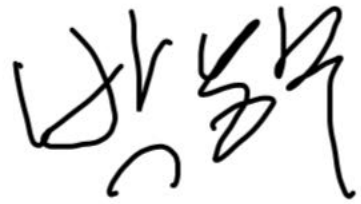
[T17]Depth camera 기반 안면인식 보안 시스템

201511249 김영운

201511261 박성현

201511291 장유준

201511302 최연규



지도교수 : 박능수 교수님

목차

프로젝트 개요

프로젝트명 : Depth camera 기반 안면인식 보안 시스템

프로젝트 목적 : 안면인식 보안 시스템에서 ToF 카메라를 통해 사용자 얼굴의 깊이 정보를
받아오고 실제 사용자의 얼굴이 맞는지를 확인하여 인증한다.

개발환경 : OS : Windows 10
PL : Python 3.8 / C++
SDK : Intel RealSense SDK 2.0
IDE : Pycharm, Vscode

요구사항

◈ 기능 요구사항

1. 카메라 시스템
   : 카메라 on/off

2. 안면 정보 설정
   1. 관리자모드 설정
      : 허가된 관리자는 UI를 통해 관리자모드로 진입할 수 있어야 한다.
   2. 안면 정보 등록
      : 안면 정보를 등록하는 기능을 제공해야 한다.
   3. 안면 정보 삭제
      : 안면 정보를 삭제하는 기능을 제공해야 한다.

3. 보안 해제
   1. 안면 인증
      : 안면 정보를 등록한 사용자를 구분하고 인증할 수 있어야 한다

요구사항

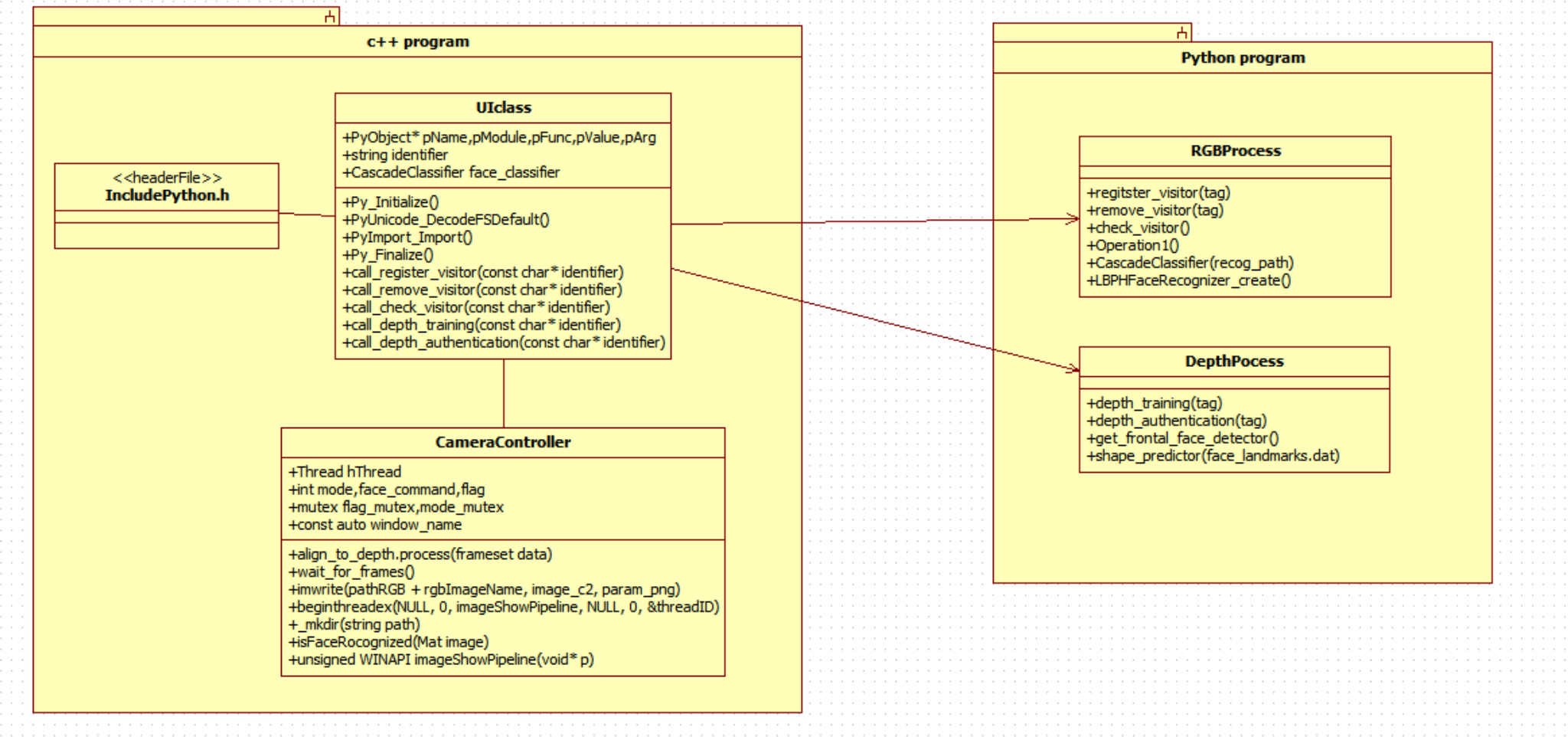
◈ 비기능 요구사항

1. 빠른 안면 인식 시간
   : 안면 인식의 처리 시간이 5초 이내여야 한다.

2. 안면 비교 정확도
   : 안면 인식 결과의 정확도는 1/50,000 수준이 되어야 한다.

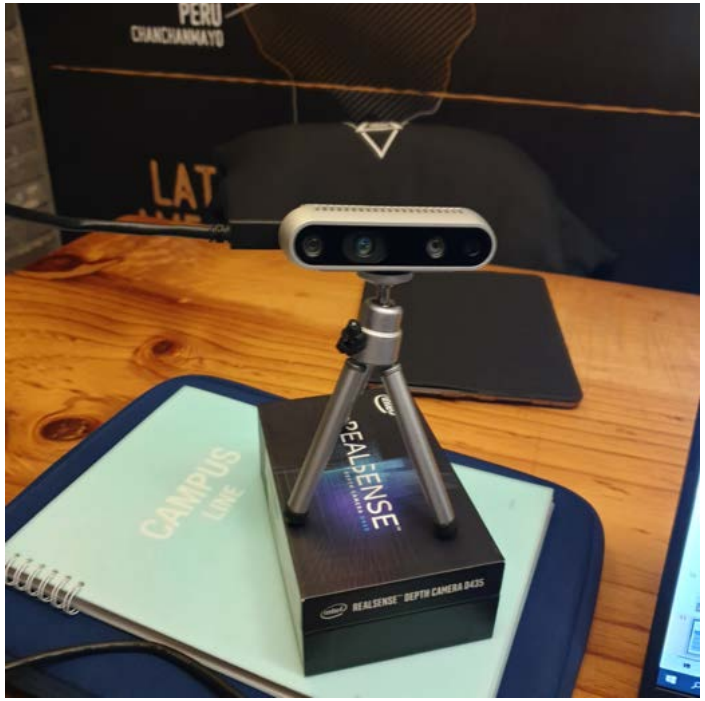
3. 사용자의 변화를 학습
   : 사용자가 인증을 시도할 때마다 Depth 정보를 학습시켜야 한다.

Deployment Diagram

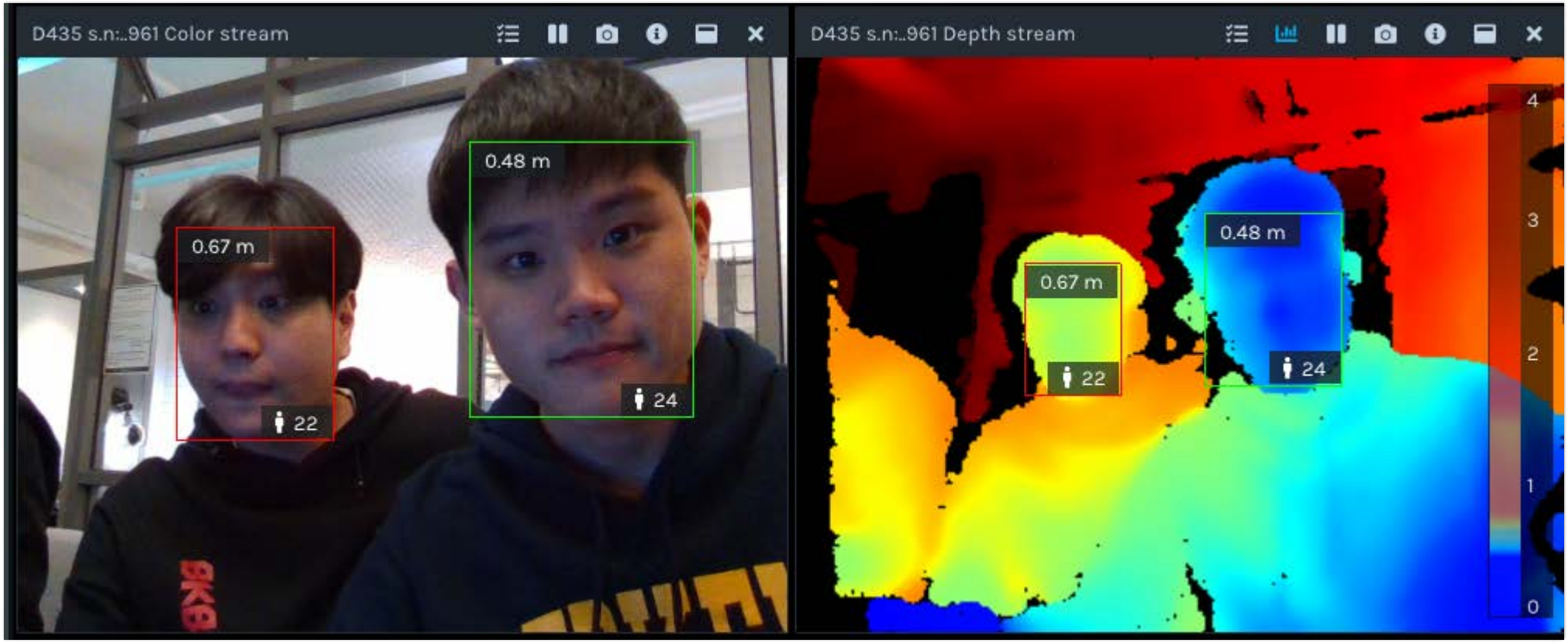


Cost Software / Hardware

- Software : Intel RealSense SDK 2.0
- Hardware : Intel RealSense D435 ToF Camera



