Face ID

202104033 제갈륜

202104022 신성우

201904020 윤건용

목차

1. 프로그램 필요성

- 얼굴은 개인을 고유하게 식별하는 특징을 가지고 있고 이러한 특성을 통해 새로운 인증 방식이나 강력한 보안 수단으로 활용하고자 이 프로그램을 개발하게 되었다.

- 현재 개발된 얼굴인식은 데이터에 편향이 존재할 경우, 특정 인종, 성별, 연령 등에 대해 인식 오류가 발생할 수 있다. 이를 해결하고자 얼굴 인식 방식에 있어 얼굴에 68개의 점을 찍고 눈, 코, 입, 눈썹 등의 위치에 따라 각 점들의 거리를 구하여 얼굴을 인식하게 된다. 이는 머리길이나 피부색 등을 구별하지 않고 얼굴의 특징을 통해 비교하는 방식으로 편향성 문제에 대한 해결방안을 제시한다.

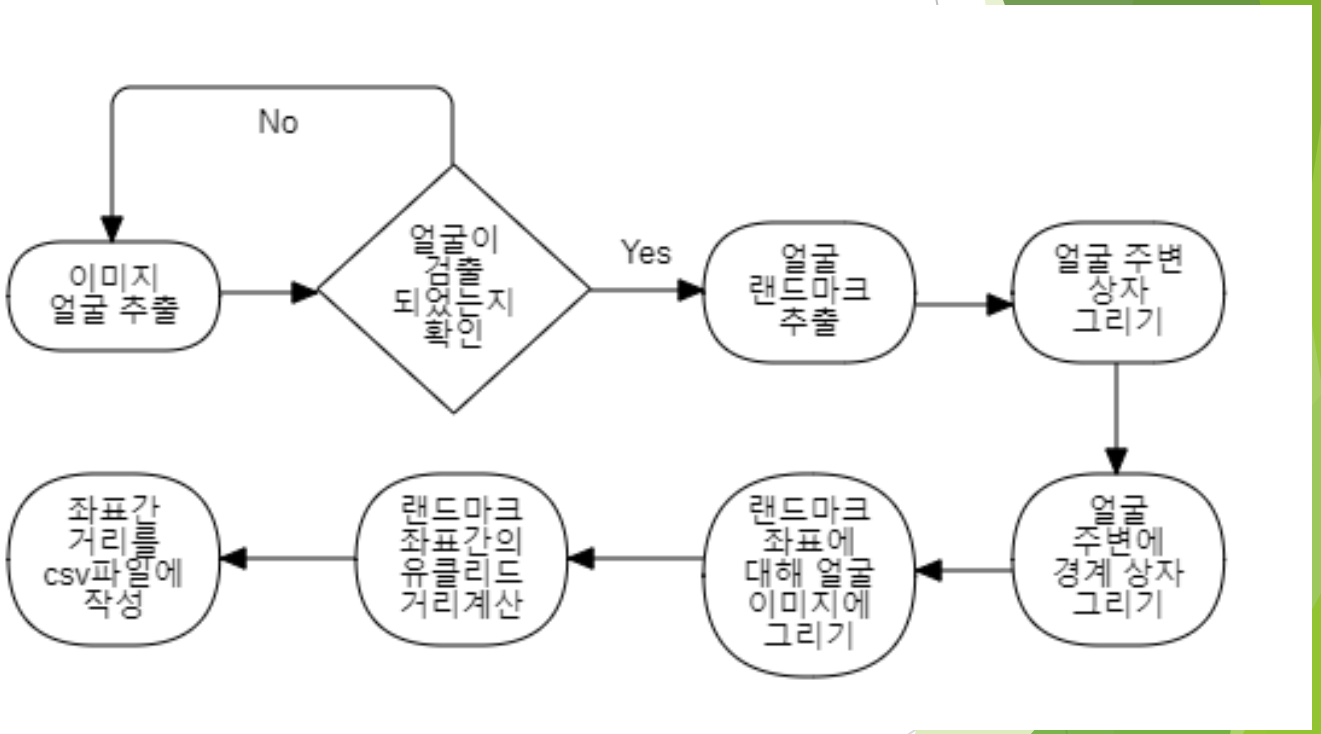
2. 프로그램 소개

- 제목: FaceID
- 목적: 사람 얼굴을 인식하여 얼굴에 68개의 점을 찍고 눈, 코, 입, 눈썹 등의 위치에 따라 각 점들의 거리를 추출하여 학습하고 사람을 판별한다.
- 요약: 한 사람의 다양한 각도로 찍은 사진에서 각각의 점들의 거리를 구하여 그 값을 통해 다른 사진과 비교하여 일치한지 불일치한지 판단한다.
- 기대효과: 비밀번호나 카드키와 같은 인증수단과 달리 얼굴인식은 개인의 얼굴 특징을 기반으로 인증을 하기 때문에 상대적으로 안전한 인증 방식이다. 또한 비밀번호와 패턴 입력과 같은 번거로운 인증을 없애고 사용자의 편의성을 높여준다.

