

    #accept()함수로 입력만 받아주고 이후 알고리즘은 핸들러에게 맡긴다.
    notice_thread = threading.Thread(target=handle_notice, args=(client_socket, addr))
    notice_thread.daemon = True
    notice_thread.start()

    receive_thread = threading.Thread(target=handle_receive, args=(client_socket, addr))
    receive_thread.daemon = True
    receive_thread.start()
```

그림 4-2-2. Server.py중 일부

- Server는 기본적으로 다수의 Client에게 Data를 받아야 하기 때문에 thread로 Client를 관리하게 해준다.

```

23 def handle_receive(client_socket, addr):
24     print("connect with " + str(addr))
25     count = 0
26     string = str(addr)
27     #ip와 socketnum 얻어오기
28
29     dir_name = os.getcwd()
30     dir_name += "/Client" +str(user_list.index(addr))
31     #받아온 client별 폴더 이름 지정
32
33     try:
34         shutil.rmtree(dir_name) #디렉토리가 존재하면 삭제
35     except Exception as e:
36         print(dir_name)
37
38     os.makedirs(dir_name)#디렉토리 생성
39
40     while 1:
41         file_length = client_socket.recv(1024)
42         #file 크기를 받는 과정
43         client_socket.send(str.encode())
44         file_length = file_length.decode()
45         #file_length에는 server가 받아온 file 크기가 저장됨
46         #print(file_length)
47         if not file_length:
48             break
49
50         length = 0
51         data = []
52         #file을 붙이는 과정
53         while 1:
54             d = client_socket.recv(4096)
55             data.append(d)
56             #data에 4096바이트씩 받아오면서 연결해줌
57             length += len(d)
58             #print(length)
59             if int(length) == int(file_length):
60                 #print('break')
61                 break
62
63         stri = dir_name + "/" + string + str(count) + ".jpg" #jpg 파일 생성
64         count+=1
65         #print(str(addr) + " : " + stri)
66         with open(stri, 'wb') as f:
67             for a in data:
68                 f.write(a)
69
70     print("disconnect with " + str(addr))
71     user_list.remove(addr)

```

그림 4-2-3. Server.py중 일부

- Client의 연결이 요청되면 handle\_receive함수를 통해 연결을 진행하고 Client별로 폴더를 만들어 그 폴더 안에 이미지 파일로 저장하게 된다.

- Client에게 File의 크기를 먼저 받게 되고, 그 File의 크기만큼 Data를 전달 받게 되면 Data를 jpg형식으로 Client 폴더안에 저장하게 된다.



### 4-3) Detection

#### Face Detection



그림 4-3-1. dlib

- dlib 라이브러리는 C++로 작성된 toolkit이지만, python 패키지로도 설치해 사용할 수 있다. 특히 HOG(Histogram of Oriented Gradients)을 사용하여 얼굴 검출하는 기능이 많이 사용되고 있다.
- HOG는 픽셀 값의 변화로 파악할 수 있는 **영상의 밝기가 변하는 방향을 그래디언트(gradient)**로 표현하고, 이로부터 **객체의 형태**를 찾아낼 수 있다.
- dlib은 사람의 얼굴에 **68개의 점(특징점)**을 찍고, 점을 기반으로 추출하여 점의 위치를 128개의 벡터화(이는 deep learning의 결과물)를 통하여 저장할 수 있다. 만약 같은 사람의 얼굴이 입력 값으로 주어진다면, 128개의 벡터가 비슷한 숫자가 나온다. 즉, 나온 숫자를 바탕으로 유사성을 구할 수 있으며, 이를 활용하여, **사람 인식(Recognition)**에도 활용하였다.
- 이러한 과정을 구현하기 위해, dlib 패키지의 **face\_locations()** 함수를 활용하여 **68개의 점**을 추출한다. 또한, **face\_encodings()** 함수는 dlib와 numpy에 쉽게 접근할 수 있도록 wrapping해 주는데, 이 함수의 결과물로서 **128개의 벡터값**들이 제공된다.
- **사람의 얼굴을 구분**하기 위해서는 128개의 벡터값들을 서로 비교하

여 같은 사람인지 다른 사람인지를 판단해야한다. 이러한 벡터 값들을 유사도를 비교하기 위해서 **Euclidean distance(유클리드 거리)**를 이용하여 두 벡터의 간의 거리를 구하였다. 이러한 기능은 face\_recognition의 **face\_distance()** 함수가 지원을 해준다. 구해진 거리의 결과 값은 0~1사이 값으로 나온다. 구해진 거리의 결과 값을 통해 기존에 저장되어 있던 벡터들과 새로운 입력 값의 벡터를 비교하여 만약, 가장 높은 숫자를 보유하면서 **Similarity Threshold(유사 임계값)**보다 낮다면 같은 얼굴로 판단하며, Similarity Threshold보다 높다면 다른 얼굴, 즉, 기존의 사람이 아닌 새로운 사람이라고 판단한다.

- 보통 서양인의 얼굴에서는 **Similarity Threshold의 수치**는 0.4~0.45가 적당하다고 알려져있으며, 프로젝트에서는 동양인 위주 얼굴인점과 라즈베리파이의 카메라 화질을 고려하여 Similarity Threshold의 수치를 **0.35로 설정**하였다. 이때, 카메라 화질을 고려하는 이유는, face\_encodings()을 통해 구해진 벡터들은 얼굴의 표정, 각도, 이미지 크기, 조명 상태 등 여러 변수에 의해 달라지기 때문에, Euclidean distance로 구한 값이 낮아질 수 있기 때문이다.

- 현재 얼굴 탐색에서 가장 많이 활용되고 있지만, 이외에도 보행자 검출 등에 활용할 수 있다.

## 사용코드

```
for q in range(2): # range(~~~) -> client 수
    if src_file == "0":
        total_img = './client'+str(q)
        frame_img = os.listdir(total_img)
        frame_img.sort()
        src_file = './client'+str(q) + '/' + frame_img[0]

    src = cv2.VideoCapture(src_file)

    if not src.isOpened():
        print("cannot open inputfile", src_file)
        exit(1)
```

그림 4-3-2. 작품의 수만큼 분석하는 code

- 기능을 구현하기 위해, **face\_classifier.py**와 **person\_db.py**를 구현했다.
- 그림 4-3-2는 각 작품마다 찍힌 frame을 분석하기 위해 작품 별로

분석하기 위해 경로를 설정하는 과정이다.

- 그림 4-3-3은 person\_db.py를 통해 기존에 분석한 사람의 얼굴을 face\_encodings()을 통해 벡터화 한 값을 파일로 저장하였으며, pdb.load\_db(result\_dir)를 통해 기존에 저장된 벡터를 불러온다. 이를 바탕으로, 새로운 입력 값과 비교할 준비를 한다.

```
# person DB 불러오기
result_dir = "result" + str(q)
pdb = PersonDB()
pdb.load_db(result_dir)
pdb.print_persons()
```

그림 4-3-3. 기존 저장된 벡터를 불러오는 code

- 라즈베리파이로부터 받은 이미지에서 얼굴 인식하는 방법으로 face\_recognition의 face\_encodings를 통해 인식을 하며, 해당 값을 faces list에 저장한다. 이때, 인식된 얼굴이 여러개일 경우 faces list에 여러개의 값이 저장된다. 그 후 faces list를 반환한다.

```
def detect_faces(self, frame, frame_name):
    boxes = self.locate_faces(frame)
    if len(boxes) == 0:
        return []

    # faces found
    faces = []
    now = datetime.now()
    str_ms = now.strftime('%Y%m%d_%H%M%S.%f')[:-3] + '-'
    encodings = face_recognition.face_encodings(frame, boxes)
    for i, box in enumerate(boxes):
        face_image = self.get_face_image(frame, box)
        face = Face(frame_name, face_image, encodings[i])
        face.location = box
        faces.append(face)
    return faces
```

그림 4-3-4. 얼굴 인식 code

```

def compare_with_known_persons(self, face, persons):
    if len(persons) == 0:
        return None

    # 유클리드 거리 구하기
    encodings = [person.encoding for person in persons]
    distances = face_recognition.face_distance(encodings, face.encoding)
    index = np.argmin(distances)
    min_value = distances[index] # distance 최소값 구하기
    # distance의 최소값이 similarity_threshold보다 작으면 같은 사람의 얼굴임
    if min_value < self.similarity_threshold:
        persons[index].add_face(face)
        persons[index].calculate_average_encoding() # 얼굴의 face_encoding의 평균
        face.name = persons[index].name
    return persons[index]

```

그림 4-3-4. 얼굴 비교 code

- 그림 4-3-4의 `compare_with_known_persons()`를 통해 새롭게 얼굴이 인식되면, 기존에 알고 있던 사람들의 정보의 `face_encodings()`과 유클리드 거리를 사용하여 값을 구한다. 이때, 가장 거리가 가까운 사람의 거리가 `similarity_threshold`보다 작으면 같은 얼굴이라고 판단하고, `person_DB`에 얼굴을 추가한다. 그 후 새로 추가된 얼굴을 포함하여 `face_encodings()`을 업데이트한다.

- 그림 4-3-5를 통해 `compare_with_unknown_faces()`를 통해 모르는 얼굴은 `unknown_faces`에 따로 저장 해놓는다. 새로 인식된 얼굴과 `unknown_faces`과의 유클리드 거리를 구한다. 이때, 가장 가까운 얼굴와의 distance가 `similarity_threshold` 보다 작다면 두 얼굴은 같은 사람의 얼굴이라 판단하고, 새로운 사람을 만든다. 하지만, distance가 `similarity_threshold` 보다 크다면 `unknown_faces`에 추가한다.