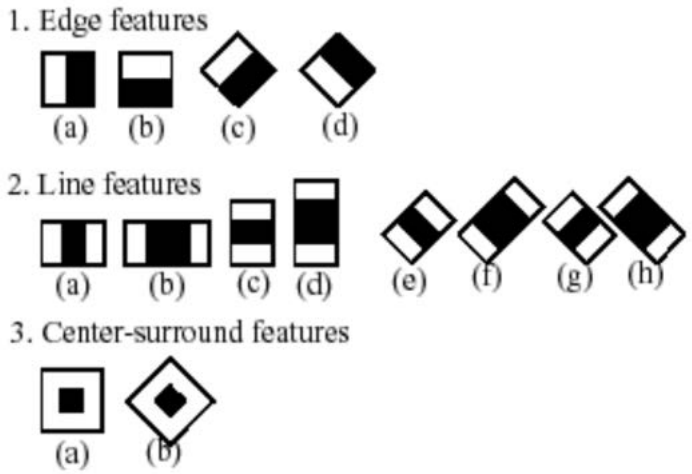
Intel RealSense Camera

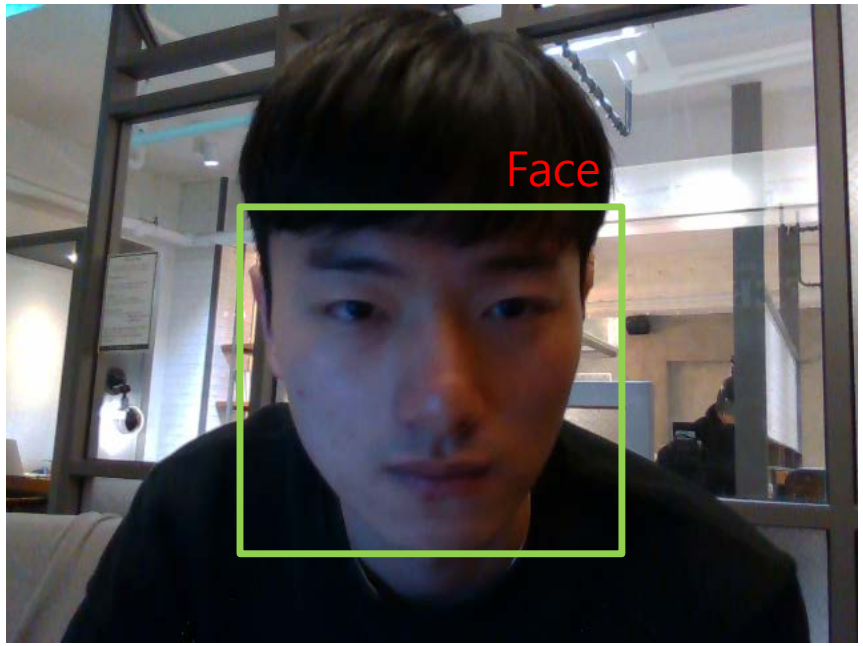


RGB Face Model Generation

얼굴 검출(Haar-based face cascade)

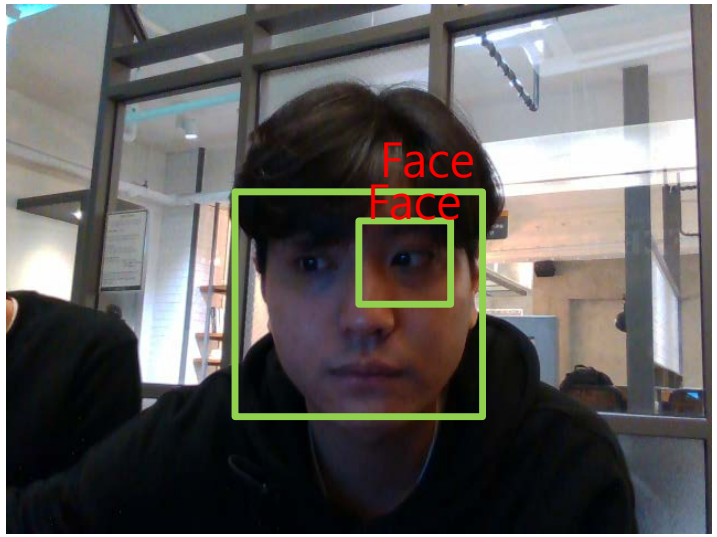
- Haar-based cascade는 단순한 특징들을 조합하여 객체를 찾아낸다. 받아들인 이미지를 gray-scale로 변환한 후 영역의 화소값 차이를 이용한다. 이 분류기는 수 천개의 얼굴 영상과 수 만개의 얼굴이 아닌 영상을 사용하여 학습되었다.





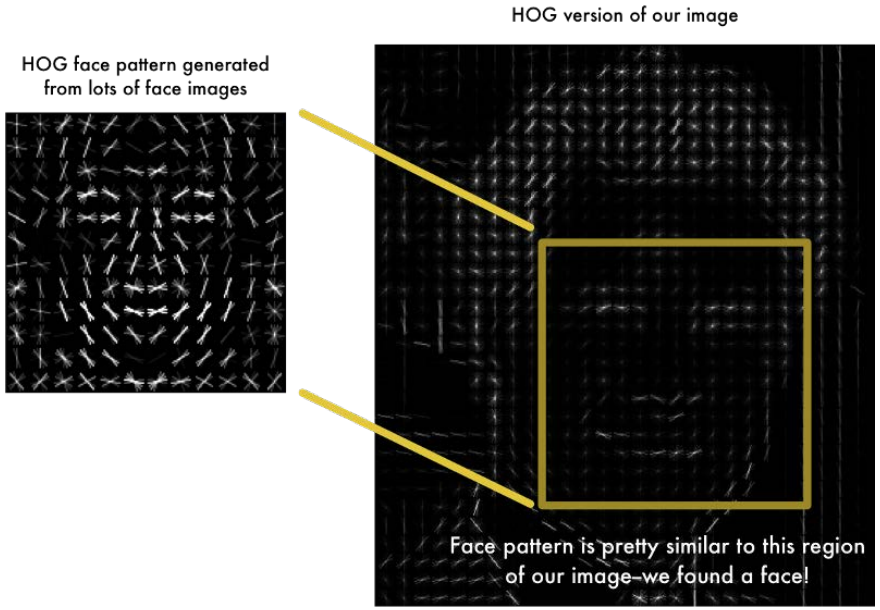
얼굴 검출(Haar-based face cascade)

- 카메라와 방문자 사이의 거리에 따라 얼굴이 아닌 곳의 패턴이 얼굴과 일치하여 얼굴로 인식하는 경우도 있는데, 이는 사용자가 항상 같은 거리에서 얼굴을 등록 및 인식한다는 가정을 하였고, 이때 최소 사이즈를 지정 하여 아래 사진과 같이 얼굴이 아닌 부분을 얼굴로 인식하는 경우를 방지하였다.



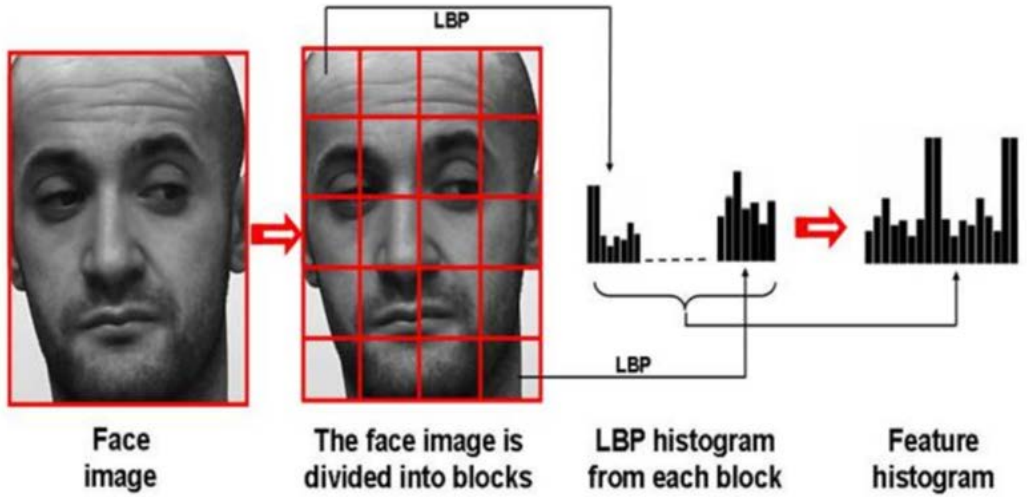
특징 검출(dlib shape predicator)

- dlib 라이브러리에서는 얼굴 탐색을 위해 HOG특성을 활용하거나 또는 대신 학습된 CNN모델을 사용할 수 있다.
- HOG는 픽셀값의 변화로 파악할 수 있는 영상 밝기 변화의 방향을 그래디언트(gradient)로 표현하고, 이로부터 객체의 형태를 찾아낼 수 있다. 얼굴 탐색 이외에도 보행자 검출 등에 활용할 수 있다.



