

영상 속 인물 혹은 캐릭터 detection 과  
대상과의 신뢰도 판별



SINCE 1947  
**SEO KYEONG**  
UNIVERSITY

제출일	2024.06.16	(3 조) 임주형
과목	컴퓨터비전	(3 조) 이세비
담당교수	김진현	(3 조) 최하은

## < 목 차 >

### I. 서론

1. 주제 선정 이유	3
-------------	---

### II. 구현 세부사항 (\_\_init\_\_.py)

1. File 메뉴	4
2. Video_Control_menu (Video_Control.py)	4
3. Casecade_menu (Select_Casecade.py)	7
4. Threshold (Threshold.py)	8
5. Geotrans (Geotrans.py)	9
6. Contour (Contour.py)	11
7. K-Means (K_means.py)	12

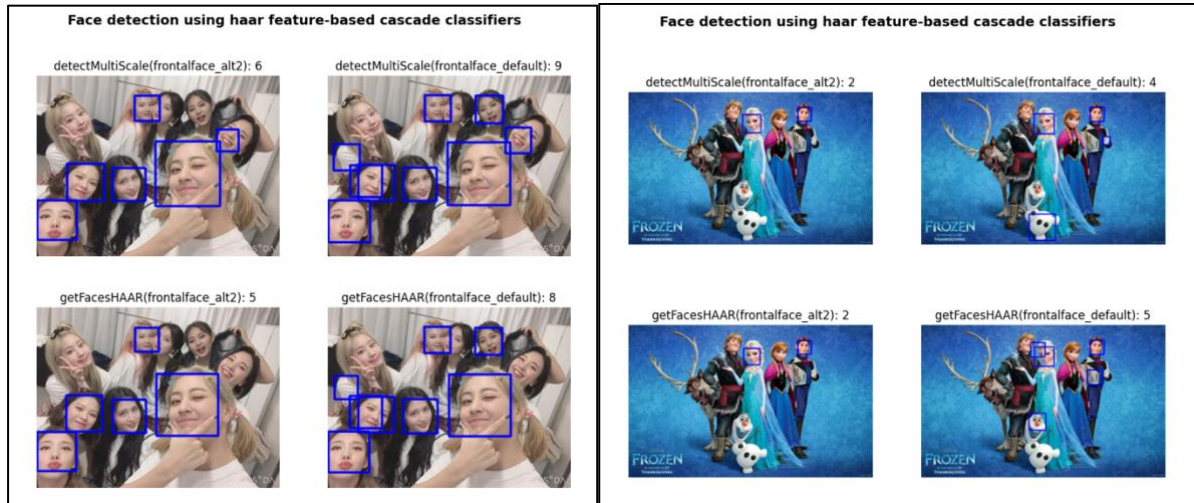
### III. 연구 과정

1. download.py	13
2. ai_test.py	14
3. TextDetection.py	15

IV. 결론 및 고찰	16
-------------	----

## I. 서론

### 1. 주제 선정 이유

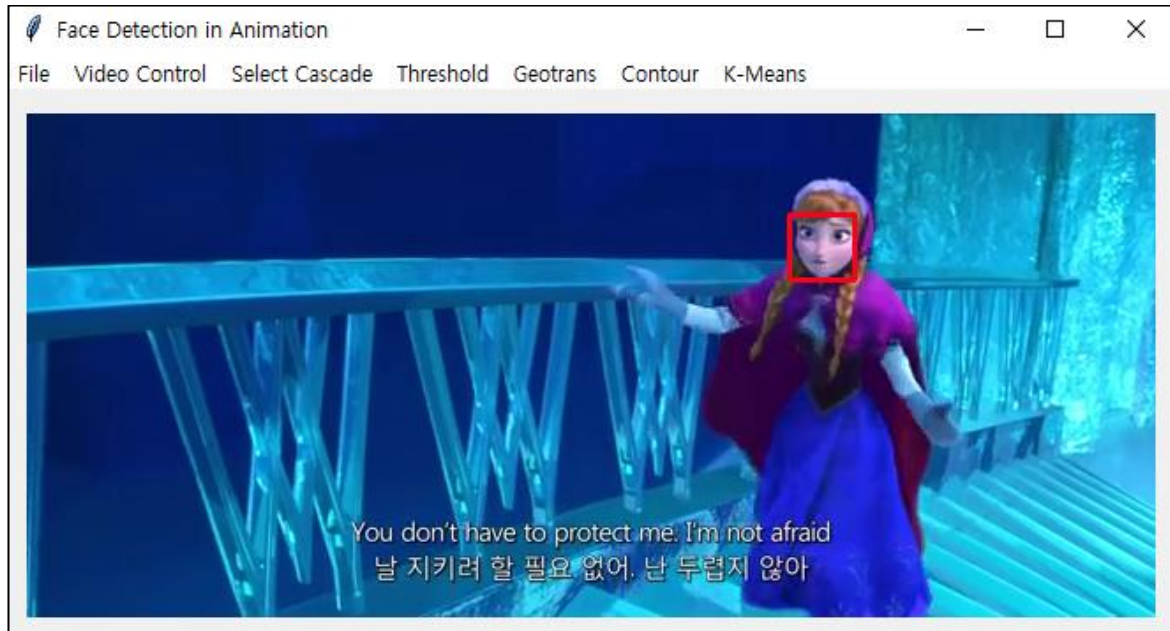


Cascade classifier 객체를 선언하고 detectMultiScale()과 getFacesHAAR()를 이용하여 애니메이션 캐릭터를 인식하여 보였다. 사람에 대해 인식은 정확하다고 파악할 수 있지만 애니메이션화 된 인물에서도 인식이 효과적으로 인식된다고 파악이 어렵다. 데이터 학습 차이가 있는 것 같다. 얼굴 인식 모델은 일반적으로 실제 사람 얼굴을 인식하도록 학습되어 사람 얼굴의 다양한 특징을 효과적으로 인식할 수 있지만, 애니메이션 캐릭터는 이러한 학습 데이터에 포함되지 않는 경우가 많아 인식률이 떨어질 수 있다.

영화에 나오는 주인공들을 대상으로 detection 하여 해당 대상과의 유사도를 출력하여 어떠한 점이 다른지 확인할 수 있도록 한다. 또한 영상을 Tkinter를 이용하여 영상 또는 이미지를 다양한 방식으로 처리할 수 있도록 하여 사용자에게 직관적인 인터페이스를 제공해 이미지 처리와 분석 작업이 더욱 효율적으로 이루어질 수 있도록 하였다.

## II. 구현 세부사항

Tkinter 를 사용하여 구축된 이 GUI 는 비디오 제어, 임계값 처리, 얼굴 검출, 기하 변환, 윤곽선 그리기 및 K-평균 군집화 등 다양한 이미지 처리 작업을 위한 사용자 친화적인 인터페이스를 제공한다.



<\_\_init\_\_.py>

먼저, GUI 의 최상위 메뉴를 생성하여 사용자는 파일 메뉴, 비디오 제어 메뉴, 얼굴 검출기 선택 메뉴, 임계값 처리 메뉴, 기하 변환 메뉴, 윤곽선 메뉴 및 K-평균 군집화 메뉴에 접근할 수 있도록 한다.

### 1. File 메뉴(+ Exit 메뉴)

```
# 메뉴 생성
menu = Menu(root)
root.config(menu=menu)

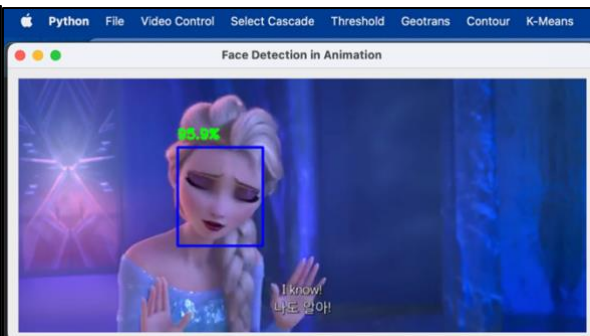
file_menu = (Menu(menu))
