DLIP Final Team Project

Gait Analysis using Openpose based VGG-19

Date: ***2023-06-19***

Author: ***21700435 Yang Taekyun***, ***21800198 Kim Haewon***, ***22000532 Lee Seungjae***

Github: DLIP_Project_GaitAnalysis_2023

Demo Video: Gait Analysis

Introduction

This project is a lab of how to analyze gait using python openpose. The reason for choosing gait analysis is that it is inexpensive, and if the analysis proceeds successfully, we can determine whether it is normal or pathological walking.

Requirement

Hardware

- NVDIA graphic cards

Software Installation

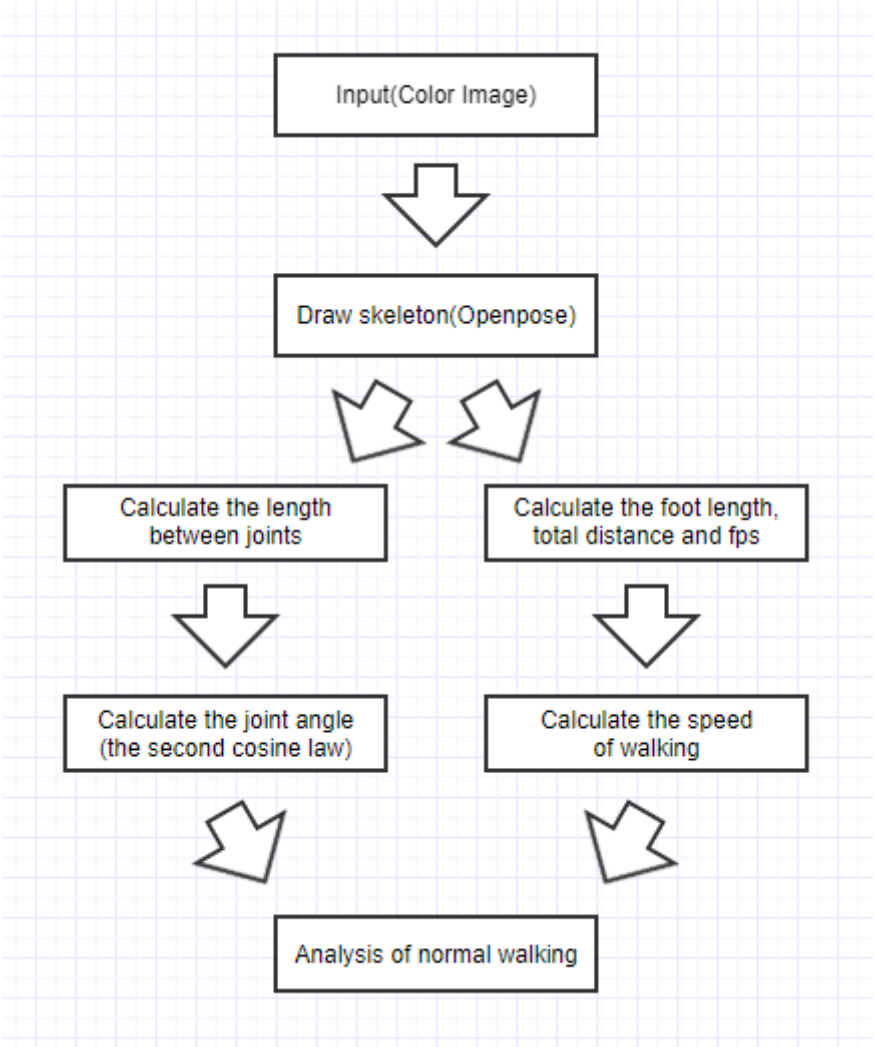
- Python3.9
- CUDA 11.8.0
- cudatoolkit
- Anaconda
- PyTorch 11.2.1
- CuDNN 8.2.1
- Openpose 1.7.0
- CMake: 3.26.4