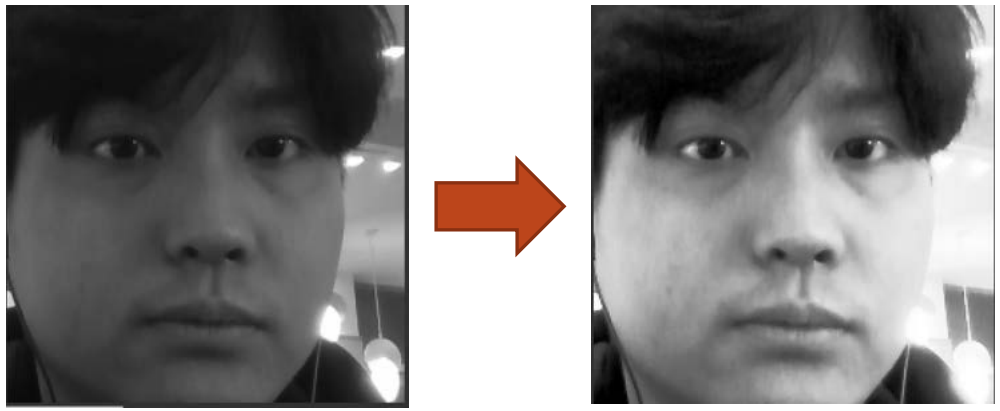
얼굴 인식(LBPH)

- LBPH(Local Binary Patterns Histogram)은 이미지의 middle-level-feature인 질감과 얼굴인식에 사용하는 알고리즘 이다.
- 이미지를 여러 개의 블록으로 분할한 후 각 블록에서 LBP히스토그램을 구하고 그것들을 통해 얼굴 인식 프로그램을 구현 할 수 있다.



얼굴 인식(LBPH)

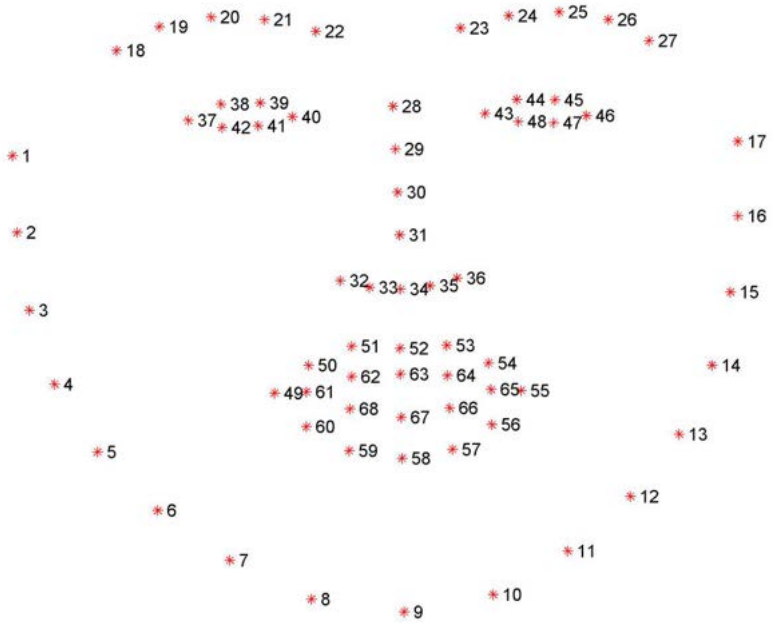
- 방문자는 얼굴을 등록할 때 RealSense Cameara를 통해 얼굴 정보를 등록한다. 이 때 얼굴을 인식한 경우 얼굴 정보를 저장하는데, RealSense Camera가 잘 호환되는 얼굴 인식 알고리즘은 dlib와 haar가 있다. 이 중 haar가 더 좋은 성능을 보여주었기 때문에 이를 이용하여 얼굴 정보를 저장하고, 같은 정보에 대해 모델링을 하기 위해 haar를 통해 얼굴 정보를 받아오고, lbph알고리즘을 이용하여 모델링을 진행하였다. gray_scale에 대해 사용되는 알고리즘이지만, 이후 depth 데이터를 통해 다시 한번 등록자인지에 대한 여부를 확인하기 때문에 속도적인 측면을 고려하여 사용하였다.
- 명암의 영향을 최소화 하여 특징을 더 잘 뽑아낼 수 있도록 이미지를 사전에 평활화하여 넣어주었다.

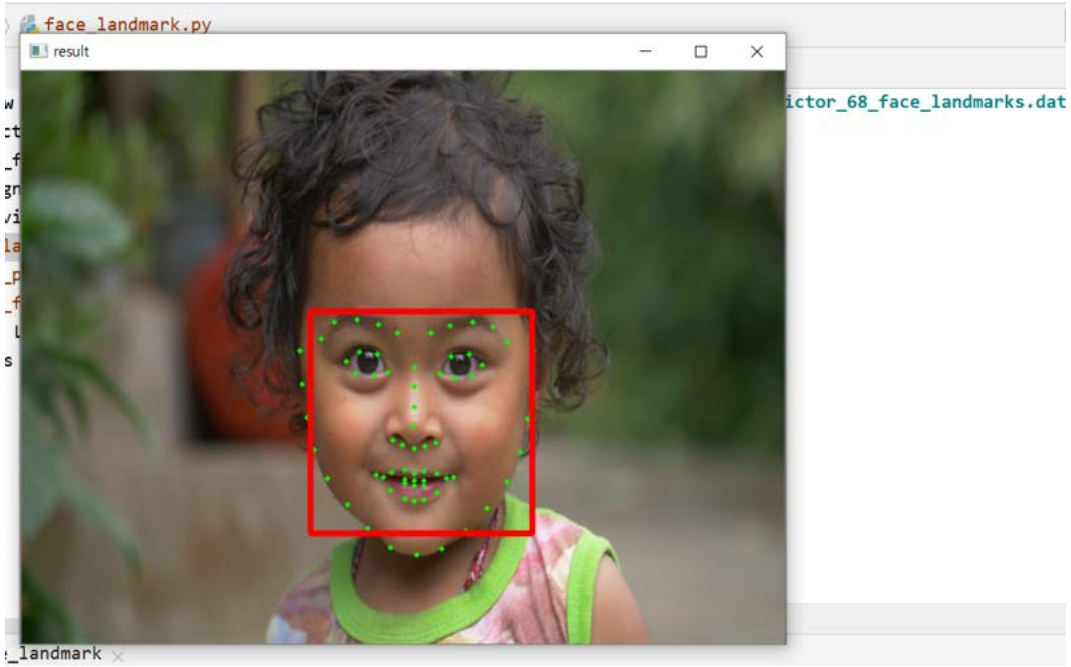


Depth Face Model Generation

얼굴 및 특징 검출(dlib Face landmarks)

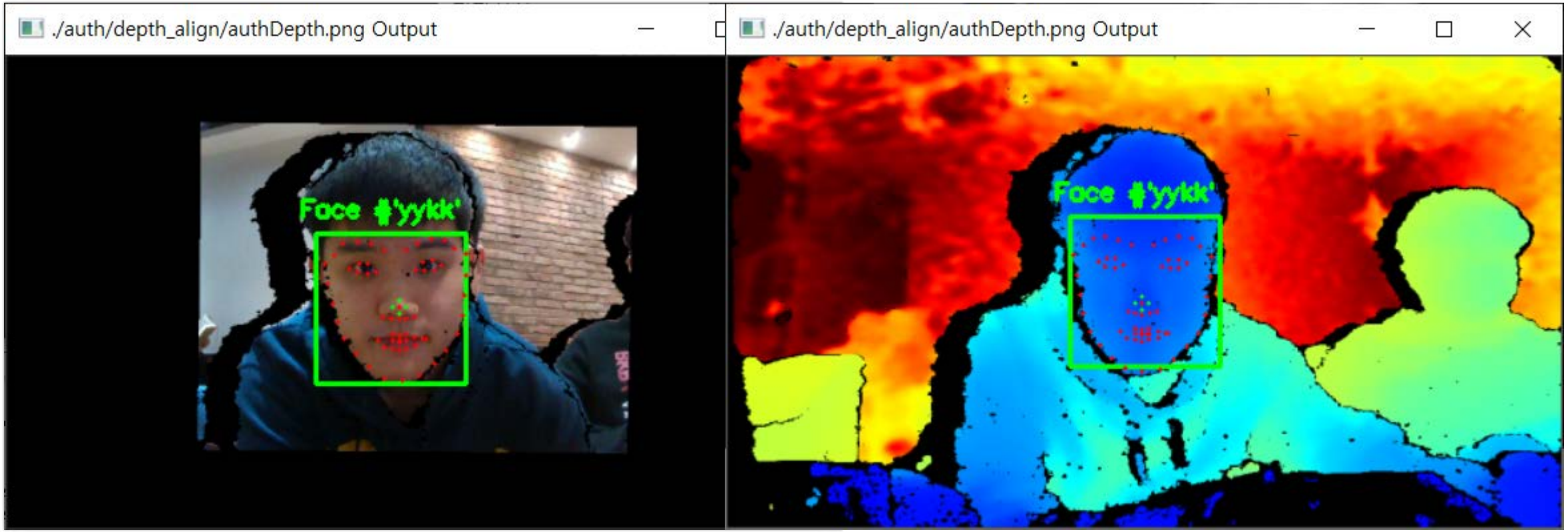
- dlib의 Facial Landmark를 이용하여 68개의 얼굴 정보를 추출한다.





