Key Points 탐지

최석재 *lingua@naver.com*

키 포인트 탐지

- 키 포인트 탐지는 사람의 신체 구조와 자세를 추정하는 기술
- 포즈 추정(Pose Estimation) 기술이라고도 한다

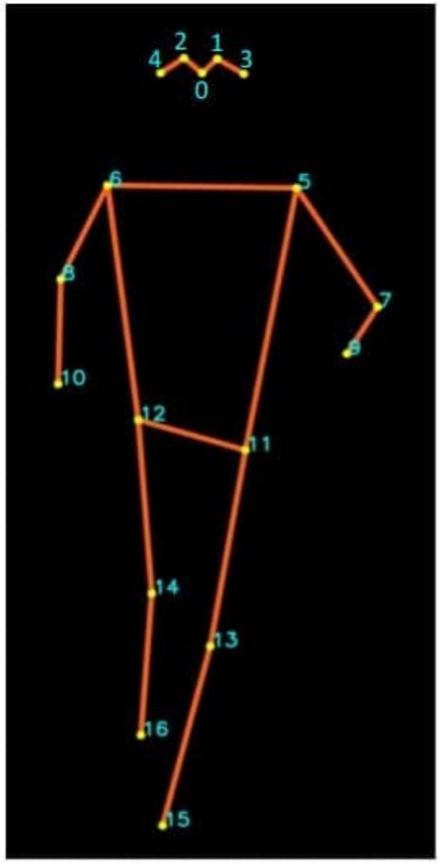
- Ultralytics는 객체 탐지의 경우와 같이 모델의 크기에 따라
- yolov8n-pose, yolov8s-pose, yolov8m-pose, yolov8l-pose, yolov8x-pose의
- 다섯 가지 포즈 추정 모델을 발표하였다

- 기본적인 사용방법은 객체 탐지의 경우와 유사하다

Ultralytics Key Points

- Ultralytics 포즈 추정 모델은 다음과 같은 16개의 키 포인트를 탐지한다

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



Colab에서 YOLO v8 사용하기

- Colab에서의 사용 방법으로 알아본다

구글 드라이브와 연결하기

- from google.colab import drive
- drive.mount('/content/gdrive')

토치비전 설치

- !python -m pip install torch torchvision

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: torch in /usr/local/lib/python3.9/dist-packages (2.0.0+cu118)
Requirement already satisfied: torchvision in /usr/local/lib/python3.9/dist-packages (0.15.1+cu118)
Requirement already satisfied: sympy in /usr/local/lib/python3.9/dist-packages (from torch) (1.11.1)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.9/dist-packages (from torch) (2.0.0)
Requirement already satisfied: networkx in /usr/local/lib/python3.9/dist-packages (from torch) (3.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.9/dist-packages (from torch) (4.5.0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.9/dist-packages (from torch) (3.1.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from torch) (3.11.0)
Requirement already satisfied: lit in /usr/local/lib/python3.9/dist-packages (from triton==2.0.0->torch) (16.0.1)
Requirement already satisfied: cmake in /usr/local/lib/python3.9/dist-packages (from triton==2.0.0->torch) (3.25.2)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from torchvision) (2.27.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.9/dist-packages (from torchvision) (8.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from torchvision) (1.22.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from jinja2->torch) (2.1.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (2.0.12)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (1.26.15)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->torchvision) (3.4)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.9/dist-packages (from sympy->torch) (1.3.0)
```

ultralytics 설치

- !pip install ultralytics

```
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (4.41.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2022.7.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2023.7.22)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (3.12.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (4.7.1)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (1.11.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.7.0->ultralytics) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.7.0->ultralytics) (3.25.2)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.7.0->ultralytics) (16.0.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.2.2->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.7.0->ultralytics) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.7.0->ultralytics) (1.3.0)
Installing collected packages: ultralytics
Successfully installed ultralytics-8.0.145
```

모델 저장 경로 설정

- %cd /content/gdrive/MyDrive/pytest_img/YOLO/

```
/content/gdrive/MyDrive/pytest_img/YOLO
```

경로 확인

- !ls # sample.mp4 파일이 있으면 정상

모델 다운로드

- from ultralytics import YOLO

- model = YOLO("../models/yolov8m-pose.pt")

경로 설정

- seconds = 1 # 동영상을 읽을 시간을 초 단위로 설정

- video_path = '/content/gdrive/MyDrive/pytest_img/YOLO/walking.mp4' # video path
- output_dir = '/content/gdrive/MyDrive/pytest_img/YOLO/results' # 결과 저장 폴더

예측 함수 정의

- 포즈 추정을 하기 위해서는 사람의 위치를 찾아야 하므로
- 객체 탐지에서 사용했던 예측 함수가 필요하다. 바운딩 박스와 함께 키포인트 탐지 분석도 수행한다

- import torch
- def predict(frame, iou=0.7, conf=0.25):

