0227 진행상황

코드

```
from ultralytics import YOLO
import cv2
import numpy as np



# 모델 불러오기
model = YOLO("C:/capstone/yolo/runs/detect/train7/weights/best.pt")



# 이미지 경로
image_path = "C:/capstone/yolo/0227test01.jpg"



# 추론 진행
results = model.predict(source=image_path, conf=0.35)



# 이미지 읽기
image = cv2.imread(image_path)
image_height, image_width = image.shape[:2]
image_area = image_width * image_height



# 클래스 이름

class_names = [

    'person_all', 'head', 'face', 'eye', 'nose', 'mouth', 'ear', 'hair',

    'neck', 'body', 'arm', 'hand', 'leg', 'foot', 'button', 'pocket',
'sneakers', 'man_shoes', 'woman_shoes'
```

```
]



# 탐지 결과 저장

detections = {name: [] for name in class_names}



# 탐지 결과 처리

for result in results:

    boxes = result.boxes

    for box in boxes:

        xyxy = box.xyxy.cpu().numpy()[0]

        conf = box.conf.cpu().numpy()[0]

        cls_id = int(box.cls.cpu().numpy()[0])

        label = class_names[cls_id]

        x1, y1, x2, y2 = map(int, xyxy)

        width, height = x2 - x1, y2 - y1

        detections[label].append([x1, y1, x2, y2, width, height, conf])



########################################

# 전체 크기 세분화

########################################

def classify_overall_size(person_area, image_area):

    ratio = person_area / image_area
```

```
    if ratio < 0.3:
        return "매우 작음"
    elif ratio < 0.6:
        return "작음"
    elif ratio < 0.8:
        return "보통"
    elif ratio < 1.0:
        return "큼"
    else:
        return "매우 큼"


########################################
# 'person_all' 크기 및 위치 분석
########################################
if 'person_all' in detections and len(detections['person_all']) > 0:
    person_box = detections['person_all'][0]
    x1, y1, x2, y2 = person_box[:4]
    person_area = person_box[4] * person_box[5]
    person_size = classify_overall_size(person_area, image_area)
    # 수평 치우침
    person_center_x = (x1 + x2) / 2
```

```
    image_center_x = image_width / 2

    tolerance = image_width * 0.05

    if person_center_x < image_center_x - tolerance:

        bias_horizontal = "수평.왼쪽으로 치우침"

    elif person_center_x > image_center_x + tolerance:

        bias_horizontal = "수평.오른쪽으로 치우침"

    else:

        bias_horizontal = "수평.중앙"

    # 수직 위치

    person_center_y = (y1 + y2) / 2

    image_center_y = image_height / 2

    vertical_tolerance = image_height * 0.05

    if abs(person_center_y - image_center_y) < vertical_tolerance * 0.5:

        bias_vertical = "수직.과도한 정중앙"

    elif person_center_y < image_center_y - vertical_tolerance:

        bias_vertical = "수직.상단"

    elif person_center_y > image_center_y + vertical_tolerance:

        bias_vertical = "수직.하단"

    else:

        bias_vertical = "수직.중앙"

    # 특수 가장자리 분석

    cut_off = []
```

```
    if y1 <= 0:
        cut_off.append("상 절단")
    # if y2 >= image_height:
    #     cut_off.append("하 절단")
    if x1 <= 0:
        cut_off.append("좌 절단")
    if x2 >= image_width:
        cut_off.append("우 절단")
    cut_off_status = ", ".join(cut_off) if cut_off else "절단 없음"
    detections['person_all'][0].extend([person_size, bias_horizontal,
bias_vertical, cut_off_status])
else:
    person_size = "not 검출됨"
    bias_horizontal = "not 검출됨"
    bias_vertical = "not 검출됨"
    cut_off_status = "not 검출됨"

########################################
# head 기준 기대 크기 비율
########################################
if 'head' in detections and len(detections['head']) > 0:
    head_box = detections['head'][0]
```

```
    head_w, head_h = head_box[4], head_box[5]
    expected = {
        'eye':   {'w': 0.233 * head_w, 'h': 0.143 * head_h},
        'nose':  {'w': 0.067 * head_w, 'h': 0.143 * head_h},
        'mouth': {'w': 0.333 * head_w, 'h': 0.029 * head_h},
        'ear':   {'w': 0.1   * head_w, 'h': 0.229 * head_h},
        'hair':  {'w': 1.333 * head_w, 'h': 0.229 * head_h},
        'neck':  {'w': 0.067 * head_w, 'h': 0.286 * head_h},
        'face':  {'w': 0.6   * head_w, 'h': 0.7   * head_h},
        'body':  {'w': 1.5   * head_w, 'h': 0.3   * head_h},
        'arm':   {'w': 0.25  * head_w, 'h': 1.2   * head_h},
        'hand':  {'w': 0.15  * head_w, 'h': 0.15  * head_h},
    }
else:
    expected = {}


########################################
# 크기 분류 함수
########################################
def classify_size(detected, expected):
    if detected < expected * 0.6:
```

```
        return "매우 작음", -2
    elif detected < expected * 0.8:
        return "작음", -1
    elif detected <= expected * 1.2:
        return "평균", 0
    elif detected <= expected * 1.4:
        return "큼", 1
    else:
        return "매우 큼", 2


def combine_status(score_w, score_h):
    total = score_w + score_h
    if total <= -3:
        return "매우 작음"
    elif total == -2:
        return "작음"
    elif total <= 1:
        return "보통"
    elif total == 2:
        return "큼"
    else:
        return "매우 큼"
```

```
for part in expected.keys():
    if part in detections:
        for box in detections[part]:
            w, h = box[4], box[5]
            status_w, score_w = classify_size(w, expected[part]['w'])
            status_h, score_h = classify_size(h, expected[part]['h'])
            combined = combine_status(score_w, score_h)
            box.append(combined)

if 'neck' in detections:
    for neck_box in detections['neck']:
        neck_w, neck_h = neck_box[4], neck_box[5]
        status_w, score_w = classify_size(neck_w, expected['neck']['w'])
        status_h, score_h = classify_size(neck_h, expected['neck']['h'])
        neck_status = combine_status(score_w, score_h)
        neck_box.append(neck_status)

########################################
# 필압 및 선 분석
########################################
```

