안면 인식 보안 장치

이지훈, 김도현
국민대학교 전자공학부
andy5263@kookmin.ac.kr, dohyungo100@kookmin.ac.kr

요 약

안면 인식에 대해서 배우고 이를 활용하여 안면 인식 보안 장치를 만들었습니다. Haar알고리즘으로 학습시킨 것을 활용하여 얼굴을 인식했고 얼굴의 눈, 코, 입, 턱과 같은 특징점을 잡아서 유사도를 확인하고 이를 보안 장치에 활용하였습니다. 이를 활용하여 70~80퍼센트의 정확도를 가진 보안 장치를 만들 수 있었고 성별이나 가족관계에서 발생하는 정확도 차이의 특징을 확인할 수도 있었습니다.

1. 서론

안면 인식에 대해 배우고, 활용할 수 있는 분야 중에서 보안 장치에 활용하는 것이 좋을 것 같다고 생각했습니다. 보안 장치의 경우 비밀번호를 입력해야 한다는 특성이 물리적으로 노출될 수 있기 때문에 안면 인식을 보안 장치에 사용하면 효과적인 보안 장치를 만들 수 있을 것입니다. 또한 수업시간에 실습했던 코드들을 활용하여 진행할 수 있을 것이라고 생각했습니다.

노트북에 내장되어 있는 웹캠을 활용하기 위해 VS코드 환경에서 진행하였고 수업시간에 배웠던 라이브러리와 함수들 사용하여 진행하다 실행속도와 더 많은 기능을 사용하기 위해 다양한 라이브러리를 활용하였습니다.

2. 수행 내용

저희 조의 안면 인식 보안 장치가 어떻게 동작하는지 설명하고, 동작시키기 위해 사용된 기능들에 대해서 설명하겠습니다. 코드에 대해서도 설명하고, 동작했을 때의 결과를 첨부하겠습니다.

2.1 안면 인식 보안 장치의 동작 과정1

안면 인식 보안 장치의 동작과정에 대해 설명하겠습니다. 우선 출입시키고자 하는 사람에 대한 사진을 저장해 놓습니다. 한 사람당 20개의 사진을 저장합니다.

이 후에 프로그램을 실행시키면 웹캠화면이 등장합니다. 화면에는 저장되어 있는 이미지와 비슷한 얼굴은 인식 한 경우 초록색 사각형과 함께 그 이름과 문이 열렸다는 표시를 하게 됩니다. 만약에 저장된 사진과 일치하는 사진이 없다면 빨간색 사각형이 그려지게 됩니다. 이렇게 일치하는 사진이 없었다면 그 사진을 저장합니다. 그리고 이후에 그 사람이 다시 나타났을 경우 이전에 방문했던 사람임을 알려줍니다.

일치하지 않는 사람을 출입시키고 싶으면 이전에 저장한 이미지를 출입할 수 있는 폴더로 불러옵니다.

2.2 동작시키기 위해 사용한 기능

Haar 알고리즘으로 학습된 파일을 활용하여 동작시킵니다. known 폴더와 unknown 폴더에 저장된 이미지를 불러와 배열에 저장합니다. 이후에 웹캠을 열고 화면에 출력시킵니다. 웹캠을 동작시키면서 얼굴을 인식하는 동작을 진행합니다.

위에서 불러왔던 Haar 알고리즘으로 학습된 파일을 활용하여 얼굴을 인식합니다. 얼굴을 인식한 후에는 known 폴더에 있던 이미지를 가져와 유사도를 확인합니다. 일정 유사도 이상이 확인 된 경우 중지합니다. 그리고 그 이미지의 이름을 불러와 얼굴에 초록색 사각형을 표시하고 일치했던 이미지의 이름과 유사도(%)를 출력합니다.