Dataset

Project Procedure

Setup

Flow Chart

- 

About the Cycle of the Gait

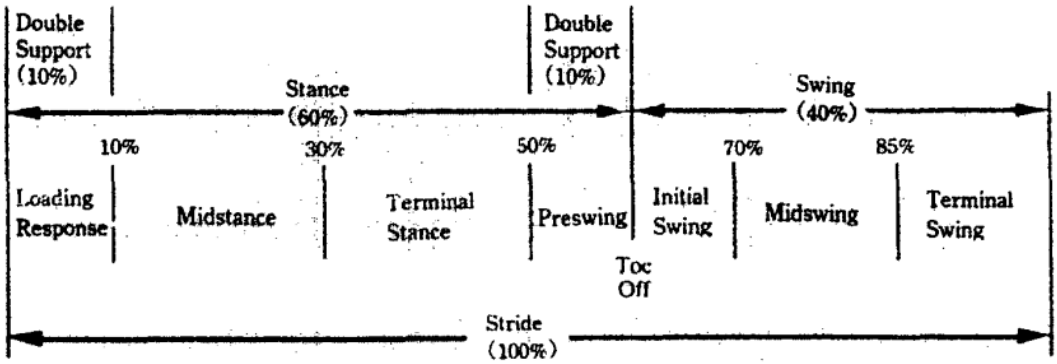
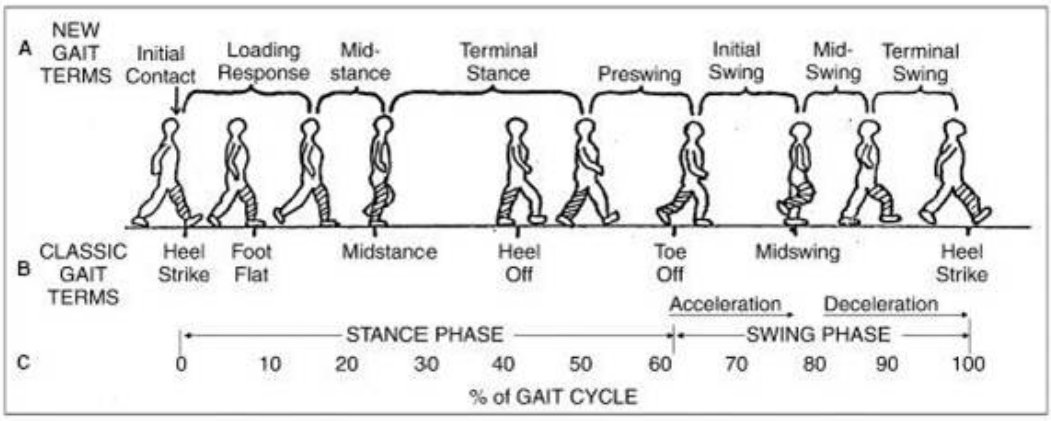
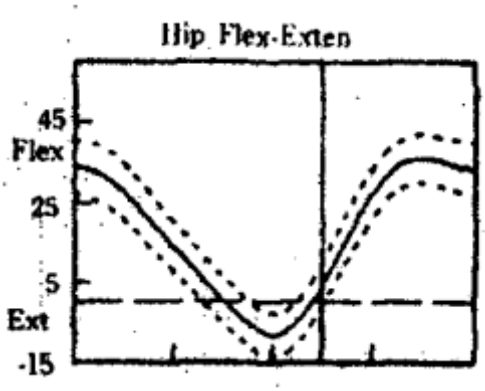
- 

Fig. 1. Gait cycle terminology used for walking.