```

def compare_with_unknown_faces(self, face, unknown_faces):
    if len(unknown_faces) == 0:
        # this is the first face
        unknown_faces.append(face)
        face.name = "unknown"
        return

    encodings = [face.encoding for face in unknown_faces]
    distances = face_recognition.face_distance(encodings, face.encoding)
    index = np.argmin(distances)
    min_value = distances[index]

    if min_value < self.similarity_threshold:
        # two faces are similar -> create new person with two faces
        person = Person()
        newly_known_face = unknown_faces.pop(index)
        person.add_face(newly_known_face)
        person.add_face(face)
        person.calculate_average_encoding()
        face.name = person.name
        newly_known_face.name = person.name
        return person
    else:
        # unknown face
        unknown_faces.append(face)
        face.name = "unknown"
        return None

```

그림 4-3-5. 얼굴 비교 code

- person\_db 모듈은 Face, Person, PersonDB Class로 구성되어있다.

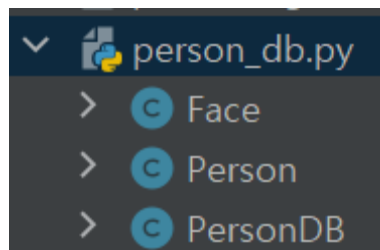


그림 4-3-6. person\_db 구성

- **Face Class**의 기능으로는 얼굴을 **pickle** 라이브러리를 사용하여 파일로 저장하는 기능과 **pickle** 라이브러리로 저장된 **face\_encodings** 정보들을 읽어온다.

- **Person Class**에서 **add\_face()**는 입력값의 **face**를 저장하는 기능이며, **calculate\_average\_encoding()**은 새롭게 저장된 **face**의 **encodings**값을 기존의 저장되어있던 **encodings** 값들과 함께 평균을 내어 해당 사람의 얼굴 정보를 정확하게 해주고자 설계되었다. **distance\_statistics()**는 기존에 저장되어 있는 **face\_encodings** 정보들과 입력으로 들어온 **face\_encodings**의 값과의 모든 유클리드 거리를 구하여, 그 중 가장 최소 값을 반환한다. **save\_faces()**는 **faces**라는 기존에 저장되어 **encodings** 정보들 리스트에 입력 받은 **encodings** 값을 저장한다.

- **PersonDB Class**는 **load\_db()**를 통해 **pickle**로 저장된 기존의 정보를 읽어오며, **save\_encodings()**을 통해 **dir\_name**의 경로에 **encodings** 정보들을 **pickle**로 저장한다. **save\_db()**를 통해 **directory** 경로를 받아 해당 경로가 없다면 폴더를 만들고 그 안에 이미지를 저장하며 있다면 그 경로에 바로 이미지를 저장한다. 이 때, 저장되는 이미지의 이름은 이후 설명할 체류 시간을 구하기 위하여 라즈베리파이로부터 받은 이미지 파일의 이름이 아닌, 해당 파일의 **metadata** 정보인 '만든 시간'을 가져와 이름을 주었다.

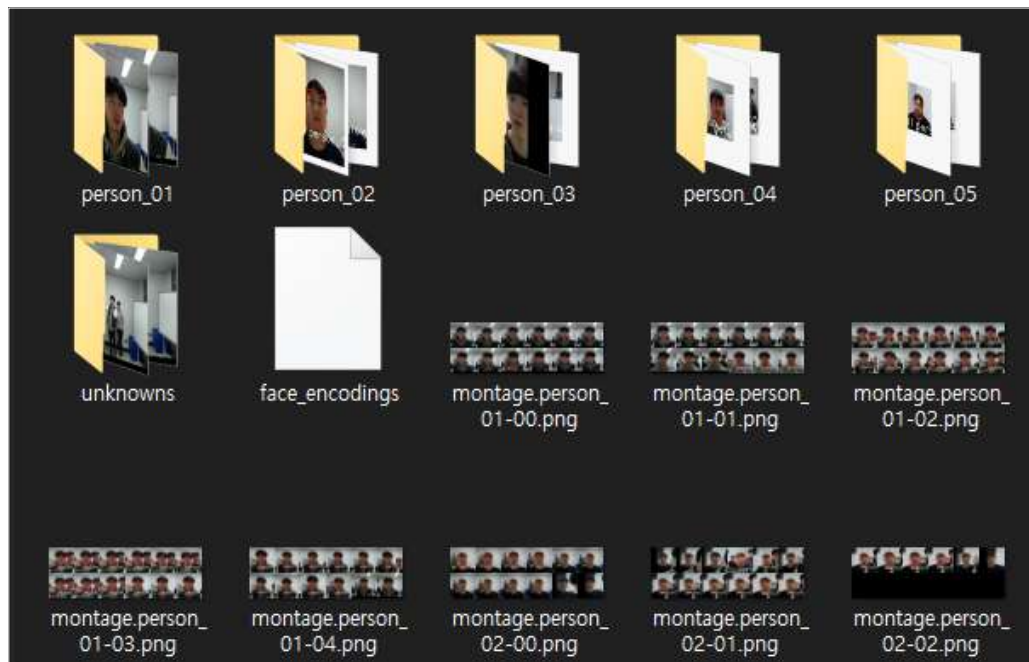


그림 4-3-7. 디렉토리 별 구분

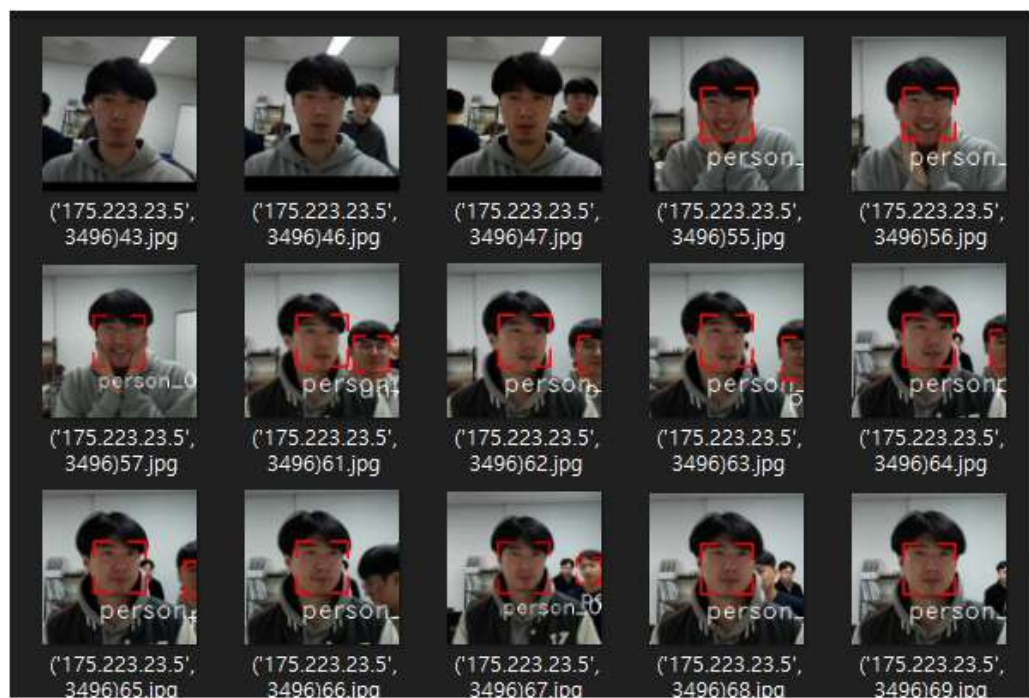


그림 4-3-8. 구분된 디렉토리 내부



#### 4-4) 동선 분석

##### Tensorflow



그림 4-4-1. TensorFlow

- **Tensorflow**는 구글이 개발하고 오픈소스로 공개한 머신러닝 프레임워크이다. 머신러닝과 딥러닝을 일반인도 쉽게 사용할 수 있도록 기능들을 제공하고 있고, 현재 머신러닝 프레임워크 중 가장 높은 점유율을 차지하고 있다. GPU버전은 NVIDIA사의 CUDA를 사용하여 training 속도를 가속화할 수 있다. Tensorflow 2.0이 출시되면서 딥러닝 초보자들이 쉽게 접근하고, 사용할 수 있는 ‘Keras’ 프레임워크가 내장되며 딥러닝 프레임워크의 대세가 되었다. 우리 프로젝트에서는 Tensorflow 2.3.0version을 사용하였다.

##### Tensor 자료형

- **Tensor**는 Numpy의 ndarrays와 유사한 것으로 GPU에서도 사용할 수 있다. Tensor는 N차원 배열이며, Tensorflow는 Tensor 연산을 위한 다양한 함수들을 제공한다.

##### YOLO(You Only Look Once)

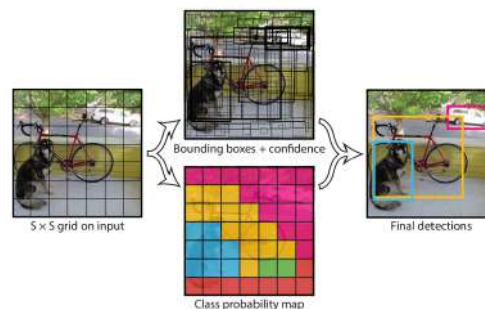


그림 4-4-2. YOLO

- 본 프로젝트에서는 딥러닝에 기반한 object detection algorithm으로 **YOLOv4**를 사용했다. YOLO는 sliding window 방식을 사용하는 object detection algorithm들의 높은 연산량으로 인한 **속도 저하 문제**를 해결한 딥러닝 모델이다.
- YOLO는 **Real-time object detection** algorithm으로 다른 algorithm보다 더 뛰어난 처리 속도를 보여준다. 기존의 sliding window방식은 이미지보다 작은 Window를 계속 이동시키며 Image와의 연산을 통해 얻은 score를 이용하여 object detection을 수행하였다. Score를 한 번 만에 계산하지 못하고 여러 번 계산해야 하는 다른 딥러닝의 algorithm 특성상 속도 측면에서 overhead가 있을 수밖에 없는 방식이었다.
- YOLO는 **Real-time object detection algorithm**으로는 기존의 algorithm보다 더 뛰어난 처리 속도를 보여준다. 기존의 sliding window방식은 이미지보다 작은 Window를 계속 이동시키며 Image와의 연산을 통해 얻은 **score**를 이용하여 **object detection**을 수행하였다. Score를 여러 번 계산해야 하는 algorithm의 특성 상 속도 측면에서 **overhead**가 있을 수밖에 없는 방식이었다.
- YOLO는 input image를 균일한 region으로 나눈 뒤, 나눠진 region에 대하여 bounding box와 class probability를 하나의 regression 문제로 간주하여 **단 한 번만 이미지를 보고 예측**한다. Image에 대한 연산이 한 번만 이루어져도 된다는 장점 때문에 YOLO는 기존의 다른 Algorithm들 보다 **월등히 빠른 속도**를 가질 수 있게 되었다.
- 현재 YOLO의 가장 높은 버전은 **v5**이다. 하지만 v5의 경우, 본 프로젝트에서 사용하는 프레임워크인 Tensorflow가 아닌 **Pytorch**로 구현이 되어있어 사용할 수 없었다. 따라서, 본 프로젝트는 91개의 class로 구성되어 있는 coco dataset에서 Average Precision이 약 43%인 v4를 사용하였다.