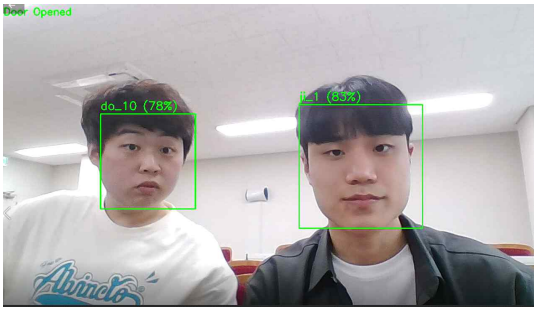

그림 1. 보안 장치가 해제된 경우

만약 일치하지 않은 경우 unknown 폴더에 있는 이미지와 비교합니다. unknown 폴더에는 이전에 일치하지 않았던 이미지를 저장해 두었던 폴더입니다. 여기서 유사도가 일정 이상이면 위와 같이 얼굴에 정보를 출력합니다. 이 때 보안 장치가 해제될 때 표시되는 초록색이 아닌 파란색으로 표시되게 됩니다.

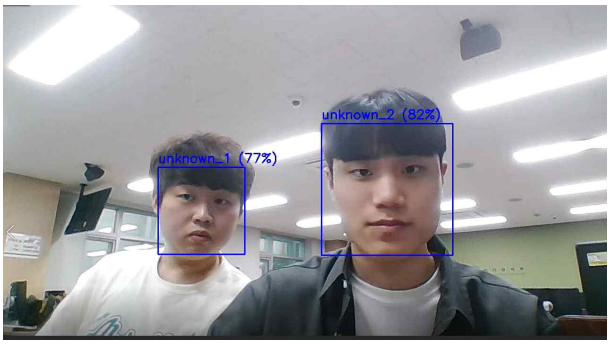

그림 2. 이전에 방문했던 사람인 경우

마지막으로 두 과정에서 모두 일치하지 않았던 경우에는 얼굴에 붉은 색 사각형으로 표시한 후에 unknown 폴더에 이미지를 저장합니다. 바로 최신화되기 때문에 이후에 얼굴에 unknown_i 번의 이름으로 표시되는 것을 확인할 수 있습니다.

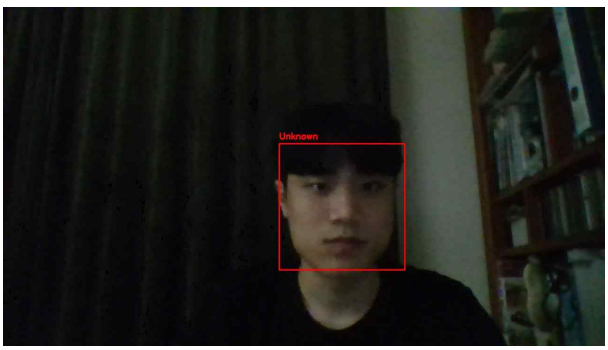

그림 3. 처음 방문했던 사람인 경우

2.3 작성한 코드 설명

작성한 코드에 대해서 설명하겠습니다.

```python
import cv2
import os
import face_recognition

known_dir = 'known'
unknown_dir = 'unknown'
threshold = 70  # 임계값

# known 디렉토리에서 알려진 얼굴 이미지를 불러옵니다.
known_faces = []
known_names = []
for file_name in os.listdir(known_dir):
    if file_name.endswith('.jpg') or file_name.endswith('.jpeg') or file_name.endswith('.png'):
        image = face_recognition.load_image_file(os.path.join(known_dir, file_name))
        face_encoding = face_recognition.face_encodings(image)
        if len(face_encoding) > 0:
            known_faces.append(face_encoding[0])
            name = os.path.splitext(file_name)[0]  # 파일 이름에서 확장자를 제거하여 이름으로 사용합니다.
            known_names.append(name)

# unknown 디렉토리에서 알려진 얼굴 이미지를 불러옵니다.
unknown_faces = []
unknown_names = []
for file_name in os.listdir(unknown_dir):
    if file_name.endswith('.jpg') or file_name.endswith('.jpeg') or file_name.endswith('.png'):
        image = face_recognition.load_image_file(os.path.join(unknown_dir, file_name))
        face_encoding = face_recognition.face_encodings(image)
        if len(face_encoding) > 0:
            unknown_faces.append(face_encoding[0])
            name = os.path.splitext(file_name)[0]  # 파일 이름에서 확장자를 제거하여 이름으로 사용합니다.
            unknown_names.append(name)
```

