



연구논문/작품 중간보고서

2020 년도 제 2 학기

| 논문/작품 | ○논문() ○작품(V) ※ 해당란에 체크 | |
|-----------------------------------|---|--------------------------------|
| 제목 | 멀티 카메라 얼굴인식을 통한 아파트 현관문과 엘리베이터 시스템 | |
| GitHub URL | https://github.com/GiHyeon-Choi/GiHyeon-Choi-2013314639- | |
| 평가등급 | 지도교수 수정보완 사항 | 팀원 명단 |
| A, B, F 중 택 1 (지도교수가 부여) | ○ ○ ○ | 최기현 (인) 학번 : 2013314639 |

2020 년 9 월 23 일

지도교수 : 이 호 준 서명

1. 서론

이 프로젝트는 3D 카메라를 이용해 아파트 주민들의 얼굴을 전면이미지와 측면이미지로 저장하고 아파트를 출입할 때 카메라에 얼굴을 보여주면 얼굴 인식 알고리즘을 통해 주민의 신상을 파악하고 공동 현관문의 자동문과 엘리베이터를 작동시키는 시스템이다.

2020년 상반기에 한국에 상륙한 코로나19 바이러스(COVID-19)가 빠른 속도로 확산되고 사람들은 가능한 타인과 접촉을 꺼리는 생활을 하게 된다. 그리고 이번엔 코로나 바이러스가 최대 2주 동안 숙주 없이 생존 할 수 있다는 결과가 발표되고 사람들은 여러 사람이 만지는 물건에 접촉하는 걸 최대한 피하는 경향이 생겼다. 그럼에도 불구하고 반드시 여러 사람이 접촉하는 물체를 만져야 하는 경우가 있는데 그것은 바로 아파트 공동 현관문과 엘리베이터의 버튼들이다. 이러한 상황에서 안면인식을 통해 주민들은 버튼을 누르지 않고 자기가 살고 있는 집까지 이동할 수 있는 방법을 프로젝트로 만들게 되었다.

먼저, 현재 안면인식 기술의 경우 크게 2D와 3D 방식으로 나누어 지는데 2D 안면인식은 얼굴의 평면 사진에서 이목구비의 하이라이트 부분들을 체크 해서 판단하는 방식이고 3D 안면인식은 최근의 Apple Face ID가 대표적으로 적외선이나 입체카메라 등으로 얼굴의 굴곡을 3D depth image로 추출해 판단하는 방식이다. 2D와 3D 안면인식 방식에 대해서는 둘 다 장단점이 있지만 정확도 면에서는 3D 안면인식이 훨씬 높다. 또한 3D 안면인식은 사진 등으로 속이는 행위인 스푸핑(Spoofing)에 2D 안면인식보다 강하다. 그러나 3D 안면인식이 많이 활성화 되지 못한 것은 3D depth image를 가져오는 카메라가 고가여서 였는데 이러한 장단점을 타협해서 2D 카메라를 전면과 측면에 두개 설치해서 비용과 성능 면에서 2D 카메라와 3D 카메라의 중간 정도에 해당하는 효과를 낼 수 있다고 생각되어 프로젝트를 진행하게 되었다.

이번 프로젝트의 목표는 두개의 카메라를 사용해서 전면, 측면 두 얼굴을 평가하는 알고리즘을 만들어서 현재 얼굴이 아파트의 주민이면 현관 자동문을 열어주고 엘리베이터를 해당 주민이 살고 있는 세대로 이동시켜주고 아파트의 주민이 아니라고 판단 되면 출입을 거부하는 기능을 메인으로 한다. 자동문과 엘리베이터의 경우 이번 프로젝트에서 실제로 준비하기에는 곤란하므로 자동문과 엘리베이터의 작동을 상황에 따라 알맞게 애니메이션을 재생시키는 방식을 채택한다. 또한 두개의 카메라를 고정시키고 그 카메라들이 있는 장소는 가능하면 균일한 빛 방향과 세기가 조성되는 환경이어야 한다. 카메라가 준비되면 얼굴이 카메라에 잡혔다고 판별되면 자동으로 캡처하는 기능을 필요로 하고 자동으로 캡처된 얼굴은 DB의 아파트 주민들의 얼굴과 대조해서 판별하는 알고리즘을 거쳐 아파트 주민이면 자동문과 엘리베이터를 작동시키고 주민이 아니면 출입을 거절하고 또는 새로운 주민으로 얼굴을 등록시키는 메뉴를 실행 할 수 있게 한다. 이때 주민들의 편의성을 위해 얼굴인식에 소요되는 시간은 너무 길어서는 안된다. 마지막으로 자동문과 엘리베이터의 작동에 관해서는 자동문은 닫기는 순간에 방금 주민의 얼굴인식에 사용한 임시 정보들을 모두 삭제한다. 그리고 엘리베이터의 경우 엘리베이터가 작동하고 있는 중 또 다른 주민이 얼굴인식을 시키는 상황이 있을 수 있으므로 엘리베이터 작동 정보는 Queue 방식을 사용한다.