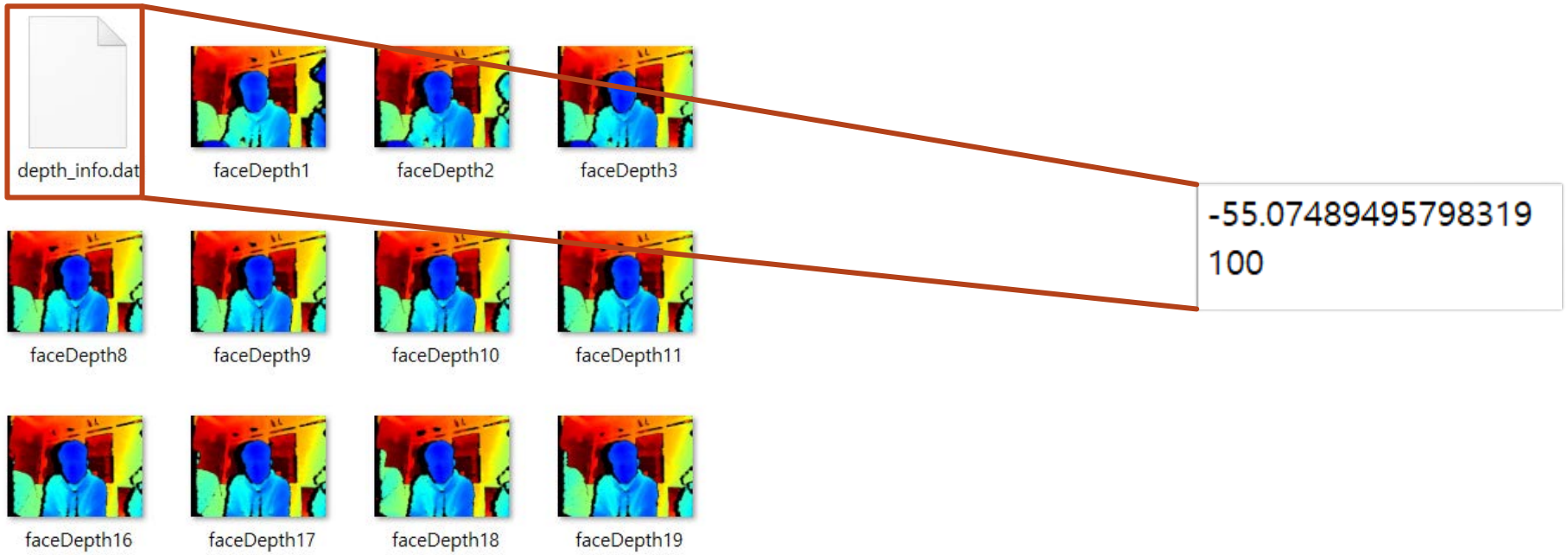
Depth Image로 Mapping

- 얼굴윤곽을 비롯하여 눈썹, 눈, 코, 입의 좌표를 추출한다.
- 추출된 좌표를 RGB 이미지에서 Depth 이미지로 사상한다.



Depth 정보 학습 및 저장

- 가까이 있는 물체는 B에 가깝게, 멀리 있는 물체는 R에 가깝게 표현되는 카메라의 특성을 이용하여 RGB 중 B값을 통해 사용자별 안면의 Depth 정보를 추출할 수 있따.
- 100개의 Depth 정보를 분석하여 mean값을 구하고 .dat 파일로 저장한다.



2nd Implementation

- User Interface
- RGB Face Model Generation / Authentication / Remove
- Depth Face Model Generation / Authentication / Remove

User Interface - PyObject

C++ 연동을 위해 PyObject 사용

```
PyObject* pName, * pModule, * pFunc, * pValue, * pArg;
```

파이썬 코드와 매핑하기 위한 PyObject 변수 생성

```
Py_Initialize();

pName = PyUnicode_DecodeFSDefault("process");
//cout << "0x01" << endl;
pModule = PyImport_Import(pName);
//cout << "0x02" << endl;
```

process.py 프로세스 생성 및 현재 C++ 프로그램에서 함수 호출

PyObject

```
if (Py_IsInitialized()) {

    if (pModule != NULL) {
        //cout << "파일 찾음" << endl;
        pFunc = PyObject_GetAttrString(pModule, "depth_training");//실행할 함수인 test_func을 PyObject에 전달
                                                  //cout << "0x03" << endl;
        pArg = Py_BuildValue("(z)", (const char*)id);//문자열 hello를 담은 매개변수 만든다
                                               //cout << "0x04" << endl;
        pValue = PyObject_CallObject(pFunc, pArg);
        //cout << "0x05" << endl;
        PyObject* objectsRepresentation = PyObject_Repr(pValue); // 객체를 문자열로 표현한다.
        PyObject* str = PyUnicode_AsEncodedString(objectsRepresentation, "utf-8", "~E~"); // 객체를 인코딩한다.
        string result = PyBytes_AsString(str); // string으로 변환한다.
```

- 생성한 파이썬 프로세스의 depth_training() 함수에 매개변수를 할당함
- depth_training()함수 호출 뒤 반환 값은 pValue에 저장되고, 인코딩 과정을 거쳐 result에 c++ string type으로 저장됨

멀티스레딩

카메라 인터페이스 노출을 위해 추가적인 스레드 생성

```
HANDLE hThread;
unsigned threadID;

hThread = (HANDLE)_beginthreadex(NULL, 0, imageShowPipeline, NULL, 0, &threadID);

if (hThread == 0) return -1;
```

hThread 스레드를 생성하여 imageShowPipeline() 를 실행

imageShowPipeline 내부에서는 UI 입력에 따라 카메라에서 들어오는 프레임을 제어하고 이식된 파이썬 모듈을 관리

멀티스레딩

```
unsigned WINAPI imageShowPipeline(void* p) {


    rs2::colorizer color_map;
    rs2::align align_to(RS2_STREAM_COLOR);
    rs2::align align_to_depth(RS2_STREAM_DEPTH);

    face_cascade_name = "C:\\opencv\\sources\\data\\haarcascades\\haarcascade_frontalface_alt.xml";
    if (!face_classifier.load(face_cascade_name)) { printf("--(!)Error loading face cascade\n"); return -1; };

    rs2::pipeline pipe;
    pipe.start();

    namedWindow(window_name, WINDOW_AUTOSIZE);
```

스레드에서 Realsense카메라를 먼저 로드하고, 실패하면 Error 메세지 출력

RGB Face Model Generation

- 카메라에서 저장한 데이터들이 있는 폴더에 접근하여 존재하는 모든 등록자들에 대해 폴더 별로 라벨링을 한 후 모델을 생성한다.

RGB Face Model Authentication

- 생성된 모델을 통해 유사도가 가장 높은 라벨 번호와 유사도(confidence)를 반환 받는다. 유사도는 여러 테스트를 한 결과 70이하 정도일 때 사람을 잘 인식하는 것으로 나와, 70으로 설정하였다. register_visitor.py와 마찬가지로 평활화 작업을 거친 후 모델에서 유사도를 측정한다.
  - 70보다 높은 경우 등록되지 않은 사용자로, 반환받은 라벨 번호를 파기하고, 'notauth'라는 값을 메인 프로그램에 반환한다.
  - 70보다 낮은 경우 등록된 사용자로, 해당하는 라벨을 통해 알아낸 방문자의 폴더의 이름을 depth_process.py에 전달할 수 있도록 메인 프로그램에게 반환한다.

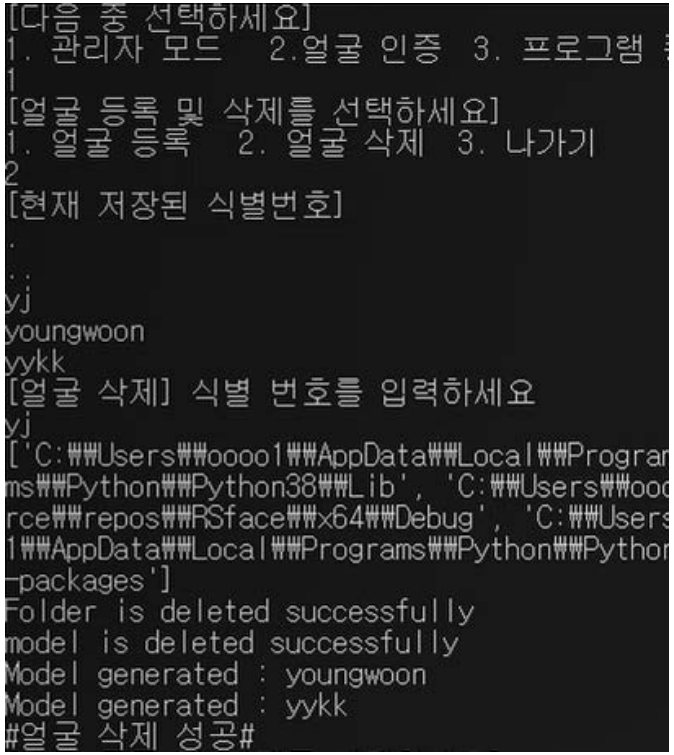
visitor2

```
May be this person is :  visitor2
result of conf : 33.538608
```

visitor2에 대한 conf값

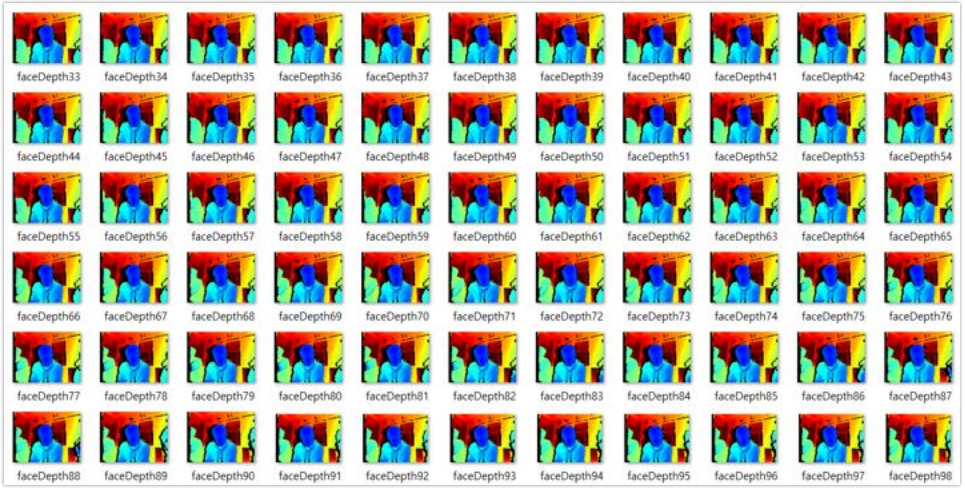
RGB Face Model Remove

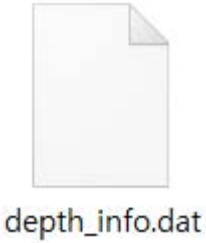
- 메인 프로그램에서 전달받은 매개변수를 통해 등록자들의 정보가 저장되어 있는 폴더 중 해당 폴더를 삭제한 후 모델을 재생성한다.
- 만약 등록자의 정보가 잘못되었다면 메인프로그램에 0을 반환하여 존재하지 않는 정보임을 알려준다.