그림 4. 코드 1

다음 코드는 안면 인식 보안 장치를 동작시키기 전에 저장되어있는 이미지를 불러오는 단계입니다. known에는 보안 장치가 해제되어야 하는 사람의 이미지가 미리 저장되어 있고 이를 불러옵니다. unknown에는 이전에 저장되지 않았던 사진들이 있고 이를 불러옵니다. 이후에 아래 코드를 실행시키면서 이 배열 안에 새로운 사람이 등장하였을 때 저장합니다.

```python
video_capture = cv2.VideoCapture(0)

while True:
    ret, frame = video_capture.read()
    if not ret:
        break

    # 프레임에서 얼굴 인식
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)
```

그림 5. 코드 2

이 코드는 웹캠을 동작시킨 후의 반복문 첫번째 부분입니다. 반복문의 처음에는 얼굴을 인식하는 단계입니다.

```python
for face_encoding, face_location in zip(face_encodings, face_locations):
    # known_faces와 비교하여 유사도 계산
    matches = face_recognition.face_distance(known_faces, face_encoding)
    min_distance = min(matches)
    min_distance_percent = (1 - min_distance) * 100
```

그림 6. 코드 3

이 부분은 얼굴을 인식한 이후에 known 에 저장된 값과 비교하며 유사도를 계산하는 부분입니다. 이곳에서 보안 장치를 해제하기 위해 저장된 이미

지와의 유사도를 얻게 됩니다.

```
if min_distance_percent > threshold:
    matched_index = matches.argmin()
    name = known_names[matched_index]
    (top, right, bottom, left) = face_location

    # 얼굴 부분에 사각형 표시 (초록색)
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
    # 이름과 유사도 표시 (초록색)
    text = f"{name[:2]} ({min_distance_percent:.2f}%)"
    cv2.putText(frame, text, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
    # Door Opened 표시 (초록색)
    cv2.putText(frame, "Door Opened", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0), 2)
```

그림 7. 코드 4

이 코드는 위에서 얻은 유사도를 이전에 설정해 놓았던 70% 보다 높은 경우에서 실행하는 코드입니다. 이 조건에서 보안 장치를 해제됩니다. 초록색을 활용하고 얼굴에 사각형, 이름과 유사도, Door Opened를 표시합니다.

```
else:
    (top, right, bottom, left) = face_location

    matches = face_recognition.face_distance(unknown_faces, face_encoding)
    min_distance = min(matches)
    min_distance_percent = (1 - min_distance) * 100

    if min_distance_percent > threshold:
        matched_index = matches.argmin()
        name = unknown_names[matched_index]

        # 얼굴 부분에 사각형 표시 (파란색)
        cv2.rectangle(frame, (left, top), (right, bottom), (255, 0, 0), 2)
        # 이름과 유사도 표시 (파란색)
        text = f"{name} ({min_distance_percent:.2f}%)"
        cv2.putText(frame, text, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2)
```

그림 8. 코드 5

이 부분은 위에서 보안 장치가 해제되는 조건을 만족하지 못한 경우입니다. 이 경우 unknown에 있는 이미지와 비교를 하여 이전에 방문한 적이 있는 사람인지 확인 할 수 있습니다. 유사도가 일정 이상인 경우 몇 번째에 방문했던 사람인지 파란색을 활용하여 얼굴에 사각형, 이름과 유사도를 보여줍니다.

```
else:
    # 얼굴 부분에 사각형 표시 (빨간색)
    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
    # Unknown 표시 (빨간색)
    cv2.putText(frame, "Unknown", (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

    # unknown 이미지 저장
    unknown_image = frame[top:bottom, left:right]
    unknown_image_path = os.path.join(unknown_dir, f"unknown_{len(unknown_faces)}.jpg")
    cv2.imwrite(unknown_image_path, unknown_image)

    # 얼굴 이미지 저장
    face_image_path = os.path.join(unknown_dir, f"face_{len(unknown_faces)}.jpg")
    cv2.imwrite(face_image_path, frame)

    # unknown_faces와 unknown_names에 추가
    unknown_faces.append(face_encoding)
    unknown_names.append(f"unknown_{len(unknown_faces)}")
```