3. 데이터 분석

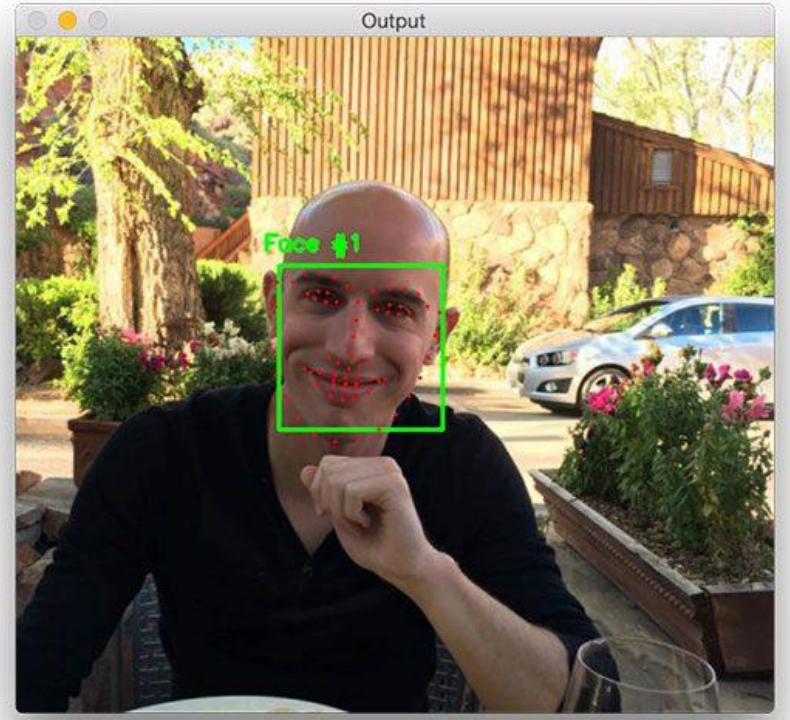
- 이러한 순서로 각각의 이미지에 대한 landmark_distances.csv 파일 생성

- 얼굴영역 잘라내어 얼굴 위치를 같게 한다.



3. 데이터 분석

- 랜드마크 점 예시



3. 데이터 분석

인덱스	이름	데이터 타입	값	설명
0	landmark	연속형	정수형	Landmark의 번호
1	1	연속형	정수형	Landmark 1번과의 거리
n	1	연속형	정수형	Landmark n번과의 거리

Landmark	1	2	3	4	5	6	7	8	...
1	0	32	64.19502	96.74709	128.7245	160.854	188.5099	211.4048	...
2	32	0	32.38827	65.11528	97.29337	129.8075	158.2403	182.4829	...
3	64.19502	32.38827	0	32.75668	65	97.71898	126.7162	152.1217	...
4	96.74709	65.11528	32.75668	0	32.28002	65.19202	94.82616	121.6059	...
5	128.7245	97.29337	65	32.28002	0	33.10589	63.51378	91.96739	...
6	160.854	129.8075	97.71898	65.19202	33.10589	0	31.14482	61.40033	...
...	...	...	...	...	...	...	...	...	...

4. 사용 알고리즘

선현회귀 모델을 통해 학습시키는 random state값을 변화시켜
얼굴에 맞는 최적의 random state값을 찾는다.

가장 최적의 random state값을 가지고 mse, mae, R-squared 값을 구한다.