menu.add_cascade(label="File", menu=file_menu)
file_menu.add_command(label="Exit", command=root.quit)
```

### 2. Video\_control\_menu 메뉴 ( + Start, + Stop, + Visualization Rect, + Unvisualization Rect, + Visualization Confidence, + Unvisualization Confidence)

```
# 시작 및 중지 버튼 추가
video_control_menu = Menu(menu)
menu.add_cascade(label="Video Control", menu=video_control_menu)
video_control_menu.add_command(label="Start", command=V.start_video)
video_control_menu.add_command(label="Stop", command=V.stop_video)
video_control_menu.add_command(label="Visualization Rect", command=V.turn_on_detection_rect)
video_control_menu.add_command(label="Unvisualization Rect", command=V.turn_off_detection_rect)
video_control_menu.add_command(label="Visualization Confidence", command=V.turn_on_confidence)
video_control_menu.add_command(label="Unvisualization Confidence", command=V.turn_off_confidence)
```



Visualization Rect 메뉴 실행



Visualization Confidence 메뉴 실행

#### - Video\_Control.py

```
# 모델 파일 경로
caffe_model_path = "ssd_models/res10_300x300_ssd_iter_140000_fp16.caffemodel"
prototxt_path = "ssd_models/deploy.prototxt"

# 네트워크 로드
net = cv2.dnn.readNetFromCaffe(prototxt_path, caffe_model_path)

# 상태 변수
is_running = True           # 영상이 재생 중인가
is_running_rect = True      # face detection 사각형 출력 여부
is_showed_confidence = False # DNN을 통한 신뢰도 출력 여부
confidence_threshold = 0.3  # 이 이상의 신뢰도일 경우만 출력

# 프레임 생성
video_frame = tk.Frame(root, width=800, height=600)
video_frame.pack(side=tk.TOP, padx=10, pady=10)

# 라벨 생성
label = Label(video_frame)
label.pack()
```