그림 9. 코드 6

이 코드는 위의 두 조건을 모두 만족하지 못한 경우로, 보안 장치를 해제할 수 있는 사람도 이전에 방문한 적이 있는 사람도 아닌 경우입니다. 이 경우에는 얼굴에 빨간 사각형을 표시하고 이미지를 저장합니다. 저장한 이미지를 위에서 설정했던 unknown 디렉토리에도 저장하여 바로 언제 방문했던 사람인지 표시할 수 있도록 합니다.

```
    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
```

그림 10. 코드 7

마지막 코드로 키보드의 q를 누르는 것으로 프로그램을 종료시키고 웹캠을 닫을 수 있습니다.

3. 실험 결과 및 분석

실험을 진행하면서 평균적으로 70 퍼센트 정도의 유사도로 설정하면 자신의 이미지를 활용하여 보안 장치를 해제시킬 수 있었습니다. 이미지의 표본이 많지 않았기 때문에 더 자세하고 많은 정보를 얻기는 힘들었습니다.

가족과 주변 친구들을 활용하여 프로그램을 실행해 보았습니다. 성별이 달라진 경우에는 동성인 경우보다 10~20퍼센트 정도 유사도가 낮아지는 경우도 확인할 수 있었습니다. 그리고 가족의 경우에는 가족이 아닌 경우보다 유사도가 5~10퍼센트 정도 높게 나오는 것도 확인할 수 있었습니다.

실험을 진행하면서 실행속도를 높이면 정확도가 감도하고 정확도를 높이면 실행속도가 낮아져 웹캠 영상에서 딜레이가 발생하였습니다. 어떤 방법으로 할지 고민하던 중 보안 장치인 만큼 정확도가 더 중요하다고 생각하여 정확도를 우선하는 방법으로 프로그램을 만들었습니다.

4. 결론

Haar 알고리즘으로 학습시킨 것을 활용하여 얼굴을 인식했고 얼굴의 눈, 코, 입, 턱과 같은 특징점을 잡아서 유사도를 확인하고 이를 보안 장치에 활용하였습니다. 이를 활용하여 70~80퍼센트의 정확도를 가진 보안 장치를 만들 수 있었고 성별이나 가족관계에서 발생하는 정확도 차이의 특징을 확인할 수도 있었습니다. 그리고 방문했던 사람을 저장해 놓기 때문에 자주 방문하던 사람을 출입시키고자 할 때 추가하는 것이 쉬워지고 언제 방문했는지 확

인 할 수도 있었습니다.

참고문헌

[1] ChatGPT (코드를 작성할 때는 ChatGPT의 도움을 많이 받았습니다.)
[2] http://dlib.net/ (dlib 사이트, 최초에 dlib를 설치하고 이를 활용하여 진행하였을 때 설치와 코드를 작성하는데 사용하였습니다.)
[3] https://github.com/ageitgey/face_recognition/blob/master/README_Korean.md (face_recognition 라이브러리가 어떻게 동작하는지 확인하기 위해 활용하였습니다.)
[4] https://viso.ai/computer-vision/deepface/ (Deepface 라이브러리를 사용하려고 했을 때 활용한 사이트입니다.)
[5] https://matplotlib.org/ (Matplotlib을 활용하려고 했을 때 방문했던 사이트입니다.)

- 이 외에는 파일을 가져오기 보다는 ChatGPT의 도움을 받아서 코드를 작성하고, 수정하고, 더 정확한 방법을 찾는 과정으로 진행하였습니다.