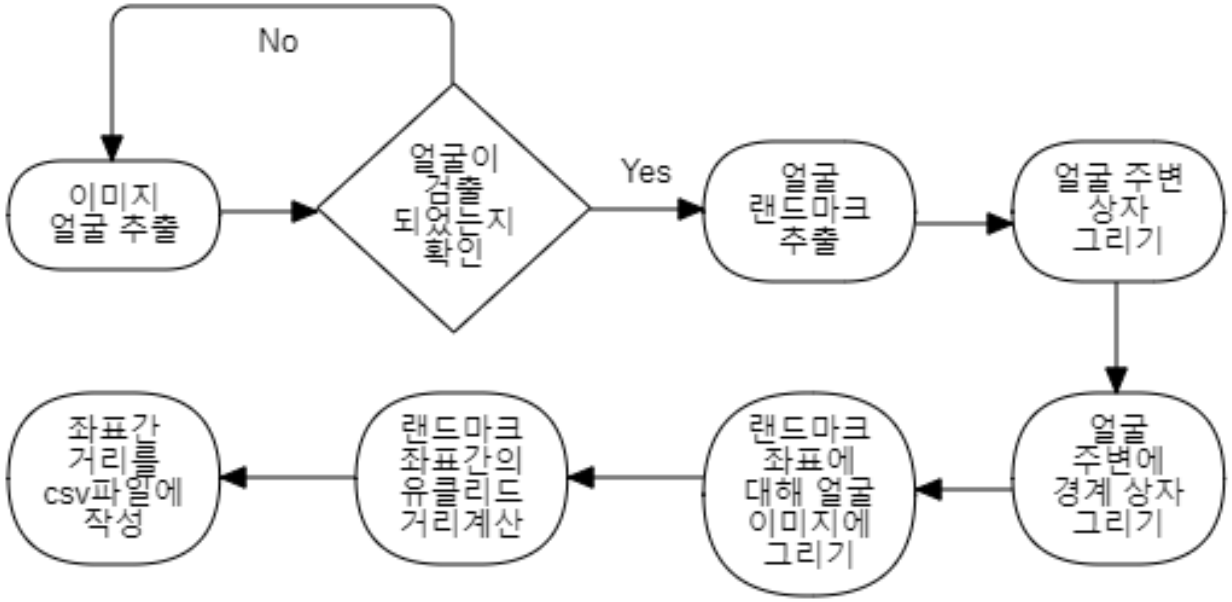
Radom state값을 가지고 판별할 데이터를 학습시켜
성능평가 값을 구하고 이 성능평가 값을 가지고 판별한다.

5. 순서도 및 구현

- 데이터 추출순서도(CSV 만들기)

- 이미지를 가져와 얼굴영역을 구분하고

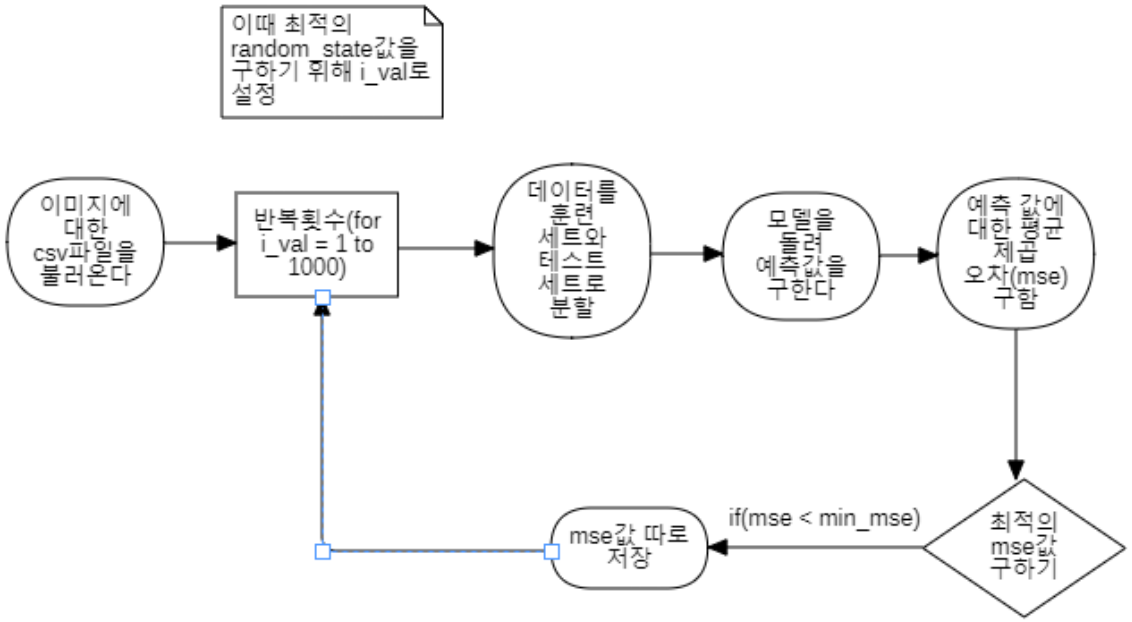
얼굴에 점을 찍어 유클리드 거리 공식을 이용하여 좌표간 거리를 구하여 데이터를 추출하고 csv파일을 만들고 저장한다.



5. 순서도 및 구현

- 최적의 random_state 구하기

- 선형회귀 모델을 이용해 데이터의 예측값을 구한다.
- 데이터를 예측값을 구하며 최적을 mse, mae, R-squared값을 구한다.