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(gray, 50, 150)

edge_density = np.count_nonzero(edges) / (image_height * image_width)

pressure_status = "강한 필압" if edge_density > 0.05 else "약한 필압"



# 필압 변화 분석

regions = [edges[:image_height//2, :], edges[image_height//2:, :]]

densities = [np.count_nonzero(region) / (region.shape[0] * region.shape[1])
for region in regions]

pressure_var = np.std(densities)

pressure_change = "과도한 변화" if pressure_var > 0.02 else "적당한 변화"



lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=50, minLineLength=30,
maxLineGap=10)

if lines is not None:

    lengths = [np.linalg.norm((line[0][0]-line[0][2], line[0][1]-line[0]
[3])) for line in lines]

    avg_line_length = np.mean(lengths)

    line_status = "전체적 강한 선" if avg_line_length > 100 else "일부분 강한
선"

else:

    line_status = "약한 선"



# 지면선 존재 여부
```

```
ground_line_exists = False

if lines is not None:

    for line in lines:

        x1_l, y1_l, x2_l, y2_l = line[0]

        dx = x2_l - x1_l

        dy = y2_l - y1_l

        angle = np.degrees(np.arctan2(dy, dx))

        if abs(angle) < 10 or abs(angle - 180) < 10:

            if y1_l > image_height * 0.8 and y2_l > image_height * 0.8:

                ground_line_exists = True

                break



########################################

# 선 특징 분석

########################################

line_lengths = []

if lines is not None:

    for line in lines:

        x1_l, y1_l, x2_l, y2_l = line[0]

        length = np.sqrt((x2_l - x1_l)**2 + (y2_l - y1_l)**2)

        line_lengths.append(length)
```

```
    avg_length = np.mean(line_lengths)

    line_length_status = "길다" if avg_length > 100 else "짧다"

else:

    line_length_status = "정보 없음"



contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

curvature_scores = []

for cnt in contours:

    epsilon = 0.01 * cv2.arcLength(cnt, True)

    approx = cv2.approxPolyDP(cnt, epsilon, True)

    curvature_scores.append(0 if len(approx) <= 3 else 1)

if curvature_scores:

    shape_status = "직선적" if np.mean(curvature_scores) < 0.5 else "곡선적"

else:

    shape_status = "정보 없음"



########################################

# 세부 묘사

########################################

orb = cv2.ORB_create()

keypoints = orb.detect(gray, None)
```

```
num_keypoints = len(keypoints)

density = num_keypoints / image_area

if density > 0.0005:

    detail_status = "과도한 세부 묘사"

elif density < 0.0001:

    detail_status = "부족한 세부 묘사"

else:

    detail_status = "보통"



########################################

# 움직임 표현

########################################

laplacian_var = cv2.Laplacian(gray, cv2.CV_64F).var()

if laplacian_var < 100:

    movement_status = "과도한 움직임 표현"

elif laplacian_var > 300:

    movement_status = "움직임 부족"

else:

    movement_status = "적당한 움직임"



########################################

# 대칭성 분석
```

```
########################################

if 'head' in detections and len(detections['head']) > 0:

    head_box = detections['head'][0]

    x1_h, y1_h, x2_h, y2_h = head_box[:4]

    head_img = image[y1_h:y2_h, x1_h:x2_h]

    head_img_flipped = cv2.flip(head_img, 1)

    diff = cv2.absdiff(head_img, head_img_flipped)

    mean_diff = np.mean(diff)

    symmetry_status = "대칭성이 높음" if mean_diff < 20 else "대칭성이 낮음"

else:

    symmetry_status = "정보 없음"



########################################

# 투명성 분석

########################################

if image.shape[2] == 4:

    alpha_channel = image[:, :, 3]

    avg_alpha = np.mean(alpha_channel)

    transparency_status = "투명한 표현이 과도함" if avg_alpha < 100 else "보통"

else:

    transparency_status = "해당 없음"
```

```
########################################
# 머리 크기 분류
########################################
def classify_head_size(head_box, person_box=None, image_area=None):
    head_area = head_box[4] * head_box[5]
    if person_box is not None:
        person_area_val = person_box[4] * person_box[5]
        ratio = head_area / person_area_val
    elif image_area is not None:
        ratio = head_area / image_area
    else:
        ratio = 0
    if ratio < 0.2:
        return "작음"
    elif ratio < 0.35:
        return "보통"
    else:
        return "큼"


if 'head' in detections and len(detections['head']) > 0:
    head_box = detections['head'][0]
```

```
    if 'person_all' in detections and len(detections['person_all']) > 0:

        person_box = detections['person_all'][0]

        head_size_class = classify_head_size(head_box,
person_box=person_box)

    else:

        head_size_class = classify_head_size(head_box,
image_area=image_area)

else:

    head_size_class = "검출되지 않음"