```
    results = model(
        source = frame,
        device = "0" if torch.cuda.is_available() else "cpu",
        iou = iou,                  # 바운딩 박스 필터링 신뢰도 기준
        conf = conf,                # 모델이 탐지한 객체에 대한 최소 신뢰도 기준
        verbose = False,            # 추가 정보 출력 여부
    )
    result = results[0]             # 첫 번째 프레임의 이미지 (현재 1개씩의 프레임만 전달됨)
    return result
```

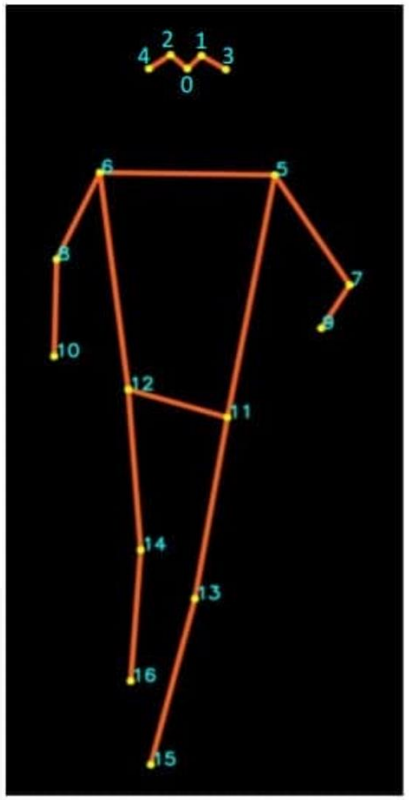
iou: Intersection Over Union
예측된 경계 상자와 실제 경계 상자 사이의 겹치는 부분을 의미

키 포인트 예측 함수 정의 (1/3)

- import cv2
- import numpy as np

- def draw_keypoints(result, frame):

```
    # 신체 부위를 따라 키 포인트를 연결하는 리스트
    connections = [
        ([4, 2, 0, 1, 3], (0, 255, 0)),          # 얼굴 부위. 초록
        ([10, 8, 6, 5, 7, 9], (255, 0, 0)),      # 두 팔. 파랑
        ([6, 12, 11, 5], (255, 0, 255)),         # 몸통. 보라
        ([12, 14, 16], (0, 165, 255)),           # 오른 다리. 주홍
        ([11, 13, 15], (0, 165, 255))            # 왼 다리. 주홍
    ]
```




```
def draw_keypoints(result, frame):
    # 신체 부위를 따라 키 포인트를 연결하는 리스트
    connections = [
        ([4, 2, 0, 1, 3], (0, 255, 0)),      # 초록
        ([10, 8, 6, 5, 7, 9], (255, 0, 0)),  # 파랑
        ([6, 12, 11, 5], (255, 0, 255)),     # 보라
        ([12, 14, 16], (0, 165, 255)),       # 주홍
        ([11, 13, 15], (0, 165, 255))        # 주홍
    ]

    for kps in result.keypoints:
        kps = kps.data.squeeze()
        nkps = kps.cpu().numpy()

        for group, color in connections:
            for i in range(len(group) - 1):
                idx1, idx2 = group[i], group[i + 1]
                x1, y1, score1 = nkps[idx1]
                x2, y2, score2 = nkps[idx2]

                if score1 > 0.5 and score2 > 0.5:
                    point1 = (int(x1), int(y1))
                    point2 = (int(x2), int(y2))

                    cv2.circle(frame, point1, 3, (0, 0, 255), cv2.FILLED)
                    cv2.putText(frame, str(idx1), point1, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

                    cv2.circle(frame, point2, 3, (0, 0, 255), cv2.FILLED)
                    cv2.putText(frame, str(idx2), point2, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

                    cv2.line(frame, point1, point2, color, 2)

    return frame
```

키 포인트 예측 함수 정의 (2/3)

```
for kps in result.keypoints:                    # predict()가 분석한 키 포인트의 원소를 순회
    kps = kps.data.squeeze()                    # 크기가 1인 불필요한 차원 제거
    nkps = kps.cpu().numpy()                    # 넘파이 배열로 변환

    for group, color in connections:
        for i in range(len(group) - 1):
            idx1, idx2 = group[i], group[i + 1] # 키 포인트와 이어야 하는 인접 키 포인트
            x1, y1, score1 = nkps[idx1]         # x좌표, y좌표, 신뢰도 추출
            x2, y2, score2 = nkps[idx2]         # x좌표, y좌표, 신뢰도 추출
```