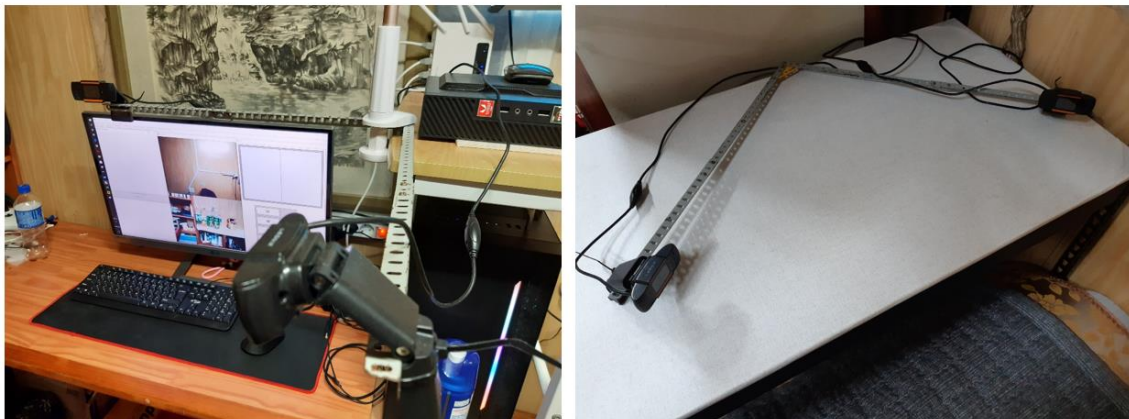


그림 1. 프로젝트의 멀티 카메라 장치

프로젝트에 사용된 장치로서 이미지 프로세싱과 데이터베이스, 코드 실행 등으로 사용할 PC와 정면과 측면에 각각 카메라가 달려있는 형태의 프레임을 준비했다. 두개의 카메라는 PC와 USB(Universal Serial Bus)를 통해 연결되어 있고 가격은 개당 1만원 정도의 금액으로 동일한 모델을 구입 하였다. 카메라가 배치되는 곳은 가능하면 조금 어두운 환경에 배경이 단색으로 되어 있어 얼굴인식에 방해 될 외부 요소들을 배제하는 편이 이상적이다.

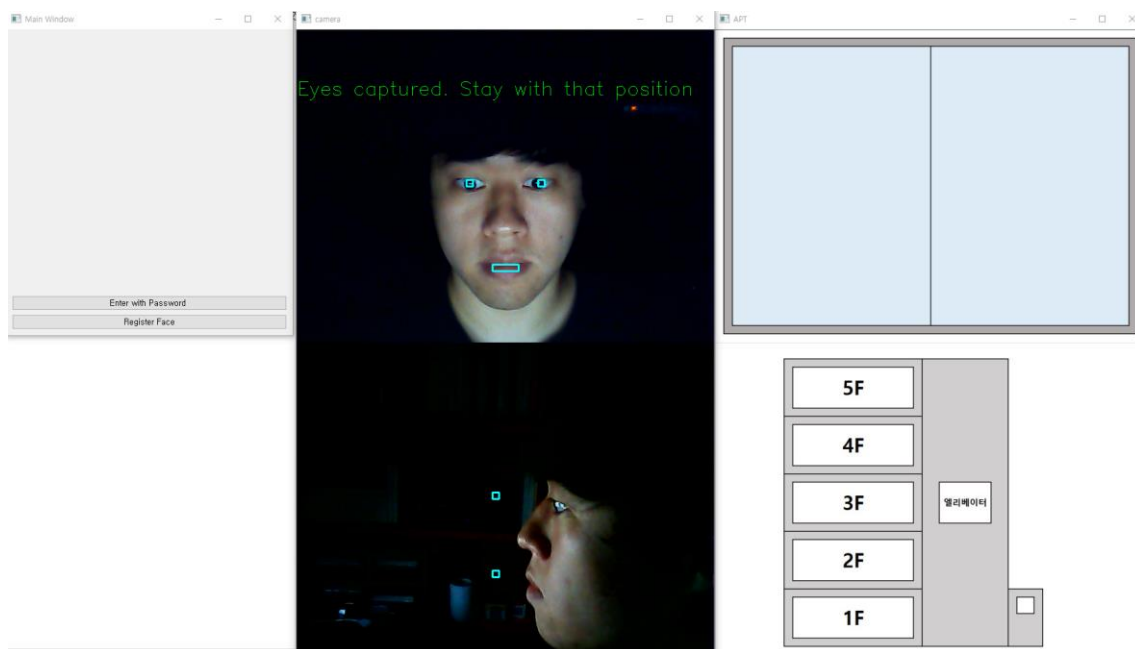





그림 2. 프로젝트의 실행 화면

프로젝트는 9개의 Python 코드 파일과 empty.jpg 파일 하나, 그리고 자동문과 엘리베이터의 동작을 시각화 하기 위해 필요한 이미지 파일들이 포함된 door_img와 floor_img 폴더가 있다.

메인 함수가 실행되면 메인 함수는 PyQt5로 구현된 조작패널 윈도우와 OpenCV로 구현된 camera와 APT 창이 각각 threading을 통해 실행되어서 각자의 기능들이 독립된 스레드로 병렬 실행되어 처리결과를 주고 받는 구조로 되어 있다.

| | | |
|------|------------------|--|
| 운영체제 | Windows 10 64bit |  |
| 언어 | Python 3.8 |  |
| DB | MySQL 6.0 |  |

| | |
|-----|---------|
| API | OpenCV |
| | Pillow |
| | PyQt5 |
| | PyMySQL |

| | |
|----|------------------------------------|
| HW | AMD Ryzen 3 3300X 4-Core Processor |
| | RAM 16.0GB |
| | LEO-C480 USB 카메라 x2 |

그림 3. 프로젝트의 실행 환경

이 작품을 위해 구성한 환경은 Windows 10 64bit , Python 3.8 , MySQL 6.0 이고 API는 OpenCV , Pillow , PyQt5 , PyMySQL 이 사용되었다.














| | | | | |
|---|-------------------|--------------------|-------------|-----|
|  | door_img | 2020-09-15 오전 1:18 | 파일 폴더 | |
|  | faces | 2020-09-22 오후 7:10 | 파일 폴더 | |
|  | floor_img | 2020-09-15 오전 2:55 | 파일 폴더 | |
|  | apt_module.py | 2020-09-22 오전 7:29 | Python File | 4KB |
|  | border_maker.py | 2020-09-22 오전 5:38 | Python File | 3KB |
|  | capture_module.py | 2020-09-22 오전 7:44 | Python File | 8KB |
|  | comparison.py | 2020-09-22 오전 7:12 | Python File | 4KB |
|  | download_faces.py | 2020-09-22 오전 7:20 | Python File | 1KB |
|  | empty.jpg | 2020-09-18 오전 8:18 | 이미지(jpg) 파일 | 2KB |
|  | main.py | 2020-09-22 오전 7:31 | Python File | 1KB |
|  | MainWindow.py | 2020-09-22 오전 7:34 | Python File | 4KB |
|  | PwWindow.py | 2020-09-22 오전 7:36 | Python File | 2KB |
|  | RegWindow.py | 2020-09-22 오전 7:35 | Python File | 3KB |

그림 4. 프로젝트의 파일들

또한 Python 코드 파일들과 폴더는 위 그림과 같다. 폴더 door_img와 floor_img는 apt_module.py 가 사용하는 이미지가 있고 faces에는 주민들의 사진을 임시로 저장한다.