```

pred = [interpreter.get_tensor(output_details[i]['index']) for i in range(len(output_details))]
if FLAGS.model == 'yolo_v3' and FLAGS.tiny == True:
    boxes, pred_conf = filter_boxes(pred[1], pred[0], score_threshold=0.25, input_shape=tf.constant([input_size, input_size]))
else:
    boxes, pred_conf = filter_boxes(pred[0], pred[1], score_threshold=0.25, input_shape=tf.constant([input_size, input_size]))
else:
    saved_model_loaded = tf.saved_model.load(FLAGS.weights, tags=[tag_constants.SERVING])
    infer = saved_model_loaded.signatures['serving_default']
    batch_data = tf.constant(images_data)
    pred_bbox = infer(batch_data)
    for key, value in pred_bbox.items():
        boxes = value[:, :, 0:4]
        pred_conf = value[:, :, 4:]

boxes, scores, classes, valid_detections = tf.image.combined_non_max_suppression(
    boxes=tf.reshape(boxes, (tf.shape(boxes)[0], -1, 1, 4)),
    scores=tf.reshape(
        pred_conf, (tf.shape(pred_conf)[0], -1, tf.shape(pred_conf)[-1])),
    max_output_size_per_class=50,
    max_total_size=50,
    iou_threshold=FLAGS.iou,
    score_threshold=FLAGS.score
)

pred_bbox = [boxes.numpy(), scores.numpy(), classes.numpy(), valid_detections.numpy()]

image2 = cv2.imread('./data/white.png') # pred_bbox[0][0][0] 좌표 1 정확도 2 클래스 '0'이 사탕
for i in range(len(pred_bbox[0][0])):
    # print(1)
    if pred_bbox[0][0][i][0] == 0.:
        continue
    if pred_bbox[2][0][i] == 0.:
        x=int(((pred_bbox[0][0][i][1]+pred_bbox[0][0][i][3]) / 2)*1024)
        y=int(((pred_bbox[0][0][i][0]+pred_bbox[0][0][i][2]) / 2)*478)
        image2 = cv2.circle(image2, (x,y), 5, (0,0,255), -1)
cv2.imwrite('./data/white.png', image2)
image = utils.draw_bbox(original_image, pred_bbox)
# image = utils.draw_bbox(image_data*255, pred_bbox)
image = Image.fromarray(image.astype(np.uint8))
image.show()
image = cv2.cvtColor(np.array(image), cv2.COLOR_BGR2RGB)
cv2.imwrite(FLAGS.output, image)

```

그림 4-4-3. Yolo를 사용한 동선분석 code

- combined\_non\_max\_suppression() 함수를 이용해서 batch\_data(image파일의 데이터)를 tensorflow값 boxes(인식된 물체의 x,y좌표를 저장하는 box값), scores(인식된 물체의 정확도), classes(인식된 물체의 종류), valid\_detection(물체가 유효한지 아닌지 판별하는 값)로 변환시킨 후에 numpy함수를 통해서 실제 사용할수 있는 값으로 변환을 해준다.
- 좌표값으로는 x최소값, y최소값, x최대값, y최대값의 순서로 들어가게 되며 여기서 x값은 실제로 우리가 인식하는 x축과는 반대인 y축의 값이기 때문에 x에는 2,4번째 값을 넣게 되고 y에는 1,3번째 값을 넣어 계산을 하게한다.
- 또한 x,y좌표가 최상단 좌측을 0,0을 기준으로 최하단 우측을 1,1로 잡고 0~1사이의 비율값이 들어가기 때문에 '좌표값\*사진의 해상도'를 계산하면 실제 사진 상의 위치의 좌표가 찍히게 된다.

- 동선분석 결과

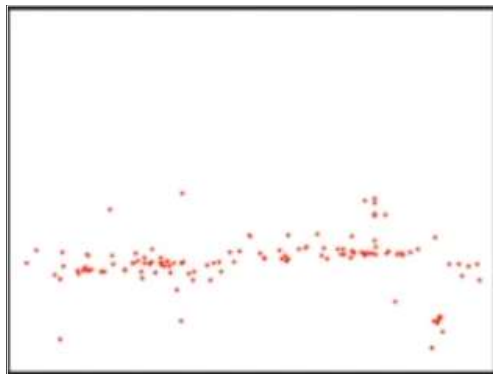
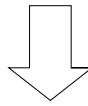
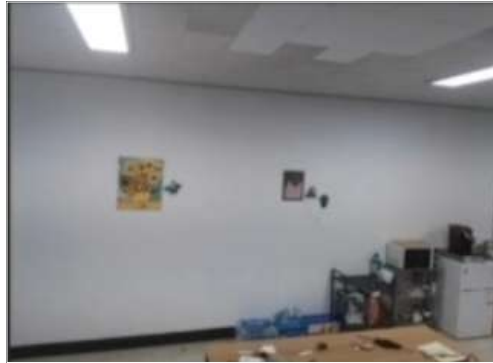


그림4-4-4. 동선분석 결과

- 위와 같이 관람객이 머무른 자리에 점이 찍히게 되고, 이 정보로 관람객이 많이 머무른 구간을 확인할 수 있다.

#### 4-5) 표정, 나이 및 성별 분석

##### Multi-task learning model

- 본 프로젝트에서 관람객의 정보를 분석하는데 가장 핵심적인 부분을 담당하는 딥러닝 모델이다. 본 프로젝트는 **관람객을 분석한 데이터**를 가격 책정 등 마케팅에서 사용할 수 있도록 데이터화 하는 것이 목표이기 때문에 실시간분석으로 수행할 필요가 없다. 따라서, 후처리를 수행하지만 그럼에도 분석을 효율적으로 수행해야한다. 다시말해 분석 프로그램이 너무 무겁지 않아야 한다.

- 현존하는 딥러닝 모델 중 이미지를 분석하는 데는 **CNN(Convolutional Neural Network)**이 **최고의 성능**을 보여준다는 것은 자명한 사실이다. 하지만 보통의 CNN이라면 한 개의 클래스(카테고리)에서만 구분할 수 있다. 예를 들면, 동물의 종류, 사람인지 아닌지 등이다. 본 프로젝트에서 사용하는 multi-task learning의 경우, 세 가지 클래스인 연령대, 성별과 표정에 대해 한 번의 분석으로 결과를 얻을 수 있어서 세 가지의 딥러닝 모델을 각각 사용하는 것 보다 훨씬 효율적이고 **추론 속도** 또한 훨씬 빠르다.

- 우선 **성별**과 **연령대**, 그리고 **표정**에 대한 각각을 input으로 받고 이를 합쳐서 **CNN shared block**이라는 블록에 input으로 입력한다. 이 블록은 CNN의 모델 중 VGGNet, ResNet이나 DenseNet과 같은 모델로 구성될 수 있다. 이 shared block을 지난 후, 네트워크는 세가지 분기로 나누어지는데 이는 성별, 연령대, 표정이다. 각 분기별로 학습을 진행한 후 각각의 학습에 해당하는 loss function을 가진다. 각각의 분기들은 각각의 **loss function**으로 **back-propagation**을 수행하여 **오차를 줄이는 방향으로** 가중치들을 수정하며 학습을 한다. 이에 대한 파이프라인은 아래에 있다.

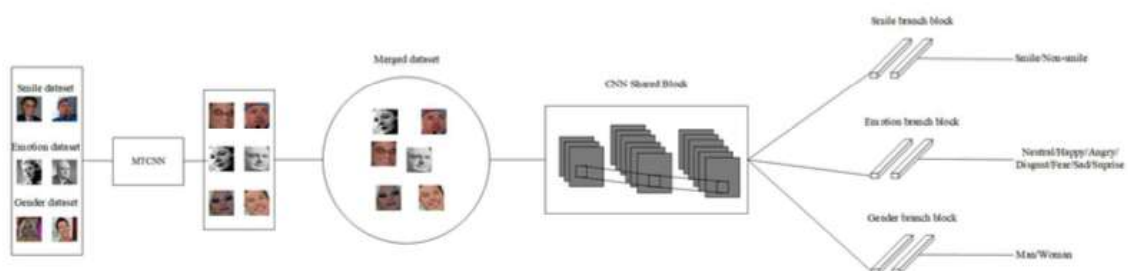


그림 4-5-1. Multi-task learning pipeline

- 본 모델은 다른 CNN모델과는 다르게 shared block을 사용하였는데, 이것이 이 모델의 성능을 더 높이는데 도움을 주었다. 이는 여러데이터 셋에서 공통된 특징들을 학습할 수 있다. 따라서, 단일 작업(1개의 클래스에 대한 분류작업)에 대한 모델보다 더 일반화되었고, 더 정확한 분석을 수행할 수 있게 되었다.

- 본 모델은 GoogLeNet의 inception module과 ResNet 그리고 VGGNet에서 영감을 얻어 shared block을 설계하였고, BKNet의 architecture를 사용하였다. 아래 그림 4-5-2는 **shared block의 구조**를 보여주고 있다.

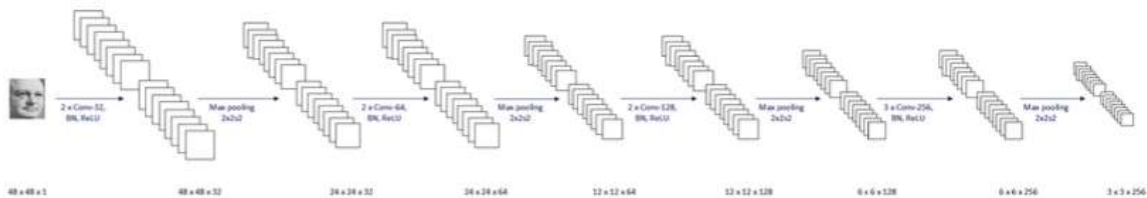


그림 4-5-2. CNN Shared Block for Multi-task learning

- 이 모델의 성능은 세 가지 기능으로 나뉜 만큼, 각각의 성능을 따로 측정해야 한다. 첫 번째로 표정 인식의 경우, **96.23%**의 정확도를 얻을 수 있었다. 이는 GENKI-4K와 FERC-2013과 IMDB의 데이터 셋을 사용하여 훈련한 결과이다. 다음으로 성별 구분의 경우 IMDB 데이터 셋을 사용하여 훈련을 수행했는데, **92.20%**의 정확도를 보였다. 마지막으로 연령대의 경우, 아래와 같이 데이터 셋 자체에 편중이 있었기 때문에 20~30대의 경우에 더 뛰어난 성능을 보였다.

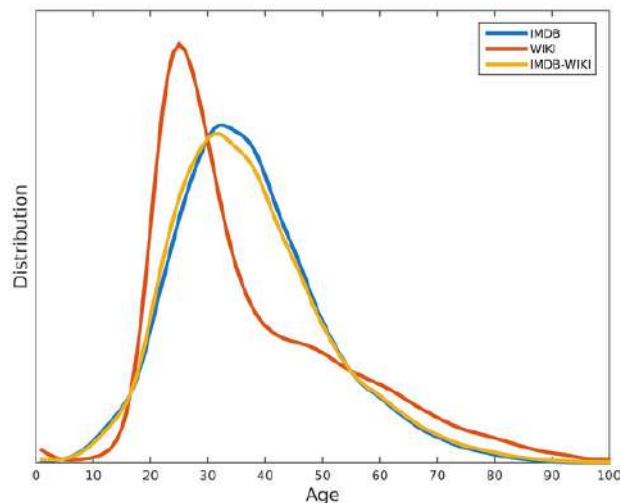


그림 4-5-3. Age's distribution of dataset

## 사용 코드