########################################

# 얼굴 분석: 뒤통수 (+옆모습)

########################################

if 'face' in detections and len(detections['face']) > 0:

    face_detected = True

    num_eyes = len(detections.get('eye', []))

    has_nose = 'nose' in detections and len(detections['nose']) > 0

    has_mouth = 'mouth' in detections and len(detections['mouth']) > 0

    if num_eyes == 0 and not has_nose and not has_mouth:

        face_status = "뒤통수"

    # elif num_eyes == 1:

    #     face_status = "옆모습"
```

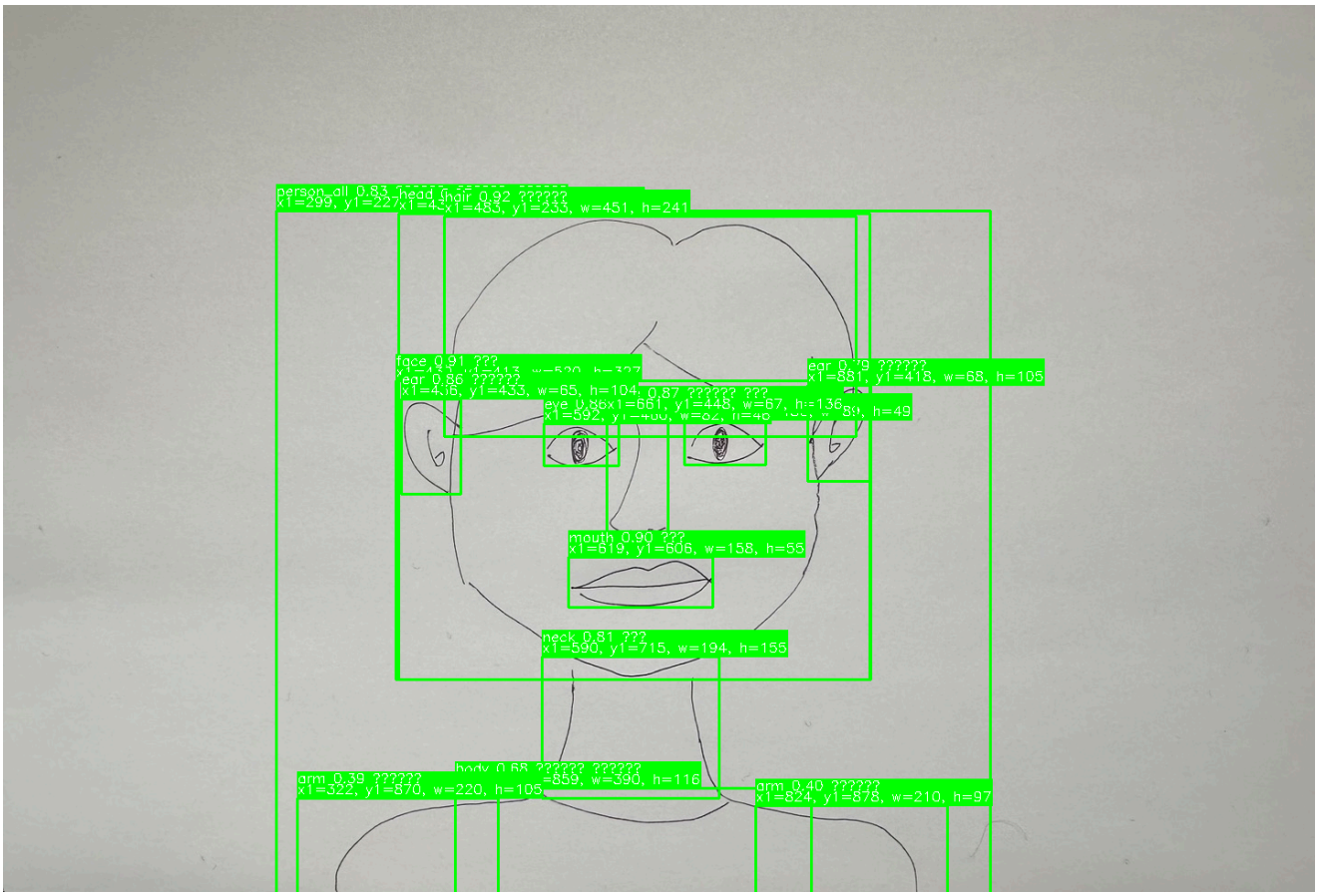
```
    else:

        face_status = "정면 또는 기타"

else:

    face_status = "검출되지 않음"



########################################

# 머리와 몸의 단절

########################################

if 'head' in detections and len(detections['head']) > 0 and 'body' in
detections and len(detections['body']) > 0:
    head_box = detections['head'][0]
    body_box = detections['body'][0]
    gap = body_box[1] - head_box[3]
    head_height = head_box[5]
    threshold = 0.1 * head_height
    if gap > threshold and 'neck' not in detections:
        disconnection_status = "머리와 몸의 단절"
    else:
        disconnection_status = "연결됨"
elif 'head' in detections and len(detections['head']) > 0 and ('body' not in
detections or len(detections['body']) == 0):
    disconnection_status = "몸 생략"
elif 'body' in detections and len(detections['body']) > 0 and ('head' not in
detections or len(detections['head']) == 0):
    disconnection_status = "머리 생략"
else:
    disconnection_status = "머리와 몸 모두 미검출"