Depth Face Model Generation

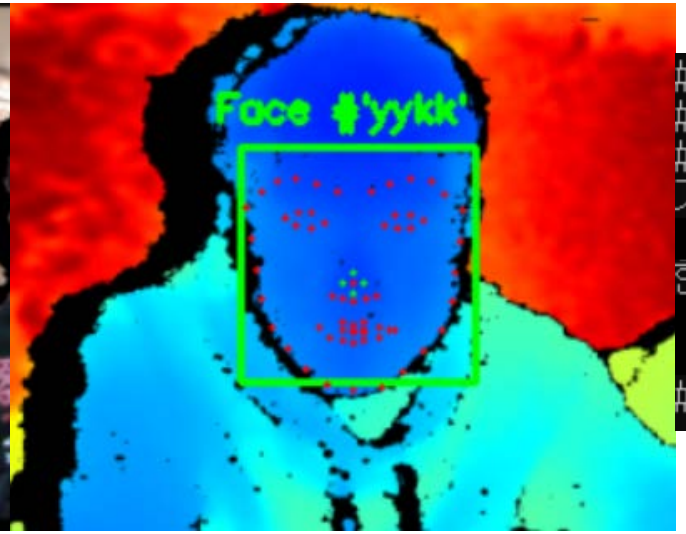
- 100장의 사용자 Depth 정보에서 Face landmarks를 추출하고 Depth 정보의 상대적인 차이를 구한다.(코 주위 값의 평균 – 눈 주위 값의 평균)
- 100개의 Depth 차이 정보의 mean값을 구하여 .dat 파일로 저장한다.





Depth Face Model Authentication

- 사용자 얼굴이 인식되면 먼저 RGB Face Model Authentication를 통해 등록된 사용자인지 확인하고, 등록된 사용자의 경우 저장된 Depth Face Model을 확인하여 Depth 정보를 확인한다.
- Depth Threshold(임계값)이 저장된 사용자의 값보다 +-20 범위 내에 있으면 올바른 사용자로 인식하여 인증 성공 값을 반환한다.



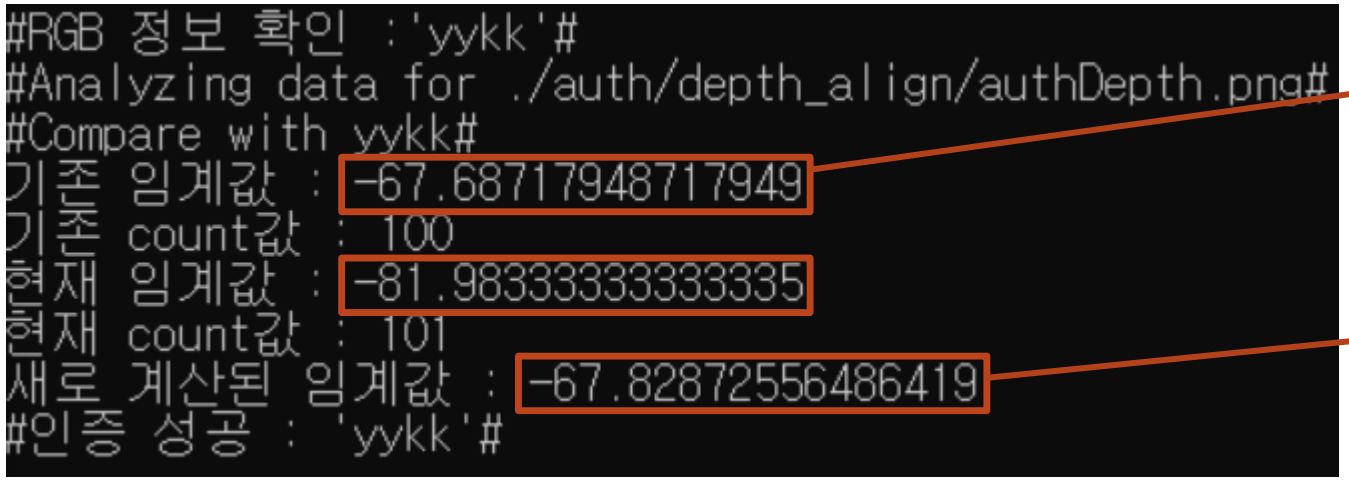

```
#RGB 정보 확인 :'yykk'#
#Analyzing data for ./auth/depth_align/authDepth.png#
#Compare with yykk#
기존 임계값 : -67.68717948717949

현재 임계값 : -81.98333333333335


#인증 성공 : 'yykk'#
```

Depth Face Model Machine Learning

- 만약 해당 사용자의 안면 정보로 인증을 성공할 경우 해당 이미지의 Depth 정보를 Threshold(임계값)에 반영한다.
- 인증 성공 횟수가 많아질수록 임계값이 현재 얼굴에 맞게 변화된다.



-67.68717948717949
100
기존 사용자의 Threshold

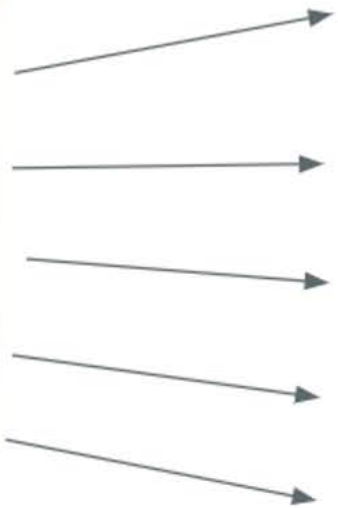
-67.82872556486419
101
새로운 인증 데이터가 반영된 Threshold

Depth Face Model Machine Remove

- 삭제할 사용자가 매개변수로 전달되면 Depth Face Model과 해당 사용자의 Depth Image를 모두 삭제한다.

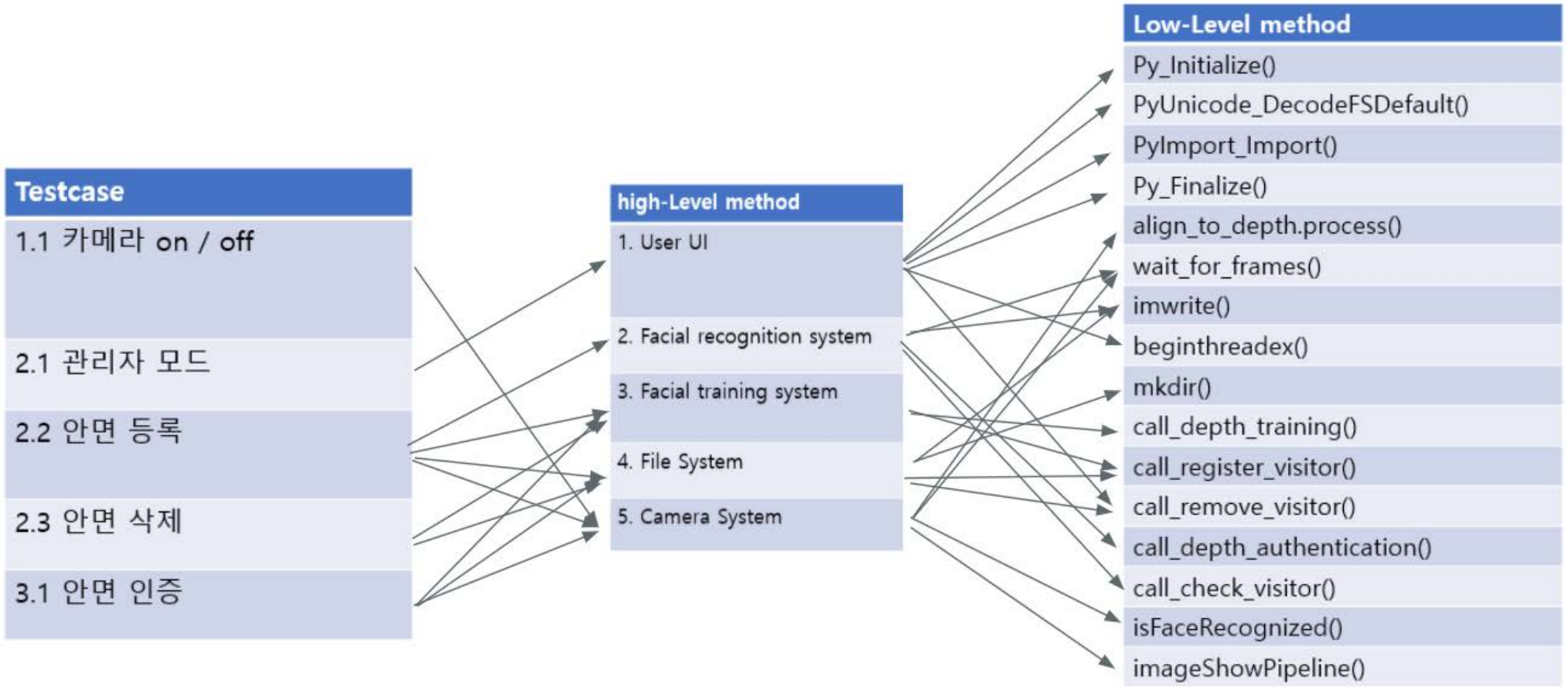
System test – 추적성 분석

Requirement	Pass/Fail criteria
1.1 카메라 on / off	카메라 기능 작동, 사진 촬영 기능 정상 작동
2.1 관리자 모드	관리자 모드 정상 진입
2.2 안면 등록	카메라로부터 RGB image, Depth image 정상 촬영 후 파일 시스템에 정상 저장
2.3 안면 삭제	파일시스템에 안면정보 출력 후 정상 삭제
3.1 안면 인증	안면 정보 정상 인증