DDNN 모델 파일과 네트워크 구조 파일을 로드한다.

```
def turn_on_detection_rect():
    global is_running_rect
    is_running_rect = True
! usage
def turn_off_detection_rect():
    global is_running_rect
    is_running_rect = False
! usage
def turn_on_confidence():
    global is_showed_confidence
    is_showed_confidence = True
! usage
def turn_off_confidence():
    global is_showed_confidence
    is_showed_confidence = False
```



- turn\_on\_detection\_rect(): 얼굴 검출 사각형 출력을 활성화
- turn\_off\_detection\_rect(): 얼굴 검출 사각형 출력을 비활성화
- turn\_on\_confidence(): 신뢰도 출력을 활성화
- turn\_off\_confidence(): 신뢰도 출력을 비활성화

```
def show_frame():
    if not is_running:
        return

    ret, frame = cap.read() # 비디오 캡처
    if not ret:
        root.after(10, show_frame) # 비디오가 끝났을 때 대기
        return

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # 흑백 이미지로 변환
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    # 얼굴에 사각형 그리기
    if is_running_rect:
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (25, 0, 255), 2)

    if is_showed_confidence:
        frame = detect_and_display(frame.copy(), confidence_threshold)

    # OpenCV 이미지 포맷을 PIL 이미지 포맷으로 변환
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)

    # Tkinter 라벨에 이미지 업데이트
    label.imgtk = imgtk
    label.configure(image=imgtk)
    label.after(ms=10, show_frame) # 10밀리초마다 업데이트
```

비디오 스트림을 읽어와서 얼굴을 검출하고, 이를 Tkinter 라벨에 출력하는 함수

```
def detect_and_display(frame, confidence_threshold=0.7):
    h, w = frame.shape[:2] # 프레임(영상)의 높이와 너비를 지정합니다.
    blob = cv2.dnn.blobFromImage(frame, scalefactor=1.0, size=(300, 300), mean=[104., 117., 123.], swapRB=False, crop=False)
    net.setInput(blob)
    detections = net.forward()

    # detections.shape[2] - 탐지된 객체의 수
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2] # 현재 객체의 신뢰도
        if confidence > confidence_threshold:
            # 객체 사각형 좌표 출력
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            cv2.rectangle(frame, (startX, startY), (endX, endY), (255, 0, 0), 2)

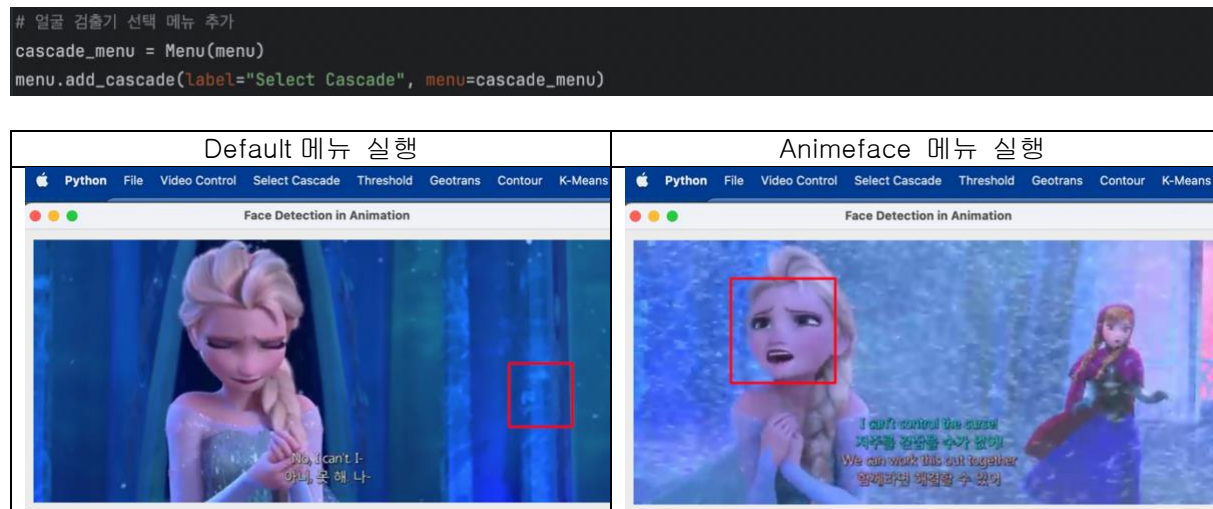
            # 신뢰도 표시
            text = "{:.1f}%".format(confidence * 100)
            y = startY - 10 if startY - 10 > 10 else startY + 10
            cv2.putText(frame, text, org=(startX, y), cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.5, color=(0, 255, 0), thickness=2)

    return frame
```

DNN을 사용하여 얼굴 검출 신뢰도를 출력한다.

