

# 2020-1 인터넷기술융합

## ICT 융합 기술 보고서

1713941 조혜민

### -목차-

#### I. 얼굴 인식 기술에 대한 문헌 조사 및 비교

- A. 문헌 조사
- B. 문헌 비교

#### II. Amazon Rekognition 플랫폼을 활용한 융합 프로젝트 구현

- A. 이미지 및 비디오 분석 플랫폼 <Amazon Rekognition>
  - 1. 소개 및 선택 동기
  - 2. 얼굴 인식 기술
  - 3. 얼굴 인식 기술 활용 방안
- B. 소프트웨어 설치, 튜토리얼 예제 실행 및 결과
  - 1. 소프트웨어 설치
  - 2. 예제 실행 및 결과
  - 3. 예제 실행 과정 중 발생한 문제와 해결 방법
- C. 얼굴 인식 기술을 활용한 출입 관리 및 제한 구역 감시 시스템
  - 1. 프로젝트 설계
  - 2. 구현
  - 3. 결과 분석 및 발전 방향

#### III. 참고 문헌

## I. 얼굴 인식 기술에 대한 문헌 조사 및 비교

### A. 문헌 조사

#### 1. 사용자 인증을 위한 딥러닝 기반 얼굴인식 기술 동향 [1]

개인별 서비스를 제공하기 위해 인증이 요구되는 시점에서 생체인증 서비스가 다양해지고 있고 그 중에서도 편리성이 높은 얼굴인식이 사용자 인증에 많이 활용되고 있다.

얼굴인식 기술은 사전에 입력된 여러 이미지 등 영상에서 얼굴 영역 추출하는 단계, 각 이미지에서 추출된 얼굴 영역에서 특징 추출하는 단계, 새로운 이미지와의 매칭을 통한 식별하는 단계의 3단계를 거쳐 이미지로부터 사용자를 식별하는 기술이다. 단계 별로 사용되는 기술을 알아보았다.

얼굴 영역 추출: 입력 영상에서 얼굴 부분만을 검출하는 단계로 개인의 얼굴이 고정된 모양을 갖지 않기 때문에 다양한 변형이 가능하여 얼굴 영역 추출에 어려움이 있다. 얼굴 템플릿에 기반하여 얼굴 영역을 추출하는 방법과 눈, 코, 입과 같은 얼굴 요소의 위치를 기반으로 특징 벡터를 계산하여 얼굴 부분을 추출 하는 방법을 주로 사용한다.

얼굴 영역에서 특징 추출: 얼굴 영상으로부터 식별을 효율적으로 높이기 위해 얼굴 구성 요소인 눈, 코, 입을 찾아 특징 (Feature) 추출한다. 고차원 신호의 차원을 낮추어 통계적 방법으로 특징 추출하는 방법(PCA)과 선형 판별 분석 방법(LDA)이 사용된다. 이 방법은 계산 속도가 빠르고 모델이 저장된 DB의 크기가 크지 않아도 된다는 장점이 있다.

새로 입력된 이미지와의 매칭: 특징 벡터 값을 DB에 저장하고, 등록된 특징 벡터 값과 비교하여 식별한다. 최근에는 SVM 기법을 이용하여 얼굴의 작은 영역에서 요소만 추출한 후 특징 벡터로 활용하여 인식하는 기술이 사용되고 있다.

얼굴 인식 기술 예시: 구글의 FaceNet, 페이스북 DeepFace가 대표적인 얼굴인식기술이다. 카메라를 통해 개인의 얼굴을 찍으면 Face API가 제공되는 서버에 전달이 되고, HOG 를 통해 얼굴을 인식, 재배치하여 128개의 벡터를 추출한다. 추출한 값으로 SVM으로 분류하여 얼굴을 인식하는 기술이다.

#### 2. 얼굴 인식 및 인증 알고리즘 성능분석 [2]

이 논문에서는 대표적인 얼굴 인식 및 인증 알고리즘의 성능과 정확도에 대한 분석을 실시 하였다. 얼굴 인식 기술에서는 Adaboost를 이용한 Haar-like 기반 알고리즘과 LBP 기반 알고리즘에 대해서 성능 분석을 실시 하였고, 얼굴 인증 기술에서는 LBPH 기반 알고리즘, Eigenface 기반 알고리즘, Fisherface 기반 알고리즘에 대해서 성능을 분석하였다.

먼저 얼굴 인식 부분에서는 LBP 기반 알고리즘이 Haar-like 기반 알고리즘 보다 학습 속도, 인식 속도 각각 1.95배, 3.8배의 좋은 성능을 보여 주었다. 하지만, 인식률은 Haar-like 기반 알고리즘이 LBP 기반 알고리즘 보다 8% 정도 우수한 것으로 결과를 확인 할 수 있었다. 얼굴 인식을 사용하려는 용도에 따라 속도와 인식률을 고려하여 두 알고리즘 중 하나를 선택하여 사용하면 좋을 것으로 보인다.

얼굴 인증 알고리즘 중에서는 학습 속도 부분에서 Eigenface 기반 알고리즘 대비 LBPH 기반 알고리즘은 약 2.15배 정도 빠르고 Fisherface 기반 알고리즘은 약 1.74배가 빠른 것으로 분석되었고, 인증 속도는 Eigenface 기반 알고리즘 대비 LBPH 기반 알고리즘이 약 5.76배가 빨랐으며, Fisherface 기반 알고리즘은 약 15.29배로 월등히 빠른 것으로 나타났다. 이 결과로 보면 실시간 영상처리와 같은 빠른 처리가 요구되는 부분에서는 Fisherface 기반 알고리즘이 가장 적합하고, 정확하게 얼굴 인증이 요구되는 시스템에서는 LBPH 기반 알고리즘을 선택하는 것이 좋은 결과를 얻을 수 있을 것으로 보인다.

얼굴 인증 및 인식에서 인식률은 많은 양의 다양한 학습 및 테스트 데이터를 이용하여 더 높은 인식률을 낼 수 있도록 학습이 가능하다. 하지만, 학습을 통해서 인식 속도를 높일 수는 없다. 인식 속도는 결국 알고리즘의 속도와 비례하는 것이다. 바로 이런 점을 고려하여 알고리즘 선택해야 한다.

### 3. 일반 카메라 영상에서의 얼굴 인식을 향상을 위한 얼굴 특징 영역 추출 방법 [3]

카메라로 촬영한 영상에서 기존 얼굴 특징 성분을 추출하는 방법에는 많은 어려움이 존재하는데, 대표적으로 얼굴 영상에서 불필요한 화소 값들이 존재하는 경우, 이 화소 값들에 의해 얼굴 특징 점들의 위치가 변하여 얼굴 인식 단계에서 오인식을 유발한다.

이러한 문제점을 해결하기 위해 본 논문에서 제안하는 방법은 카메라로 입력된 영상에서 얼굴 검출을 실행한 후 얼굴 영역에서 눈과 입 검출, 양 쪽 눈과 입 영역을 이용한 기준점 검출 및 얼굴 외곽선 검출, 얼굴 외곽선을 이용한 불필요 영역 제거, 얼굴 영상 크기 정규화, LDA기반 특징 추출, ANN기반 얼굴 특징 학습 및 인식 순서로 처리하는 방법을 제안하였다.