########################################

# 시각화

########################################
```

```
for label in detections:

    for box in detections[label]:

        x1, y1, x2, y2, width, height, conf = box[:7]

        if label == 'person_all':

            status = f"{box[7]}, {box[8]}, {box[9]}, {box[10]}" if len(box)
> 10 else ""

        else:

            status = box[7] if len(box) > 7 else ""

        text1 = f"{label} {conf:.2f} {status}" if status else f"{label}
{conf:.2f}"

        text2 = f"x1={x1}, y1={y1}, w={width}, h={height}"



        font = cv2.FONT_HERSHEY_SIMPLEX

        scale = 0.5

        thickness = 1

        (text1_width, text1_height), _ = cv2.getTextSize(text1, font, scale,
thickness)

        (text2_width, text2_height), _ = cv2.getTextSize(text2, font, scale,
thickness)

        box_height = text1_height + text2_height + 5

        box_width = max(text1_width, text2_width)



        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

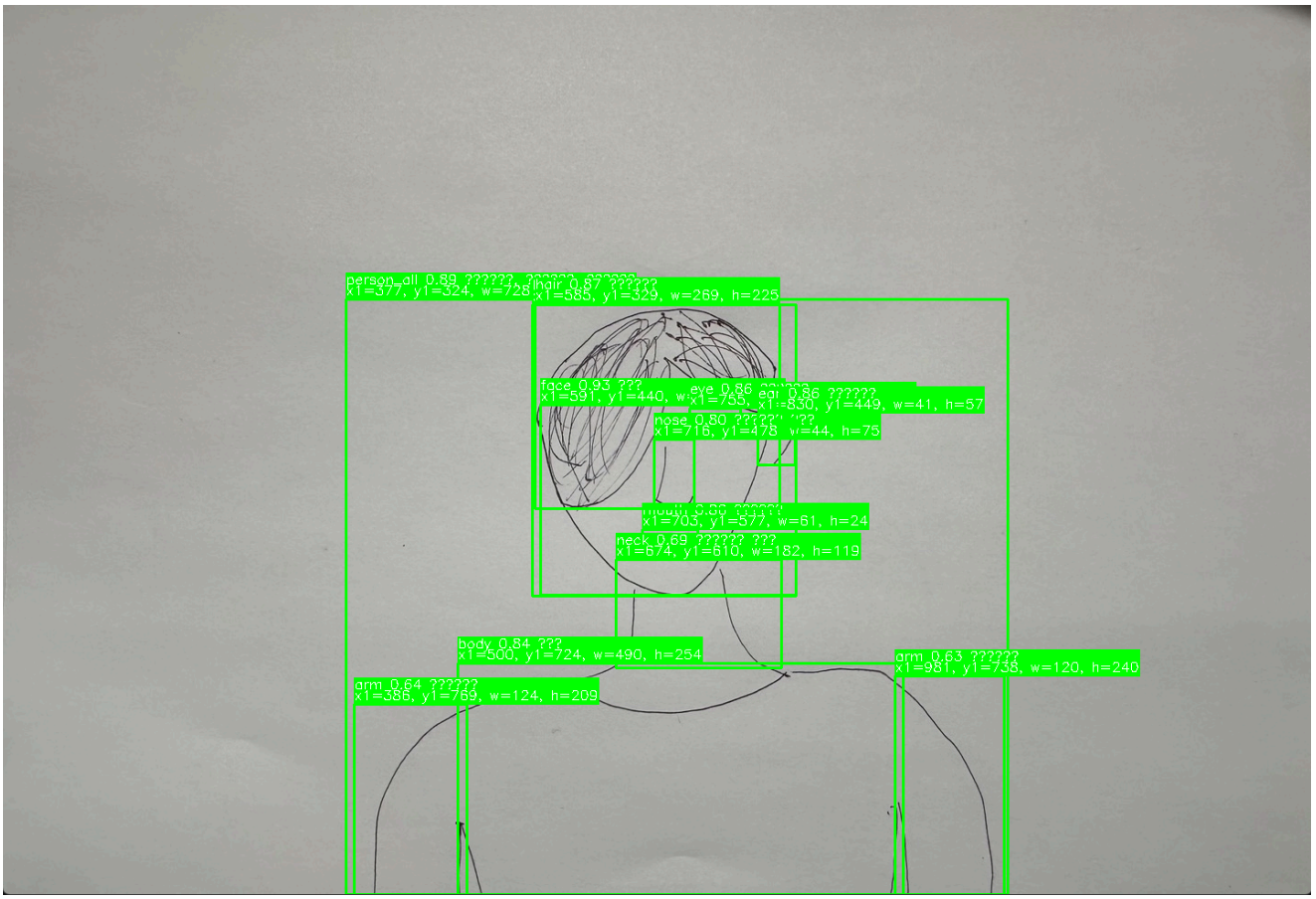
```
        cv2.rectangle(image, (x1, y1 - box_height), (x1 + box_width, y1),
(0, 255, 0), -1)

        cv2.putText(image, text1, (x1, y1 - text2_height - 5), font, scale,
(255, 255, 255), thickness)

        cv2.putText(image, text2, (x1, y1 - 5), font, scale, (255, 255,
255), thickness)



cv2.imshow("Detection Results", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

cv2.imwrite("detection_result_10.jpg", image)



########################################

# 결과 출력

########################################

print("\n\n### 내용적 분석 결과 ###")

print(f"Person_all: {person_size}, {bias_horizontal}, {bias_vertical},
{cut_off_status}")

if 'head' in detections and len(detections['head']) > 0:

    print(f"Head: 존재, {head_size_class}, {disconnection_status}")

else:

    print(f"Head: not 검출됨, {disconnection_status}")

print(f"Face: {face_status}")
```

```
for part in ['eye', 'ear', 'arm', 'hand']:
    if part in detections and len(detections[part]) > 0:
        num = len(detections[part])
        statuses = [box[7] for box in detections[part] if len(box) > 7]
        if num == 2:
            status = f"둘 존재 - {statuses}"
        elif num == 1:
            status = f"하나 생략됨 - {statuses}"
        else:
            status = "둘 생략됨"
        print(f"{part.capitalize()}s: {status}")
    else:
        print(f"{part.capitalize()}s: 둘 생략됨")


for part in ['nose', 'mouth', 'hair', 'neck', 'face', 'body']:
    if part in detections and len(detections[part]) > 0:
        status = detections[part][0][7] if len(detections[part][0]) > 7 else
"검출됨"
        print(f"{part.capitalize()}: {status}")
    else:
        print(f"{part.capitalize()}: 생략됨")
```