2. 관련연구

이 작품을 진행하기 위해 사용한 언어는 Python 3.8 이다. Python 언어는 높은 범용성과 생산성을 가지고 있고 이 작품을 구현하기 위한 유용한 API들의 생태계가 잘 갖추어져 있는 인터프리터 언어이다. 특히 Python은 OpenCV와 같은 이 작품에서 핵심이 되는 API에 관련 자료가 많고 유사한 구현 사례도 많아서 Python을 채택하였다.

이 작품에서 사용된 API는 크게 OpenCV , Pillow , PyQt5 , PyMySQL이 있는데 각각의 API가 이 작품에 활용 될 수 있는 방법은 다음과 같다.

먼저 OpenCV의 경우 C++언어와 Python 언어를 지원하는 오픈 소스 컴퓨터 비전 라이브러리 중 하나로 크로스플랫폼과 실시간 이미지 프로세싱에 중점을 둔 라이브러리이며 현재는 Python 언어로 활용하는 것이 주류가 된 라이브러리 이다. 이 작품의 목표는 실시간 얼굴을 캡처하고 알고리즘을 실행하는 것이 목표 이므로 OpenCV가 이 작품의 가장 핵심적인 API 이다.

두번째로 Pillow의 경우 Python의 대중적인 라이브러리 로서 간편한 이미지 처리에 특화되어 있다. OpenCV와 역할이 어느정도 겹치지만 OpenCV가 움직이는 화상의 이미지 처리에 특화되어 있는 반면 Pillow는 정지된 이미지의 작업에 특화되어 있다. 이러한 OpenCV와의 차별점 때문에 두 API를 동시에 사용하는 경우가 많고 이 작품에서도 OpenCV와 같이 병행해서 사용하게 되었다.

세번째로 PyQt5의 경우 파이썬으로 GUI를 구현해주는 API이다. 이 작품에서 자동문의 조작패널에 해당하는 장치를 구현하기 위해서 버튼과 메시지 박스, 키보드 입력이 있는 GUI를 필요로 한다. 그렇기 때문에 PyQt5 API를 채택하게

되었다.

마지막으로 PyMySQL API에 대해 설명하자면 PyMySQL은 오픈소스 DBMS인 MySQL을 Python에서 활용 가능하게 해주는 API로서 작품이 실행되는 장치에 설치된 DBMS가 MySQL이기 때문에 PyMySQL을 사용하게 되었다.

추가적으로 이 작품에 사용된 DBMS MySQL 6.0 버전이고 MySQL Browser를 통해 쉽게 데이터를 확인 할 수 있게 하였다. 기본적으로 schema는 facedb를 사용하고 accounts 테이블에 주민들의 정보가 저장된다. 테이블 accounts는 itemid , name , floor , front , side 라는 column으로 이루어져 있고 각각 정수 , 문자열 , 정수 , blob , blob 타입으로 구성되어 있다.

작품 구현에 필요한 API와 DBMS를 소개한 것에 이어 얼굴인식 알고리즘에 사용된 연구자료를 소개하자면 얼굴인식은 크게 인물 캡처 , 데이터 추출 , 신원 확인 세 과정으로 이루어져 있는데 인물 캡처 과정에서는 2D 센서와 3D 센서로 캡처하는 두 가지 방식이 있고 데이터 추출 방식에서는 딥러닝을 사용하는 방식과 딥러닝을 사용하지 않는 방식으로 이루어져 있다.

먼저 2D 센서를 통해 얼굴인식을 하는 방식은 얼굴의 정면만을 2D 이미지로 캡처해서 얼굴 인식을 사용하는 방식이다. 이러한 방식은 필요한 장비가 단순한 카메라 하나만 있어도 되므로 매우 저렴하지만 단점은 등록된 얼굴을 인쇄해서 카메라에 비추는 등의 Face Spoofing 행위를 해도 이것이 실제 등록된 사람의 얼굴인지 아니면 가짜인지를 구별 할 수 없다는 점이다. 또한 얼굴의 깊이를 알기 힘들게 되어 정확도 역시 떨어진다는 단점도 있다.

두번째로 3D 센서를 통해 얼굴인식을 하는 방식은 ToF 방식과 SL 방식이 있는데

전자는 삼성 갤럭시 10 등에 채택하고 후자는 아이폰이 대표적이다. ToF 방식은 레이저를 피사체에 보내고 반사되어 돌아오는 시간을 측정하여 깊이 정도를 파악하는 것이고 SL 방식은 3만개 이상의 특정 패턴의 적외선 도트를 피사체에 방사한 후 피사체 표면 모양에 따라 패턴의 변형된 정도를 분석하는 것이다. 하지만 두 3D 센서들의 공통된 특징이 있는데 둘 다 일반 카메라 센서에 비해 매우 비싼 가격을 형성하고 있다는 점이다. 앞서 말한 두 스마트폰은 매우 고가이고 대량생산을 하기 때문에 3D 센서를 채택 할 수 있었지만 이 작품 컨셉같이 소량생산에 맞춰진 제품에서 사용하기엔 일반 카메라 센서에 비해 공급처를 찾기 힘들다.

그렇기 때문에 2D 센서의 저렴한 가격이라는 장점과 3D 센서의 다방면에서의 얼굴정보 수집이라는 장점의 중간 지점으로 타협해서 2D 카메라 두개를 사용해서 전면과 측면을 동시에 평가하는 얼굴인식 시스템을 계획하게 되었다. 전면에 설치된 카메라 센서는 얼굴의 정면을 평가하고 측면에 설치된 카메라 센서는 얼굴의 측면을 평가하기 때문에 센서를 구입하는데 필요한 비용은 2D 센서와 3D 센서의 중간 정도를 하게 되고 얼굴에서 얻을 수 있는 정보량 또한 2D 센서와 3D 센서의 중간 정도이다. 이 작품은 이러한 2D 센서를 통한 얼굴인식 보단 더 높은 보안을 요구하고 3D 센서를 통한 얼굴인식 보단 더 낮은 비용을 요구하는 상황이 현재 COVID-19로 인해 상호간 접촉을 꺼리는 상황에서 불특정 다수가 필연적으로 접촉 할 수 밖에 없는 아파트 현관과 엘리베이터를 비접촉으로 작동시키고자 하는 수요와 결합 시킬 수 있다는 가능성을 포착해 만들게 되었다.