기존 얼굴 인식 방법은 얼굴 영상에서 별도의 얼굴 영역을 추출하지 않고, 바로 얼굴의 특징 성분을 추출하므로 인식단계에서 불필요한 영역이 합성된 얼굴 영상을 사용하는 경우 그 영역이 포함되어 잘못 인식되거나 학습된 얼굴 영상과의 얼굴 크기의 차이로 인해 얼굴 인식률이 평균 78% 정도로 낮게 나타났다. 또 다른 방법은 얼굴 색상 정보를 이용하여 얼굴 영역을 추출하는 방법인데, 촬영된 이미지에 따라 얼굴 영역 정보가 손실되는 문제가 발생하거나 불필요 영역이 일부 포함되는 경우가 있어 평균 86% 정도의 인식률을 나타내었다. 하지만 새로 제안하는 방법에서는 얼굴 외곽선 검출을 통해 얼굴 영역을 추출하여 얼굴 인식에 적용한 결과 94% 정도의 높은 인식 결과를 나타내었다.

### 4. 딥러닝을 이용한 얼굴인식 성능 비교 [4]

이 논문에서 비교한 얼굴 인식 방법은 기하학적 방법, 특징점을 기반으로 한 방법, 심층 학습 얼굴 인식 방법이다. 첫번째로 기하학적 방법에는 밝기 정보, 색상 정보를 이용하여 얼굴과 배경을 구분한다. 그러나 경계 부근 화소들의 밝기 값이 유사할 경우 또는 배경이 복잡한 경우 얼굴 영역을 정확히 추출할 수 없는 한계를 가진다. 다음으로 특징점 기반 얼굴인식에서 특징 추출은 눈, 코, 입 등 상관관계를 고려하여 얼굴 영역을 추출하는 방법으로 높은 안정성을 보이고 있으나 처리시간이 많이 걸린다는 단점이 있다.

몇 가지의 얼굴 인식 알고리즘 성능을 실험한 결과 주성분 분석 알고리즘과 LBP 특징점 알고리즘보다 선형판별법의 얼굴 인식률이 95.55%의 결과를 보였다. 그 이유는 선형판별법이 얼굴의 다양한 표정변화와 각도에도 영향을 받지 않는 강한 얼굴 인식 알고리즘이기 때문이다.

실제 수행 성능을 분석하기 위해 실시간 USB 카메라 영상에서 이용자의 얼굴을 검출하고 인식하는 실험을 하였다. 그 결과 GPU 임베디드 시스템을 이용한 얼굴인식 방법이 가장 효과적인 것을 확인하였다. 향후 GPU기반의 임베디드 시스템으로 실시간 얼굴 인식 시스템을 이용하여 출입통제 또는 차량 사용자 확인 등에 활용할 수 있을 것으로 보인다.

## 5. 적외선/깊이 영상을 이용한 강인한 얼굴인식 방법 [5]

얼굴 인식에 사용되는 영상 중 적외선 영상과 깊이 영상은 조명에 불변하고, 특히 깊이 영상의 경우 얼굴의 입체적 정보를 이용하므로 사진, 디스플레이를 이용한 위변조 공격으로부터 안전해지는 장점이 있다. 적외선 영상은 색상 영상과 달리 조명의 영향을 적게 받기 때문에 색상 영상을 이용하는 방법보다 얼굴 인식의 정확도가 높지만, 위변조 공격에는 취약하다. 깊이 영상은 적외선 영상과 같이 조명의 영향을 적게 받고, 거리 정보를 함께 얻기 때문에 사진을 이용한 위변조 공격으로부터 안전하다는 장점이 있다. 그러나 깊이 영상은 색상 영상이나 적외선 영상에 비해 잡음이 많이 포함되어 있어 단독으로 얼굴인식에 쓰일 때는 인식률이 떨어진다. 이러한 단점을 극복하기 위해, 두 영상의 장점을 살리고 단점을 보완 하는 얼굴 인식 방법을 제안하였다.

적외선/깊이 영상을 이용해 얼굴인식을 수행하는 과정은 다음과 같다. 먼저 깊이 영상으로부터 객체를 추출하고 추출된 객체로부터 얼굴을 검출한다. 여기서 얼굴검출의 계산량 감소 및 얼굴인식을 위한 전처리를 목적으로 깊이 영상을 이용한다. 그 후 얼굴의 주요 랜드마크 추출 결과를 바탕으로 얼굴의 포즈 및 거리를 정규화 한다. 얼굴 포즈 추정에 필요한 주요 7개의 특징점은 입, 오른쪽 눈썹, 왼쪽 눈썹, 오른쪽 눈, 왼쪽 눈, 코, 턱 이다. 다음으로 얼굴 영상으로부터 특징을 추출하고 추출된 특징 히스토그램을 데이터베이스와 비교한다. 마지막으로 얼굴의 움직임 분석하여 위변조를 판별한다. 분석한 위변조 확률이 임계치 이하인 경우 본인임을 인증한 것으로 본다. 모의 실험을 통해 앞에서 제안한 얼굴인식 방법의 성능을 측정한 결과 ERR 1.7%로 우수한 성능을 보임을 확인하였다

## 6. 얼굴 인식을 위한 효율적인 신경망에 관한 연구 [6]

기존에 얼굴 인식에 사용되던 Softmax 손실 함수에 대하여 공용 데이터셋이 아닌 직접 수집한 데이터를 활용하여 특징점 임베딩을 시각화하는 실험을 수행하여 Softmax 손실 함수의 문제점을 확인하였다. 또한, 클래스간 데이터의 불균형이 신경망의 정확도에 미치는 영향을 파악하기 위하여 클래스간 데이터의 분포를 다양하게 설정하여 정확도 테스트를 수행하였고 클래스 간 데이터의 불균형보다 데이터의 수가 더 중요하다는 점을 알아냈다.

이 연구에서는 손실 함수의 변형이 아닌, 특징점 임베딩을 효율적으로 하기 위한 방법론을 제안하였다. 실험 결과 손실 함수의 변형 없이 특징점 추출 신경망만을 변경함으로써 성능을 향상할

수 있음을 보였다. 다양한 관점에서 특징점 임베딩을 하기 위하여, U-Net, Attention 등의 구조를 사용하였다. Attention을 활용한 구조에서는 LFW, AGE-DB30의 데이터에서 정확도가 향상함을 보였다. 추가적으로, 위에서 구성한 얼굴 인식 신경망을 사용하여 API 서버를 구축함으로써 실제 상황에서 실시간으로 활용 가능함을 입증하였다.

