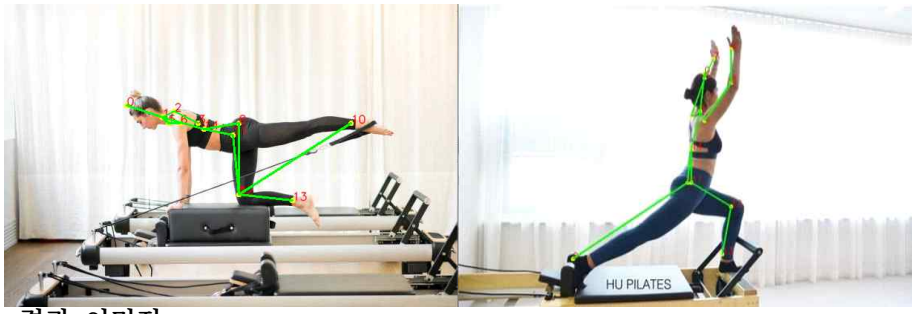
9월 5주차 - A#
(2024.09.30.~2024.10.06.)

1. openpose를 통한 영상 처리

```
import cv2

# MPII에서 각 파트 번호, 선으로 연결될 POSE_PAIRS
BODY_PARTS = { "Head": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
               "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
               "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "Chest": 14,
               "Background": 15 }

POSE_PAIRS = [ ["Head", "Neck"], ["Neck", "RShoulder"], ["RShoulder", "RElbow"],
               ["RElbow", "RWrist"], ["Neck", "LShoulder"], ["LShoulder", "LElbow"],
               ["LElbow", "LWrist"], ["Neck", "Chest"], ["Chest", "RHip"], ["RHip", "RKnee"],
               ["RKnee", "RAnkle"], ["Chest", "LHip"], ["LHip", "LKnee"], ["LKnee", "LAnkle"] ]

# 각 파일 path
protoFile = "pose_deploy_linevec_faster_4_stages.prototxt"
weightsFile = "pose_iter_160000.caffemodel"

# 위의 path에 있는 network 불러오기
net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)

# 이미지 읽어오기
image = cv2.imread("gg.jpeg")
# capture = cv2.VideoCapture(0)

# frame.shape = 불러온 이미지에서 height, width, color 받아옴
imageHeight, imageWidth, _ = image.shape

# network에 넣기위해 전처리
inpBlob = cv2.dnn.blobFromImage(image, 1.0 / 255, (imageWidth, imageHeight), (0, 0, 0), swapRB=False, crop=False)

# network에 넣어주기
net.setInput(inpBlob)

# 결과 받아오기
output = net.forward()

# output.shape[0] = 이미지 ID, [1] = 출력 맵의 높이, [2] = 너비
H = output.shape[2]
W = output.shape[3]
print("이미지 ID : ", len(output[0]), ", H : ", output.shape[2], ", W : ",output.shape[3]) # 이미지 ID

# 키포인트 검출시 이미지에 그려줌
points = []
for i in range(0,15):
    # 해당 신체부위 신뢰도 얻음.
    probMap = output[0, i, :, :]

    # global 최대값 찾기
    minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)

    # 원래 이미지에 맞게 점 위치 변경
    x = (imageWidth * point[0]) / W
    y = (imageHeight * point[1]) / H

    # 키포인트 검출한 결과가 0.1보다 크면(검출한곳이 위 BODY_PARTS랑 맞는 부위면) points에 추가, 검출했는데 부위가 없으면 None으로
    if prob > 0.1 :
        cv2.circle(image, (int(x), int(y)), 3, (0, 255, 255), thickness=-1, lineType=cv2.FILLED)       # circle(그릴곳, 원의 중심, 반지름, 색)
        cv2.putText(image, "{}".format(i), (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, lineType=cv2.LINE_AA)
        points.append((int(x), int(y)))
    else :
        points.append(None)

cv2.imshow("Output-Keypoints",image)
cv2.waitKey(0)
# 이미지 복사
imageCopy = image

# 각 POSE_PAIRS별로 선 그어줌 (머리 - 목, 목 - 왼쪽어깨, ...)
for pair in POSE_PAIRS:
    partA = pair[0]             # Head
    partA = BODY_PARTS[partA]   # 0
    partB = pair[1]             # Neck
    partB = BODY_PARTS[partB]   # 1

    #print(partA," 와 ", partB, " 연결₩n")
    if points[partA] and points[partB]:
        cv2.line(imageCopy, points[partA], points[partB], (0, 255, 0), 2)


cv2.imshow("Output-Keypoints",imageCopy)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

*사진 분석을 위해 사용된 코드



***결과 이미지**

우측의 사진은 관절 부위를 잘 연결했으나, 좌측의 사진은 잘 연결되지 않음
(자세에 따라 정확도 차이가 있음)