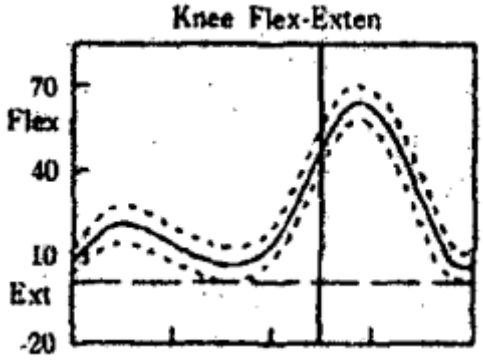
- 

About the Angle of the Joint

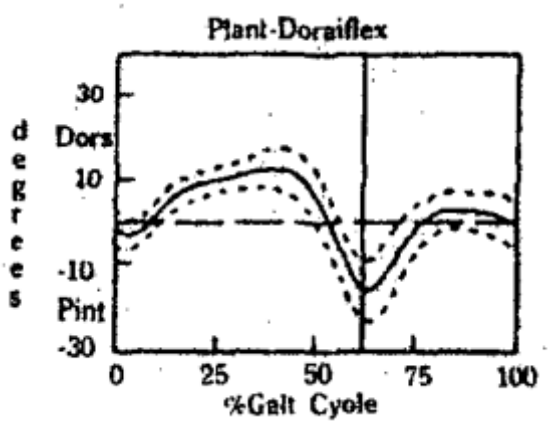
- **Hip Joint Flex-Extension**



- **Knee Joint Flex-Extension**



- **Plant-Dorsiflexion**



Calculation Requirements

1. **Initial Contact**
    1. When the event occurred
        - Occurs when **the right heel touches the ground**
    2. The angle of the joint
        - About **35º** Hip Joint Flexion [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **145º**
        - About **8º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **172º**
        - About **2º** Right Dorsiflexion [RAnkle - RHeel - RToe]
            - The angle between the calf and the foot is about **88º**
2. **Loading Response**
    1. When the event occurred
        - From the Initial Contact to **the left toe off the ground**
    2. The angle of the joint
        - About **35º -> 30º** Hip Joint Flexion [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **145º -> 150º**
        - About **8º -> 15º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **172º -> 165º**
        - About **2º** Right Dorsiflexion -> About **6º** Right Plantar Flexion [RAnkle - RHeel - RToe]
            - The angle between the calf and the foot is about **88º -> 96º**
3. **Midstance**
    1. When the event occurred
        - From the Loading Response to **the left toe being ahead of the right heel**
    2. The angle of the joint
        - About **30º -> 36º** Hip Joint Flexion [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **150º -> 174º**
        - About **15º -> 12º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **165º -> 168º**
        - About **6º** Right Plantar Flexion -> About **10º** Right Dorsiflexion [RAnkle - RHeel - RToe]
            - The angle between the calf and the foot is about **96º -> 80º**
4. **Terminal Stance**
    1. When the event occurred
        - From the Midstance to **the left heel touches the ground**
    2. The angle of the joint
        - About **6º** Hip Joint Flexion -> About **6º** Hip Joint Extension [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **174º** -> The angle between the waist and the thigh is about **174º**(However, in the opposite direction of the walking direction)
        - About **12º -> 8º** Right Knee Flexion [RHip - RKnee - RAnkle]

        - The angle between the thigh and calf about **168º -> 172º**
    - About **10º -> 12º** Right Dorsiflexion [RAnkle - RHeel - RToe]
        - The angle between the calf and the foot is about **80º -> 78º**

5. **Preswing**
    1. When the event occurred
        - From the Terminal Stance to **the right toe off the ground**
    2. The angle of the joint
        - About **6º -> 0º** Hip Joint Extension [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **174º -> 180º**
        - About **8º -> 35º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **172º -> 145º**
        - About **12º** Right Dorsiflexion -> About **6º** Right Plantar Flexion [RAnkle - RHeel - RToe]
            - The angle between the calf and the foot is about **78º -> 96º**
6. **Initial Swing**
    1. When the event occurred
        - From the Preswing to **the right toe being ahead of the left heel**
    2. The angle of the joint
        - About **0º -> 20º** Hip Joint Flexion [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **180º -> 160º**
        - About **35º -> 64º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **145º -> 116º**
        - About **6º -> 18º** Right Dorsiflexion [RAnkle - RHeel - RToe]
            - The angle between the calf and the foot is about **96º -> 108º**
7. **Midswing**
    1. When the event occurred
        - From Initial Swing to **parallel right and left feet**
    2. The angle of the joint
        - About **20º -> 37º** Hip Joint Flexion [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **160º -> 143º**
        - About **64º -> 35º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **116º -> 145º**
        - About **18º** Right Plantar Flexion -> About **3º** Right Dorsiflexion [RAnkle - RHeel - RToe]
            - The angle between the calf and the foot is about **108º -> 87º**
8. **Terminal Swing**
    1. When the event occurred
        - From Midswing to **the right heel touches the ground**
    2. The angle of the joint
        - About **37º -> 35º** Hip Joint Flexion [Neck - RHip - RKnee]
            - The angle between the waist and the thigh is about **143º -> 145º**
        - About **35º -> 8º** Right Knee Flexion [RHip - RKnee - RAnkle]
            - The angle between the thigh and calf about **145º -> 172º**
        - Aobut **3º -> 2º** Right Dorsiflexion [RAnkle - RHeel - RToe]