## 7. 딥러닝 기반 얼굴인식 라이브러리 활용 및 보완에 관한 연구 [7]

딥러닝 기반 얼굴인식 라이브러리 중 2015년 발표한 FaceNet 논문을 활용한 방법이 얼굴인식분야에서 가장 큰 정확도를 보인다는 것을 확인하였다. 이 연구에서는 이러한 FaceNet Library를 직접 구현해 보면서 보완할 점을 알아보았고, 그 결과 FaceNet 시스템을 활용한 Library를 연구한 결과 표정변화 감지의 의존성을 보완해야 함을 발견했다. 이는 얼굴인식 과정에서 사용하는 feature point가 사람의 표정을 결정짓는 부분과 밀접한 연관이 있기 때문이다. 표정이라 함은 주름으로 기인한 얼굴 명암이라는 기준을 두고 가변성을 갖는 인자이다. 얼굴인식 알고리즘들이 명암에 의존적이기 때문이다. 그리고 얼굴인식 알고리즘 중에서 명암에 가장 영향을 받지 않는 OpenFace에 사용된 알고리즘 또한 얼굴의 명암으로 인한 표정에 영향을 받는다. 따라서 얼굴인식 알고리즘에 표정분석 알고리즘을 추가로 접목시킬 필요가 있다고 본다. 또한 본 연구로 하여금 FaceNet 시스템이 얼굴인식을 벗어나 표정 분석기반 비지도 학습이 가능할 수 있음을 처음으로 시사하였다. 이 라이브러리를 사용하여 범죄자, 실종자를 찾는 데 도움을 줄 수 있고 시각장애인을 위한 시스템을 구축할 수도 있다고 본다. 우리나라는 아직 오픈소스 및 라이브러리에 대한 공헌을 개인의 선택에만 맡기고 있는 실정이다. 따라서 오픈소스 및 라이브러리 개발에 대한 제대로 된 보상과 인식전환이 국가적으로 요구된다고 본다.

## 8. 다중 생체 인식 기법을 이용한 개인 인증 기술에 관한 연구 [8]

현재 스마트폰 시장이 성장하면서 세계 생체인증시장 또한 같이 성장하고 있다. 특히 모바일의 하드웨어 성능이 높아지면서 기존의 지문인증 외에도 홍채인식 및 안면인식 등이 적용되고 있으며, 생체인증 기술은 헬스케어, 핀테크, 모바일 의료복지, 위치기반서비스, 출입관리, 자동차, 검역, 범죄수사, 공공서비스 등 광범위한 분야에서 적용이 가능하다. 향후 10년간 금융, 헬스케어, 정부 부문이 생체인식 시장의 주요 산업으로 자리매김하고 홍채, 음성, 지문 인식이 생체인식 방식 중 제일 큰 매출을 올릴 것으로 예상하고 있다.

이 연구에서 생체 인증 기술의 향상을 위해 제시한 다중 인증 방법은 다음의 순서로 진행된다. 먼저 지문 및 얼굴 정보를 등록 이후, 사용자 요청에 따라 지문과 얼굴 인증을 진행한다. 그리고 1개의 정보라도 불일치할 경우 사용(동작) 불가 상태로 넘어가며, 2가지의 인증정보가 동일하면 기능 및 동작이 작동한다. 지문과 얼굴인증의 경우 1인당 100장의 사진을 등록 및 저장 이후 인증 요청이 들어오면 100장의 사진과 비교하여 80% 이상의 매칭 비율을 보이면 인증 통과가 되는 방식을 선택하였다. 실험 결과 1인 등록 및 단일 인증의 경우 지문 및 얼굴 인증 성공률이 96%로 나타났으며, 다인(3인)등록 단일 인증 성공률은 92%로 나타났다. 이후 1인 등록 다중 인증 및

다인 등록 다중 인증에 대한 실험결과는 각각 98% 및 94%로 나타났다.

이 방법은 기존의 지문 인증이 가지고 있는 잘리거나 메마른 피부, 붓대를 감았거나 피부가 굳은 손가락은 인증이 어렵다는 단점과 얼굴 인증이 가지고 있는 조명 및 환경과 영상의 각도에 민감하며, 변장, 세월이 흐르면서 생기는 얼굴 변화, 성형수술, 쌍둥이의 유사한 얼굴 특징 등을 구분하는데 어려움이 있다는 각각의 단점을 보완하기 위한 방법이다.

## 9. 얼굴 인식 Open API를 활용한 출입자 인식 시스템 개발 [9]

현재 널리 사용되고 있는 ID 카드 기반의 출입자 인식 시스템은 ID 카드의 위변조에 취약하다.

때문에 이를 방지하기 위해서 생체 인식 기반의 출입자 인식 시스템 개발 수요가 높아지고 있다. 기존의 얼굴 인식 기반 출입자 인식 시스템은 자체적으로 얼굴 인식 모듈을 개발하여 출입자를 인식하는 방법을 주로 사용하고 있다. 하지만 기존의 방법은 제한된 수의 얼굴 인식 모델을 사용한다는 단점이 존재하기 때문에 다양한 환경에서 일정 수준의 인식률을 유지하기 위한 추가적인 개발 비용이 수반된다.

따라서 본 논문에서는 BetaFace, Lambda, KAIROS와 같은 얼굴 인식 Open API를 기반으로 출입자 인식 시스템을 개발하고자 한다. 제안하는 출입자 인식 시스템은 크게 Open API Adapter, User Interface, Client Adapter, Message Router 네 부분으로 나누어지며 동작 과정은 다음과 같다.

먼저 User Interface는 사용자로부터 명령을 입력받아 Client Adapter 내의 Client Interface에게 전달한다. Client Adapter 내의 Query Generator는 사용자로부터 전달된 명령어를 시스템에 맞게 해석한 후 내부 메시지를 생성한다. 생성된 메시지는 Message Router로 전달되어 메시지의 유효성을 검증한다. 메시지가 유효할 경우에는 Routing Manager를 통해 전달되어야 할 경로를 설정한다. 전달될 경로가 설정된 메시지는 WorkFlow Manager를 통해 해당 Adapter로 전송된다. Message Router는 출입자 인식 시스템에 카메라와 도어락 등이 추가되어 출입자 관리 시스템으로의 확장 시 내부 Adapter 또는 모듈 간의 통신을 담당하기 위해 존재한다. Open API Adapter는 기존의 Open API들과 통신하고 결과를 양상불하는 역할을 수행한다.

이 방식의 성능을 측정한 결과, 얼굴 인식률이 단일 Open API 인식률에 비해서 더 우수함을 보였다. 향후 연구로는 상황에 따라 사용하는 얼굴 인식 Open API마다 가중치를 다르게 설정하는 양상불 알고리즘을 개발하고 출입자 관리 시스템으로 확장하고자 한다.

## B. 문헌 비교, 장단점 분석

위의 논문에서는 얼굴의 특징을 추출하고 인식하는 과정에서 각각 선형 판별 분석 방법과 LBP 기반, Haar-like 기반 알고리즘을 사용하였다[1][2]. 선형 판별 분석 방법은 계산 속도가 빠르고 모델이 저장된 DB의 크기가 크지 않아도 된다는 장점이 있다. LBP 기반 알고리즘은 학습 속도와 인식 속도에서는 강점을 갖지만 인식률은 Haar-like 알고리즘보다 떨어진다는 단점이 있다.