3. 제안 작품 소개

main.py

main.py 는 이번 작품의 main 으로서 조작 패널을 작동시키는 GUI와 capture_module 과 apt_start를 threading으로 실행 시키는 코드를 포함하고 있다. threading으로 각 모듈들을 실행시키기 때문에 GUI는 다른 모듈과 동시에 실행 된다.

MainWindow.py

MainWindow는 아파트 현관의 자동문 조작 패널을 GUI로 구현한 것으로서 크게 새 주민 등록 기능과 비밀번호를 직접입력해서 자동문을 여는 기능을 포함하고 있다. Class로 이루어진 MainWindow(QMainWindow)는 GUI를 초기화 하는 initUI(self) 와 RegWindow와 PwWindow로부터 받아온 값을 사용하는 onButtonClicked1(self)와 onButtonClicked1(self) 의 def가 있으며 initUI에서 QLabel 과 QPushButton을 통해 작동결과를 메시지를 출력 부분과 얼굴등록, 비밀번호 입력 부분을 구현하였다.

RegWindow.py

RegWindow는 MainWindow의 서브 윈도우로서 새 주민이 얼굴을 등록하고자 하는 상황에서 거주 층수와 이름, 아파트 현관 비밀번호를 입력 받고 MainWindow로 입력받은 정보를 전달한다. 만약 비밀번호가 틀리거나 층수를 선택하지 않으면 등록이 실패하도록 MainWindow에서 구성되어 있다.

PwWindow.py

PwWindow 또한 MainWindow의 서브 윈도우로서 만약 현재 사람이 얼굴인식이 아닌 단순히 아파트 현관 비밀번호를 입력해 출입하고 싶을 때 사용한다. 비밀번호를 입력받고 비밀번호가 맞으면 아파트 현관문이 열리는 애니메이션이 출력되고 틀리면 MainWindow에서 비밀번호가 틀렸다는 메시지를 출력한다

capture_module.py

capture_module은 이 작품의 핵심적인 코드 중 하나로서 두 대의 카메라 센서를 작동시키고 얼굴이 포착되었다고 판단되는 사진이 캡처되면 border_maker와 comparison을 통해 얼굴을 DB의 얼굴들과 대조시키게 하는 역할을 한다. 기본적으로 capture_module 자체는 눈동자가 가이드라인에 들어오면 그 자세를 유지하라는 메시지를 표시하고 그 상태에서 약 3초간 얼굴자세가 유지되면 캡처해서 얼굴을 분석하는 코드를 실행시키기 동적 화상이기 때문에 capture_module은 현재 카메라 센서에 포착된 화상이 사람인지 아닌지 판별하는 속도가 빨라야 한다^[8]. 만약 카메라 센서에서 보내는 초당 수십장의 화상 하나하나를 너무 자세하게 분석하려 한다면 프로세서의 리소스 낭비가 심해 질 것이다. 그러므로 capture_module은 가능한 빠른 처리를 위해 양쪽 눈과 입을 가이드 라인으로 그리고 그 가이드 라인에 눈동자 있는지 여부만 판별^[9]한다. 눈동자를 검출하는 알고리즘은 양 눈동자와 그 양 옆의 흰자의 평균 색상을 검출^[6]하는 간단한 알고리즘으로 이루어져 있으며 순간적인 노이즈를 눈동자로 판별하는 것을 막기 위해 cnt 변수로 눈동자가 가이드라인에 들어온 횟수를

측정¹⁹해 일정 이상 눈동자 검출횟수가 누적되어야 현재 화상을 캡처하도록 하였다. 눈동자가 가이드라인에 들어오면 OpenCV의 message 기능으로 현재 눈동자가 포착되었으니 잠시만 그 자세를 유지 해 달라는 메시지를 표시한다. 캡처가 완료되면 얼굴분석 및 비교 알고리즘이 실행되어서 약 2~3초간 카메라 화상이 정지된다. 캡처된 얼굴의 사진은 흑백으로 변환 된 후 contrast 값이 높혀지고 불필요한 부분이 crop되고 얼굴비교 알고리즘의 속도를 높이기 위해 resize 되어서 jpg 파일로 저장 되고 바로 직후 border_maker 함수에서 사용된다.

border_maker.py

border_maker는 얼굴을 특징을 검출하는 기능을 하는 코드이다. 이 코드는 capture_module에서 캡처해서 저장한 jpg 파일의 픽셀을 분석해서 그 픽셀의 주변 픽셀들의 색상이 현재의 픽셀과 얼마나 다른지를 검출²⁰해서 변화가 클수록 그 픽셀의 위치에 더 큰 값을 저장 한 뒤 새로운 이미지로 만들어 낸다. 쉽게 말해서 얼굴의 명암을 미분하는 코드라고 할 수 있다. 이 코드는 comparison 함수에서 비교하기 편한 형태로 얼굴을 변환시켜 주는 역할을 하며 카메라 센서 주변 환경의 밝기가 달라져도 비교적 균일한 이미지로 변환이 된다. 그렇기 때문에 대체로 빛의 환경을 많이 받는 아파트 현관의 환경에서 이 코드를 통한 이미지 변환은 얼굴비교 알고리즘에 상당한 도움이 된다. border_maker를 거쳐서 변환된 이미지 jpg 파일로 저장되어 직후 comparison 함수에서 쓰게 되고 또한 DB에 등록될 이미지로 사용된다.

download_faces.py

download_faces는 DB에서 주민들의 얼굴 사진들을 현재 얼굴분석 중인 사람의 얼굴과 비교하기 위해 임시로 faces 폴더에 받아서 저장하는 기능을 수행한다. 얼굴분석과 비교가 끝나면 사진 파일들은 삭제된다.

comparison.py

comparison 코드는 capture_module과 더불어 이 작품의 핵심적인 코드로서 faces 폴더의 주민들의 얼굴과 현재 현관문 앞에 서 있는 사람의 얼굴을 비교하고 유사도를 측정하는 코드이다. comparison.py는 comparison 함수와 comparison_each 함수로 나누어져 있는데 comparison 함수에서 DB의 주민들의 정면과 측면 얼굴을 하나씩 읽으면서 comparison_each 함수로 현관 앞에 서있는 사람의 얼굴과 비교한다. comparison_each 함수는 방금 찍은 사진을 n분할 된 각 좌표계로 변환한다.