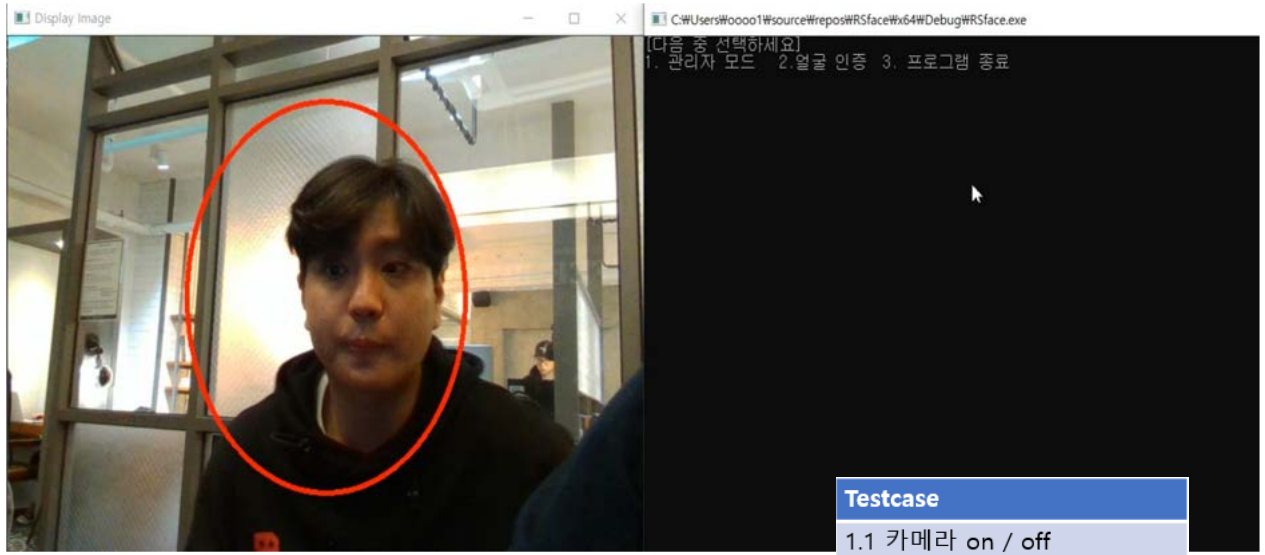


Testcase
1.1 카메라 on / off
2.1 관리자 모드
2.2 안면 등록
2.3 안면 삭제
3.1 안면 인증

System test – 추적성 분석



추적성 분석 수행 - 1.1 카메라 on / off

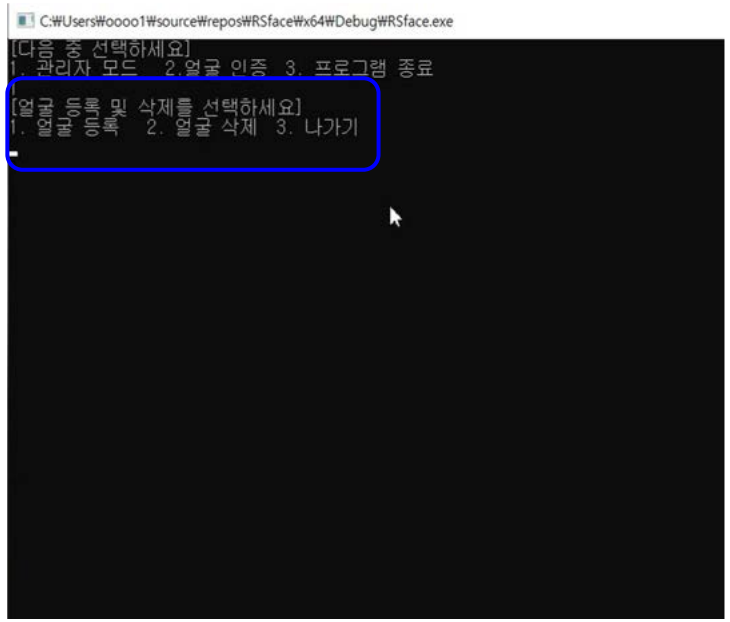


Testcase
1.1 카메라 on / off
2.1 관리자 모드
2.2 안면 등록
2.3 안면 삭제
3.1 안면 인증

high-Level method
1. User UI
2. Facial recognition system
3. Facial training system
4. File System
5. Camera System

Low-Level method
Py_Initialize()
PyUnicode_DecodeFSDefault()
PyImport_Import()
Py_Finalize()
align_to_depth.process()
wait_for_frames()
imwrite()
beginthreadex()
mkdir()
call_depth_training()
call_register_visitor()
call_remove_visitor()
call_depth_authentication()
call_check_visitor()
isFaceRecognized()
imageShowPipeline()

추적성 분석 수행 - 2.1 관리자 모드 시작

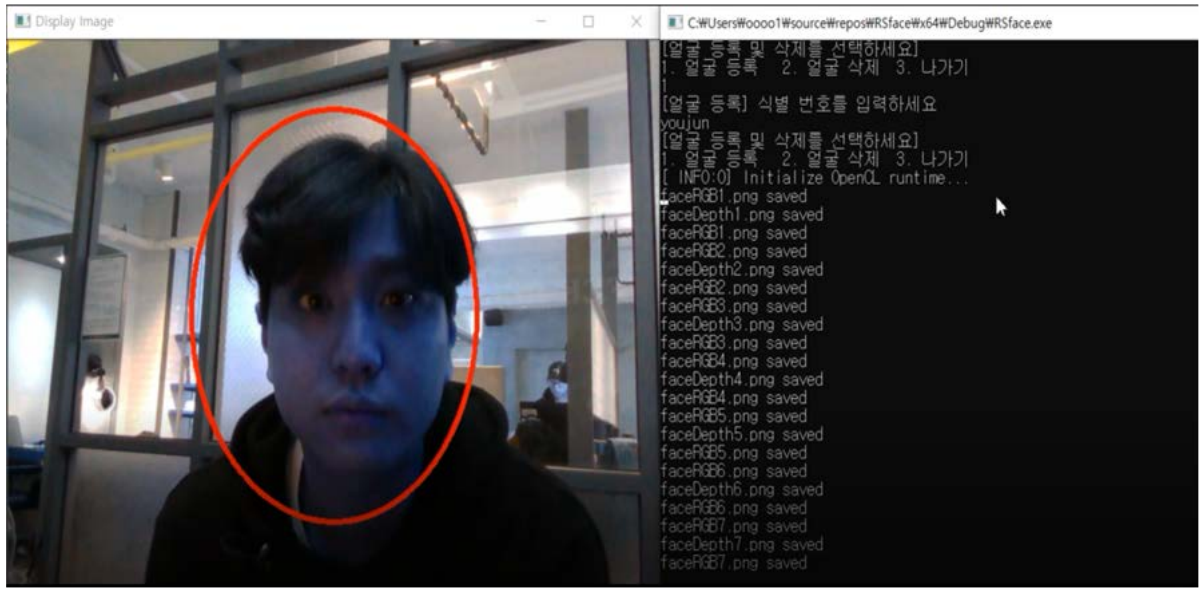


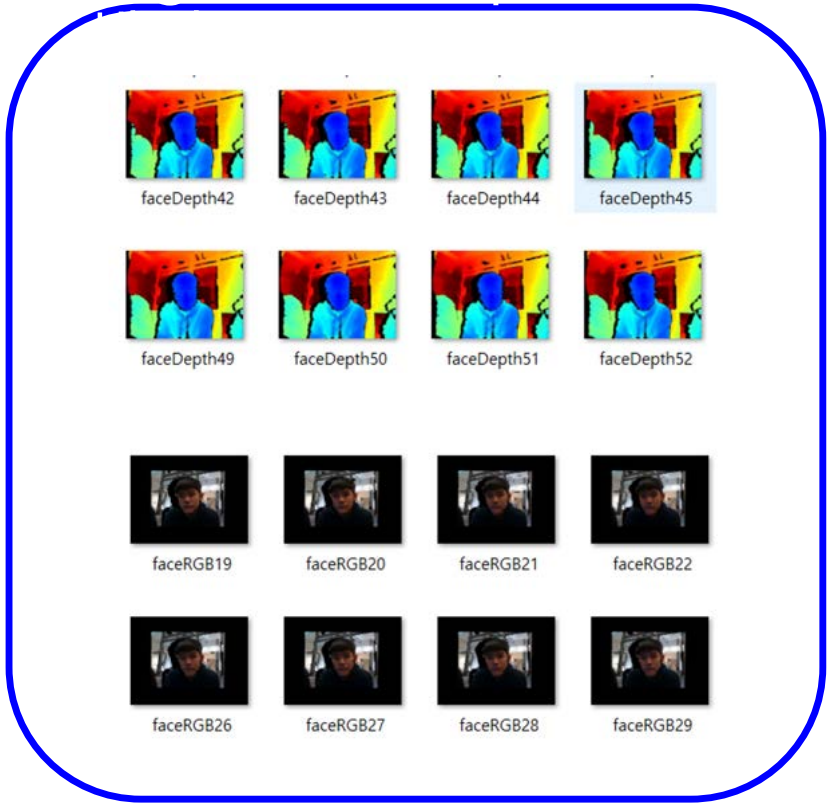
Testcase
1.1 카메라 on / off
2.1 관리자 모드
2.2 안면 등록
2.3 안면 삭제
3.1 안면 인증