이러한 단점을 보완하고 인식률을 높이기 위해 얼굴 외곽선 검출법을 함께 사용한다면 인식률을 높일 수 있을 것이다[3]. 또한 선형 판별 분석 방법은 얼굴의 각도와 표정에 영향을 적게 받기 때문에 LBP기반 알고리즘보다 더 높은 얼굴 인식률을 보인다[4].

다음으로, 얼굴 인식 알고리즘에 사용되는 Softmax 손실 함수의 문제점을 파악하기 위해 실험한 결과, 데이터의 수가 정확도에 미치는 영향이 큰 것을 알게 되었다. 따라서 손실 함수의 변형이 아닌 특징점 데이터 추출 방법을 변경하여 Softmax 함수의 성능을 향상시켰다[6].

얼굴 인식에 사용되는 영상에 대해 분석한 논문에서는 적외선 영상과 깊이 영상을 함께 사용하는 방법을 제안하였다[5]. 깊이 영상의 경우 얼굴의 입체적 정보를 이용하므로 사진, 디스플레이를 이용한 위변조 공격으로부터 안전해지는 장점이 있고, 적외선 영상은 색상 영상과 달리 얼굴 인식의 정확도가 높다는 장점이 있다. 하지만 깊이 영상에는 잡음이 많이 포함되어 있어 단독으로 쓰일 경우 인식률이 떨어진다. 적외선 영상의 경우 위변조의 위험이 있으므로, 서로의 단점을 보완하고 장점을 살리기 위해 두 영상을 모두 사용하여 얼굴 인식을 수행하였고 높은 인식률을 확인할 수 있었다.

생체 인식을 활용한 시스템을 분석한 논문에서는 다중 생체 인식 기법을 제안하였다[8]. 지문 인 증은 지문의 변화가 쉽다는 단점을 가지고 있고, 얼굴 인 증은 조명 및 환경에 민감한 단점을 가지고 있어 서로 상호 보완을 위해 지문 인 증과 얼굴 인 증을 함께 진행하였다. 1인 등록 다중 인 증의 경우 98%의 높은 성공률을 보였다. Open API를 활용한 얼굴 인식 시스템 개발을 제안한 논문도 있었다[9]. 이 시스템에서도 앞에서 살펴본 다중 생체 인식 기법[8]과 비슷하게 다중 인 증을 진행하였다. 차이점은 위의 논문에서는 지문 인식과 얼굴 인식이라는 서로 다른 생체 인식 기법을 사용하였고, 이 논문에서는 Open API를 활용한 얼굴 인 증을 여러 번 진행하였다는 점이다. 이 방법 역시 Open API 인 증을 한 번만 진행했을 때보다 더 향상된 인식률을 보였다.

## II. Amazon Rekognition 플랫폼을 활용한 융합 프로젝트 구현

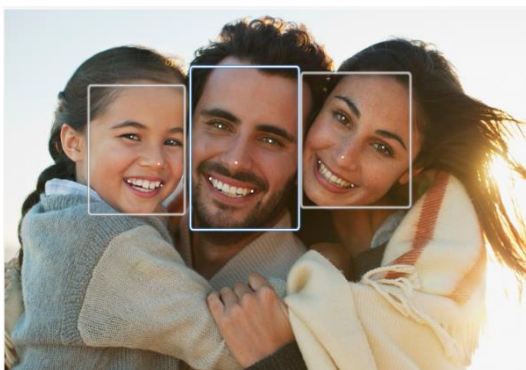
### A. 이미지 및 비디오 분석 플랫폼 <Amazon Rekognition>

#### 1. 소개 및 선택 동기




Amazon Rekognition은 Amazon에서 개발한 클라우드 기반 이미지 분석 플랫폼이다. 이 플랫폼은 확장성이 뛰어난 딥 러닝 기술을 사용하여 이미지 및 비디오 분석을 수행한다. 이미지 및 비디오에서 객체, 사람, 텍스트, 장면 및 활동을 식별하고 부적절한 콘텐츠를 탐지할 수 있다. 또한 다양한 사용자 확인, 사람 수 계산, 공공 안전 사용 사례를 위해 얼굴 탐지, 분석 및 비교하는 데 사용할 수 있는 매우 정확한 얼굴 분석 및 얼굴 검색 기능을 제공한다. 활용성이 높고 다양한 기능을 구현할 수 있어 얼굴인식 기술을 활용하기 위한 플랫폼으로 Amazon Rekognition을 선택하게 되었다.

#### 2. 얼굴 인식 기술

Amazon Rekognition에서는 이미지와 비디오에서 얼굴이 나타나는 순간을 탐지하여 아래 예시와 같이 여러 명의 얼굴을 한 번에 인식할 수 있다. 인식한 얼굴에서는 성별, 연령대, 표정, 안경 착용 여부, 머리 스타일과 같은 기본적인 속성을 확보할 수 있다. 예측한 속성은 확률의 형태로 신뢰 수준을 나타낸다. 이 확률 값에 임계치를 설정하여 일정 확률 이하인 경우에는 표시하지 않거나 지정된 다른 값을 표시하도록 설정할 수도 있다. 얼굴 감지 알고리즘에서는 이미지에서 가장 큰 100개의 면을 감지하고 각 면에 대해 상세 정보를 분석한다. 상세 정보에는 얼굴의 경계, 눈코 입 등의 좌표, 수염, 안경 등이 포함된다.



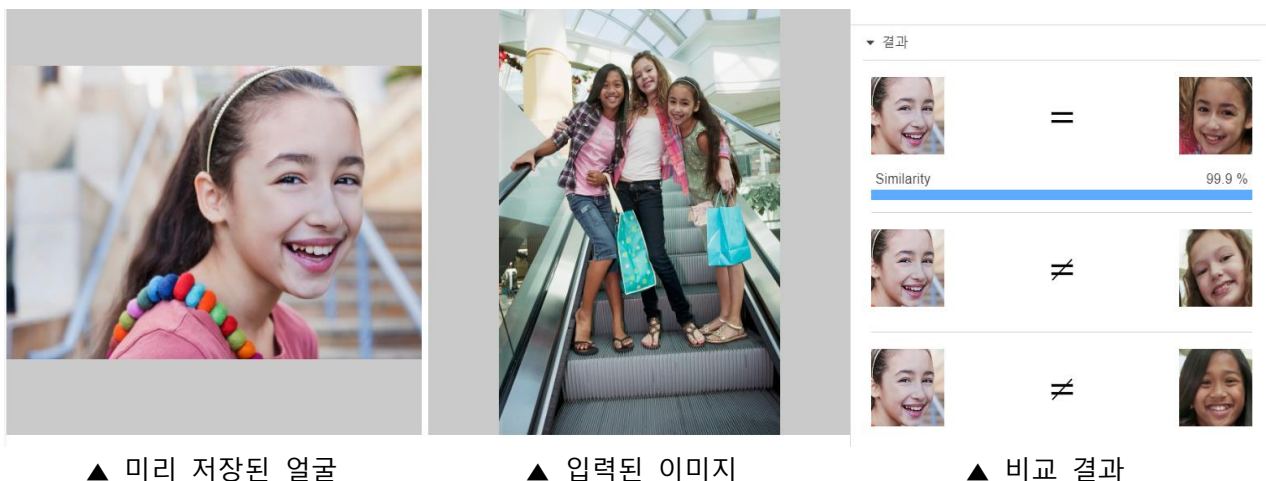
▲ 분석할 이미지

	>		
looks like a face		99.9 %	
appears to be male		99.4 %	
age range		22 - 34 years old	
smiling		99.9 %	
appears to be happy		99.7 %	
not wearing glasses		99.6 %	
	>		
looks like a face		99.9 %	
appears to be female		95.6 %	
age range		5 - 15 years old	
smiling		99.6 %	
appears to be happy		99.5 %	
not wearing glasses		99.4 %	
	>		
looks like a face		99.9 %	
appears to be female		99.1 %	
age range		22 - 34 years old	
smiling		99.9 %	
appears to be happy		99.8 %	
not wearing glasses		99.8 %	