```
def load_network():
    gpu_options=tf.compat.v1.GPUOptions(per_process_gpu_memory_fraction=0.75)
    tf.compat.v1.ConfigProto()
    tf.compat.v1.disable_eager_execution()
    sess = tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(gpu_options=gpu_options))
    x = tf.compat.v1.placeholder(tf.float32, [None, 48, 48, 1])
    y_smile_conv, y_gender_conv, y_age_conv, phase_train, keep_prob = BKNetStyle.BKNetModel(x)
    print('Restore model')
    saver = tf.compat.v1.train.Saver()
    saver.restore(sess, './save/current5/model-age181.ckpt.index')
    print('OK')
    return sess, x, y_smile_conv, y_gender_conv, y_age_conv, phase_train, keep_prob

# In[3]:

def draw_label(image, x, y, w, h, label, font=cv2.FONT_HERSHEY_SIMPLEX, font_scale=1, thickness=2):
    cv2.rectangle(image, (x, y), (x+w, y+h), (0,255,255), 2)
    cv2.putText(image, label, (x,y), font, font_scale, (255, 255, 255), thickness)

# In[4]:

def main(sess, x, y_smile_conv, y_gender_conv, y_age_conv, phase_train, keep_prob):
    # capture video
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
    detector = MTCNN()
    count = 0
    while True:
        # get video frame
        img = cv2.imread('bae.jpg')

        # detect face and crop face, convert to gray, resize to 48x48
        original_img = img
        result = detector.detect_faces(original_img)
        if not result:
            cv2.imshow("result", original_img)
            continue
        face_position = result[0].get('box')
        x_coordinate = face_position[0]
        y_coordinate = face_position[1]
        w_coordinate = face_position[2]
        h_coordinate = face_position[3]
        img = original_img[y_coordinate:y_coordinate+h_coordinate, x_coordinate:x_coordinate+w_coordinate]
        if(img.size==8):
            cv2.imshow("result", original_img)
            continue
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img = cv2.resize(img, (48, 48))
        img = (img - 128) / 255.0
        T = np.zeros([48, 48, 1])
        T[:, :, 0] = img
        test_img = []
        test_img.append(T)
        test_img = np.asarray(test_img)

        predict_y_smile_conv = sess.run(y_smile_conv, feed_dict={x: test_img, phase_train: False, keep_prob: 1})
        predict_y_gender_conv = sess.run(y_gender_conv, feed_dict={x: test_img, phase_train: False, keep_prob: 1})
        predict_y_age_conv = sess.run(y_age_conv, feed_dict={x: test_img, phase_train: False, keep_prob: 1})

        smile_label = "--" if np.argmax(predict_y_smile_conv)==0 else ":"
        gender_label = "Female" if np.argmax(predict_y_gender_conv)==0 else "Male"
        argmax_predict_age = np.argmax(predict_y_age_conv)

        label = "{}, {}, {}".format(smile_label, gender_label, argmax_predict_age)
        draw_label(original_img, x_coordinate, y_coordinate, w_coordinate, h_coordinate, label)

        cv2.imshow("result", original_img)
```

그림 4-5-4. Multi-tasking learning model code



- load\_network()는 분석 전 훈련된 정보가 저장되어있는 가중치파일인 weight 파일과 딥러닝 모델 추론의 가속화를 위한 gpu 사용을 설정하는 단계이다.

- draw\_label()은 분석 후 관람객의 정보를 지정된 좌표에 사각형 및 분석된 관람객의 정보를 그리는 함수이다.

- 수행은 MTCNN()을 호출하여 입력된 이미지에서 얼굴을 인식한 후, 세 가지의 정보를 추출하는 딥러닝을 수행한다. 이때, 얼굴이 인식되지 않으면 바로 다음사진으로 건너뛰게 된다.

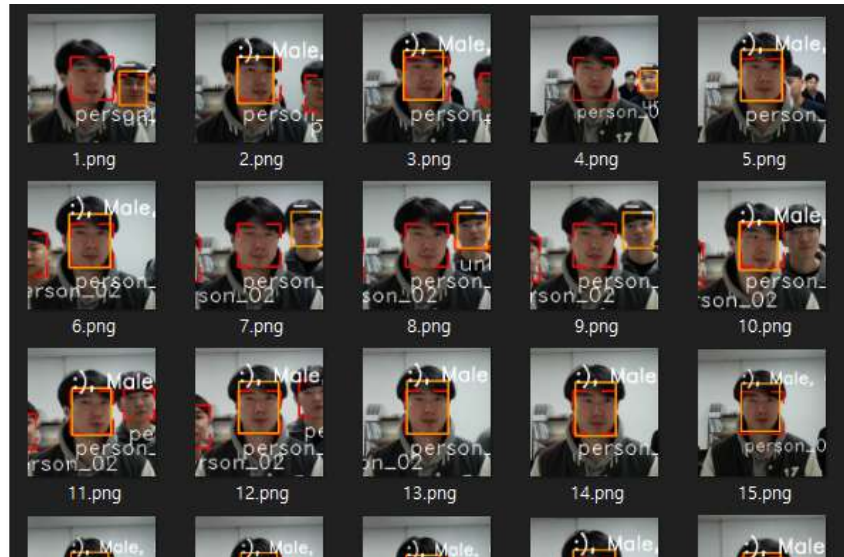


그림 4-5-5. 관람객 분석 후의 기준 이미지

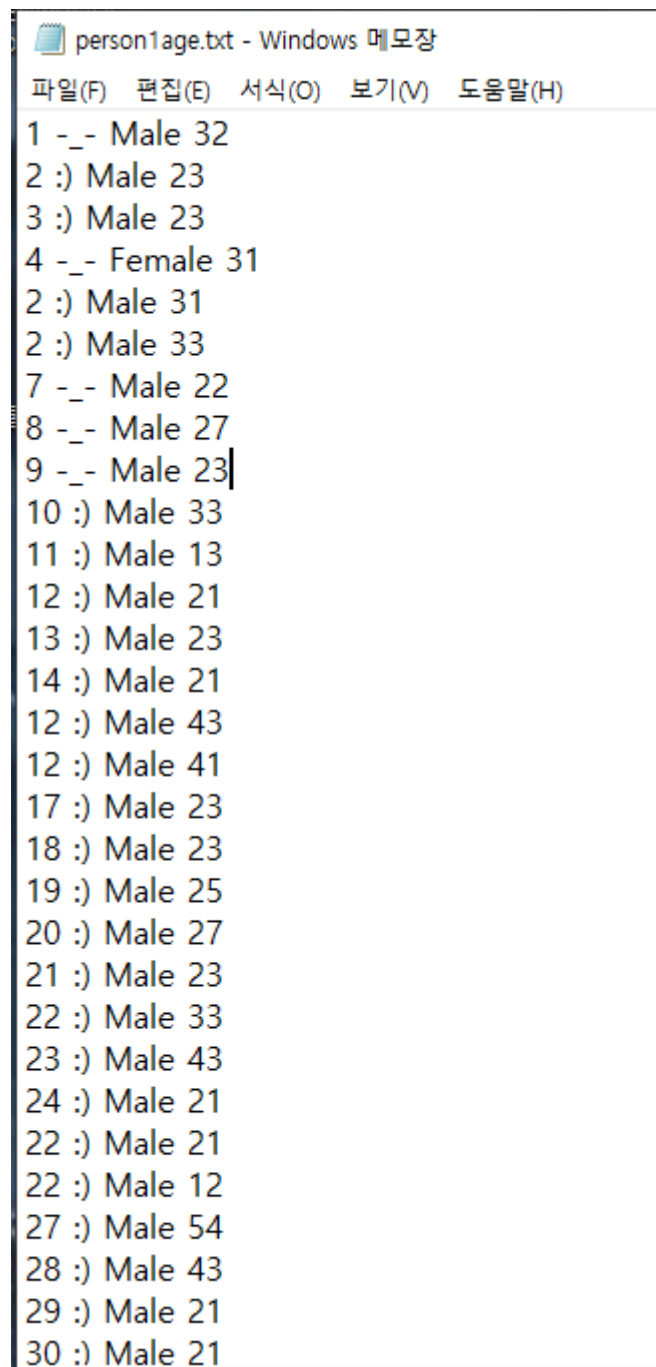


그림 4-5-6. 관람객 분석 결과(표정, 성별, 연령)

## 4-6) 체류 시간 분석

- 4-3) Detection 과정에서 라즈베리파이로부터 받은 image들의 이름을 각각의 사람별로 디렉터리에 구분해서 저장하며, 이때 image들의 이름은 라즈베리파이로가 해당 image를 저장한 시점의 시간으로 저장하기 위하여, metadata의 '만든 시간'을 'YYHHMM(연도,시간,분)'으로 저장하였다.

- 사람이 구분이 되어 있는 result 폴더의 경로로 간 후 personXX(X는 숫자)이라고 적힌 모든 디렉터리들을 탐방하여 각각의 디렉터리에 대한 이미지들의 파일명을 통해 체류시간을 분석한 뒤 이를 파일로 저장하였다.

- 체류 시간은 파일의 이름들을 전부 읽어 온 후, 오름차순으로 정렬한다. 그 후, 정렬된 시간들을 순차적으로 2개씩 비교하여 '나중에 찍힌 이미지의 시간 - 먼저 찍힌 이미지의 시간'을 구한다. 이때, 이 값이 10(초)가 넘지 않는다면 이는 계속 해당 라즈베리파이 앞에 있었다고 판단하여 체류시간에 더해진다. 그리하여, 리스트의 마지막이 되거나 10초가 넘는다면 총 더해진 체류시간을 계산하며 그 값이 체류시간이 되며 파일에 저장한다. 만약, 10초가 넘었다면 이는 잠시 자리를 비웠다가 다시 나타났다고 판단하여 이전의 누적되어있던 체류시간을 파일에 저장한다. 그리고 체류시간을 초기화하여 다시 위와 같은 알고리즘으로 체류시간을 구하게 된다.

## 사용코드