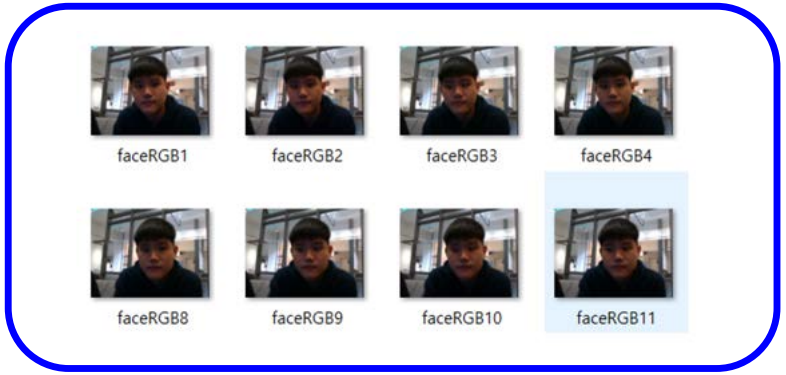
high-Level method
1. User UI
2. Facial recognition system
3. Facial training system
4. File System
5. Camera System

Low-Level method
Py_Initialize()
PyUnicode_DecodeFSDefault()
PyImport_Import()
Py_Finalize()
align_to_depth.process()
wait_for_frames()
imwrite()
beginthreadex()
mkdir()
call_depth_training()
call_register_visitor()
call_remove_visitor()
call_depth_authentication()
call_check_visitor()
isFaceRecognized()
imageShowPipeline()