▲ 이미지 분석 결과



또한 인식한 얼굴로 얼굴 비교 시스템을 구현할 수 있다. 이 기능은 인식한 얼굴이 기존에 등록된 얼굴과 일치하는지를 보여준다. 먼저 비교할 이미지를 저장하고 인식한 얼굴과 여러 특징들을 비교해 신원을 확인할 수 있다. 다음과 같이 저장된 얼굴 이미지와 입력된 이미지를 비교한다. 이 시스템에서는 이미지의 밝기, 표정, 털, 연령과 관계 없이 얼굴 골격이나 눈, 코, 입의 위치에 대한 정보를 비교하고 일치하는 확률 값으로 반환한다. 이 확률 값의 전체 일치 확률이 일정 확률 이상일 경우 같은 얼굴로 인식한다. 이 임계 확률을 적절하게 조절해야 얼굴 비교가 제대로 이루어지지 않는 경우를 방지할 수 있다.



### 3. 얼굴 인식 기술 활용 방안

#### i. 디지털 신원 확인

자동 결제 및 출입 통제를 위해 미리 저장된 신분증 사진과 현재 얼굴 이미지를 비교하여 사용자에게 대한 신원 확인을 수행할 수 있다.

#### ii. 실종자 탐색

실종자 정보를 데이터베이스에 저장하고 CCTV 영상을 비교하여 실종자 구조 작업을 도울 수 있다.

#### iii. 나이, 연령대 별 구매 패턴 분석

대형 마트나 소매점에서 얼굴 분석을 통해 나이, 연령대, 성별 별 구매 내역을 분석하여 광고 및 마케팅에 활용할 수 있다.

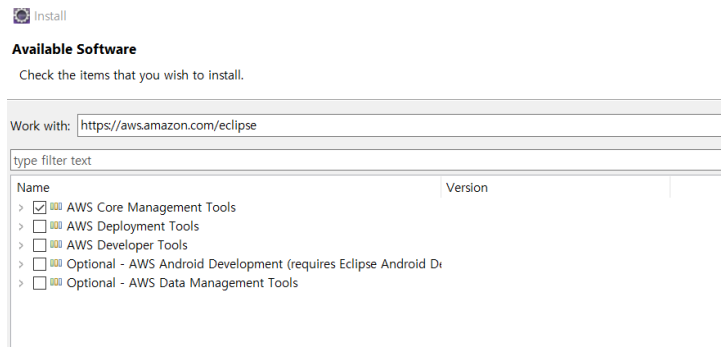
#### iv. 기타 활용 방안

이미지를 분석하고 이미지와 관련된 유명 인물의 키워드를 추출하여 검색해주는 검색 기능, 폭력적인 장면이나 잔인한 장면 등 유해한 콘텐츠를 인식하여 플래그를 지정해주는 기능 등 생체 인식 중 얼굴 인식 기술을 활용하여 다양한 서비스를 제공할 수 있다.

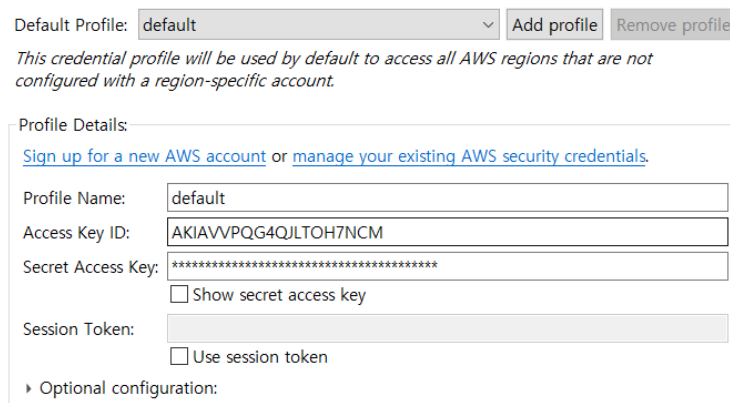
## B. 소프트웨어 설치, 튜토리얼 예제 실행 및 결과

### 1. 소프트웨어 설치

AWS를 사용하기 위해 전용 SDK를 설치하였다. Java Eclipse를 통해 개발할 예정이기 때문에 Eclipse 설치 및 실행 후 Install New Software 탭에서 aws management tool을 찾아 다운로드한다.



설치가 완료되면 Eclipse 내의 Preference 설정 창에서 다음과 같이 AWS계정 ID와 비밀번호를 입력해야 한다. 이 설정을 마치고 나면 Eclipse에서 AWS 소프트웨어를 사용할 수 있게 된다.



### 2. 예제 실행 및 결과

업로드한 이미지를 분석하여 한 개 이상의 얼굴 영역을 추출하고, 추출한 얼굴의 나이, 성별, 표정 등을 분석하여 출력해주는 얼굴 분석 예제를 실행하였다.