- The angle between the calf and the foot is about **87º -> 88º**

Installation

Prerequisites

1) CMake

Installing the CMake GUI

Download the latest version of **cmake-X.X.X-win64-x64.msi** from CMake download website

After downloading, double-click the msi file to proceed with the installation.

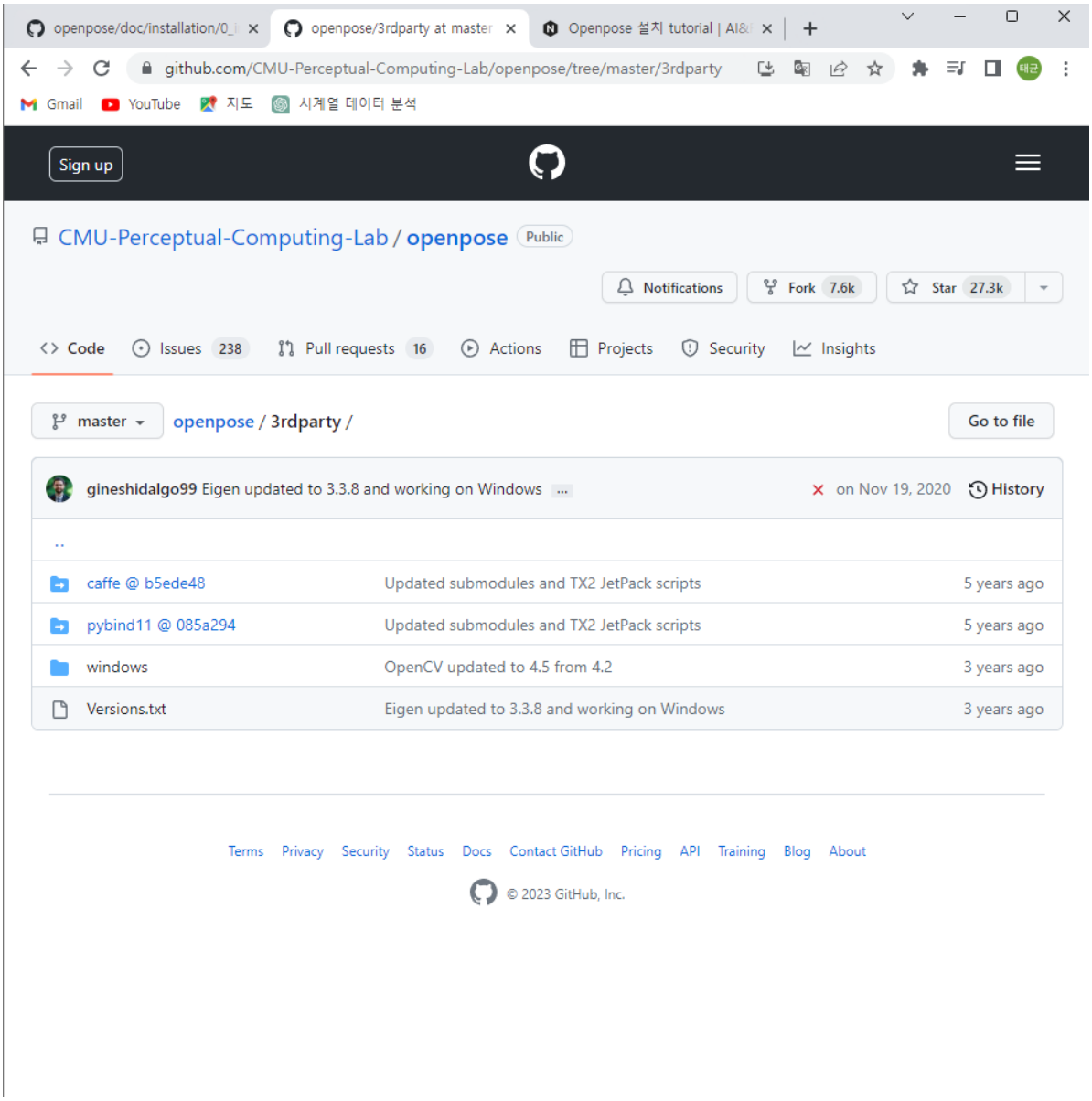
2) Visual Studio

Visual Studio to be used for build is required
Microsoft Visual Studio (VS) 2019 Enterprise
Microsoft Visual Studio (VS) 2017 Enterprise
VS 2015 Enterprise Update 3
It says to install one of them, and **Visual Studio 2019 Community** isn't officially supported either, but it's possible.