사람이 사람의 얼굴을 볼 때 미간을 중심으로 본다는 점에서 착안해서 만들어진 이 방식은 미간을 중심으로 하는 각좌표계⁷⁾로 얼굴을 저장하면 사람이 얼굴을 인식할 때 시선의 중심점에 가까운 부분 일수록 높은 밀도로 얼굴 정보를 저장시킬 수 있다. 또한 이러한 방식은 타겟 얼굴과 비교시 쉽게 사진을 회전시킬 수 있으므로 사진을 찍을 때 미세하게 각도가 기울어 진 경우에 기울어진 사진을 만들어 내기 쉽다.

이렇게 각좌표계로 구현된 얼굴사진은 대상주민의 얼굴과 여러가지 각도와 상하좌우 이동의 경우에서 픽셀 대 픽셀로 비교된다. 각각의 주민들의 얼굴을

대상으로 이러한 반복수행을 하면서 일치하는 픽셀이 있으면 약간의 가점을 주고
틀린 픽셀이 있으면 많은 감점을 해서 계속 similarity를 갱신하게 한다.

apt_module.py

apt_module은 아파트의 자동문과 엘리베이터를 OpenCV를 사용해
애니메이션으로 구현한 것으로 자동문은 얼굴인식 코드가 생성하는 txt 파일을
읽어서 내용을 읽고 이 사람이 거주민이 맞는지를 확인하고 1층에 살고 있다면
자동문만 열고 2층 이상에 살고 있다면 엘리베이터를 작동시켜서 먼저
엘리베이터를 1층으로 보낸 후 거주민이 살고있는 층으로 이동한다. 엘리베이터가
작동하는 동안 또 다른 주민이 얼굴인식을 완료하는 경우가 있을 수 있으므로
엘리베이터의 목적지는 Queue 방식으로 저장된다.

create.sql

create.sql은 facedb 스키마를 생성하고 itemid, name, floor, front, side의 column을
가진 accounts 테이블을 생성한다.

Init.sql

Init.sql은 itemid 값이 1인 빈 정보를 insert 해서 초기화 하는 역할을 한다

delete.sql

delete.sql은 현재 등록된 모든 얼굴 정보들을 삭제한다

4. 구현 및 결과분석

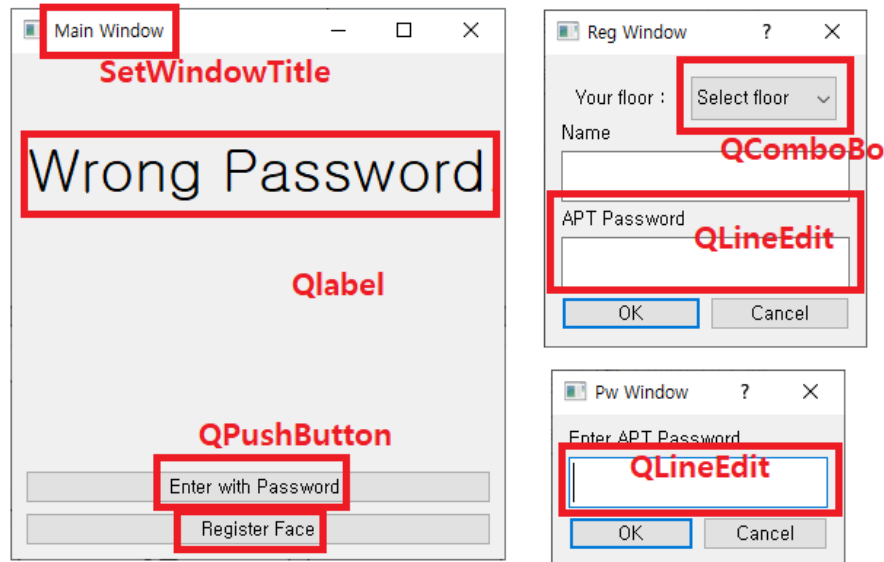


그림 5. PyQt5로 구현된 GUI의 구성

PyQt5로 구현된 GUI는 위와 같이 구성되어 있다. Main Window는 Reg Window와 Pw Window를 서브 윈도우로 가지고 있고 각각의 서브들은 Main Window의 두 버튼들을 통해 실행된다.

Main Window의 기능은 두 서브 윈도우들의 작동 결과를 메시지로 알려주는 역할을 한다. 예를 들어서 RegWindow에서 층수를 선택 하지 않으면 "Register Fail! You did not select the floor" 를 출력하고 아파트 비밀번호를 틀리면 "Register Fail! Wrong Password!" 를 출력한다.

Reg Window는 새 주민이 얼굴을 등록하고자 할 때 쓰인다. Reg Window는 거주 층수를 선택하는 QComboBox와 이름과 비밀번호를 입력하는 QLineEdit으로 이루어져 있다

Pw Window는 아파트 현관 자동문을 직접 비밀번호를 입력해 열 때 사용 되는 서브이다. 비밀번호가 맞으면 문이 열리고 틀리면 "Wrong Password"가 표시된다.