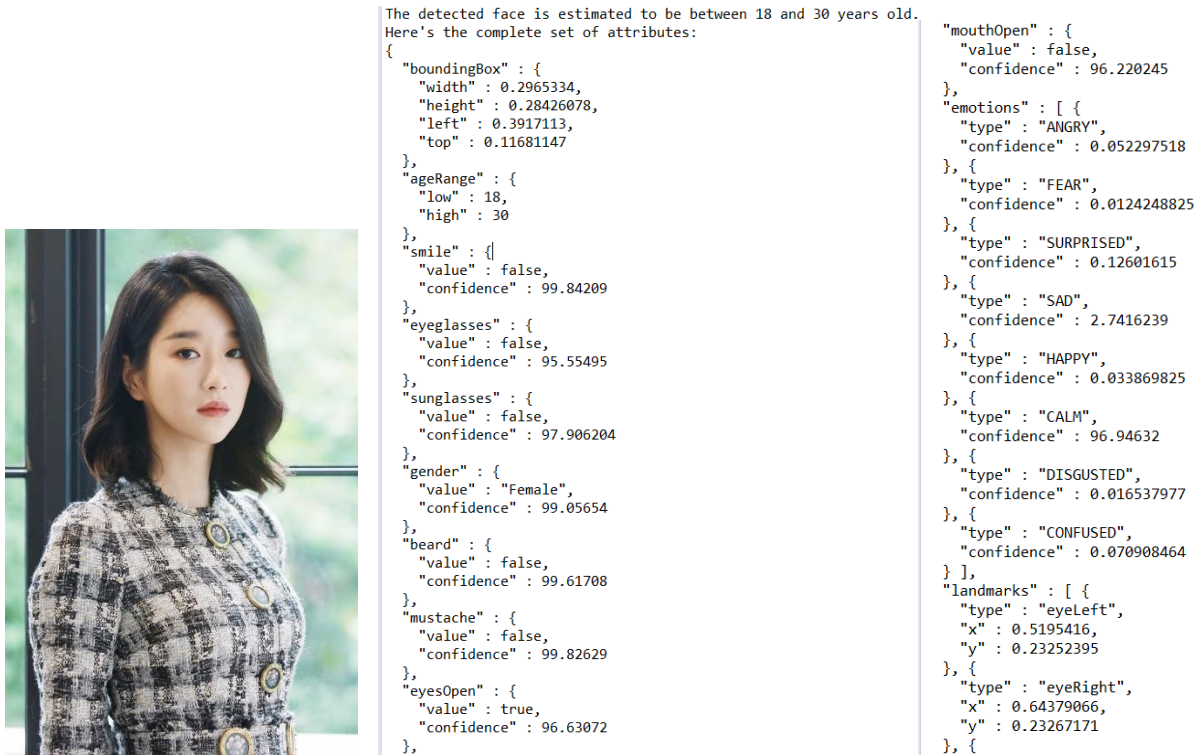
먼저 예제를 실행하기 위해 소스 이미지를 S3 클라우드에 업로드 하였다. Amazon S3 서비스에 접속하여 새로운 버킷을 만들어주고 (hyemin-rekognition-bucket) 분석할 이미지를 업로드한다.

Eclipse를 실행하여 새로운 AWS Java project를 만들고, amazon rekognition에서 제공하는 DetectFaces API를 사용하였다. 제공되는 예제 코드에 업로드한 이미지의 키값과 버킷 주소를 입력하여 이미지 분석을 수행해보았다.

사용한 소스코드:

[https://github.com/iamchm/AWS\\_Rekognition/blob/master/src/main/java/com/amazonaws/samples/DetectFaces.java](https://github.com/iamchm/AWS_Rekognition/blob/master/src/main/java/com/amazonaws/samples/DetectFaces.java)

얼굴 분석을 실행한 소스 이미지와 분석 결과는 다음과 같다.



▲ 소스 이미지 (source.jpg)

▲ 이미지 분석 결과

먼저 클라우드에서 가져온 이미지에서 얼굴 영역을 검출한다. 검출한 얼굴을 분석하여 예측한 연령대를 결과값으로 나타내고 그 외 원하는 세부 항목도 선택하여 출력이 가능하다. 세부 항목은 이미지에서 얼굴을 검출한 위치 좌표, 표정 감지, 안경 착용 여부, 성별, 헤어 스타일 등 다양한 정보를 신뢰도(confidence)와 함께 나타내도록 구성되어 있다. 실제로 수행해본 결과, 분석하기 어려울 것이라 생각했던 표정, 감정까지 비교적 정확히 분석 가능한 것을 확인하였다. 또한 얼굴의 주요 랜드마크(눈, 코, 입 위치 등)의 각 시작/끝 지점의 좌표 또한 분석이 가능하여 추후에 얼굴의 특정 요소만을 추출하는 등 여러 방향의 활용이 가능할 것으로 보인다.

### 3. 예제 실행 과정 중 발생한 문제와 해결 방법

i. Amazon Recognition을 사용하기 위해서는 다음과 같이 AWS 계정에서 AmazonRekognitionFullAccess, AmazonS3ReadOnlyAccess의 권한을 설정해주는 과정이 필요하다. 처음에 이 설정을 해주지 않고 진행하여 오류코드 400: AccessDeniedException 오류가 발생하였다. 다음과 같이 AWS 콘솔에 접속하여 계정 관리 탭의 권한 추가 버튼을 통해 이 권한들을 허용해주어 문제를 해결하였다.



▲ AWS 콘솔의 권한 추가 페이지

ii. 코드를 실행할 때 저장소가 아닌 클라우드에서 이미지를 가져오는 경우, 클라우드에 접속하여 업로드한 이미지를 퍼블릭으로 설정해주어야 이미지 불러오기가 가능하다. 이 설정을 하지 않고 실행하면 이미지 불러오기가 되지 않기 때문에 오류코드 400: InvalidS3ObjectException (Unable to get object metadata from S3. Check object key, region and/or access permissions.) 오류가 발생한다. 이미지 파일 속성에서 퍼블릭으로 설정해주어 문제를 해결하였다.



◀ 이 버튼을 클릭하여 외부에서 접근이 가능하도록 퍼블릭으로 설정해주어 문제를 해결하였다.