```
# 체류시간 계산하기
path_person_dir = './result' + str(q) + '/'

temp = os.listdir(path_person_dir)
dir_person = []
f = open('./stay_time/client'+str(q)+".txt", 'w') #체류시간 저장할 txt 이름

# person 디렉터리 이름 가져오기 -> dir_person에 저장
for i in range(len(temp)):
    if temp[i].startswith('person_'):
        dir_person.append(temp[i])

print("client"+str(q))
f.write("client"+str(q)+"\n")

for j in range(len(dir_person)):
    path_dir = './result' + str(q) + '/' + dir_person[j]
    file_list = os.listdir(path_dir)

    file_list.sort()
```

```

flag = 0
zero_flag = 0
start_time = 0
now_time = 0
print(dir_person[j])
f.write(dir_person[j]+"\n")
for i in range(len(file_list)):
    hms = file_list[i][0:6]

    if i == 0:
        start_time = (int(hms[0:2]) * 3600) + (int(hms[2:4]) * 60) + int(hms[4:6])
        flag += 1
        continue
    if flag == 0:
        bhms = file_list[i-1][0:6]
        start_time = (int(bhms[0:2]) * 3600) + (int(bhms[2:4]) * 60) + int(bhms[4:6])
        flag = 1
        zero_flag = 1
    if i != len(file_list) - 1:
        continue

    now_time = (int(hms[0:2]) * 3600) + (int(hms[2:4]) * 60) + int(hms[4:6])
    bhms = file_list[i-1][0:6]
    before_time = (int(bhms[0:2]) * 3600) + (int(bhms[2:4]) * 60) + int(bhms[4:6])

    if i == len(file_list) - 1:
        print(now_time - start_time)
        f.write(str(now_time - start_time)+"\n")

    if flag == 1 and now_time - before_time > 10: # 시간 분리 기준 == 10초가 넘으면 다른데 갔다왔다고 생각
        if zero_flag == 1:
            flag=0
            zero_flag=0
            f.write("0\n")
            continue
        residence_time = before_time - start_time
        print(residence_time)
        f.write(str(residence_time)+"\n")
        flag = 0
        zero_flag = 0

f.close()

```

그림 4-6-1. 체류 시간 분석 코드

client0.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) !

```

client0
person_01
18
15
person_02
8
5
person_03
2
person_04
4
person_05
9

```

그림 4-6-2. 작품1에서의  
관람객체류시간

## 4-7) 성별, 나이에 대한 Visualization

- 4-3)Detection과 4-5)표정, 나이 및 성별 분석을 통해서, 각각의 구분된 사람의 이미지들에 대한 분석된 표정과 나이 및 성별이 파일로 저장되어 저장되어있다.
- 각각의 사람에 대한 정보가 저장되어있는 파일들을 읽고, 저장되어있는 나이와 성별의 최빈값을 구한다. 이때, 최빈값으로 구한 이유는 흔들려서 찍힌 사진, 어두운 사진 등 환경에 따른 불안정한 값들을 최대한 제거하기 위해 최빈값으로 설정함으로 인해 정확도를 높이기 위해서이다.
- 시각화를 위해 matplotlib의 pyplot 라이브러리와 numpy 라이브러리를 이용하였다.

### 사용코드

```
import matplotlib.pyplot as plt
import numpy as np
import os

path_person_dir = './textfile/'
temp = os.listdir(path_person_dir)
dir_person = []
values = [0,0,0,0,0,0,0,0,0,0] #age 저장리스트
val = [0,0] #성별 저장리스트 index0 : male, index1 : female

# person 디렉터리 이름 가져오기 -> dir_person에 저장
for i in range(len(temp)):
    if temp[i].startswith('person'):
        dir_person.append(temp[i])
sex = {'Male': 0, 'Female': 0}

def f1(x):
    return sex[x]
```

```
for i in range(len(dir_person)):
    f = open(path_person_dir + dir_person[i], 'r')
    test = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    age = []
    print(dir_person[i])
    while True:
        line = f.readline()
        if not line:
            break
        person = line.split(" ")
        if len(person) < 3:
            continue
        elif len(person) >= 5:
            person = person[1:]
            #print(person[2])
            sex[person[2]] += 1
            age.append(int(person[3]))

    key_max = max(sex.keys(), key=f1) #Male or Female
    if(key_max == 'Male'):
        val[0] += 1
    else:
        val[1] += 1
```

```

for i in range(len(age)):
    test[int(age[i]/10)] += 1
print(test)

max_age=0
max_index=0
for i in range(len(test)):
    if max_age < test[i]:
        max_index = i
        max_age = test[i]

test.clear()
avg_age = round(sum(age_0.0)/len(age)) #평균 나이
values[max_index] += 1

x = np.arange(10)
years = ['0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '90-99']
plt.bar(x, values)
plt.xticks(x, years)
plt.show()