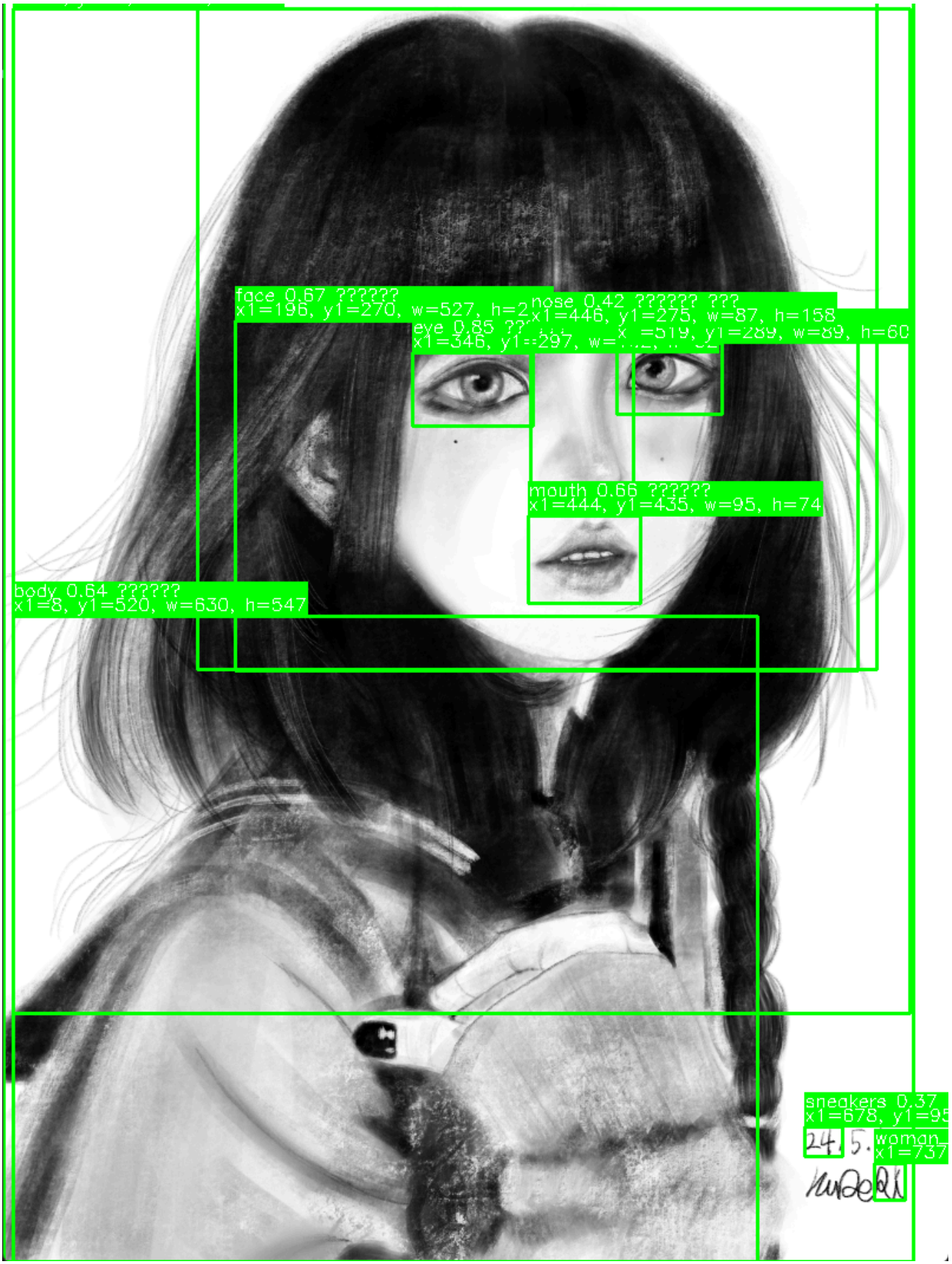
```
print("\n### 형식적 분석 결과 ###")

print(f"필압 상태: {pressure_status}, 변화: {pressure_change}")

print(f"선 분석 상태: {line_status}")

print(f"선 길이 상태: {line_length_status}")

print(f"선 형태 상태: {shape_status}")

print(f"세부 묘사 상태: {detail_status}")

print(f"움직임 상태: {movement_status}")

print(f"대칭성 상태: {symmetry_status}")

print(f"투명성 상태: {transparency_status}")

print(f"지면선 존재: {'있음' if ground_line_exists else '없음'}")
```