## C. 얼굴 인식 기술을 활용한 출입 관리 및 제한 구역 감시 시스템

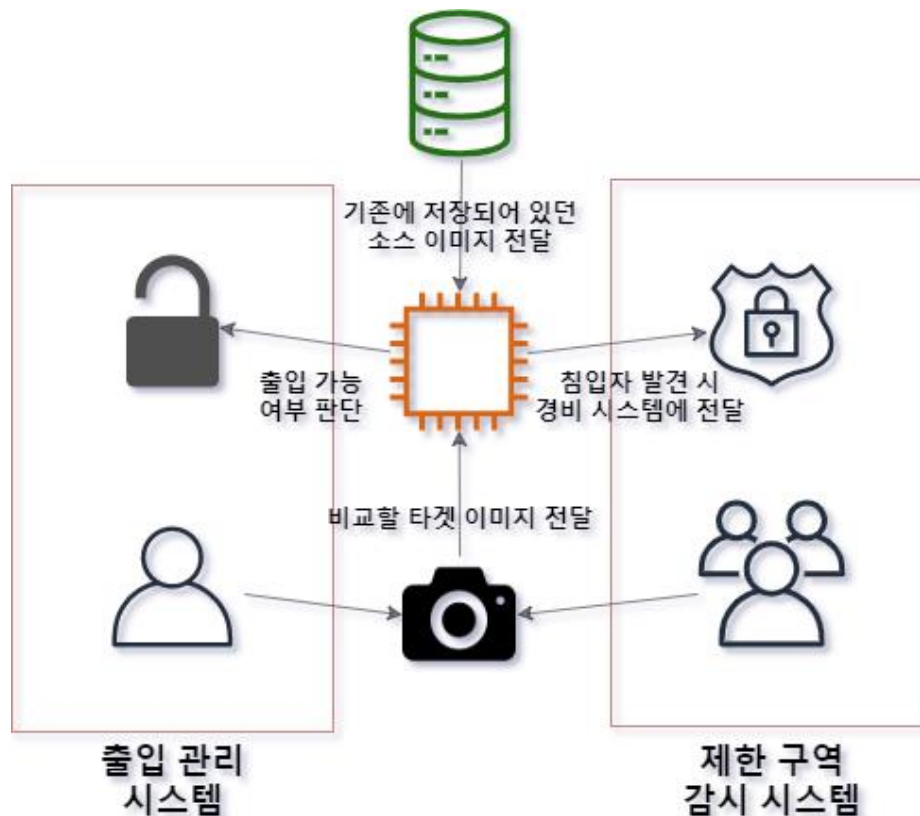
### 1. 프로젝트 설계

회원제 도서관과 같이 보안이 요구되는 시설은 현재 대부분 ID 카드를 이용하여 본인 인증 및 출입 허가가 이루어진다. 하지만 ID 카드는 분실이나 도난의 위험이 높아 강력한 보안을 제공하는 데에는 한계가 있다. 그래서 최근 각광받고 있는 생체 인식 기술을 활용하여 보안성을 높인 출입 시스템을 설계하기로 하였다. 문헌 조사에도 확인할 수 있었듯이 흔히 사용하는 지문 인식은 다양한 변수가 존재하여 오류가 발생할 가능성이 높다. 따라서 변화가 적고 인증 방법이 편리한 얼굴 인식 기술을 사용하여 본인 인증이 필요한 곳에 설치할 수 있는 무인 출입 관리 시스템을 설계하였다. 또한 같은 기술을 활용하여 특정 인물 외 출입이 제한된 구역을 실시간으로 감시하여 허가되지 않은 침입자를 찾아낼 수 있는 시스템을 설계하였다.

먼저 출입 관리 시스템은 출입구에 카메라를 설치하여 출입을 원하는 사람의 사진을 얻는다. 이 사진을 시스템에 전달하면 시스템에서 기존에 저장되어 있던 얼굴을 불러오게 되고 저장된 얼굴과 일치하는지 비교하여 출입 가능 여부를 결정한다.

제한 구역 감시 시스템의 경우 기존의 CCTV 영상을 일정 시간마다 시스템으로 전송한다. 시스템은 전송된 영상에 나타나는 모든 얼굴들을 검출한 후 기존에 저장되어 있던 사람이 아닌 다른 사람이 발견되면 경비 시스템에 이를 전달하여 조치를 취할 수 있게 한다.

구상한 시스템 구성도는 다음과 같다.



## 2. 프로젝트 구현

얼굴을 비교하는 프로그램은 AWS에서 제공되는 CompareFaces 코드를 수정하여 사용하였다. FileInputStream으로 이미지 파일을 불러오는 코드를 S3 클라우드 버킷의 이미지를 불러오도록 수정하였다. 이 코드를 실행하면 기존에 버킷에 저장되어 있던 소스 이미지와 새로 저장한 타겟 이미지의 얼굴을 각각 검출하여 유사도를 확률 값으로 나타내고 이 유사도가 70%를 넘을 경우 같은 얼굴로 간주하여 검출한 얼굴의 위치와 함께 결과를 출력한다.

사용한 소스코드:

[https://github.com/iamchm/AWS\\_Rekognition/blob/master/src/main/java/com/amazonaws/samples/CompareFaces.java](https://github.com/iamchm/AWS_Rekognition/blob/master/src/main/java/com/amazonaws/samples/CompareFaces.java)

\* 소스 이미지는 예제에서 사용한 source.jpg를 사용하였고 버킷 주소 역시 그대로 사용하였다.

첫 번째로 소스 이미지의 인물과 같은 인물의 다른 이미지 target1\_correct.png를 버킷에 저장하고 타겟으로 지정하여 분석을 실행하였다.



◀ target1\_correct.png

```
13 public class CompareFaces {
14
15     public static void main(String[] args) throws Exception{
16         Float similarityThreshold = 70F; //유사도 임계값
17         String sourceImage = "source.jpg";
18         //버킷에 저장된 비교 이미지 - target1_correct.png / target2_incorrect.jpg / target3_1correct_3incorrect.jpg
19         String targetImage = "target1_correct.png";
20         String bucket = "hyemin-rekognition-bucket";
21
22         AmazonRekognition rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();
23
24         //S3 버킷에서 이미지 불러오도록 코드 수정
25         CompareFacesRequest request = new CompareFacesRequest()
26             .withSourceImage(new Image()
27                 .withS3Object(new S3Object()
28                     .withName(sourceImage)
29                     .withBucket(bucket)))
30             .withTargetImage(new Image()
31                 .withS3Object(new S3Object()
```

Markers Properties Data Source Explorer Servers Snippets Console AWS Explorer

terminated - CompareFaces [Java Application] C:\Program Files\Java\jre1.8.0\_251\bin\javaw.exe (2020. 6. 25. 오후 5:58:31 - 오후 5:58:36)  
이미지의 (0.34892467, 0.11634355) 위치에 있는 얼굴이 찾으려는 얼굴과 99.54149% 일치합니다.  
찾으려는 얼굴과 일치하지 않는 얼굴이 0개 존재합니다.

실행 결과를 살펴보면 타겟 이미지의 얼굴이 소스 이미지의 얼굴과 약 99%의 높은 유사도를 보이며 이에 따라 같은 얼굴로 판단한 것을 알 수 있다. 소스 이미지와 비교했을 때 헤어 스타일에 변화가 있었음에도 불구하고 정확히 같은 얼굴로 인식하며 우수한 성능을 보여주었다.

다음으로, 소스 이미지의 인물과 다른 인물의 이미지를 버킷에 저장하여 분석을 실행하였다. 분석에 사용한 타겟 이미지 target2\_incorrect.jpg 와 실행 결과는 다음과 같다.



◀ target2\_incorrect.jpg

```
13 public class CompareFaces {
14
15     public static void main(String[] args) throws Exception{
16         Float similarityThreshold = 70F; //유사도 임계값
17         String sourceImage = "source.jpg";
18         //버킷에 저장된 비교 이미지 - target1_correct.png / target2_incorrect.jpg / target3_1correct_3incorrect.jpg
19         String targetImage = "target2_incorrect.jpg";
20         String bucket = "hyemin-rekognition-bucket";
21
22         AmazonRekognition rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();
23
24         //S3 버킷에서 이미지 불러오도록 코드 수정
25         CompareFacesRequest request = new CompareFacesRequest()
26             .withSourceImage(new Image()
27                 .withS3Object(new S3Object()
28                     .withName(sourceImage)
29                     .withBucket(bucket)))
30             .withTargetImage(new Image()
31                 .withS3Object(new S3Object()
32                     .withName(targetImage)
33                     .withBucket(bucket)))
34             .withSimilarityThreshold(similarityThreshold);
35
36         CompareFacesResult result = rekognitionClient.compareFaces(request);
37
38         System.out.println("Source Image Name: " + sourceImage);
39         System.out.println("Target Image Name: " + targetImage);
40         System.out.println("Similarity: " + result.getSimilarity());
41
42         if (result.getSimilarity() > similarityThreshold) {
43             System.out.println("The two images are of the same person.");
44         } else {
45             System.out.println("The two images are of different people.");
46         }
47     }
48 }
```

terminated - CompareFaces [Java Application] C:\Program Files\Java\jre1.8.0\_251\bin\javaw.exe (2020. 6. 25. 오후 5:59:27 - 오후 5:59:32)

찾으려는 얼굴과 일치하지 않는 얼굴이 1개 존재합니다.