```
import cv2
from pathlib import Path

# MPII에서 각 파트 번호, 선으로 연결될 POSE_PAIRS
BODY_PARTS = { "Head": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
               "LShoulder": 5, "LElbow": 6, "LWrist": 7, "RHip": 8, "RKnee": 9,
               "RAnkle": 10, "LHip": 11, "LKnee": 12, "LAnkle": 13, "Chest": 14,
               "Background": 15 }

POSE_PAIRS = [ ["Head", "Neck"], ["Neck", "RShoulder"], ["RShoulder", "RElbow"],
               ["RElbow", "RWrist"], ["Neck", "LShoulder"], ["LShoulder", "LElbow"],
               ["LElbow", "LWrist"], ["Neck", "Chest"], ["Chest", "RHip"], ["RHip", "RKnee"],
               ["RKnee", "RAnkle"], ["Chest", "LHip"], ["LHip", "LKnee"], ["LKnee", "LAnkle"] ]

# 각 파일 path

protoFile = "pose_deploy_linevec_faster_4_stages.prototxt"
weightsFile = "pose_iter_160000.caffemodel"

# 위의 path에 있는 network 모델 불러오기
net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)

#쿠다 사용 안하면 밑에 이미지 크기를 줄이는게 나을 것이다
# net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA) #백엔드로 쿠다를 사용하여 속도향상을 꾀한다
# net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA) # 쿠다 디바이스에 계산 요청


###카메라랑 연결...?
capture = cv2.VideoCapture(0) #카메라 정보 받아옴
# capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640) #카메라 속성 설정
# capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480) # width:너비, height: 높이

inputWidth=320;
inputHeight=240;
inputScale=1.0/255;


#반복문을 통해 카메라에서 프레임을 지속적으로 받아옴
while cv2.waitKey(1) <0:  #아무 키나 누르면 끝난다.
    #웹캠으로부터 영상 가져옴
    hasFrame, frame = capture.read()

    #영상이 커서 느리면 사이즈를 줄이자
    #frame=cv2.resize(frame,dsize=(320,240),interpolation=cv2.INTER_AREA)

    #웹캠으로부터 영상을 가져올 수 없으면 웹캠 중지
    if not hasFrame:
        cv2.waitKey()
        break

    #
    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]

    inpBlob = cv2.dnn.blobFromImage(frame, inputScale, (inputWidth, inputHeight), (0, 0, 0), swapRB=False, crop=False)

    imgb=cv2.dnn.imagesFromBlob(inpBlob)
    #cv2.imshow("motion",(imgb[0]*255.0).astype(np.uint8))

    # network에 넣어주기
    net.setInput(inpBlob)

    # 결과 받아오기
    output = net.forward()
```

```
    # 키포인트 검출시 이미지에 그려줌
    points = []
    for i in range(0,15):
        # 해당 신체부위 신뢰도 얻음.
        probMap = output[0, i, :, :]

        # global 최대값 찾기
        minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)

        # 원래 이미지에 맞게 점 위치 변경
        x = (frameWidth * point[0]) / output.shape[3]
        y = (frameHeight * point[1]) / output.shape[2]

        # 키포인트 검출한 결과가 0.1보다 크면(검출한곳이 위 BODY_PARTS랑 맞는 부위면) points에 추가, 검출했는데 부위가 없으면 None으로
        if prob > 0.1 :
            cv2.circle(frame, (int(x), int(y)), 3, (0, 255, 255), thickness=-1, lineType=cv2.FILLED) # circle(그릴곳, 원의 중심, 반지름, 색)
            cv2.putText(frame, "{}".format(i), (int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, lineType=cv2.LINE_AA)
            points.append((int(x), int(y)))
        else :
            points.append(None)


    # 각 POSE_PAIRS별로 선 그어줌 (머리 - 목, 목 - 왼쪽어깨, ...)
    for pair in POSE_PAIRS:
        partA = pair[0]             # Head
        partA = BODY_PARTS[partA]   # 0
        partB = pair[1]             # Neck
        partB = BODY_PARTS[partB]   # 1