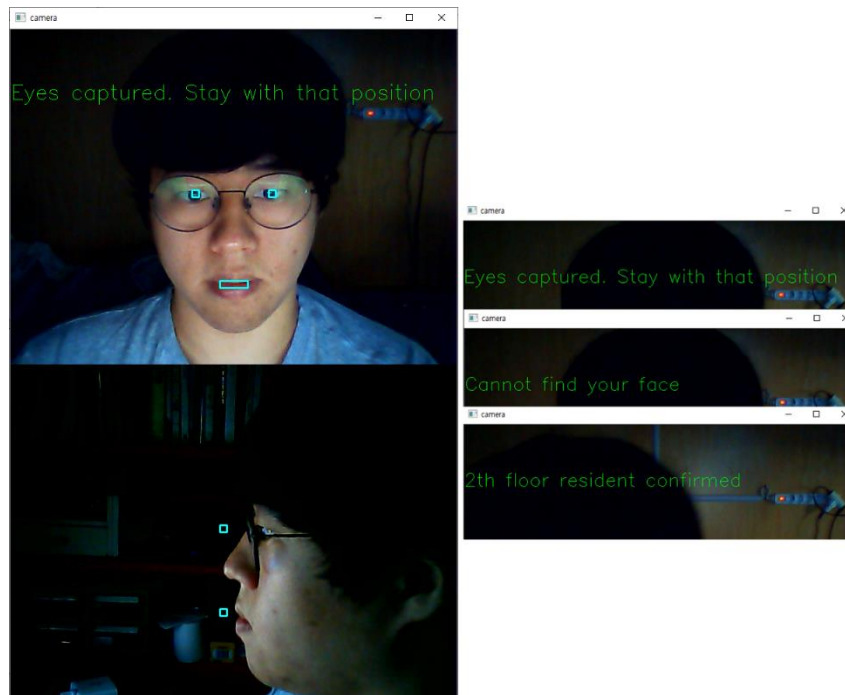


그림 6. 카메라 윈도우 모습과 메시지들

OpenCV의 imshow function으로 실행되는 camera의 경우 실시간으로 카메라를 작동시키고 눈동자가 포착되면 현재 화상을 캡처하고 흑백화와 명암 대비화를 거친 이미지를 저장한 뒤 얼굴 분석과 비교에 필요한 함수들을 실행시키는 역할을 한다.

카메라 두대를 동시에 사용하기 위해 전면 카메라인 0번 카메라와 측면 카메라인 1번 카메라를 VideoCapture 함수로 읽게 되는데 이렇게 읽은 이미지에 눈과 입의 위치를 표시한 대략적인 가이드 라인을 그린 뒤 vconcat 함수로 합친다. 이때 화상 이미지는 flip 함수를 통해 거울처럼 작동하게 해서 주민들이 얼굴 위치를 조절 할 때 불편함이 없도록 한다.

눈동자가 가이드라인에 맞춰지면 "Eyes captured. Stay with that position" 메시지가 출력되고 그 상태가 2~3초간 유지되면 얼굴 분석 및 대조 코드가 실행되어 2~4초간 화면이 정지 된다. 그리고 이 얼굴이 주민이 아니라 판단되면 "Cannot find your face"가 출력되고 주민이 맞으면 "nth floor resident confirmed"가

출력된다.

capture_module의 경우 초당 수십장의 이미지를 실시간으로 프로세싱 해야 하기 때문에 얼굴을 판별하는 코드는 양쪽 눈의 검은자와 흰자가 일정 프레임 이상 연속으로 검출되면 화면을 캡처하는 간단한 방식으로 이루어져 있다. 또한 측면 사진의 경우 가이드 라인에서 좌우로 약간 벗어나기 쉬운데 이러한 경우는 캡처 후 border_maker 코드가 측면사진의 수평 위치를 재조정 하는 방식으로 보완 된다.

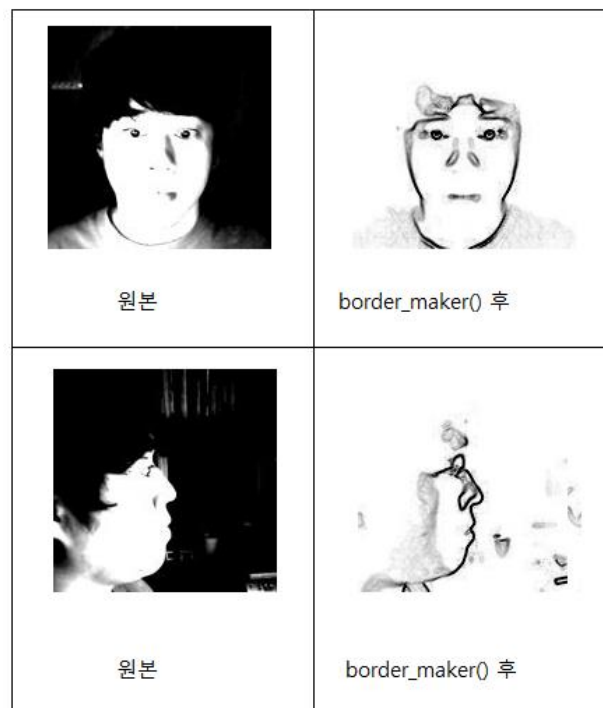


그림 7. 경계선 그리기 함수를 거치기 전과 후

코드 border_maker.py는 capture_module이 저장한 흑백화와 명암 대비화가 완료된 jpg 이미지를 나중에 comparison 코드가 사용하기 쉽도록 명암 변화의 경계선을 그려준다. 그림에서 알 수 있듯이 border_maker 함수를 거친 사진은 명암의 경계일수록 진한 선으로 그려준다.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 2 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

그림 8. 경계선 그리기 알고리즘의 간략화

이 함수를 구현하기 위해서 고안한 알고리즘은 위의 간략화 된 그림 처럼 한 픽셀에서 상하좌우로 자신과 다른 명암의 픽셀이 얼마나 있는지를 계산한다. 흑백의 정도는 0에서 255까지가 있으므로 border_maker 함수는 한 픽셀의 명암과 상하좌우로 있는 픽셀들의 명암 차를 절대값으로 누적한다. 한마디로 말해서 border_maker는 사진의 명암을 미분하는 함수이다.

또한 border_maker 함수는 측면의 사진의 위치를 재조정 하는 역할을 한다. 앞서 말했듯이 얼굴을 캡처할 때 정면사진은 대체로 가이드라인에 맞게 찍히지만 측면 사진은 좌우가 약간 틀어지는 경우가 많다. 그렇기 때문에 측면사진의 윤곽선 위치들의 평균 수평위치를 측정하여 중앙에 가깝게 재조정 해준다.

코드 border_maker의 과정이 끝나면 capture_module은 download_faces와 comparison를 실행하게 된다. 여기서 download_faces는 DB에 있는 주민들의 얼굴을 다운로드 받아서 faces 폴더에 저장하고 comparison 함수는 faces의 주민들 얼굴과 현재 얼굴을 비교하는 과정을 실행한다. 이 과정에서 쓰이는 사진들은 모두 border_maker를 거친 사진들이기 때문에 border_maker 함수는 얼굴이 캡처 되는 순간에만 실행된다.