=> DNN 은 이미지 처리와 컴퓨터 비전 작업에서 높은 정확도와 정밀도를 제공한다. 이미지에서 복잡한 패턴과 특징을 추출하는 데 매우 효과적이다. 단순한 경계나 색상 정보뿐만 아니라, 얼굴의 다양한 특징을 정확히 인식하고 분류할 수 있다. DNN 모델은 GPU 가속과 최적화된 연산을 통해 실시간 성능을 제공한다. 영상 출력에서 실시간으로 얼굴을 검출하고 신뢰도를 계산하는 데 적합하다.

### 3. Cascade\_menu 메뉴 (+ default, + Animeface)



#### - Select\_Cascade.py

```
# 얼굴 검출기 목록
cascade_files = {
    "Default": "cascades/haarcascades/haarcascade_frontalface_default.xml",
    "Animeface": "cascades/lbpcascades/lbpcascade_animeface.xml"
}

# 캐스케이드를 변경
1 usage
def change_cascade(cascade):
    global face_cascade, selected_cascade
    selected_cascade = cascade
    face_cascade = cv2.CascadeClassifier(cascade_files[cascade])
```


얼굴 검출을 위해 사용하는 캐스케이드 분류기를 변경하는 기능을 제공

- **haarcascade\_frontalface\_default.xml**  
Haar Cascade Classifier 기술을 사용하고, 주로 실제 사람 얼굴을 검출하는데 사용된다. 밝기와 어두움의 차이를 기반으로 얼굴의 특징을 인식하여 애니메이션 캐릭터와 같이 과장되거나 단순화된 얼굴에서는 검출 정확도가 낮다.
- **Lbpcascade\_animeface.xml**

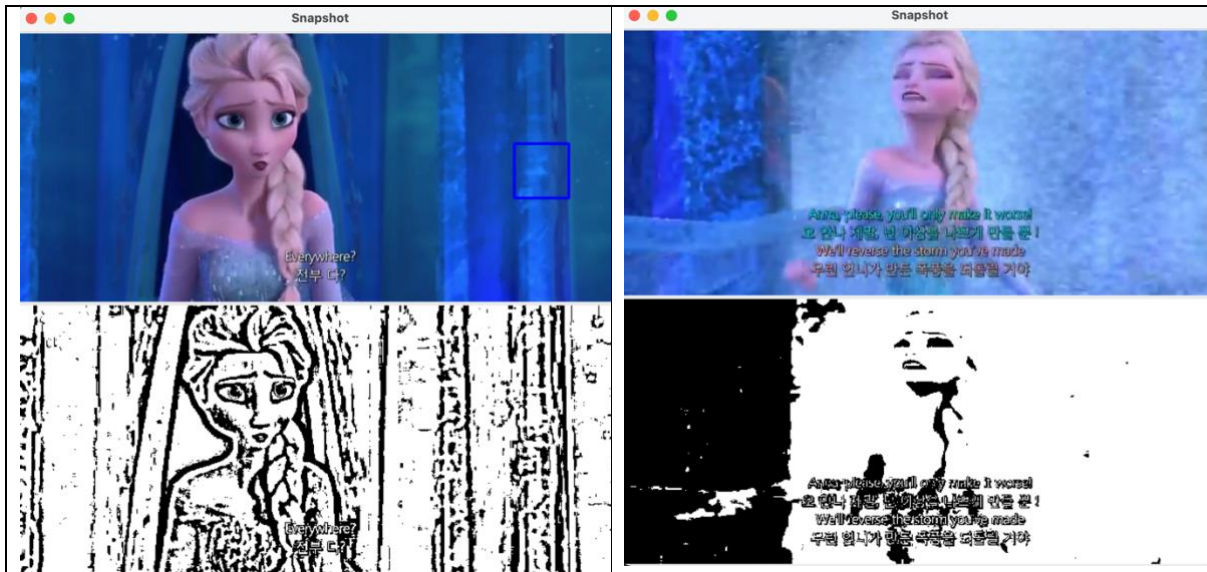
LBP(Local Binary Patterns) Casecade Classifier 기술을 사용하여 주로 이미지의 텍스처 정보를 이용하여 얼굴을 인식한다. 이미지의 각 픽셀을 중심으로 주변 픽셀과의 관계를 이진패턴으로 나타내어 특징을 추출하여 단순화된 특징에 잘 작동하여 주로 애니메이션 캐릭터 얼굴을 검출하는데 사용된다. 다양한 텍스처와 색상의 변화에 민감하여 실제 사람 얼굴에는 적합하지 않을 수 있다.