```python
def draw_keypoints(result, frame):
    # 신체 부위를 따라 키 포인트를 연결하는 리스트
    connections = [
        ([4, 2, 0, 1, 3], (0, 255, 0)),      # 초록
        ([10, 8, 6, 5, 7, 9], (255, 0, 0)),  # 파랑
        ([6, 12, 11, 5], (255, 0, 255)),     # 보라
        ([12, 14, 16], (0, 165, 255)),       # 주홍
        ([11, 13, 15], (0, 165, 255))        # 주홍
    ]
    for kps in result.keypoints:
        kps = kps.data.squeeze()
        nkps = kps.cpu().numpy()

        for group, color in connections:
            for i in range(len(group) - 1):
                idx1, idx2 = group[i], group[i + 1]
                x1, y1, score1 = nkps[idx1]
                x2, y2, score2 = nkps[idx2]

                if score1 > 0.5 and score2 > 0.5:
                    point1 = (int(x1), int(y1))
                    point2 = (int(x2), int(y2))

                    cv2.circle(frame, point1, 3, (0, 0, 255), cv2.FILLED)
                    cv2.putText(frame, str(idx1), point1, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

                    cv2.circle(frame, point2, 3, (0, 0, 255), cv2.FILLED)
                    cv2.putText(frame, str(idx2), point2, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

                    cv2.line(frame, point1, point2, color, 2)

    return frame
```

크기가 1인 차원은 해당 원소가 하나밖에 없는 것으로
구조를 단순화를 위해 해당 차원을 제거하여 가독성을 높인다
예) (1, 255, 255, 1) → (255, 255) 로 변환

kps 내용

- kps의 내용을 출력하면 다음과 같다

```
conf: tensor([[0.9350, 0.9083, 0.6095, 0.9356, 0.4266, 0.9984, 0.9855, 0.9952, 0.9063, 0.9856, 0.8625, 0.9982, 0.994
data: tensor([[[2.9856e+02, 4.4462e+02, 9.3495e-01],
        [3.1446e+02, 4.3530e+02, 9.0834e-01],
        [3.0225e+02, 4.3261e+02, 6.0954e-01],
        [3.4813e+02, 4.5198e+02, 9.3560e-01],
        [0.0000e+00, 0.0000e+00, 4.2664e-01],
        [3.1024e+02, 5.2730e+02, 9.9840e-01],
        [3.6586e+02, 5.1904e+02, 9.8554e-01],
        [2.5419e+02, 6.1045e+02, 9.9518e-01],
        [3.9133e+02, 6.2386e+02, 9.0628e-01],
        [2.4071e+02, 6.9154e+02, 9.8558e-01],
        [4.0072e+02, 7.1518e+02, 8.6253e-01],
        [3.0388e+02, 6.9715e+02, 9.9819e-01],
        [3.4122e+02, 6.9130e+02, 9.9463e-01],
        [2.9238e+02, 8.3021e+02, 9.9461e-01],
        [3.3795e+02, 8.2296e+02, 9.8474e-01],
        [3.0380e+02, 9.5142e+02, 9.6207e-01],
        [3.1092e+02, 9.3780e+02, 9.3197e-01]]])
```

각 키 포인트의 신뢰도

각 키 포인트의 (x, y) 좌표와 신뢰도

nkps의 내용

- kps data의 차원을 정리한 넘파이 배열 결과는 아래와 같다

```
nkps: [[     298.56      444.62     0.93495]
 [     314.46       435.3     0.90834]
 [     302.25      432.61     0.60954]
 [     348.13      451.98      0.9356]
 [          0           0     0.42664]
 [     310.24       527.3      0.9984]
 [     365.86      519.04     0.98554]
 [     254.19      610.45     0.99518]
 [     391.33      623.86     0.90628]
 [     240.71      691.54     0.98558]
 [     400.72      715.18     0.86253]
 [     303.88      697.15     0.99819]
 [     341.22       691.3     0.99463]
 [     292.38      830.21     0.99461]
 [     337.95      822.96     0.98474]
 [      303.8      951.42     0.96207]
 [     310.92       937.8     0.93197]]
```

키 포인트 예측 함수 정의 (3/3)