y = np.arange(2)
sex = ['Male', 'Female']
plt.bar(y, val)
plt.xticks(y, sex)
plt.show()

```

그림 4-7-1 성별,나이 시각화 사용 코드

- 나이, 성별에 대한 시각화 차트를 나타내면 아래와 같다

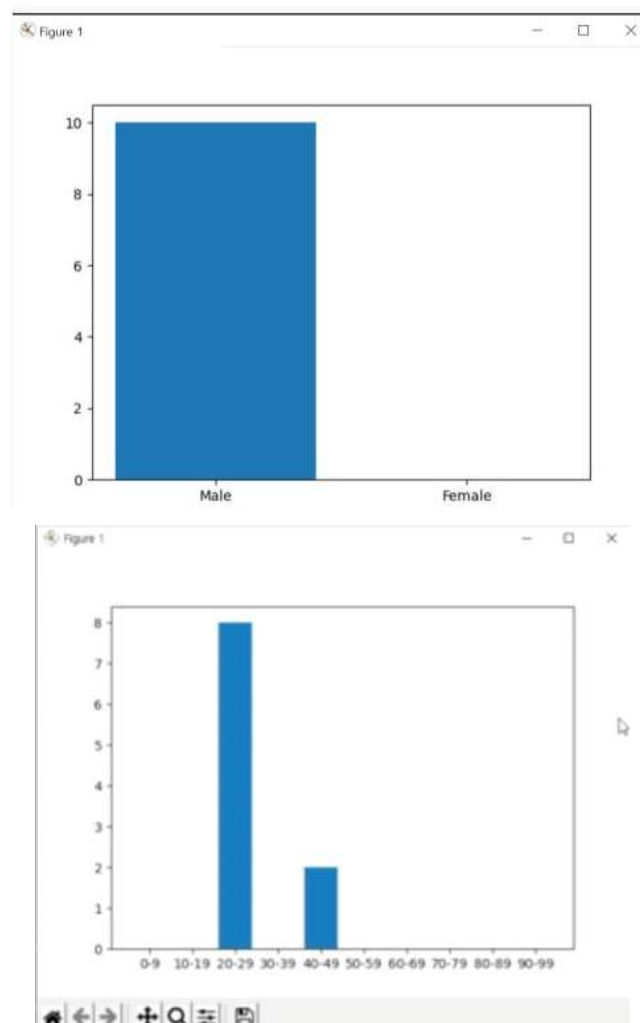


그림 4-7-2. 시연을 바탕으로 한  
성별, 나이그래프

## 5. 기대 효과 및 활용 방안

### 1) 관람객의 반응을 분석하여 미술품의 수요를 구체화

- 기존의 미술품의 가격은 보통 호당 가격으로 책정이 된다. 보통 엽서 크기를 1호라 칭하고, 세로의 크기의 변화에 따라 인물화, 풍경화, 해경화로 구분한다. 10호면 1호의 3배정도이고, 10호면 1호의 7배 정도이다. 이를 기준으로 작품의 가격이 책정된다. 하지만 이렇게 작품의 크기만으로 가격이 책정된다면 100호보다 10호의 작품이 미적 가치가 뛰어나다 하더라도 크기가 크다는 이유로 100호의 작품이 더 비싼 가격이 매겨질 것이다. 또한, 작품의 작가, 시대적 배경 등 여러 가지를 고려했을 때 호당가격은 실제 작품의 가치를 반영했다고 하기 어렵다. 따라서, 본 프로젝트는 미술품의 가치에 합리적인 가격책정 기준인 수요와 공급 체계를 적용하기 위해 수요자들의 반응을 데이터화 한다. 이는 미술품의 합리적인 가격 책정에 상당한 도움이 될 것으로 기대한다.

### 2) 소비자 데이터 분석 분야 진출

- 현재 우리는 많은 광고들과 마케팅을 하루에도 수천 번 마주하곤 한다. 또한, 기업들은 해당 기업의 제품에 대한 소비자들의 반응을 토대로 다음 제품의 목표를 잡을 수 있을 것이다. 본 프로젝트를 미술품에 제한하지 않고, 확장한다면 'b8ta'와 같이 기업의 판매 부스에 본 프로젝트를 적용한다면 기업들은 소비자들의 반응을 객관적으로 분석할 수 있고, 좀 더 서비스 증진에 반영할 수 있을 것이다. 또한, 베타테스트와 같이 소비자의 반응이 중요한 부분에 큰 도움을 줄 것이라고 기대한다.

### 3) 소비자의 서비스 만족도 상승

- 기업 차원에서 소비자의 반응을 반영한다면 소비자 측면에서 좀 더 합리적이고, 질 좋은 서비스와 상품들을 이용하게 될 것이기 때문에 소비자의 만족도가 상승할 것으로 보인다.

## - 향후 일정

☆ RE: 동계인턴십관련

보낸사람 VIP 배재현<jayangie@naver.com>  
받는사람 <actz@naver.com>

대표님 안녕하십니까

팀원들이 지금 최종 결정을 해서 명단 보내드립니다.

2016110316 이상민 12.15 ~ 02.10

1명 동계인턴십 신청합니다.

감사합니다.

-----Original Message-----

From: <actz@naver.com>

To: "배재현" <jayangie@naver.com>;

Cc:

Sent: 2020-12-02 (수) 14:15:56 (GMT+09:00)

Subject: RE: 동계인턴십관련

서류는 빠를수록 좋을것같아요

원래는 저번달 11월11일에 마무리된것을 교수님께서 비공개로 넣어주시는것 같더라구요

팀원들 연락답는데로 작성해서 보내주면 교수님께 바로 보낼게요

- 팀원 중 1명(이상민 학우)이 겨울방학동안 바네컴퍼니와 인턴십을 진행하여 본 프로젝트의 code refactoring과정을 거친 후, 프로젝트의 완성도를 높일 예정이다.



## - 참고 사항

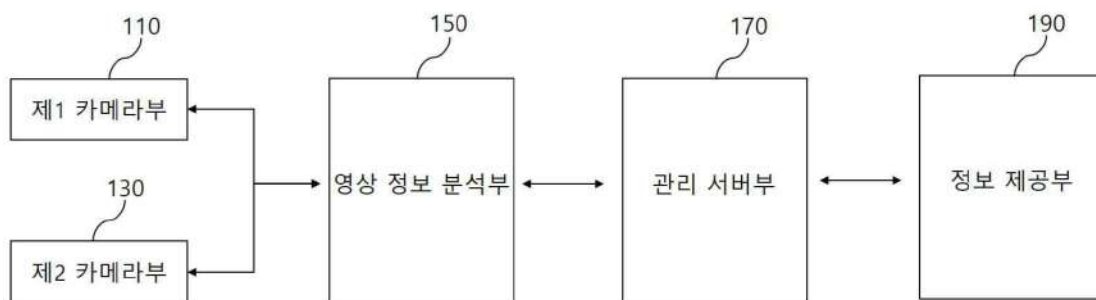
- 프로젝트 진행 당시 바네컴퍼니의 기술 특허등록 중이어서 논문 투고를 계획했으나 불가능했음을 알려드립니다. 아래는 특허의 내용입니다.

(54) 발명의 명칭 관람객에 관한 데이터를 이용한 미술품 경매 시스템 및 이를 이용한 미술품 경매 방법

### (57) 요약

관람객에 관한 데이터를 이용한 미술품 경매 시스템 및 이를 이용한 미술품 경매 방법을 개시한다. 본 발명의 관람객에 관한 데이터를 이용한 미술품 경매 시스템은 복수의 미술품이 전시된 전시 공간에 출입하는 관람객의 동작을 식별하도록 설치된 제1 카메라부, 복수의 미술품에 인접하게 개별적으로 설치되고, 해당 미술품에 접근하는 관람객을 식별하도록 설치된 제2 카메라부, 제1 카메라부와 제2 카메라부로부터 관람객에 대한 이미지 정보를 수신하고, 수신된 관람객에 대한 이미지 정보를 분석하여 미술품별 관람객들의 관심 정보를 연산하기 위한 영상 정보 분석부 및 영상 정보 분석부에서 분석된 관람객들의 관심 정보를 비밀시적으로 저장하고, 사용자의 정보 요청이 있으면 기 저장된 관람객들의 관심 정보를 독출하여 해당 사용자에게 전달하는 관리 서버부를 포함한다.

### 대표도 - 도1



### 발명의 설명

#### 기술 분야

[0001] 본 발명은 미술품 경매 시스템 및 이를 이용한 미술품 경매 방법에 관한 것으로써, 보다 상세하게는 관람객에 관한 데이터를 이용한 미술품 경매 시스템 및 이를 이용한 미술품 경매 방법에 관한 것이다.

#### 배경 기술

[0003] 미술품에 대한 관심은 꾸준히 지속되어 왔다. 유명 예술가의 작품뿐만 아니라 신진 예술가의 작품도 대중에게 알려려는 다양한 시도가 이루어지고 있다. 이러한 미술품은 희소성을 갖고 있다.

[0004] 이러한 미술품에 대한 관심은 단순히 감상하는 것에 그치지 않고, 미술품을 소장하려는 움직임으로 이어지고 있다. 미술품은 희소성으로 인하여 객관적인 가치 산정이 이루어지기 어렵고, 구매자나 판매자 사이의 가치에 대한 관점이 달라서 서로 만족할 만한 가치 산정이 어렵다.

[0005] 유명 미술품의 경우에는 저명한 기관이 오프라인 경매를 운영하고, 오프라인 경매에서 주로 거래가 이루어진다. 오프라인 경매 방식은 경매 참가자가 경매 대상 미술품에 대해서 가장 높은 가격을 부를 경우에 낙찰되는 방식이다. 이러한 전통적인 경매 방식은 유명 미술품이나 유명인의 소장품 등에서 보편적으로 적용되고 있다.