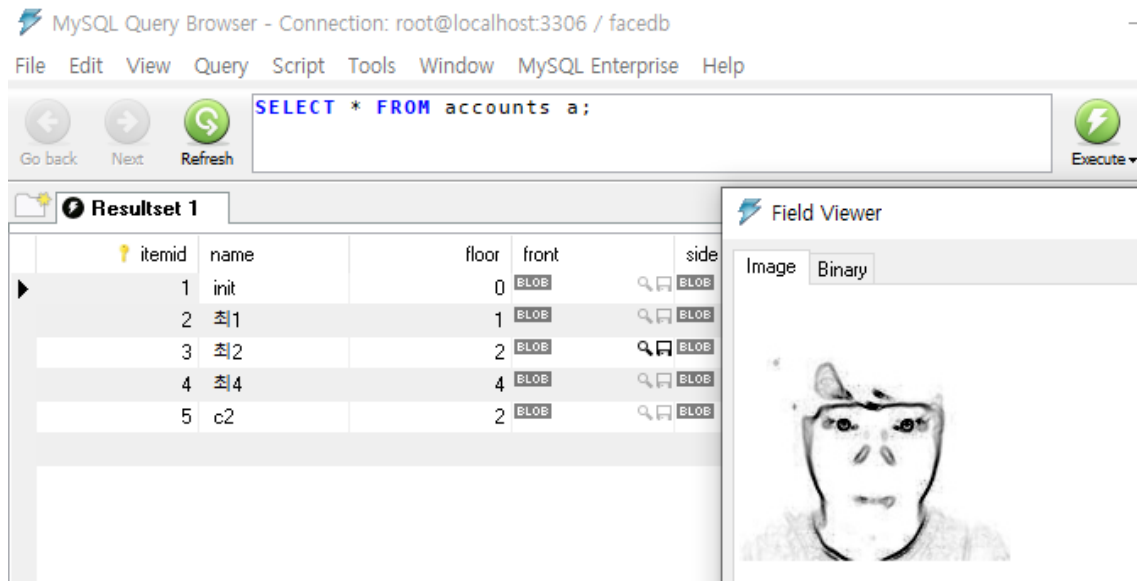


그림 9. MySQL Browser로 본 facedb의 accounts 테이블

코드 `download_faces.py`는 facedb 스키마의 accounts 테이블에 있는 모든 주민들의 얼굴을 다운로드 받아 faces 폴더에 저장한다. 이때 이미지는 blob으로 저장되어 있고 이미지 파일의 크기는 평균 5kb 정도로서 한 사람당 정면과 측면 얼굴 두개 씩 저장되므로 row 하나당 10kb 정도의 용량이다. `download_faces`가 끝나면 `comparison` 함수가 실행된다.



그림 10. 얼굴 이미지를 n분할 각좌표로 변환한 모습

얼굴비교에 사용된 방식은 얼마나 많은 픽셀이 일치하는가를 평가하는 단순한 방식이지만 비교에 사용되는 방금 캡처한 사람의 얼굴에 먼저 적용되는 이미지 프로세싱은 새로운 방식을 적용하였다. 사람의 얼굴을 n 분할 한 각좌표계에 담는 방식이다. 이러한 방식을 고안한 이유는 사람이 사람의 얼굴을 볼 때 미간을 주시하므로 미간에 가까울수록 안면인식에 중요 할 것이라는 추측 때문이다. 미간에 가까운 지점일수록 윤관이 뚜렷 해지고 얼굴을 평가하는데 중요하지 않은 부분들은 자연적으로 deemphasized 된다. 또한 이러한 방식의 다른 장점은 얼굴의 이미지를 회전시키거나 축소, 확대 또는 상하좌우 이동 모두 쉬워진다는 점이다. 각좌표계로 구현된 얼굴을 다시 평면좌표계로 그릴 때 radius를 조절하는 방법으로 확대축소를 할 수 있고 시작 angle을 조절하는 식으로 시계방향 또는 반시계 방향으로 미세한 조절이 가능하다.

각좌표계로 옮겨진 얼굴은 각 주민들과 얼굴을 비교할 때 확대, 축소, 회전 또는 평면이동 되어서 다양한 변형이 생성되어 평면좌표계로 그려진 후 각 주민들의 얼굴과 비교된다. 이때 확대, 축소, 회전, 평면이동의 범위를 정하는 수치를 조절할 수 있어서 유사도를 검출하는 알고리즘의 정확도를 높힐 수 있다. 하지만 범위를 너무 크게 하면 for문의 실행 횟수가 기하급수적으로 늘어나서 소요시간이 크게 늘어 날 수 있다는 점을 고려해야 한다.

추가적으로 comparison_each 함수가 반환하는 각 사진들에 대한 유사도인 similarity의 예시를 몇 개 소개 하자면 다음과 같다.









| | | |
|---|---|-------------------------|
|  사진1 |  사진1 | 유사도 3993 같은 사진 |
|  사진1 |  사진2 | 유사도 864 동일인물 다른사진 |
|  사진1 |  사진3 | 유사도 325 동일인물 안경착용 |
|  사진1 |  사진4 | 유사도 0 전혀 다른 사진 |

그림 11. 각좌표로 변환된 얼굴 이미지와 여러 조건의 대상 이미지들 간의 유사도

위 그림에서 볼 수 있듯이 픽셀의 일치도를 검사하기 때문에 동일한 사진 두개를 비교대상으로 실행하면 유사도가 매우 높게 나온다. 그리고 동일인물이지만 다른 사진 두개를 넣고 실행하면 유사도가 높게 나오지만 동일 사진을 넣었을 때 보다는 많이 낮은 수치를 보여준다. 또한 동일인물이 안경을 착용하고 인식시키면 유사도가 많이 낮아진다. 마지막으로 전혀 다른 사진을 넣으면 0을 출력한다.

이러한 테스트 결과를 바탕으로 similarity의 수치가 몇 이상이어야 동일인물로 판단할지 결정해야 했는데 500으로 설정하고 실행시켰을 때가 상당히 적당한 결과를 보여주었다.