```
if score1 > 0.5 and score2 > 0.5:                # 두 키 포인트의 신뢰도가 0.5보다 큰 경우에만
    point1 = (int(x1), int(y1))                  # 키 포인트
    point2 = (int(x2), int(y2))                  # 인접한 키 포인트

    cv2.circle(frame, point1, 3, (0, 0, 255), cv2.FILLED)    # 각 키 포인트를 빨간색 원으로 그림
    cv2.putText(frame, str(idx1), point1, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)
                                                             # 각 키 포인트를 빨간색 텍스트로 표시
    cv2.circle(frame, point2, 3, (0, 0, 255), cv2.FILLED)    # 인접 키 포인트
    cv2.putText(frame, str(idx2), point2, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

    cv2.line(frame, point1, point2, color, 2)                # 두 키 포인트를 선으로 연결

return frame                                     # 키 포인트와 연결된 선이 그려진 frame 반환
```

키 포인트 예측 함수 정의 (3/3)

```python
def draw_keypoints(result, frame):
    # 신체 부위를 따라 키 포인트를 연결하는 리스트
    connections = [
        ([4, 2, 0, 1, 3], (0, 255, 0)),      # 초록
        ([10, 8, 6, 5, 7, 9], (255, 0, 0)),  # 파랑
        ([6, 12, 11, 5], (255, 0, 255)),     # 보라
        ([12, 14, 16], (0, 165, 255)),       # 주황
        ([11, 13, 15], (0, 165, 255))        # 주황
    ]

    for kps in result.keypoints:
        kps = kps.data.squeeze()
        nkps = kps.cpu().numpy()

        for group, color in connections:
            for i in range(len(group) - 1):
                idx1, idx2 = group[i], group[i + 1]
                x1, y1, score1 = nkps[idx1]
                x2, y2, score2 = nkps[idx2]

                if score1 > 0.5 and score2 > 0.5:
                    point1 = (int(x1), int(y1))
                    point2 = (int(x2), int(y2))

                    cv2.circle(frame, point1, 3, (0, 0, 255), cv2.FILLED)
                    cv2.putText(frame, str(idx1), point1, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

                    cv2.circle(frame, point2, 3, (0, 0, 255), cv2.FILLED)
                    cv2.putText(frame, str(idx2), point2, cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)

                    cv2.line(frame, point1, point2, color, 2)

    return frame
```

출력 프레임 계산

- import cv2

- capture = cv2.VideoCapture(video_path)
- if capture.get(cv2.CAP_PROP_POS_FRAMES) == capture.get(cv2.CAP_PROP_FRAME_COUNT):
  capture.set(cv2.CAP_PROP_POS_FRAMES, 0)

현재 비디오의 위치가 비디오의 전체 프레임 수와 같은지,
즉, 비디오의 마지막 프레임에 도달했는지 검사

마지막 프레임에 도달했다면,
프레임의 위치를 비디오의 시작점으로 재설정하여 다시 재생할 준비를 함

- fps = capture.get(cv2.CAP_PROP_FPS) # 초당 프레임 수 얻기
- total_frames = int(fps * seconds) # seconds 시간 동안의 프레임 수
- print('total 프레임:', total_frames) # 출력하게 되는 전체 프레임 수. 25

실행 (1/2)

- import os

- if not os.path.exists(output_dir):
    os.makedirs(output_dir)

- frame_count = 0
- while frame_count < total_frames:
    ok, frame = capture.read()
    if not ok:
        print("프레임 읽기에 실패했습니다. 종료.")
        break

실행 (2/2)

while 문 안에 있음

```
result = predict(frame)                          # 바운딩 박스와 키포인트 분석
results = draw_keypoints(result, frame)          # 분석된 내용으로 키포인트 연결

output_path = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
cv2.imwrite(output_path, results)
print("Saved to:", output_path)

frame_count += 1
key = cv2.waitKey(10)
if key == ord('q'):
    print("사용자가 종료를 요청했습니다.")
    break
```

- capture.release()
- cv2.destroyAllWindows()

```
import os

if not os.path.exists(output_dir):
    os.makedirs(output_dir)

frame_count = 0
while frame_count < total_frames:
    ok, frame = capture.read()
    if not ok:
        print("프레임 읽기에 실패했습니다. 종료.")
        break

    result = predict(frame)
    results = draw_keypoints(result, frame)

    output_path = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
    cv2.imwrite(output_path, results)
    print("Saved to:", output_path)

    frame_count += 1

    key = cv2.waitKey(10)
    if key == ord('q'):
        print("사용자가 종료를 요청했습니다.")
        break

capture.release()
cv2.destroyAllWindows()
```

분석 결과 확인

- 앞에서 지정한 결과 저장 경로를 확인한다
- 전반적으로 잘 인식되었다