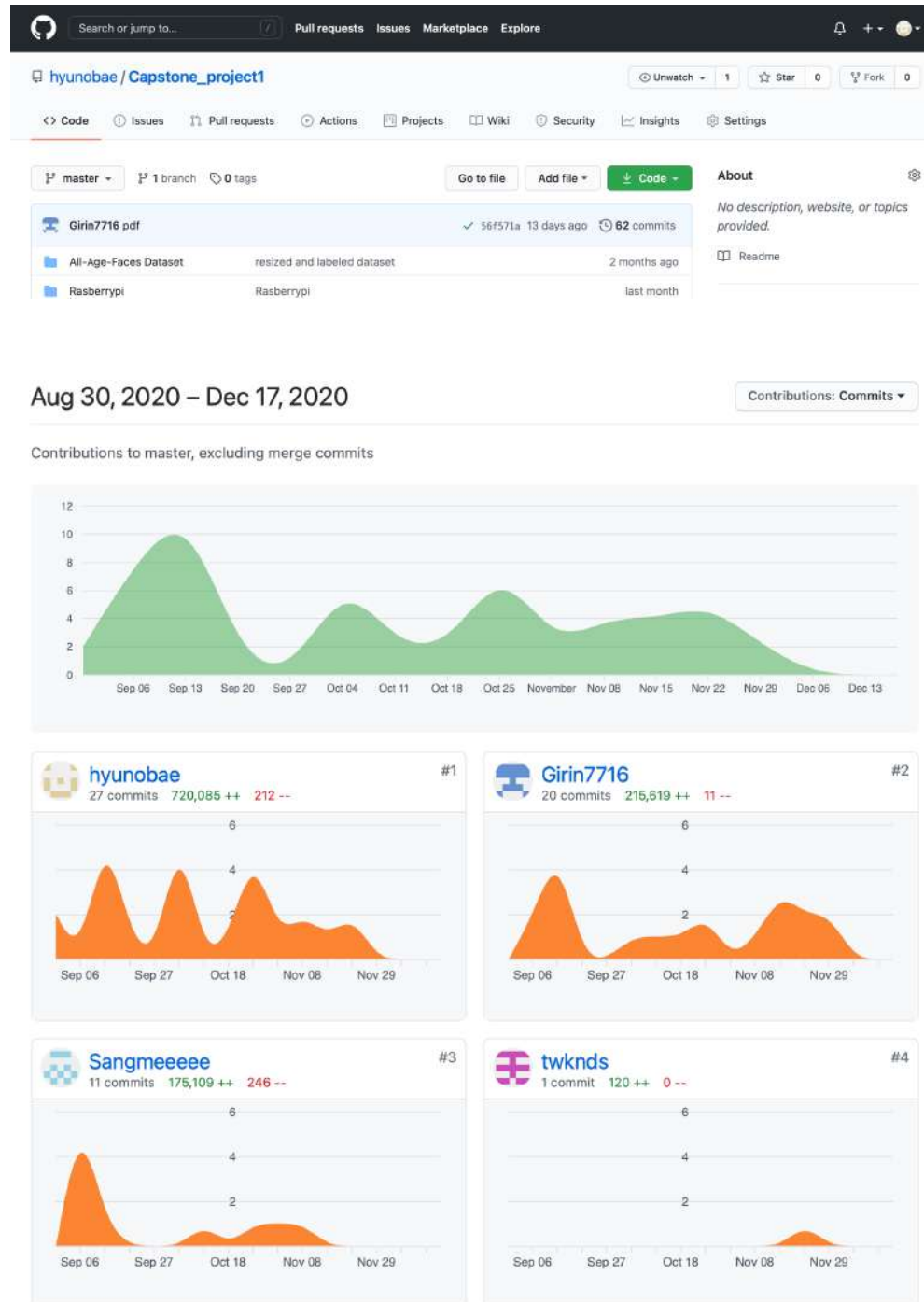
[0006] 하지만, 대중적인 미술품의 거래 수요가 생겨나면서 이러한 전통적인 미술품 경매 방식으로는 신진 예술가의 작품이나 덜 유명한 예술품에 대한 합리적인 가치 결정이 이루어지기 어려운 문제점이 있다.

[0007] 특히, 유명하지 않은 미술품에 대한 수요자가 어떤 기호를 갖고 있는지, 그들은 어떤 미술품에 대해 호감을 갖는지, 미술품에 대한 개인적 호감이 미술품을 구입하기 위한 행동으로 이어지는지, 그렇다면 어떻게 가치를 산정해야 거래가 이루어지는지 이를 결정하기 위해서는 미술품을 감상하는 관람객에 관한 개인적인 행동 정보를 분석하여 빅데이터로 생성하여 미술품 경매에 활용하기 위한 새로운 시스템 및 방법에 대한 개발의 필요성이 요구된다.

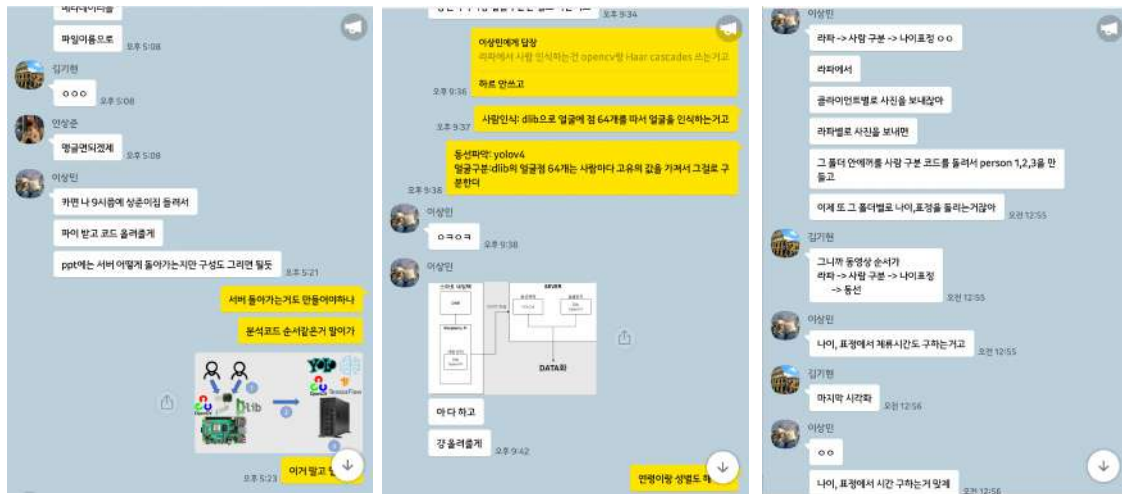
## 6. 협업 내역

### 1) 협업 도구

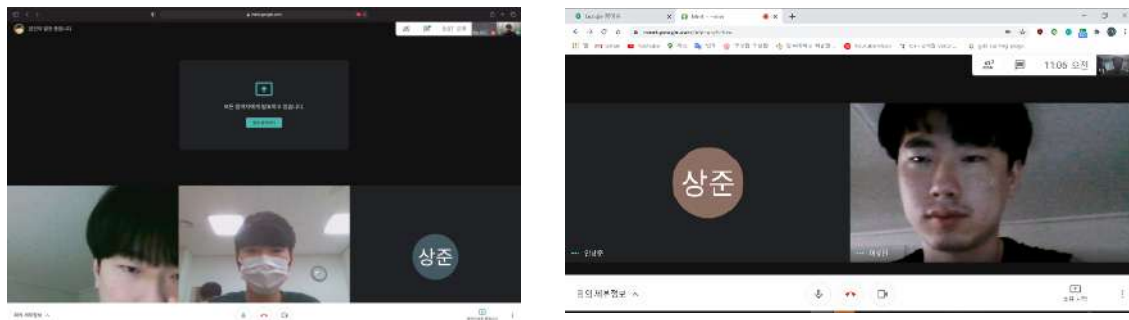
- github



- 카카오톡



- google meets



## 2) 협업 기록

- 대면 회의 시 사진



## • 멘토님과의 연락

☐ ☆ 📧	배재현	RE: 동계인턴십관련 🔍 📧
☐ ☆ 📧	장영민	동계인턴십관련 🔍 📧
☐ ☆ 📧	배재현	RE: 결과보고서 양식.hwp 🔍 📧
☐ ☆ 📧	장영민	결과보고서 양식.hwp 🔍 📧
☐ ☆ 📧	배재현	회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	회의록 및 도서리스트 입니다. 🔍 📧
☐ ☆ 📧	장영민	RE: 회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	RE: 대표님 회의록입니다. 🔍 📧
☐ ☆ 📧	장영민	RE: 대표님 회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	대표님 회의록입니다. 🔍 📧
☐ ☆ 📧	배재현	프로젝트 수행 계획서입니다. 🔍 📧
☐ ☆ 📧	배재현	Re: 9.8일 회의록 보내드립니다. 🔍 📧
☐ ☆ 📧	actz@naver.com	Re: 9.8일 회의록 보내드립니다. 🔍 📧
☐ ☆ 📧	배재현	9.8일 회의록 보내드립니다. 🔍 📧
☐ ☆ 📧	배재현	Re: 대표님 산학프로젝트 관련 연락드립니다. 🔍 📧
☐ ☆ 📧	actz@naver.com	Re: 대표님 산학프로젝트 관련 연락드립니다. 🔍 📧
☐ ☆ 📧	배재현	대표님 산학프로젝트 관련 연락드립니다. 🔍 📧
☐ ☆ 📧	배재현	RE: 산학 프로젝트 관련 문의드립니다. 🔍 📧

- 주 1회 미팅 외에 용무가 생긴다면 메일로 연락하였음

## • 멘토님과의 연락

### 회의록

#### 1. 회의 개요

일시	20.09.17	장소	경북대학교 창업보육센터
작성일	20.09.19		
참석자	바네컴퍼니 대표 장영민 / 경북대 컴퓨터학부 학부생 배재현 외 3명		
안건	영상분석을 통한 방문객 분석 프로그램 개발을 위한 3차 미팅		

#### 2. 회의 내용

회의내용	- 현재 개발 상황 보고 - 라즈베리파이로 스마트 네임택 구현함 - 스마트 네임택에서 얼굴 인식 시 촬영 후 서버로 전송 - 얼굴 표정인식은 긍정적인 부분만 뽑아내도 괜찮아 보임
향후일정	- 라즈베리파이 - 서버 사진 전송을 위한 통신 개발 - caffe 모델의 훈련을 위한 전반적인 준비
특이사항	

### 회의록

#### 1. 회의 개요

일시	20.09.08	장소	경북대학교 창업보육센터
작성일	20.09.10		
참석자	바네컴퍼니 대표 장영민 / 경북대 컴퓨터학부 학부생 배재현 외 3명		
안건	영상분석을 통한 방문객 분석 프로그램 개발을 위한 2차 미팅		

#### 2. 회의 내용

회의내용	- 실시간 영상 처리의 필요성은 없음. - 데이터 값의 처리 방식을 결정함. - 동선처리 방식은 관람자들의 동선을 시각화 할 수 있게 동선을 선으로 나타내어 겹치는 방식으로 처리함. - 스마트 네임택의 카메라는 관람객의 관람을 방해하지 않아야함. - 동선처리 카메라의 크기는 상관없음. - 5개의 카메라 샘플을 제시 1. MAix CUBE 2. OpenMV-H7 3. ESP32-CAM 4. SIPEED Maix-Face V1.1 5. ESP-EYE DevKit
결정사항	- 스마트네임택의 카메라는 OpenMV-H7을 이용하여 OpenMV를 사용한 딥러닝 모델을 탑재 예정 - 카메라에서 Data를 처리 후, 서버로 보내는 영상 처리 방식 - 데이터 값은 하루단위로 처리하여 차트로 나타냄 - 현장에 탑재되는 동선처리 카메라는 라즈베리 파이 + 카메라 이용 - 멘토님과의 미팅은 매주 목요일 5시
향후일정	- OpenMV-H7 구비 및 OpenMV IDE 학습 - OpenMV에 사용 할 수 있는 딥러닝 모델 탐색 - 참여 학생들간의 미팅은 화상으로 수시로 가지기로 함



## 회의록

### 1. 회의개요

일	시	장	소
20.09.04		장	소
20.09.04			
참석자	바네키퍼니 대표 장영민 / 경북대 컴퓨터학부 학부생 배재현 외 3명		
안건	영상분석을 통한 방문객 분석 프로그램 개발을 위한 1차 미팅		

### 2. 회의내용

회의내용	<ul style="list-style-type: none"> <li>- 미술작품의 가격 측정에 대해 수요와 공급의 법칙을 적용시키기 위해 프로그램을 개발</li> <li>- 나이, 성별, 체류시간들을 Data화 하기위해 프로그램 개발 진행</li> <li>- 기존 개발 내용을 수정해서 최적화 해보기</li> <li>- 스마트 네임택에 사용되는 카메라는 소형으로 관객이 몰라야함</li> <li>- 영상처리 방식에는 크게 두가지가 존재하기 때문에 어느 방식으로 할지 결정하기</li> <li>1. 카메라 자체에서 Data를 처리한 후, Data만 보내기</li> <li>2. 영상만을 보내서 그 후 Data를 처리하기</li> <li>- 제안서에 제시된 내용 말고도 재미있는 아이디어 생각해보기</li> <li>- 오픈소스를 활용 할 경우 대부분 서양인 기준으로 나와있기 때문에 동양인을 대상으로 Data를 모을 것</li> </ul>