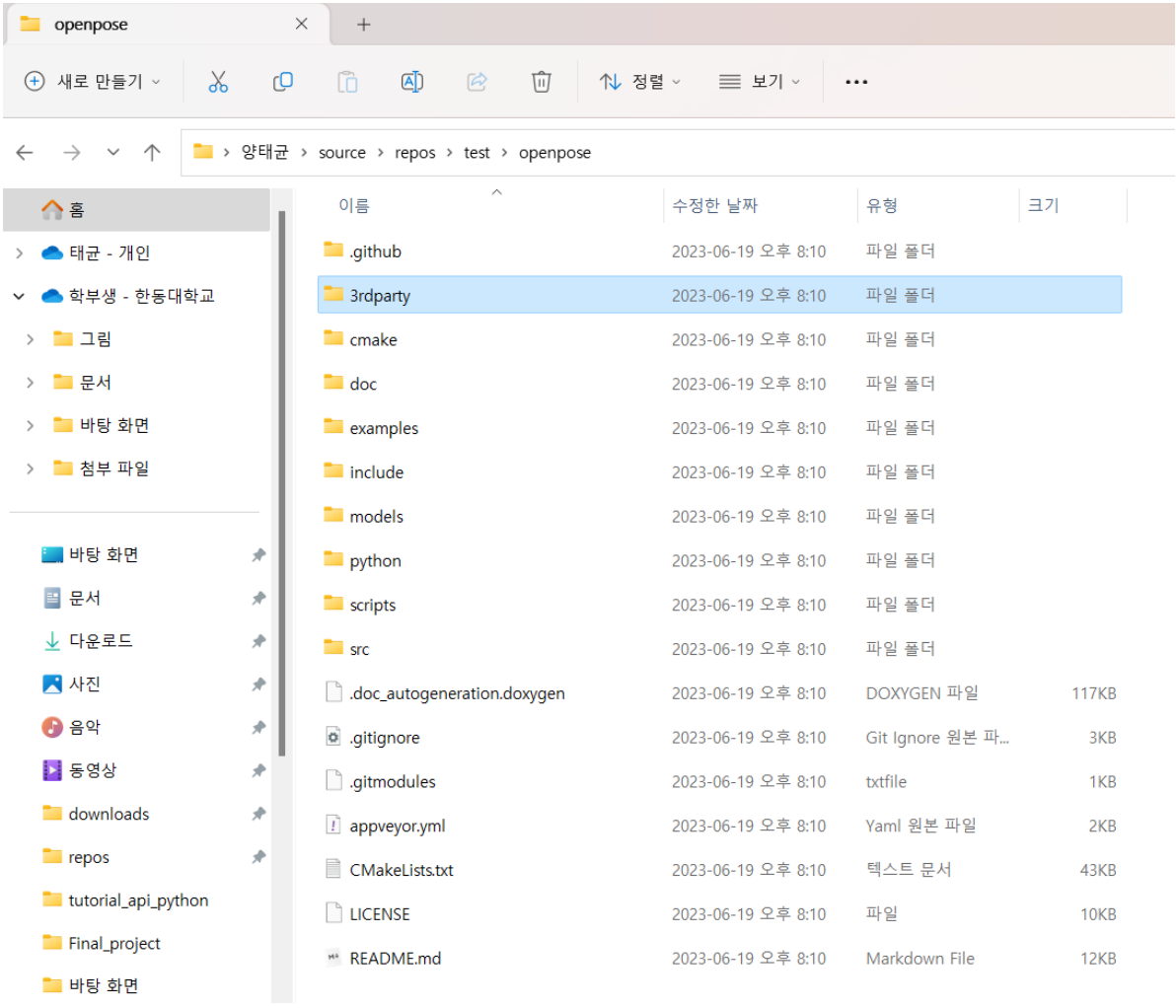
Openpose Installation

Download the Openpose zip file from Openpose download website

1) caffe, pybind11



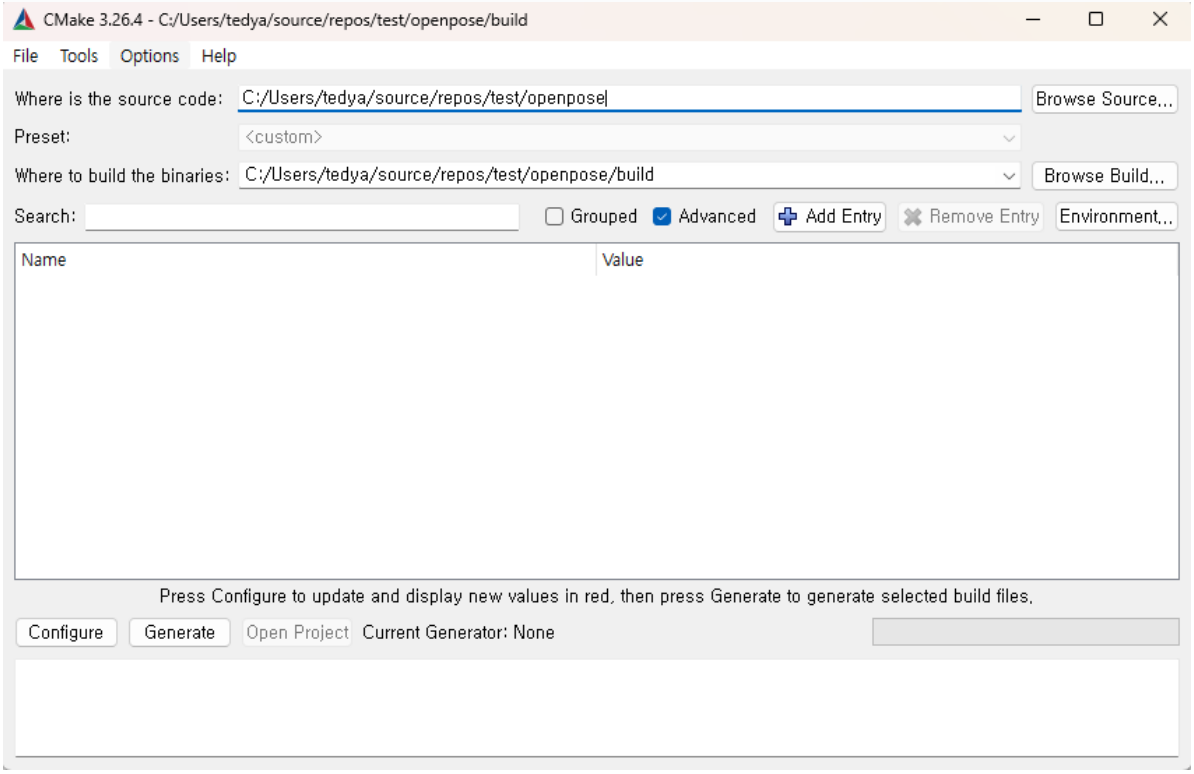
Then, click the 3rdparty folder in github where you downloaded openpose. After downloading caffe and pybind11, extract them to the 3rdparty folder. After that change the folder names to caffe and pybind11 respectively.