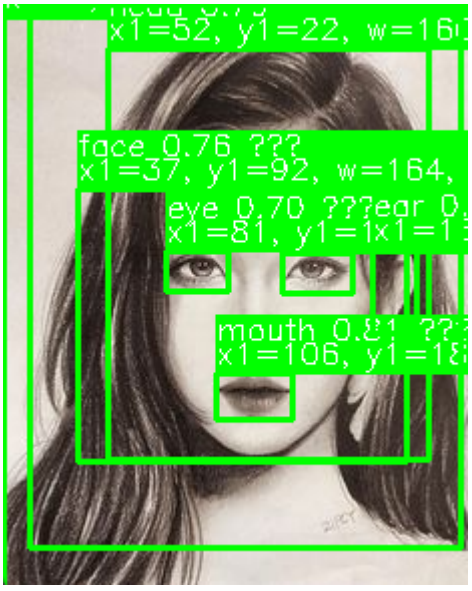

```
image 1/1 C:\capstone\yolo\0222test02.jpg: 448x640 1 사람전체, 1 머리, 1 얼굴, 2 눈s, 1 코, 1 입, 2 귀s, 1 머리카락, 1 목, 1 상체, 2 팔s, 22.4ms
Speed: 2.0ms preprocess, 22.4ms inference, 62.2ms postprocess per image at shape (1, 3, 448, 640)
### 신체 부위 분석 결과 ###
Person_all: 작음, 중앙, 하단, 절단 없음
Head: 존재, 큼, 연결됨
Face: 정면 또는 기타
Eyes: 둘 존재 - ['작음', '작음']
Ears: 둘 존재 - ['보통', '보통']
Arms: 둘 존재 - ['보통', '보통']
Hands: 둘 생략됨
Nose: 매우 큼
Mouth: 큼
Hair: 보통
Neck: 큼
Face: 큼
Body: 매우 작음

--- 추가 분석 결과 ---
필압 상태: 약한 필압, 변화: 적당한 변화
선 분석 상태: 일부분 강한 선
선 길이 상태: 짧다
선 형태 상태: 곡선적
세부 묘사 상태: 보통
움직임 상태: 움직임 부족
대칭성 상태: 대칭성이 높음
투명성 상태: 해당 없음
지면선 존재: 있음
```



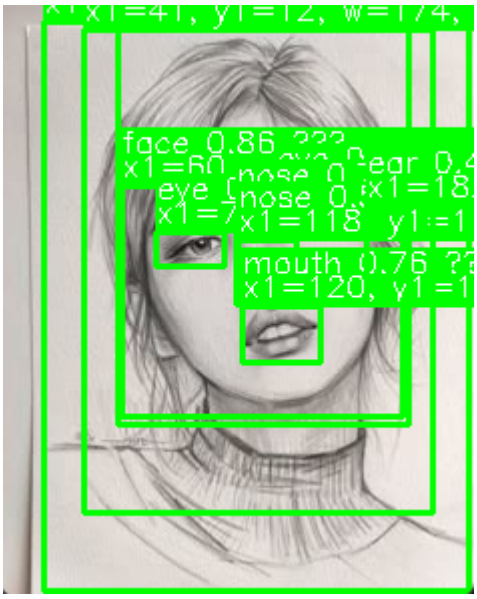

```
image 1/1 C:\capstone\yolo\0224test01.jpg: 448x640 1 사람전체, 1 머리, 1 얼굴, 1 눈, 1 코, 1 입, 1 귀, 1 머리카락, 1 목, 1 상체, 2 팔s, 22.6ms
Speed: 2.6ms preprocess, 22.6ms inference, 81.3ms postprocess per image at shape (1, 3, 448, 640)
### 신체 부위 분석 결과 ###
Person_all: 작음, 중앙, 하단, 절단 없음
Head: 존재, 작음, 연결됨
Face: 정면 또는 기타
Eyes: 하나 생략됨 - ['보통']
Ears: 하나 생략됨 - ['보통']
Arms: 둘 존재 - ['보통', '보통']
Hands: 둘 생략됨
Nose: 매우 큼
Mouth: 보통
Hair: 보통
Neck: 매우 큼
Face: 큼
Body: 큼

--- 추가 분석 결과 ---
필압 상태: 약한 필압, 변화: 적당한 변화
선 분석 상태: 일부분 강한 선
선 길이 상태: 짧다
선 형태 상태: 직선적
세부 묘사 상태: 보통
움직임 상태: 움직임 부족
대칭성 상태: 대칭성이 낮음
투명성 상태: 해당 없음
지면선 존재: 없음
```


face 0.67 ??????
x1=196, y1=270, w=527, h=2
nose 0.42 ?????? ???
x1=446, y1=275, w=87, h=158
eye 0.85 ??
x1=346, y1=297, w=
x =519, y1=289, w=89, h=60
mouth 0.66 ??????
x1=444, y1=435, w=95, h=74
body 0.64 ??????
x1=8, y1=520, w=630, h=547
sneakers 0.37
x1=678, y1=95
24
5.
woman
x1=737