#### 4. Threshold 메뉴(+ Take Snapshot, + cv2.threshold, + cv2.adaptiveThreshold, + threshold\_otsu)

```
# 스냅샷 메뉴 추가
snapshot_menu = Menu(menu)
menu.add_cascade(label="Threshold", menu=snapshot_menu)
snapshot_menu.add_command(label="Take Snapshot", command=take_snapshot)
snapshot_menu.add_command(label="cv2.threshold", command=lambda: T.threshold_image("cv2.threshold"))
snapshot_menu.add_command(label="cv2.adaptiveThreshold", command=lambda: T.threshold_image("cv2.adaptiveThreshold"))
snapshot_menu.add_command(label="threshold_otsu", command=lambda: T.threshold_image("threshold_otsu"))
```

Snapshot 메뉴 실행	Cv2.threshold 메뉴 실행
	
Cv2.adaptiveThreshold 메뉴 실행	Threshold_otsu 메뉴 실행





## – Threshold.py

```
def threshold_image(threshold_type):
    ret, frame = cap.read()
    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

        # Threshold 메뉴의 버튼에 따라 다른 작업이 이루어 짐
        # 각 Threshold 과정을 인지하기를 바람
        if threshold_type == "cv2.threshold":
            _, thresh = cv2.threshold(gray, thresh: 127, maxval: 255, cv2.THRESH_BINARY)
        elif threshold_type == "cv2.adaptiveThreshold":
            thresh = cv2.adaptiveThreshold(gray, maxValue: 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, blockSize: 11, C: 2)
        elif threshold_type == "threshold_otsu":
            _, thresh = cv2.threshold(gray, thresh: 0, maxval: 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

        # 캡처된 스냅샷 창에 표시
        show_snapshot(frame, thresh)
```

- cv2.threshold : 고정 임계값을 사용하여 픽셀 값이 127 보다 크면 255, 아니면 0 으로 설정
- cv2.adaptiveThreshold : 적응형 임계값을 사용하여 이미지의 작은 영역에 대해 각각 다른 임계값을 계산하여 적용
- threshold\_otsu : Otsu 의 방법을 사용하여 자동으로 최적의 임계값을 계산하여 적용

## 5. Geotrans 메뉴 (+ Translation, + Rotation, + Affine Transformation, + Perspective Transformation)

```
# Geotrans 메뉴 추가
geotrans_menu = Menu(menu)
menu.add_cascade(label="Geotrans", menu=geotrans_menu)
geotrans_menu.add_command(label="Translation", command=G.translation)
geotrans_menu.add_command(label="Rotation", command=G.rotation)
geotrans_menu.add_command(label="Affine Transformation", command=G.affine_transformation)
geotrans_menu.add_command(label="Perspective Transformation", command=G.perspective_transformation)
```



– Geotrans.py

```

def translation():
    ret, frame = cap.read()
    if ret:
        rows, cols, _ = frame.shape
        M = np.float32([[1, 0, 50], [0, 1, 50]]) # 포인트
        translated = cv2.warpAffine(frame, M, dsize: (cols, rows)) # 포인트
        show_snapshot(translated)

def rotation():
    ret, frame = cap.read()
    if ret:
        rows, cols, _ = frame.shape
        M = cv2.getRotationMatrix2D(center: (cols / 2, rows / 2), angle: 45, scale: 1) # 포인트
        rotated = cv2.warpAffine(frame, M, dsize: (cols, rows)) # 포인트
        show_snapshot(rotated)

1 usage
def affine_transformation():
    ret, frame = cap.read()
    if ret:
        rows, cols, _ = frame.shape
        pts1 = np.float32([[50, 50], [200, 50], [50, 200]]) # 포인트
        pts2 = np.float32([[10, 100], [230, 80], [100, 250]]) #
        M = cv2.getAffineTransform(pts1, pts2) #
        transformed = cv2.warpAffine(frame, M, dsize: (cols, rows)) # 포인트
        show_snapshot(transformed)

def perspective_transformation():
    ret, frame = cap.read()
    if ret:
        rows, cols, _ = frame.shape
        pts1 = np.float32([[56, 65], [368, 52], [28, 387], [389, 390]]) # 포인트
        pts2 = np.float32([[0, 0], [250, 0], [0, 300], [250, 300]]) #
        M = cv2.getPerspectiveTransform(pts1, pts2) #
        transformed = cv2.warpPerspective(frame, M, dsize: (cols, rows)) # 포인트
        show_snapshot(transformed)

```

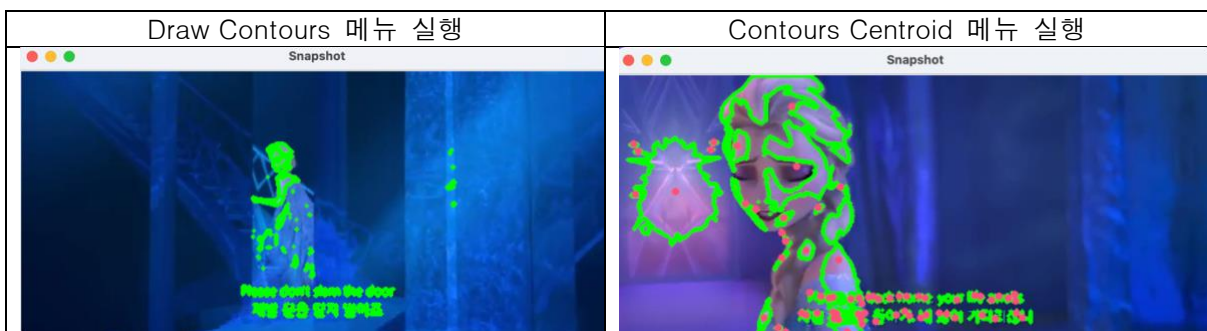
- Translation (평행이동) : 이미지를 오른쪽과 아래로 50 픽셀로 이동
- Rotation (회전) : 이미지 중심을 기준으로 45 도 회전
- Affine Transformation (어파인 변환) : 원본삼각형 pts1 을 변환 후 삼각형 pts2 로 변환
- Perspective Transformation (투시변환) : 원본사각형 pts1 을 변환 후 사각형 pts2 로 변환