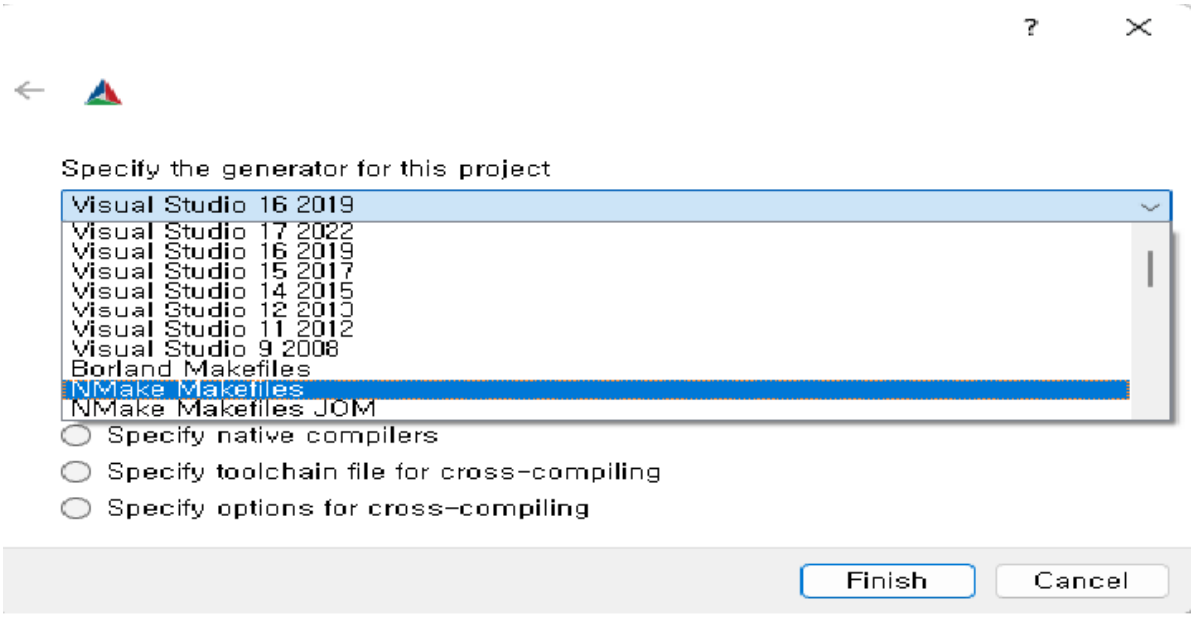
2) CMake configuration

Execute the code below in windows powershell to launch CMake GUI.

```
cd {OpenPose_folder}
mkdir build/
cd build/
cmake-gui ..
```