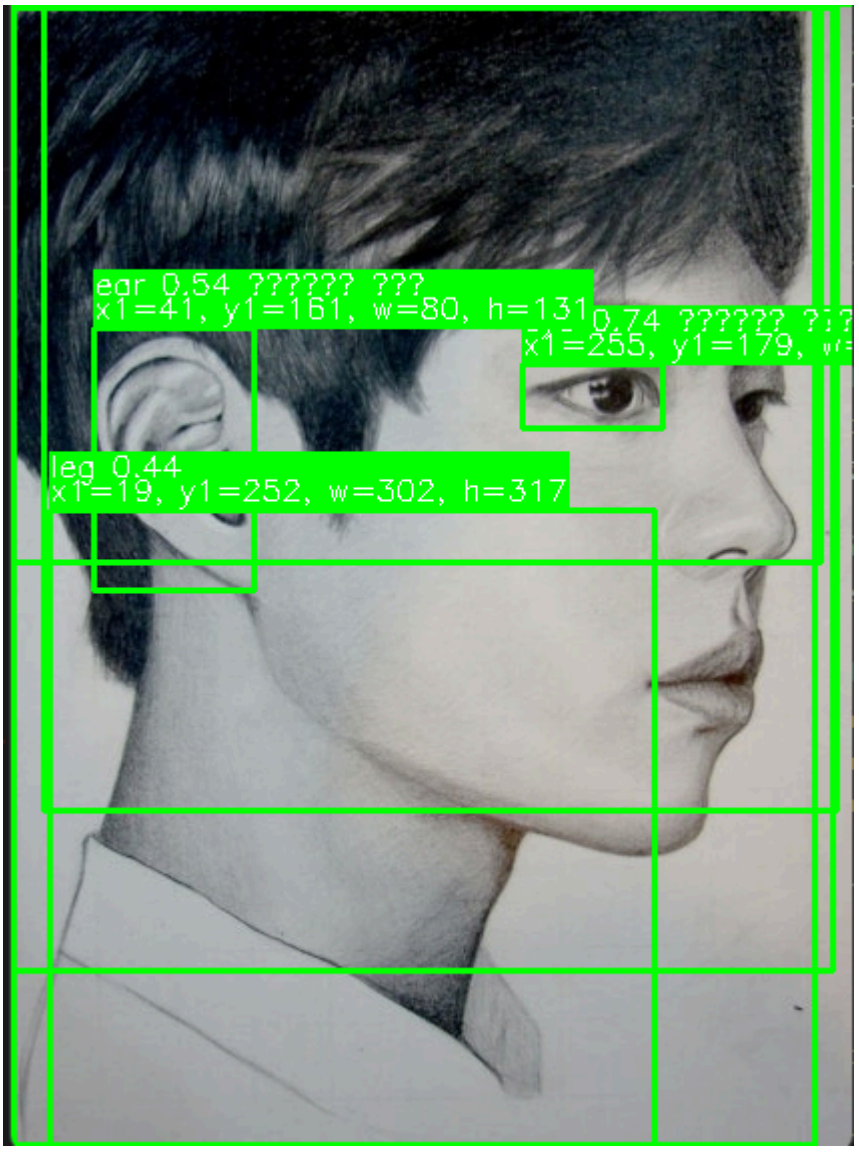
```
image 1/1 C:\capstone\yolo\0227test01.jpg: 640x480 1 사람전체, 1 머리, 1 얼굴, 2 눈s, 1 코, 1 입, 1 머리카락, 1 상체, 1 운동화, 1 여자구두두, 23.1ms
Speed: 3.2ms preprocess, 23.1ms inference, 69.3ms postprocess per image at shape (1, 3, 640, 480)


### 내용적 분석 결과 ###
Person_all: 큼, 수평.중앙, 수직.과도한 정중앙, 상 절단, 좌 절단
Head: 존재, 큼, 연결됨
Face: 정면 또는 기타
Eyes: 둘 존재 - ['작음', '작음']
Ears: 둘 생략됨
Arms: 둘 생략됨
Hands: 둘 생략됨
Nose: 매우 큼
Mouth: 보통
Hair: 큼
Neck: 생략됨
Face: 보통
Body: 보통

### 형식적 분석 결과 ###
필압 상태: 강한 필압, 필압 변화: 적당한 변화
선 분석 상태: 일부분 강한 선
선 길이 상태: 짧다
선 모양 상태: 곡선적
세부 묘사 상태: 과도한 세부 묘사
움직임 상태: 움직임 부족
대칭성 상태: 대칭성이 낮음
[illegible]
지면선 존재: 있음
```