5. 순서도 및 구현

- 구현

```
for i in range(image_start, image_end+1):
    image_path = f"IMAGE/{i}.jpg"


    if os.path.exists(image_path):
        image = cv2.imread(image_path)
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        print(f"Error: No face detected in the image face_{i}.")
        continue

    faces = detector(gray)

    for j, face in enumerate(faces):
        landmarks = predictor(gray, face)
        landmarks = np.array([(landmarks.part(i).x, landmarks.part(i).y) for i in range(68)])

        x, y, w, h = face.left(), face.top(), face.width(), face.height()
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

        for (x, y) in landmarks:
            cv2.circle(image, (x, y), 1, (0, 0, 255), -1)

        distances = pdist(landmarks)
        distance_matrix = squareform(distances)

        with open(f"landmark_distances/landmark_distances({i}).csv", "w", newline="") as csvfile2:
            writer2 = csv.writer(csvfile2)
            header = ["Landmark"] + [str(i) for i in range(1, 69)]
            writer2.writerow(header)
            writer2.writerows((i+1, *[f"{d:.8f}" for d in row]) for i, row in enumerate(distance_matrix))
```

5. 순서도 및 구현

- 구현

```
file_numbers = range(image_start, image_end+1)
i_values = range(1, 100)
for i_val in i_values:
    for file_number in range(image_start, image_end + 1):
        distances_file = f"landmark_distances/landmark_distances({file_number}).csv"

        if not os.path.exists(distances_file):
            continue
        data = pd.read_csv(distances_file)

        # 특징과 라벨 분리
        X = data.drop(columns=["Landmark"])
        y = data["Landmark"]

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=i_val)

        # 선형 회귀 모델 생성 및 훈련
        model = LinearRegression()
        model.fit(X_train, y_train)

        # 검증 데이터를 사용하여 성능평가: 예측
        test_predictions = model.predict(X_test)

        # R-squared 계산
        r2 = r2_score(y_test, test_predictions)
        # print(f"R-squared: {r2:.4f}")

        # 평균 제곱 오차(MSE) 계산
        mse = mean_squared_error(y_test, test_predictions)
        # print(f"Mean Squared Error: {mse:.4f}")

        # 평균 절대 오차(MAE) 계산
        mae = mean_absolute_error(y_test, test_predictions)
        # print(f"Mean Absolute Error: {mae:.4f}")

        if mse < min_mse and mae < min_mae and r2 > min_r2:
            min_mse = mse
            min_mae = mae
            min_r2 = r2
            min_i = i_val
            min_file_number = file_number
```

```
# 결과 출력
print(f"가장 작은 MSE: {min_mse}")
print(f"가장 작은 MAE: {min_mae}")
print(f"가장 큰 R-squared: {max_r2}")
print(f"최소 MSE에 해당하는 i 값: {min_i}")
print(f"가장 작은 MSE에 해당하는 파일 번호: {min_file_number}")
```

✓ 0.0s

```
가장 작은 MSE: 0.04259587124337109
가장 작은 MAE: 0.17443148525575122
가장 큰 R-squared: 0.9999205603375609
최소 MSE에 해당하는 i 값: 75
가장 작은 MSE에 해당하는 파일 번호: 241
```

6. 결론

- 1. 문제점 - 이미지의 각도에 따라 값이 달라져 판별이 불가능하다.
- 해결방법 - 다양한 각도의 얼굴을 판별할 수 있는 코드를 추가해서 얼굴 각도에 따른 계산을 달리할 수 있도록 만들 수 있다.
- 2. 어려웠던 점 - Cnn 모듈을 사용하지 않고 다른 방법으로 이미지를 구별해낼지 막막했다.
- 극복방법 - 많은 논문과 조언을 참고하여 극복하였다.
- 3. 보충할 점 - 실시간 영상처리를 통해 다양한 각도의 얼굴을 판별할 수 있게 한다.

참고문헌

- Juhong, Aniwat & Pintavirooj, C. (2017). Face Recognition Based on Facial Landmark Detection.10.1109/BMEiCON.2017.8229173. https://www.researchgate.net/publication/321579006_Face_Recognition_Based_on_Facial_Landmark_Detection

- Medvedev, Iurii & Shadmand, Farhad & Cruz, Leandro & Gonçalves, Nuno. (2021). Towards Facial Biometrics for ID Document Validation in Mobile Devices. Applied Sciences. 11. 10.3390/app11136134. https://www.researchgate.net/publication/363367177_Towards_Facial_Biometrics_for_ID_Document_Validation_in_Mobile_Devices₩

- "Facial landmarks with dlib, OpenCV, and Python", pyimagesearch, April 3, 2017, accessed July 3, 2021, https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

- "NicoTrombon/acial-landmarks", Github, Apr 19, 2017, https://github.com/nicoTrombon/facial-landmarks.git