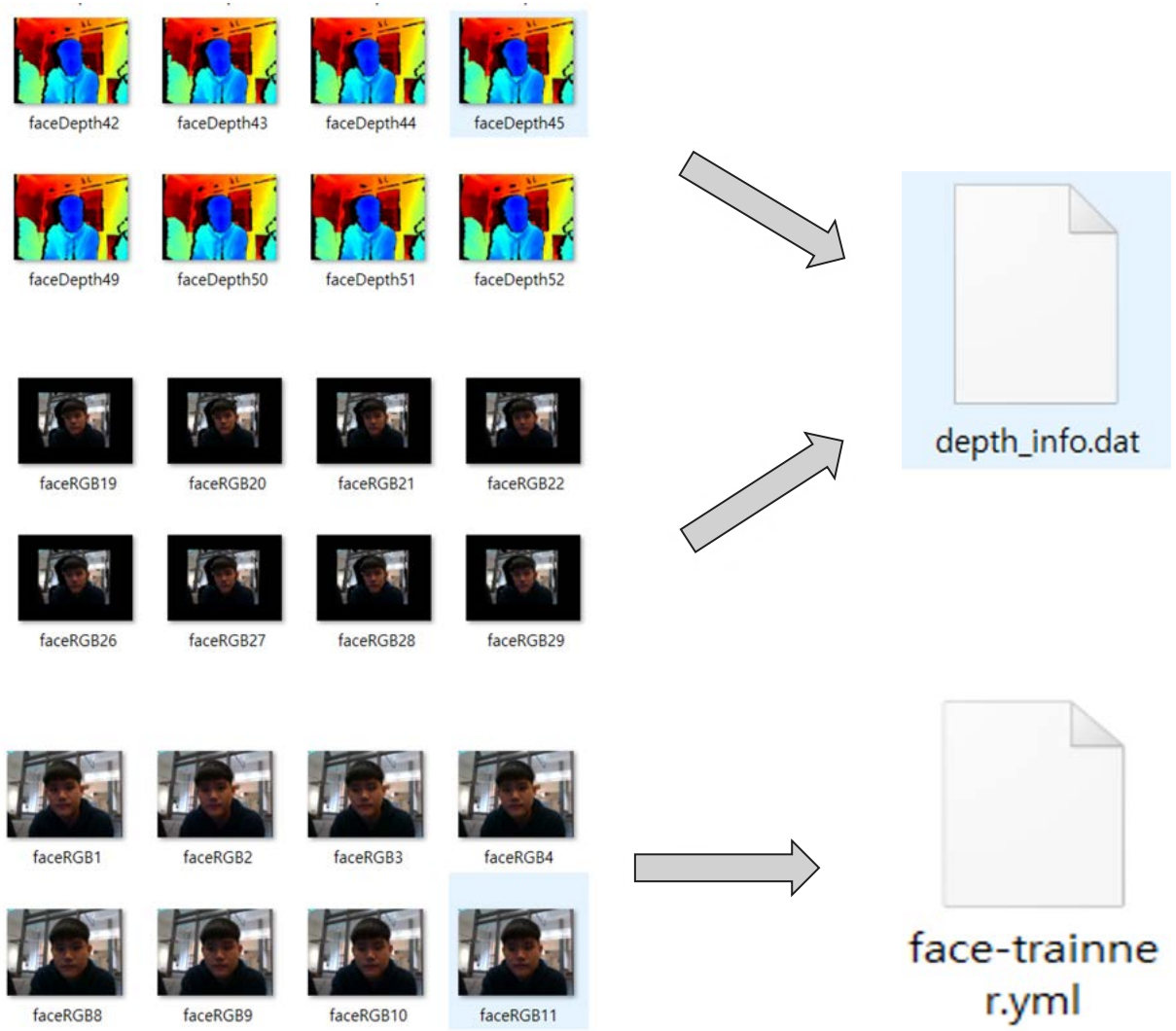
추적성 분석 수행 - 2.2 안면등록



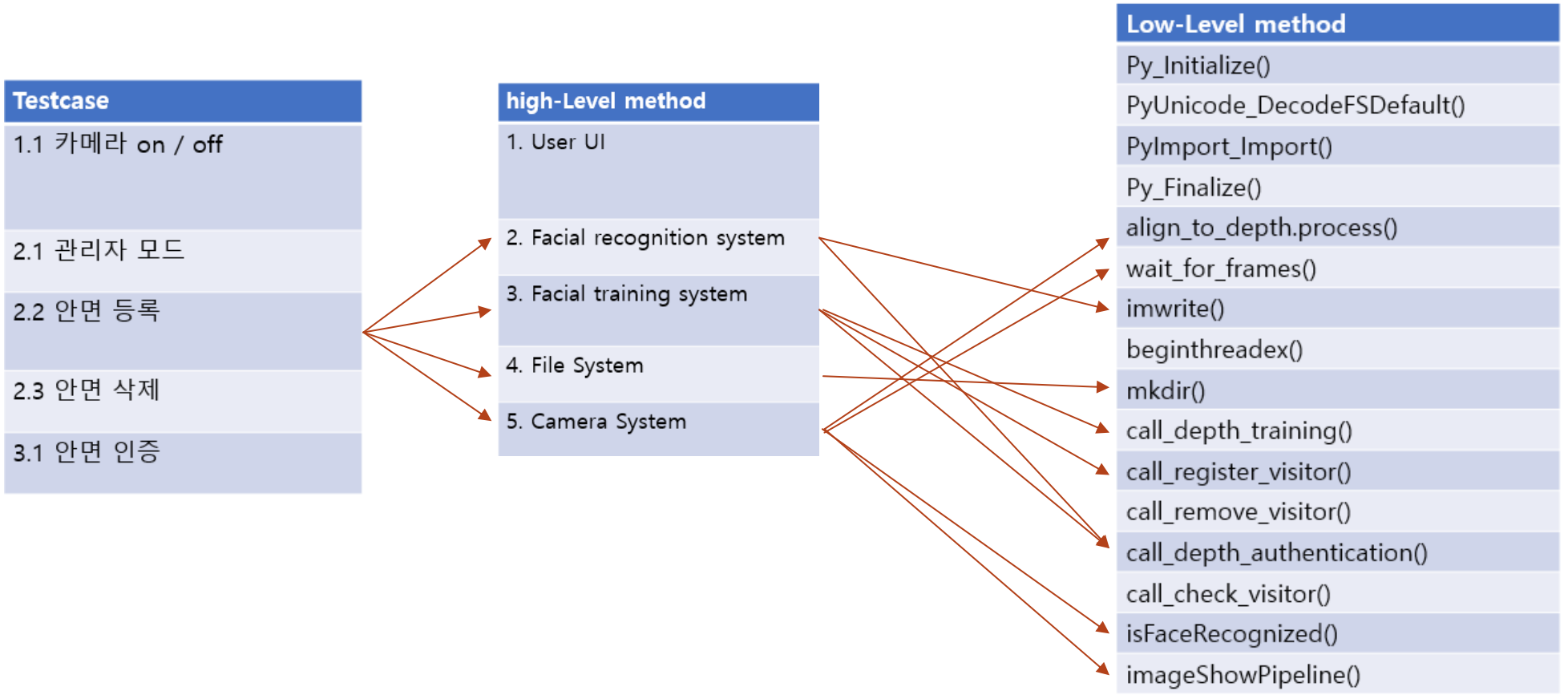




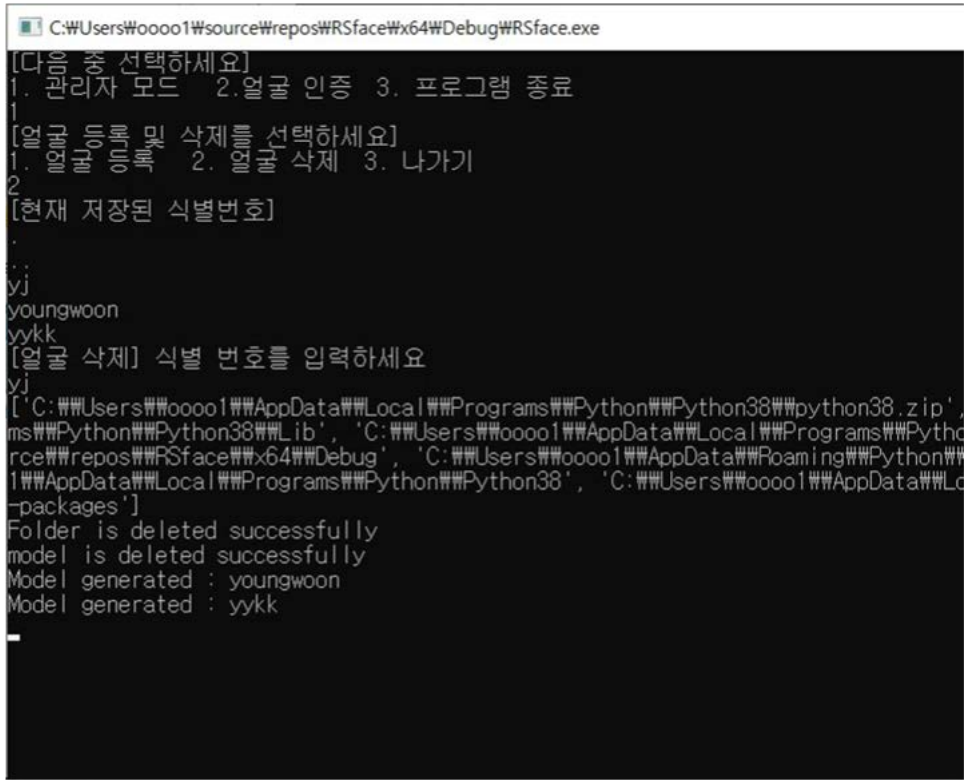
추적성 분석 수행 - 2.2 안면등록

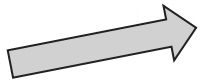


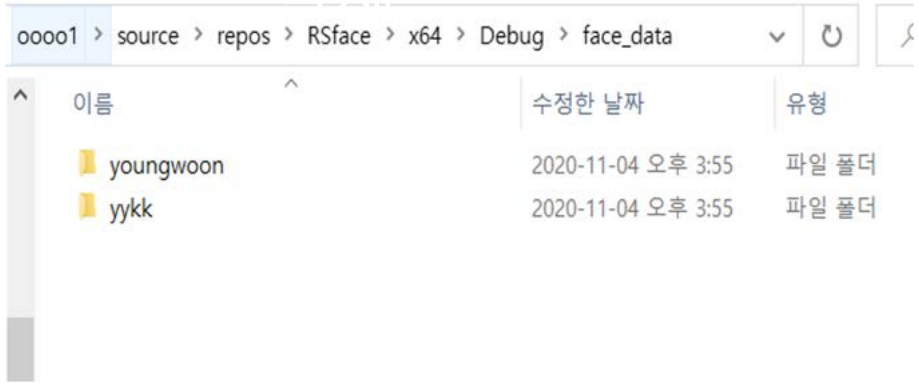
추적성 분석 수행 - 2.2 안면등록

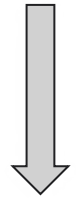


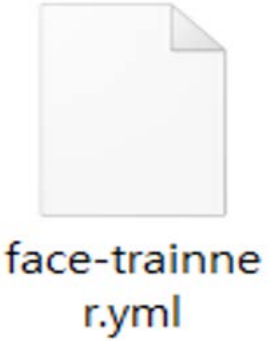
추적성 분석 수행 - 2.3 안면삭제



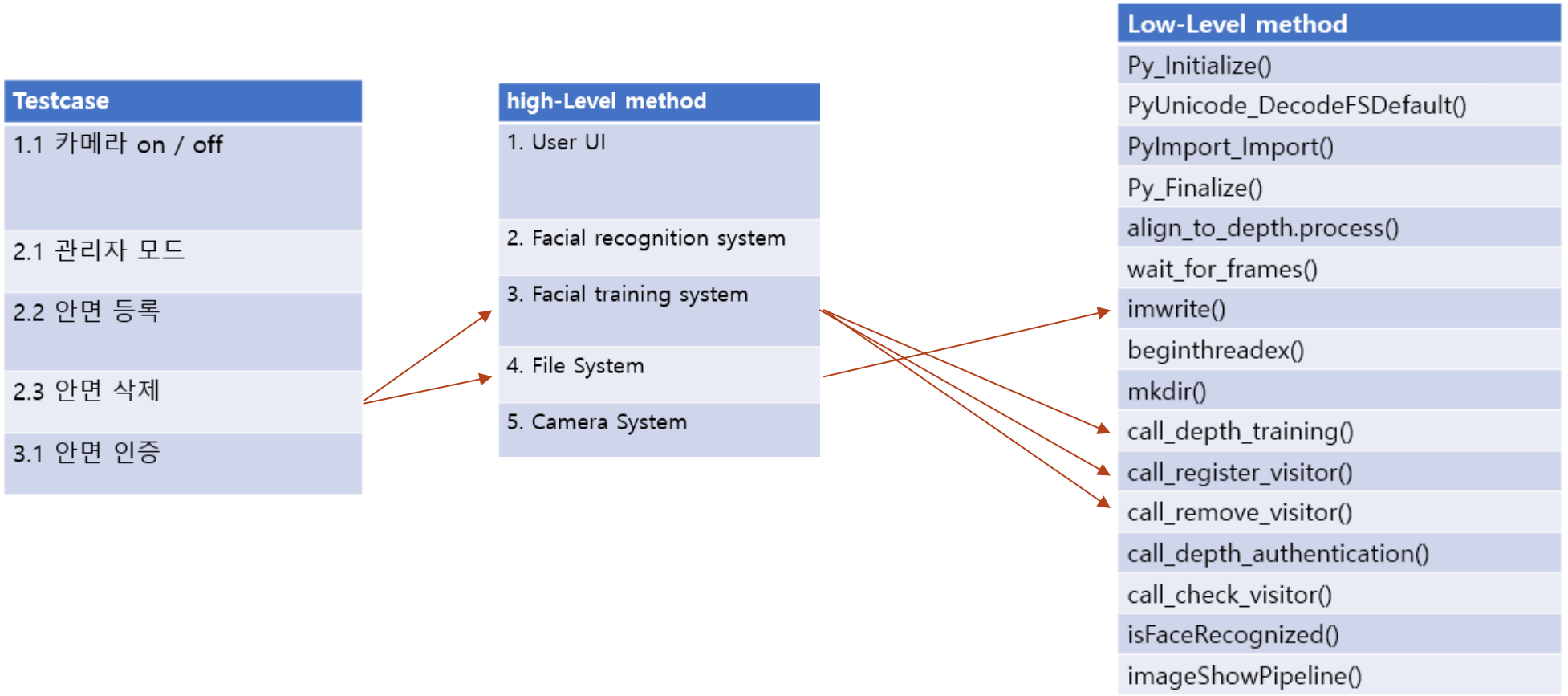




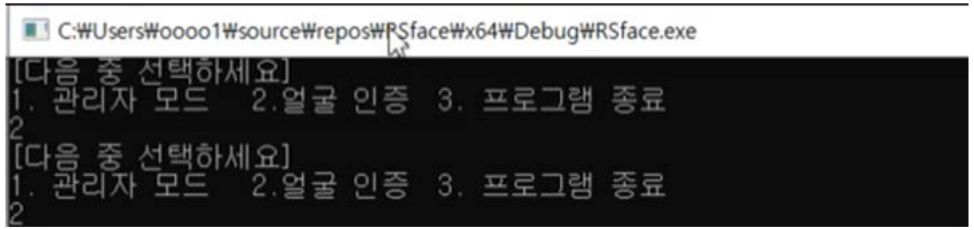


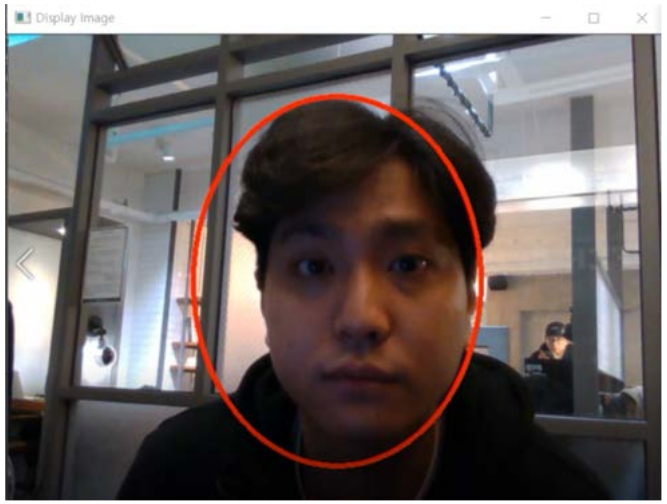


추적성 분석 수행 - 2.3 안면삭제



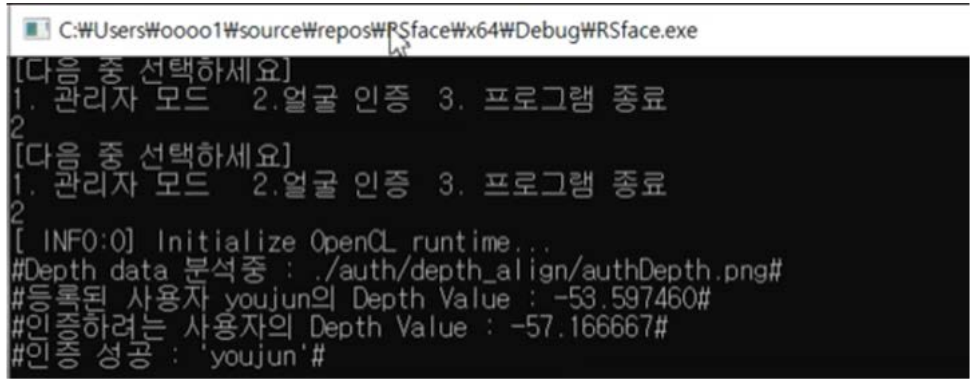
추적성 분석 수행 - 3.1 안면 인증





face-trainner.yml

depth_info.dat



정상 인증

추적성 분석 수행 - 3.1 안면 인증