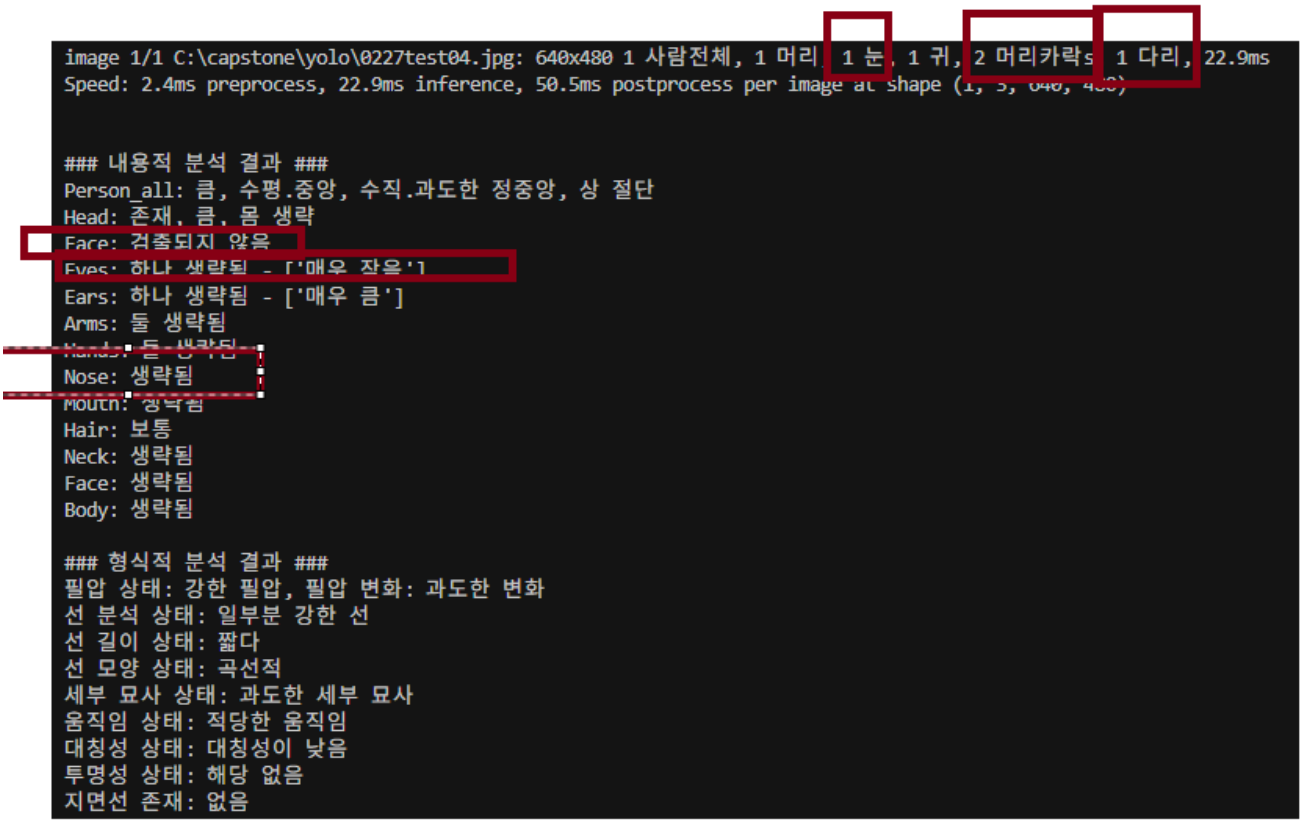



```
image 1/1 C:\capstone\yolo\0227test02.jpg: 640x544 1 사람전체, 1 머리, 1 얼굴, 2 눈s, 1 입, 1 귀, 1 머리카락, 21.9ms
Speed: 2.4ms preprocess, 21.9ms inference, 56.1ms postprocess per image at shape (1, 3, 640, 544)


### 내용적 분석 결과 ###
Person_all: 큼, 수평.중앙, 수직.과도한 정중앙, 좌 절단, 우 절단
Head: 존재, 큼, 몸 생략
Face: 정면 또는 기타
Eyes: 둘 존재 - ['보통', '보통']
Ears: 하나 생략됨 - ['보통']
Arms: 둘 생략됨
Hands: 둘 생략됨
Nose: 생략됨
Mouth: 보통
Hair: 큼
Neck: 생략됨
Face: 큼
Body: 생략됨

### 형식적 분석 결과 ###
필압 상태: 강한 필압, 필압 변화: 과도한 변화
선 분석 상태: 일부분 강한 선
선 길이 상태: 짧다
선 모양 상태: 곡선적
세부 묘사 상태: 과도한 세부 묘사
움직임 상태: 움직임 부족
대칭성 상태: 대칭성이 낮음
투명성 상태: 해당 없음
지면선 존재: 없음
```



```
image 1/1 C:\capstone\yolo\0227test03.jpg: 640x512 1 사람전체, 1 머리, 1 얼굴, 2 눈, 2 코, 1 입, 1 귀, 1 머리카락, 34.4ms
Speed: 2.2ms preprocess, 34.4ms inference, 61.1ms postprocess per image at shape (1, 3, 640, 512)


### 내용적 분석 결과 ###
Person_all: 큼, 수평.중앙, 수직.과도한 정중앙, 절단 없음
Head: 존재, 큼, 몸 생략
Face: 정면 또는 기타
Eyes: 둘 존재 - ['작음', '보통']
Ears: 하나 생략됨 - ['큼']
Arms: 둘 생략됨
Hands: 둘 생략됨
Nose: 매우 큼
Mouth: 큼
Hair: 큼
Neck: 생략됨
Face: 큼
Body: 생략됨

### 형식적 분석 결과 ###
필압 상태: 강한 필압, 필압 변화: 적당한 변화
선 분석 상태: 일부분 강한 선
선 길이 상태: 짧다
선 모양 상태: 곡선적
세부 묘사 상태: 과도한 세부 묘사
움직임 상태: 움직임 부족
대칭성 상태: 대칭성이 낮음
투명성 상태: 해당 없음
지면선 존재: 없음
```


ear 0.54 ????? ???
x1=41, y1=161, w=80, h=131
0.74 ????? ?
x1=255, y1=179,
leg 0.44
x1=19, y1=252, w=302, h=317



남은 과제

- **형식적 분석**
  - **특정 부분의 세부 묘사**
  - **왜곡 및 생략**: 일반적인 왜곡, 극단적인 왜곡.
  - **불필요한 내용 추가**: 불필요한 요소 검출.
- **내용적 분석**
  - **머리**: 모자(가림).
  - **얼굴**: 옆모습.
  - **눈**: 가림(옆모습인가 vs 가려졌는가), 윤곽만 묘사, 진한 눈동자, 눈썹.
  - **코**: 코 모양.
  - **귀**: 귀걸이 착용.
  - **입**: 담배/파이프 사용, 일직선, 비웃음, 웃음, 벌림.
  - **머리카락**: 많은 숱, 적은 숱, 세부 묘사.
  - **어깨**: 각진.
  - **팔**: 팔짱.
  - **손**: 원모양, 뾰족모양.
- **점수 및 시각화**

남은 과제 기능(ex. 모자, 귀걸이, 입 모양 등)은 yolo모델에 추가 클래스 학습이 필요하거나 복잡한 이미지 처리가 요구됨: yolo 모델 확장(라벨데이터와 그림데이터 추가 수집 필요) 또는 CV 기법(ex.

텍스처 분석, 형태 분석)을 통해 구현 -> 추가 공부 중