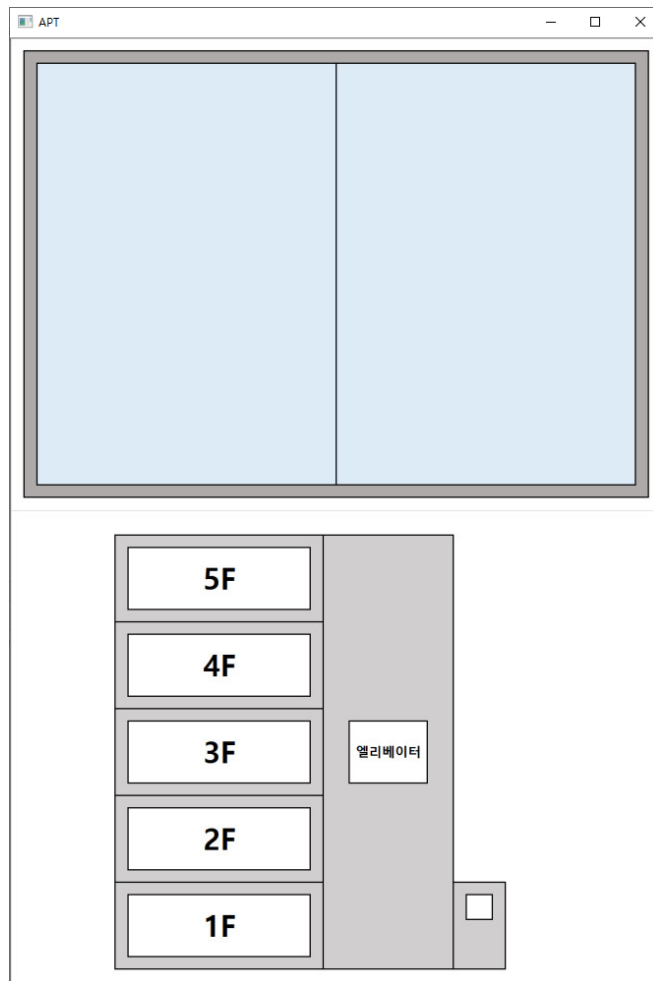


그림 12. APT 윈도우의 디자인

마지막으로 APT 창은 얼굴인식이 완료되고 생성되는 txt 파일을 토대로 거주민이 어떠한 물체에도 접촉하지 않고 본인의 세대 까지 도달할 수 있도록 자동문과 엘리베이터를 작동 되는 걸 시각화 한 창이다. 그림의 윗 부분이 자동문이고 비밀번호가 일치하거나 얼굴인식이 통과되면 문이 열리는 애니메이션이 재생된다. 아랫부분은 엘리베이터의 시각화이고 Queue 방식으로 이동위치를 저장하고 작동한다.

5. 결론 및 소감

이번 프로젝트를 진행하면서 기존 계획과 많이 바뀐 부분이 있었다. 가장 큰 변경점은 3D 카메라를 사용하는 방식에서 멀티 카메라로 얼굴인식을 하게 바뀌었다는 점과 딥러닝을 사용해서 얼굴인식을 하려던 것을 픽셀 비교로 바뀌게 된 점이다. 전자의 경우 장치를 구하지 못하고 관련 한국어 자료가 없어서 변경하였고 후자의 경우 관련 지식이 부족해서 시도하기 힘들어서 였다. 하지만 그럼에도 불구하고 변경된 방안도 기존의 계획 보다는 장점이 몇가지 있었다. 우선 제작에 필요한 장치의 금액이 많이 저렴했다는 점이고 실행속도가 예상보다 빠르면서 얼굴인식 정확도가 기대 이상으로 높았다는 점이다.

또한 프로젝트를 진행하면서 현실의 여러 사물의 정보를 컴퓨터가 처리하기 쉬운 형태로 재가공 하는 것이 생각보다 매우 어려운 일이라는 것을 느꼈다. 이 프로젝트를 진행하면서 가장 커다란 부담으로 다가온 것이 어떤 방식으로 사람의 얼굴을 서로 비교하는지에 대한 것이다. 사람은 사람의 얼굴을 쉽게 구별 할 수 있지만 그것이 [a,b,c] 가 $n*m$ 크기 만큼 담긴 리스트의 형태가 되고 그것을 사용해 컴퓨터가 서로 얼마나 유사한지 판별 할 수 있는 형태로 재가공 하는 것은 쉬운 일이 아니었다.

처음 주제를 선정했을 때는 도전 해볼만 하다고 생각했지만 막상 해보니 많은 부분에서 막혀서 그다지 많은 기능을 구현하지 않았음에도 몇달에 걸쳐 프로젝트를 하게 되었다. 이번 프로젝트를 계기로 본인이 얼마나 자기계발을 소홀히 했는지 느끼게 되었고 또한 새로운 분야들에 대한 트렌드도 알게 되었다.

6. 참고문헌

1. **Somar Boubou ; Tatsuo Narikiyo ; Michihiro Kawanishi** ,2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM) , Object recognition from 3D depth data with Extreme Learning Machine and Local Receptive Field
2. **Youngjae Yun ; Donghyeon Seo ; Donghan Kim** , 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) , Recognition of transparent objects using 3D depth camera
3. **Jukka Komulainen ; Abdenour Hadid ; Matti Pietikäinen** , 2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS) , Context based face anti-spoofing
4. **Mark Draelos ; Qiang Qiu ; Alex Bronstein ; Guillermo Sapiro** , 2015 IEEE International Conference on Image Processing (ICIP) , Intel realsense = Real low cost gaze
5. **Peng Zhang, Fuhao Zou, Zhiwen Wu, Nengli Dai, Skarpness Mark, Michael Fu, Juan Zhao, Kai Li**; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019, FeatherNets: Convolutional Neural Networks as Light as Feather for Face Anti-Spoofing
6. **조수장, 권세현, 황승진, 황호현, 유지연**; 한국정보통신학회 2018 년도 춘계학술대회, 2018, OpenCV 를 이용한 영상에서의 특정 영역 검출
7. **오재현, 곽노준**; 전자공학회논문지-SP, 2010, 극좌표계 변환에 기반한 얼굴 인식 방법
8. **이호근, 정성태**; 정보과학회논문지 : 소프트웨어 및 응용, 2005 , 빠른 얼굴 검출을 이용한 실시간 얼굴 인식 시스템
9. **김진필, 조영복**; 한국디지털콘텐츠학회 논문지, 2018 , OpenCV 를 이용한 홍채 영상에서 병변 추출 영상처리 기법