## 6. Contour 메뉴 (+ Draw Contours, + Contours Centroid)

```

# Contour 메뉴 추가
contour_menu = Menu(menu)
menu.add_cascade(label="Contour", menu=contour_menu)
contour_menu.add_command(label="Draw Contours", command=C.draw_contours)
contour_menu.add_command(label="Contours Centroid", command=C.moments)

```



– Contour.py

```
def draw_contours():
    ret, frame = cap.read()
    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, thresh: 127, maxval: 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(binary, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE) # 포인트

        # Contour 그리기
        contour_image = cv2.drawContours(frame.copy(), contours, -1, color: (0, 255, 0), thickness: 3) # 포인트

        show_snapshot(contour_image)

def moments():
    ret, frame = cap.read()
    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, thresh: 127, maxval: 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        # Contour 그리기
        contour_image = cv2.drawContours(frame.copy(), contours, -1, color: (0, 255, 0), thickness: 3)

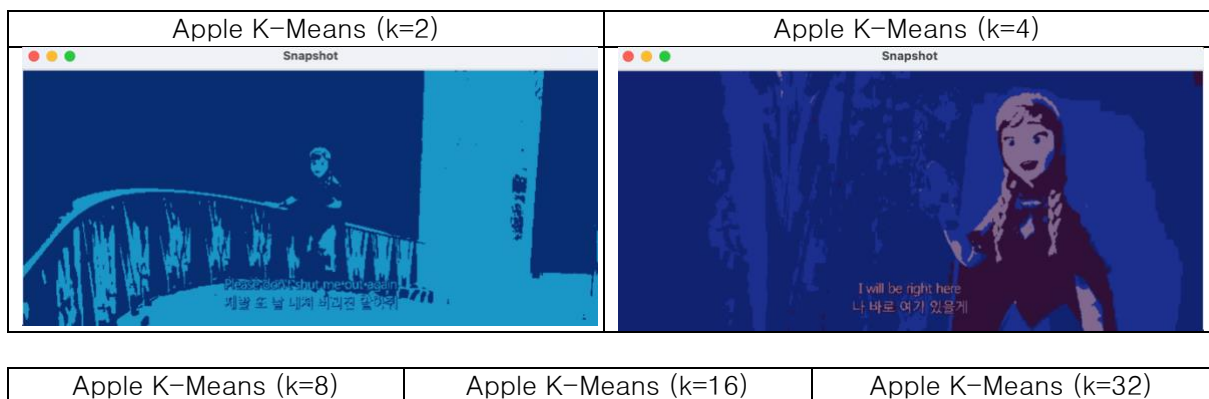
        # 각 Contour의 중심점 계산 및 표시
        for cnt in contours:
            M = cv2.moments(cnt) # 포인트
            if M["m00"] != 0:
                cX = int(M["m10"] / M["m00"])
                cY = int(M["m01"] / M["m00"])
                cv2.circle(contour_image, center: (cX, cY), radius: 4, color: (125, 80, 255), -1)

        show_snapshot(contour_image)
```

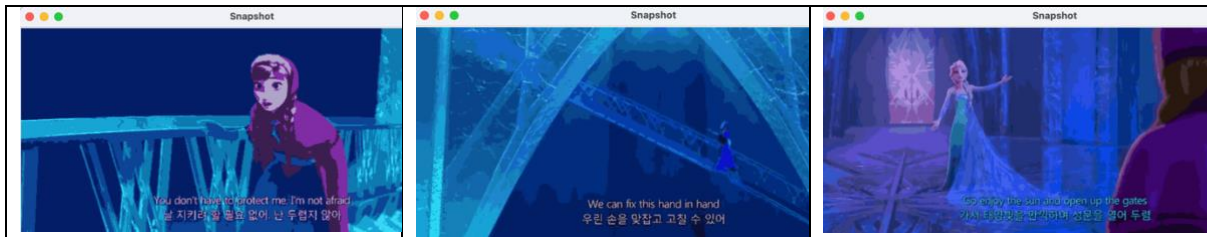
- Draw\_contours() : 윤곽선 그리기
- Momoents() : 윤곽선 중심점 계산하여 중심점을 원으로 표시