결정사항	<ul style="list-style-type: none"> <li>- 2차 미팅시 카메라 샘플을 보고 최적화된 모델을 선정</li> <li>- 파이썬과 함께 OpenCV를 사용함</li> <li>- 최종적으로 3개의 작품, 4대의 카메라 환경을 조성하고 1대의 카메라는 천장에 달아서 관람객의 동선을 파악, 나머지 3대의 카메라는 스마트네임택으로 이용</li> </ul>
향후일정	20.09.08에 2차 미팅이 예정되어 있음
특이사항	대표님은 대구에서 작업을 하시고 개발자분들은 서울쪽에 있음

## 회의록

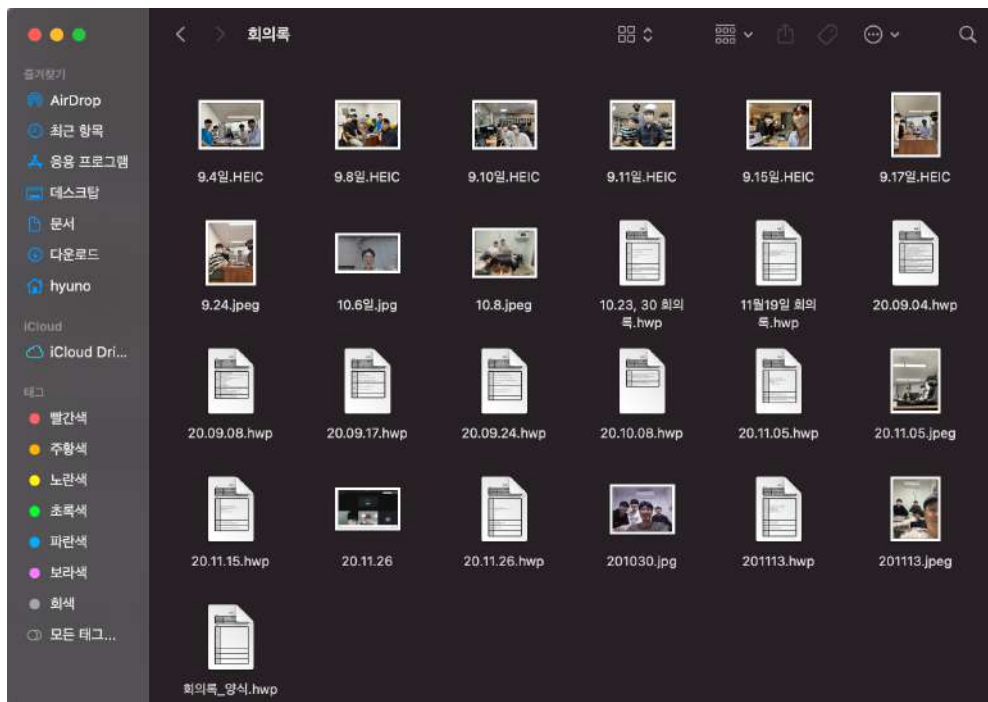
### 1. 회의개요

일	시	장	소
20.11.05		장	소
20.11.05			
참석자	바네키퍼니 대표 장영민 / 경북대 컴퓨터학부 학부생 배재현 외 1명		
안건	영상분석을 통한 방문객 분석 프로그램 개발을 위한 미팅		

### 2. 회의내용

회의내용	<ul style="list-style-type: none"> <li>- 최종발표 데모 영상 실제 미술관에서 촬영</li> <li>- 라즈베리파이 부팅 시 자동으로 서버로 사진 보내기</li> <li>- 표정인식 정확도 높이기</li> <li>- 예러 디렉팅</li> <li>- 스마트 네임택(케이스) 디자인 확인</li> </ul>
결정사항	<ul style="list-style-type: none"> <li>- 모든 과정이 자동으로 이루어져야함</li> <li>-&gt; 라즈베리파이에서 서버로 보내서 처리를 하는 과정</li> </ul>
향후일정	11월 중, 데모영상 촬영을 위해 미술관 전시회 참여 예정

- 멘토님과 주 1회 대면회의를 진행하고 회의록 작성



회의록 증빙자료

• 연구실 안전교육 이수증명서

2020. 11. 29.

연구실안전관리시스템



## 안전교육 이수증명서

▶ 교육생 정보

성명	안상준	학(사)번	2016115631	소속	컴퓨터학부
----	-----	-------	------------	----	-------

▶ 안전교육 이수정보

번호	교육구분	과정명	이수일자	이수시간	이수번호
1	정기	2020년도 하반기 안전교육	2020.11.29	3	20201129_434194
				총 이수시간	3

위와 같이 안전교육을 이수하였음을 증명합니다.

2020년 11월 29일

경북대학교 연구실안전관리센터



연구실안전관리시스템

2020. 9. 2. 오후 3:32



## 안전교육 이수증명서

▶ 교육생 정보

성명	배재현	학(사)번	2016116090	소속	컴퓨터학부
----	-----	-------	------------	----	-------

▶ 안전교육 이수정보

번호	교육구분	과정명	이수일자	이수시간	이수번호
1	정기	2020년도 하반기 안전교육	2020.09.02	3	20200902_434200
				총 이수시간	3

위와 같이 안전교육을 이수하였음을 증명합니다.

2020년 09월 02일

경북대학교 연구실안전관리센터



## 안전교육 이수증명서

### ▶ 교육생 정보

성명	김기현	학(사)번	2016113934	소속	컴퓨터학부
----	-----	-------	------------	----	-------

### ▶ 안전교육 이수정보

번호	교육구분	과정명	이수일자	이수시간	이수번호
1	정기	2020년도 하반기 안전교육	2020.10.09	3	20201009_434217
총 이수시간				3	

위와 같이 안전교육을 이수하였음을 증명합니다.

2020년 10월 09일

경북대학교 연구실안전관리센터장



## 안전교육 이수증명서

### ▶ 교육생 정보

성명	이상민	학(사)번	2016110316	소속	컴퓨터학부
----	-----	-------	------------	----	-------

### ▶ 안전교육 이수정보

번호	교육구분	과정명	이수일자	이수시간	이수번호
1	정기	2020년도 하반기 안전교육	2020.12.07	3	20201207_434188
총 이수시간				3	

위와 같이 안전교육을 이수하였음을 증명합니다.

2020년 12월 07일

경북대학교 연구실안전관리센터장





- DIP 산학협력 프로젝트 결과 보고



- 본 프로젝트는 DIP(대구디지털산업진흥원)와의 연계과제로 프로젝트 결과보고회에 참여하였음

## 7. 참여 인력(세부)

지도교 수	소속	경북대학교		성명	고석주
참여인 력 (산업체)	기업명	성명	직위	전화	Email
	바네컴퍼니	장영민	대표	010-4818-1275	actz@naver.com
과 제 참 여 학 생	소속(학과)	학위과 정 (성별)	학번	성명	담당업무
	컴퓨터학부	학사과정 (남)	2016116090	배재현	<ul style="list-style-type: none"> <li>• 팀장</li> <li>• Architecture Design</li> <li>• DL director</li> </ul>
	컴퓨터학부	학사과정 (남)	2016113934	김기현	<ul style="list-style-type: none"> <li>• Face recognition directore</li> <li>• DL assistant</li> </ul>
	컴퓨터학부	학사과정 (남)	2016115631	안상준	<ul style="list-style-type: none"> <li>• Person detection director</li> <li>• DL assistant</li> <li>• Data visualization</li> </ul>
	컴퓨터학부	학사과정 (남)	2016110316	이상민	<ul style="list-style-type: none"> <li>• Raspberrypi</li> <li>• Server-client communication</li> <li>• Data visualization</li> </ul>