😄

얼굴 탐지 및 추출

mtcnn - 오래 걸림

YOLO

https://docs.ultralytics.com/ko/models/yolov8/

pytorch 기반

객체 탐지 모델

다른 사람이 이미 학습시킨 모델 사용하면 얼굴 탐지 가능

MediaPipe

안드로이드, 웹, 파이썬, ios 다 지원

https://ai.google.dev/edge/mediapipe/solutions/guide?hl=ko

웹

https://mediapipe-studio.webapps.google.com/studio/demo/face_detector

가벼우면서 성능도 괜찮음 cpu만으로 가능, tensorflow lite 기반

face detection

사진

```
import cv2
import mediapipe as mp
import os

# 경로 설정
image_path = 'id_card.jpg'  # 분석할 이미지 경로
output_dir = 'extracted_faces'
os.makedirs(output_dir, exist_ok=True)
```

```
# MediaPipe 설정
mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils

# 이미지 읽기
image = cv2.imread(image_path)
h, w, _ = image.shape

# 얼굴 감지
with mp_face_detection.FaceDetection(model_selection=1, min_detection_c
onfidence=0.6) as face_detection:
    results = face_detection.process(cv2.cvtColor(image, cv2.COLOR_BGR2
RGB))

    if results.detections:
        for i, detection in enumerate(results.detections):
            bbox = detection.location_data.relative_bounding_box
            x = int(bbox.xmin * w)
            y = int(bbox.ymin * h)
            w_box = int(bbox.width * w)
            h_box = int(bbox.height * h)

            x, y = max(0, x), max(0, y)
            face_crop = image[y:y+h_box, x:x+w_box]
            output_path = os.path.join(output_dir, f"face_{i+2}.png")
            cv2.imwrite(output_path, face_crop)
            print(f"저장됨: {output_path}")

    else:
        print("얼굴을 찾을 수 없습니다.")
```

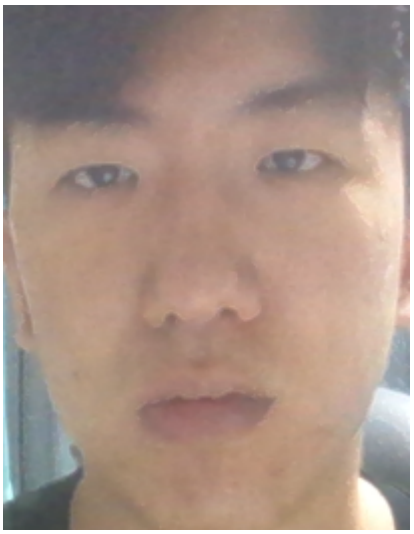
실시간 - facemesh

```
import cv2
import mediapipe as mp
import os
import time

SAVE_DIR = 'saved_faces'
os.makedirs(SAVE_DIR, exist_ok=True)
```

```
# 저장 상태 추적
saved_angles = {
    'left': False,
    'center': False,
    'right': False
}

# MediaPipe 설정
mp_face_mesh = mp.solutions.face_mesh
mp_drawing = mp.solutions.drawing_utils

def get_yaw_from_landmarks(landmarks, image_width):
    # 코와 양쪽 귀 근처 좌표로 yaw 추정
    nose_x = landmarks[1].x * image_width
    left_cheek_x = landmarks[234].x * image_width
    right_cheek_x = landmarks[454].x * image_width

    face_width = right_cheek_x - left_cheek_x
    nose_offset = nose_x - (left_cheek_x + face_width / 2)
    yaw_ratio = nose_offset / face_width
    yaw_degrees = yaw_ratio * 90  # 대략적인 각도 변환
    return yaw_degrees

cap = cv2.VideoCapture(0)

with mp_face_mesh.FaceMesh(static_image_mode=False, max_num_faces
=1, refine_landmarks=True) as face_mesh:
    while cap.isOpened() and not all(saved_angles.values()):
        ret, frame = cap.read()
        if not ret:
            break

        image = cv2.flip(frame, 1)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = face_mesh.process(rgb)
        h, w, _ = image.shape
```

```
        if results.multi_face_landmarks:
            for landmarks in results.multi_face_landmarks:
                yaw = get_yaw_from_landmarks(landmarks.landmark, w)

                # 각도 분류
                if yaw < -10:
                    direction = 'left'
                elif yaw > 10:
                    direction = 'right'
                else:
                    direction = 'center'

                # 저장 조건
                if not saved_angles[direction]:
                    # 얼굴 전체 박스 구하기 (임의 범위)
                    xs = [lm.x for lm in landmarks.landmark]
                    ys = [lm.y for lm in landmarks.landmark]
                    x_min = int(min(xs) * w)
                    y_min = int(min(ys) * h)
                    x_max = int(max(xs) * w)
                    y_max = int(max(ys) * h)

                    # 잘라내기 (경계 처리)
                    x_min, y_min = max(x_min, 0), max(y_min, 0)
                    face_crop = image[y_min:y_max, x_min:x_max]

                    timestamp = int(time.time() * 1000)
                    filename = os.path.join(SAVE_DIR, f"face_{direction}_{timestam
p}.png")
                    cv2.imwrite(filename, face_crop)
                    saved_angles[direction] = True
                    print(f"{direction.upper()} 얼굴 저장 완료: {filename}")

                # 방향 표시
                cv2.putText(image, f'Direction: {direction} ({yaw:.1f})',
                            (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        cv2.imshow("FaceMesh 각도별 얼굴 저장", image)
```

```
        if cv2.waitKey(5) & 0xFF == 27:
            break

cap.release()
cv2.destroyAllWindows()
print("모든 방향의 얼굴이 저장되었습니다.")
```



구분	`dx 방식` (눈 간 거리차)	`yaw 방식` (실제 얼굴 회전 추정)
방법	눈 좌표의 x값 차이 계산	코·눈·귀 위치로 얼굴의 **회전 각도(yaw)** 추정
원리	눈 간 거리로 비율 추정	3D 위치 기반, 실제 각도 근사
구현	쉽고 빠름 (단순 수치 비교)	약간 복잡하지만 정밀함
정밀도	낮음 (기울기에 민감, 추정치)	높음 (정확한 회전 판단)
사용 용도	간단한 분류에 적합 (좌/우/정)	얼굴 정렬, 인식 전처리에 적합