## 7. K-Means 메뉴 [ + Apply K-Means (k=2),(k=4),(k=8),(k=16),(k=32) ]

```
# K-Means 메뉴 추가
kmeans_menu = Menu(menu)
menu.add_cascade(label="K-Means", menu=kmeans_menu)
# K값에 따른 K-Means Color Quantization 버튼 추가
k_values = [2, 4, 8, 16, 32]
for k in k_values:
    kmeans_menu.add_command(label=f"Apply K-Means (k={k})", command=K.create_kmeans_button(k))
```







## - K\_means.py

```
def color_quantization(image, k):
    num_maxCount = 100 # 최대 반복 횟수
    eps = 0.2 # 수렴 기준
    num_attempts = 10 # 중심 포인트 선택 시도 횟수

    # 이미지를 2D 배열로 변환하고 모든 픽셀을 3차원 벡터로 변환한다.
    data = np.float32(image).reshape((-1, 3))

    # (최대 반복 횟수, 수렴 기준)
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, num_maxCount, eps) # 포인트

    _, label, center = cv2.kmeans(data, k, bestLabels=None, criteria=criteria, num_attempts=num_attempts, cv2.KMEANS_RANDOM_CENTERS) # 포인트

    center = np.uint8(center)
    result = center[label.flatten()]

    # 영상과 같은 shape로 만든다.
    result = result.reshape(image.shape)

    return result

def create_kmeans_button(k):
    def apply_kmeans():
        ret, frame = cap.read()
        if ret:
            quantized_image = color_quantization(frame, k)
            show_snapshot(quantized_image)
    return apply_kmeans
```

k-평균 군집화 기법을 통해 주어진 이미지의 색상을 K개의 군집으로 변환된 이미지 출력

## III. 연구 과정

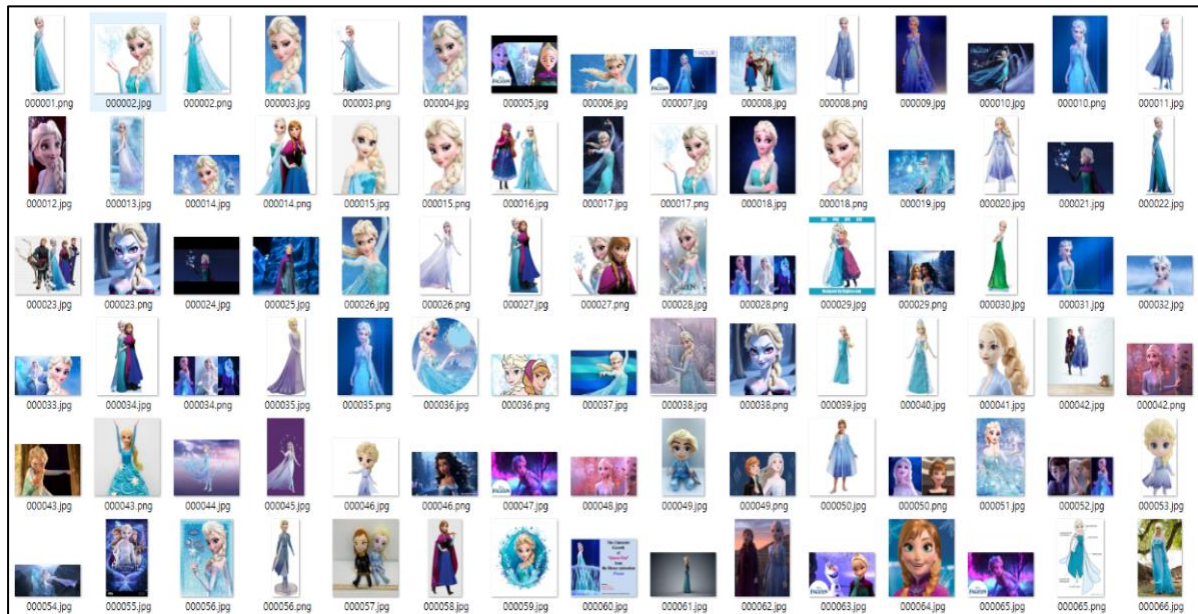
1. icrawler 라이브러리를 사용하여 Google 이미지 검색을 통해 특정 키워드에 해당하는 이미지를 다운로드하여 데이터 셋을 형성하려 시도 (download.py)

```
keywords = [
    ("Elsa Frozen", "frozen_characters/Elsa"),
    ("Elsa Frozen character", "frozen_characters/Elsa"),
    ("Frozen Elsa", "frozen_characters/Elsa"),
    ("Anna Frozen", "frozen_characters/Anna"),
    ("Anna Frozen character", "frozen_characters/Anna"),
    ("Frozen Anna", "frozen_characters/Anna")
]
```

=> Elsa, Anna 분류에 맞게 디렉토리를 나눠 저장



결론 : 분류에 맞게 이미지는 저장되나 예상 외로 저장되는 이미지 수가 너무 적었고, 주제에 맞지 않은 이미지도 포함되어 face detection 이 원활하게 진행되지 않았습니다.