소스 이미지에서 검출한 얼굴과 타겟 이미지에서 검출한 얼굴의 유사도가 70% 이하인 경우 일치하지 않는 얼굴로 판단하고 그 개수를 결과값으로 나타낸다. target2의 인물은 소스 이미지의 인물과 비슷한 헤어 스타일, 같은 성별을 가진 인물이지만 눈, 코, 입의 위치와 얼굴 골격 등의 차이를 정확하게 구분하여 다른 사람으로 인식하는 것을 확인할 수 있다.



마지막으로, 타겟 이미지에서 소스 이미지의 인물과 다른 여러 인물들의 얼굴이 함께 나타나는 경우를 실험해 보았다. 실험에 사용한 이미지 target3\_1correct\_3incorrect.jpg 는 다음과 같다.



◀ target3\_1correct\_3incorrect.jpg  
(왼쪽에서 두 번째 인물이 찾으려는 소스 이미지의 인물과 동일 인물이다.)

```
13 public class CompareFaces {
14
15     public static void main(String[] args) throws Exception{
16         Float similarityThreshold = 70F; //유사도 임계값
17         String sourceImage = "source.jpg";
18         //버킷에 저장된 비교 이미지 - target1 correct.png / target2 incorrect.jpg / target3_1correct_3incorrect.jpg
19         String targetImage = "target3_1correct_3incorrect.jpg";
20         String bucket = "hyemin-rekognition-bucket";
21
22         AmazonRekognition rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();
23
24         //S3 버킷에서 이미지 불러오도록 코드 수정
25         CompareFacesRequest request = new CompareFacesRequest()
26             .withSourceImage(new Image()
27                 .withS3Object(new S3Object()
28                     .withName(sourceImage)
29                     .withBucket(bucket)))
30             .withTargetImage(new Image()
31                 .withS3Object(new S3Object()
32                     .withName(targetImage)
33                     .withBucket(bucket)))
```

<terminated> CompareFaces [Java Application] C:\Program Files\Java\jre-8.0.251\bin\javaw.exe (2020. 6. 25. 오후 6:00:05 - 오후 6:00:11)

이미지의 (0.38352117, 0.27324948) 위치에 있는 얼굴이 찾으려는 얼굴과 98.51817% 일치합니다.  
찾으려는 얼굴과 일치하지 않는 얼굴이 3개 존재합니다.

실행 결과, 소스 이미지의 인물은 약 98%의 유사도가 나타났기 때문에 일치하는 얼굴로 판단하여 그 위치를 반환하였다. 그 외 다른 3명의 얼굴은 일치하지 않는 얼굴로 인식하고 그 숫자를 정확히 나타내는 것을 볼 수 있다.



### 3. 결과 분석 및 발전 방향

얼굴 인식 기술을 활용하여 출입 관리, 제한 구역 감시 시스템을 구축하기 위해 얼굴 비교 프로그램을 실행해보았다. 저장된 얼굴과 같은 얼굴은 헤어 스타일, 표정 변화와 관계 없이 높은 유사도를 보이는 것을 확인하였다. 또, 저장된 얼굴과 다른 얼굴은 닮은 얼굴이라 할지라도 얼굴의 특징점을 잘 비교하여 일치하지 않음을 판별하는 것을 확인하였다. 여러 명이 함께 찍힌 사진에서도 각각의 얼굴을 정확히 구분하여 검출해내었으며, 저장된 얼굴과 같은 얼굴, 다른 얼굴을 구분하여 원하는 결과를 보여주었다.

실험을 통해 얼굴 비교 프로그램이 매우 잘 작동하여 본인 인증에 충분히 활용할 수 있다는 것을 알게 되었다. 이에 따라 설계한 시스템을 실제로 구축할 수 있겠다는 가능성을 확인하였다. 이 시스템이 개발된다면 더욱 빠르고 편리한 본인 인증이 가능할 뿐만 아니라 보안 시스템에 필요한 인력을 줄일 수 있어 효율적인 인력 활용을 도울 수 있을 것이다.

다만, 클라우드에 저장하는 소스 이미지가 많아질 경우 얼굴을 하나씩 비교하게 되면 많은 시간이 걸릴 수 있는 단점이 있다. 저장된 얼굴을 성별과 연령대로 나누어 해당하는 이미지들만 비교하는 방법과 같이 단점을 보완하는 방안을 고민해봐야 할 것이다.

또한 보안과 관련된 시스템이 아니더라도, 회원제 헬스장이나 회원 혜택을 제공하는 상점들에서 얼굴 비교 프로그램을 사용할 수 있을 것이다. 직원이 따로 회원을 확인하고 필요한 서비스를 요청 받아 처리하는 대신, 스마트 미러 혹은 키오스크에 연결된 카메라로 회원을 인식하고 저장된 정보에 따라 서비스를 회원이 바로 선택할 수 있도록 하는 서비스를 도입한다면 더욱 만족스러운 고객 경험을 제공할 수 있을 것이다.

이와 같이 얼굴 인식 기술을 활용하여 구축할 수 있는 다양한 편의 시스템을 통해 편리한 일상 생활을 누릴 수 있기를 기대한다.

### Ⅲ. 참고 문헌

- [1] 사용자 인증을 위한 딥러닝 기반 얼굴인식 기술 동향 [문형진, 김계희. 산업융합연구 Vol.17 No.3. 2019. 23-29 쪽]
- [2] 얼굴 인식 및 인증 알고리즘 성능분석 [김성구. 학위논문(석사)-고려대학교 공학대학원: 전기전자컴퓨터공학과, 2018]
- [3] 일반 카메라 영상에서의 얼굴 인식을 향상 위한 얼굴 특징 영역 추출 방법 [김성훈, 한기태. 정보처리학회논문지. 소프트웨어 및 데이터 공학. Vol.5 No.5. 2016. 251-260쪽]
- [4] 딥러닝을 이용한 얼굴인식 성능 비교 [박민정. 학위논문(석사)-경성대학교 일반대학원 전기전자공학과, 2018.2]
- [5] 적외선/깊이 영상을 이용한 강인한 얼굴인식 방법 [김흥준. 학위논문(석사)-동의대학교 일반대학원: 컴퓨터 소프트웨어공학과, 2018]
- [6] 얼굴 인식을 위한 효율적인 신경망에 관한 연구 [조성만. 학위논문(석사)-서울과학기술대학교 대학원: 미디어IT공학과, 2020]
- [7] 딥러닝 기반 얼굴인식 라이브러리 활용 및 보완에 관한 연구 [유경화. 학위논문(석사)-남서울대학교 복지경영대학원: 빅데이터전문가학과, 2018]
- [8] 다중 생체 인식 기법을 이용한 개인 인증 기술에 관한 연구 [장봉환. 학위논문(박사)-단국대학교 대학원: 전자전기공학과 제어 및 신호처리 전공, 2019]
- [9] 얼굴 인식 Open API를 활용한 출입자 인식 시스템 개발 [옥기수, 권동우, 김현우, 안동혁, 주홍택. 정보처리학회논문지. 컴퓨터 및 통신시스템 Vol.6 No.4. 2017. 169-178쪽]