        #partA와 partB 사이에 선을 그어줌 (cv2.line)
        if points[partA] and points[partB]:
            cv2.line(frame, points[partA], points[partB], (0, 255, 0), 2)


    cv2.imshow("Output-Keypoints",frame)

capture.release()  #카메라 장치에서 받아온 메모리 해제
cv2.destroyAllWindows() #모든 윈도우 창 닫음
```

***노트북 캠을 활용한 실시간 자세 분석 코드**

***결과 이미지**

실시간 영상 처리의 경우 정확도가 현저히 떨어지며, 코드 및 모델의 최적화가 안된 문제도 있긴 하나, 속도가 매우 느림

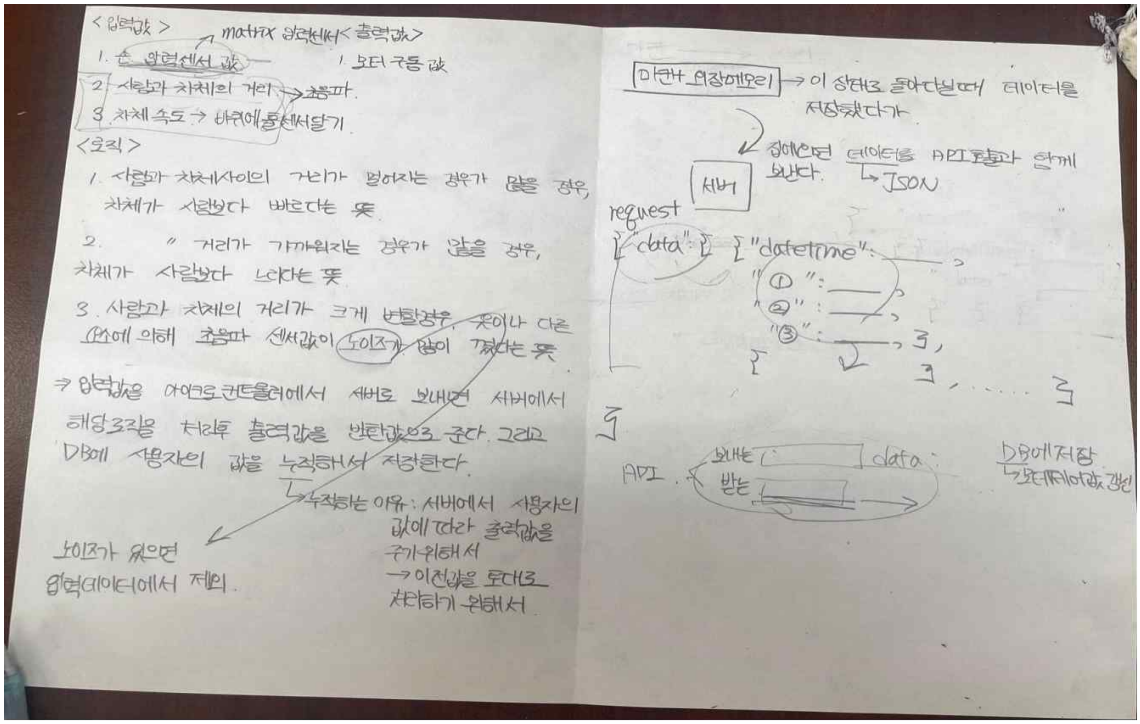
결론 : openpose를 사용한 자율 주행은 불가능할 것으로 판단

2. 대체 방안

초음파센서 값을 받아서 속도 값을 조정하는 반응형 알고리즘

입력값을 마이크로컨트롤러를 통해 받고, 외장 메모리에 저장했다가 충전할 때나, 동기화하고 싶을 때 인터넷이 연결된 상태에서 서버로 데이터를 전송

서버에서 데이터를 DB에 저장하고 아래 사진에 나와 있는 로직을 통해 변경될 모터 구동 값을 마이크로 컨트롤러로 전송



초음파센서 여러 개를 사용하여 값을 측정한다면, 정확도 향상에 도움이 될 것으로 예상

3. 진행상황

- 보조 보행기 샘플 구매
- 학교 내 창업지원단 창업동아리 지원(합격)

2024년 스타트업 칼리지 창업동아리(2차) 선정결과 안내

「2024년 스타트업 칼리지 창업동아리(2차)」에 신청해주신 모든 분들께 감사드리며, 선정 팀 및 향후 일정을 아래와 같이 안내드립니다.

□ **활동기간** : 2024.10.11.(금) ~ 2024.02.28.(금)

□ **지원내역**

○ **창업활동자금** *팀별 지원금은 협약식때 안내예정 / 개별면담 이후 사용가능

○ **창업활동공간** *개별면담 이후 사용가능

○ **멘토링 및 기타 창업에 필요한 행정지원**

○ **워크숍 및 네트워킹 데이 등 창업동아리 행사**

□ **선 정 팀**

연번	팀명	팀장명
1	소미화방	구*승
2	newplex	변*승
3	A#	이*림
4	맞스터치	조*훈
5	New Clear	김*현
6	스프링	김*민
7	방방방	박*준
8	사요나라	최*민
9	응잘보이주	최*진
10	재미있게 가자	정*진
11	생생여행	김*민

4. 향후 계획

- 추가적인 시장 조사 (인터뷰 50명 계획 중)
- 창업동아리의 멘토링 지원을 통해 아이디어 구현 구체화