2. 본래 프로그램에서 face detection 이 진행되는 동시에 해당 사각형 부분을 이미지로 저장하여 해당 이미지를 데이터셋으로 만들면 이 데이터셋을 통해 어느 정도 얼굴 인식이 가능하지 않을까 추측 (ai\_test.py)

```
output_directory = "frozen_characters/Captures/"
# 얼굴에 사각형 그리기
if is_running_rect:
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 255), 2)

    # 얼굴 영역 추출
    face_image = frame[y:y + h, x:x + w]

    # 이미지 저장
    save_path = os.path.join(output_directory,
```

```
f"frame_{frame_count}_face.jpg")
cv2.imwrite(save_path, face_image)

f.frame_count += 1
```

=> 해당 코드를 실행하여 아래와 같이 캡처 이미지들을 저장하였다.



결론 : 영상이 재생되는 동안 위의 데이터셋을 통한 face detection 은 이뤄지지 않아 해당 모듈 적용은 보류하였다.

### 3. 텍스트 인식 - pytesseract 라이브러리는 텍스트를 인식하는 모듈

해당 라이브러리를 통해 영상 자막을 detection 하여 출력 (TextDetection.py)

그러나, 영어, 한글 두 언어의 탐지 결과가 높은 정확도를 보이지 않는 경우가 다수

```

Detected English Text: = 24 B0|0F

Detected Korean Text: 는 것 뿐이야

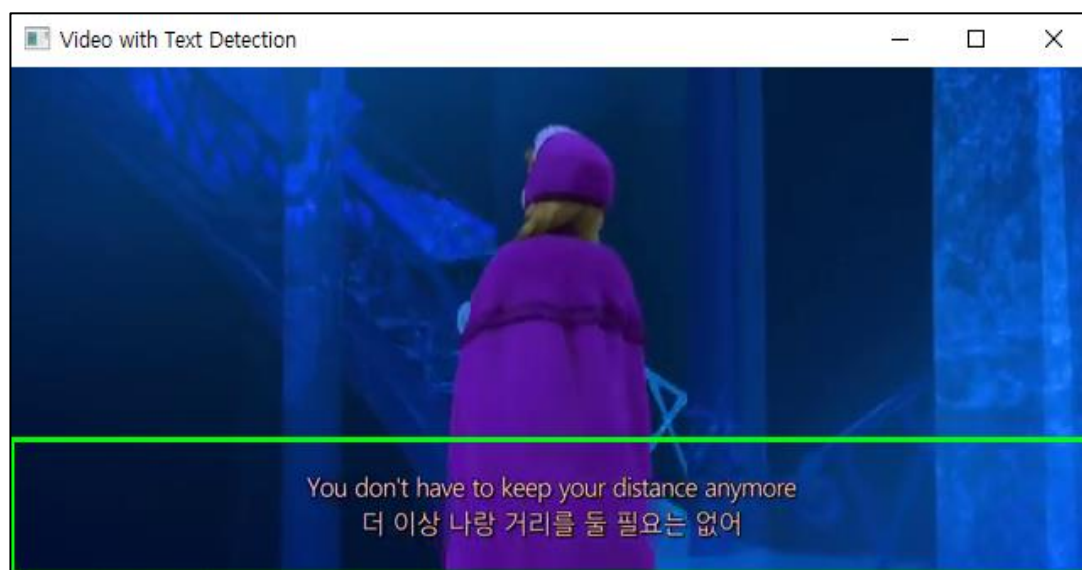
Detected English Text: don't L Tm not atrad
'a X17 BS OUT S237) tot

Detected Korean Text: 0001 \ 10001 2000
날 지키레알 필요 없어 잔 두렵지 않아

Detected English Text: ™ You don't have to protect me im not aff
SB X17121 B BQ S10}. HEIR) ROP

Detected Korean Text: | 70 000라내 10 10061 1 107 1001 21
날 지키려 할 필요 없어 난 두렵지 않아

```



=> 초록색 영역이 detection 영역

결론 : 높은 정확도를 보였다면 본래 프로그램에 포함시켜야 했겠지만 위와 같은 출력으로는 완성도가 낮아질 것을 우려하여 해당 모듈은 제외함

#### IV. 결론 및 고찰

이번 프로젝트는 OpenCV 와 Tkinter 를 사용하여 실시간으로 비디오 프레임을 처리하고 다양한 이미지 변환 및 분석을 수행하는 GUI 애플리케이션을 구현하였다. 또한 사람의 얼굴 인식이

아니라 애니메이션화 된 캐릭터를 인식함으로써 단순화되고 과장된 특징을 가진 이미지를 처리함으로써 차이점을 파악할 수 있었다.

Tkinter 를 사용한 GUI 는 복잡한 명령어 입력 없이 버튼 클릭만으로 다양한 이미지 처리 기능을 할 수 있게 하였다. 새로운 이미지 처리 알고리즘이나 기능을 더욱 쉽게 추가할 수 있고, 그 결과를 시각적으로 빠르게 확인할 수 있다. 컴퓨터 비전 분야의 연구와 응용에 기초를 제공하여, 앞으로 많은 이미지 처리 기능을 구현 가능할 것이다.