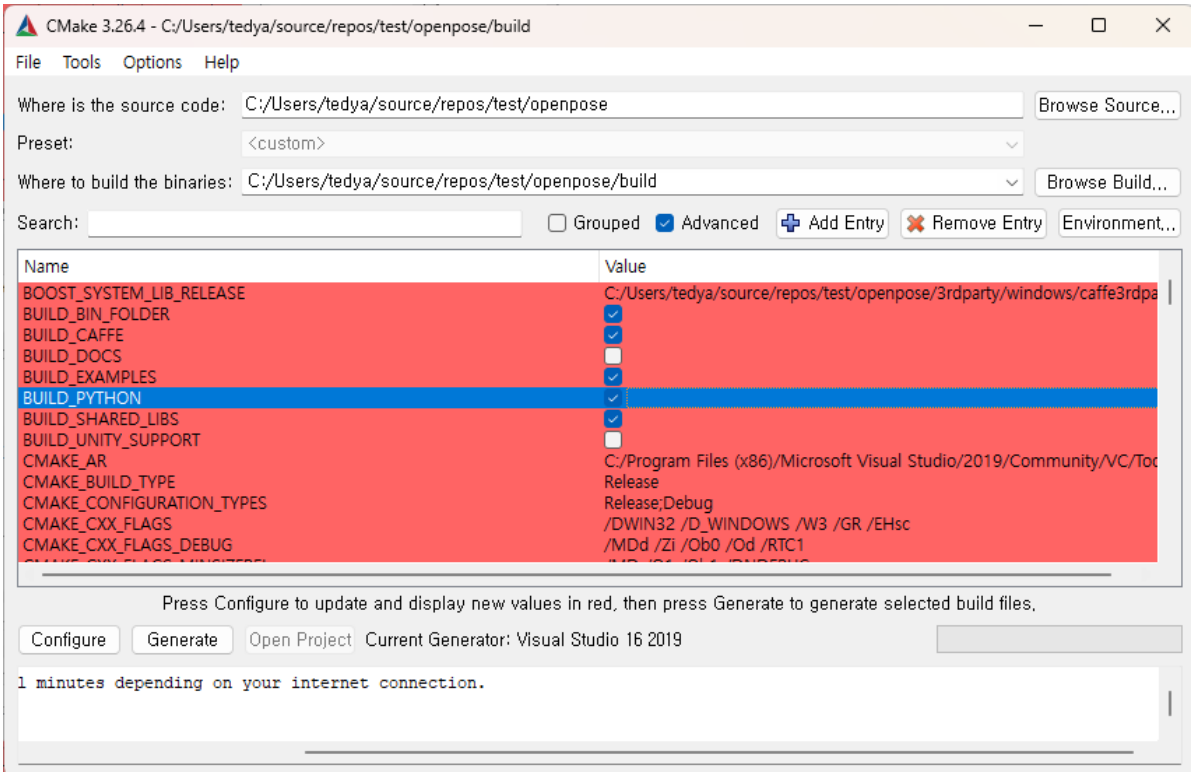


Click **Configure** at the bottom.



Choose the one that matches your version of Visual Studio. Then click **Finish**.

When it is completed, the following window appears. At this time, we use the Python API, so we check **BUILD_PYTHON**.

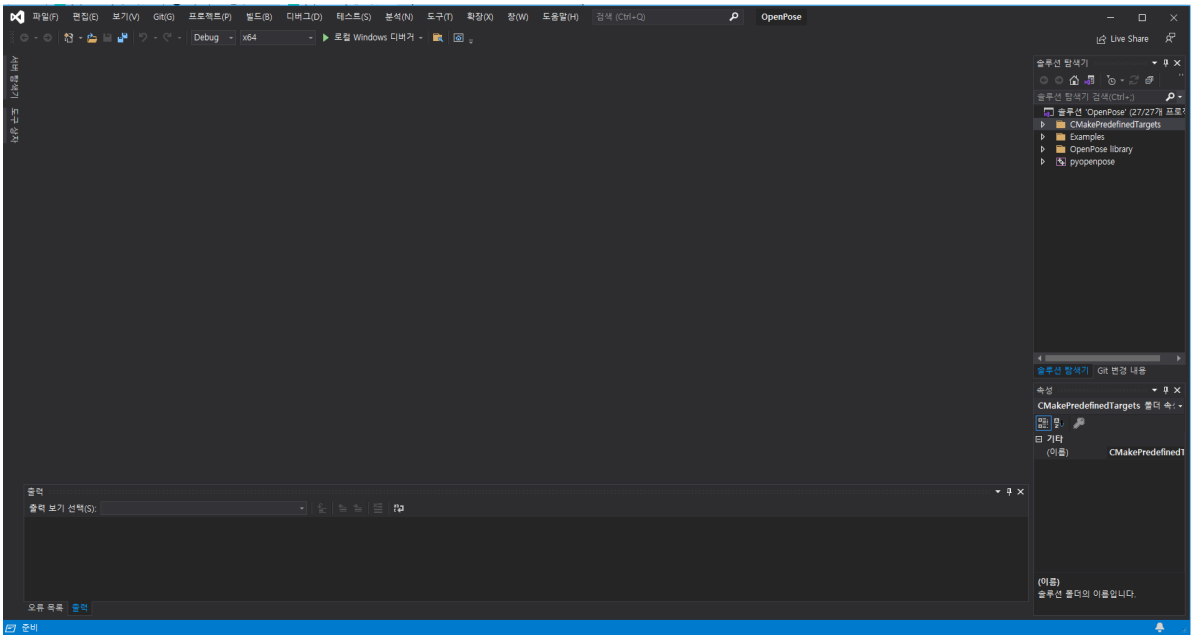


Click **Generate** at the bottom and search for python in the Search section.

And set **PYTHON_EXCUTABLE** appropriately for your python path. Click **Generate** again.

3) Project Compilation

Run the solution by double-clicking **OpenPose.sln** in the build folder. (You need to open it with the version of Visual Studio used in CMake.)

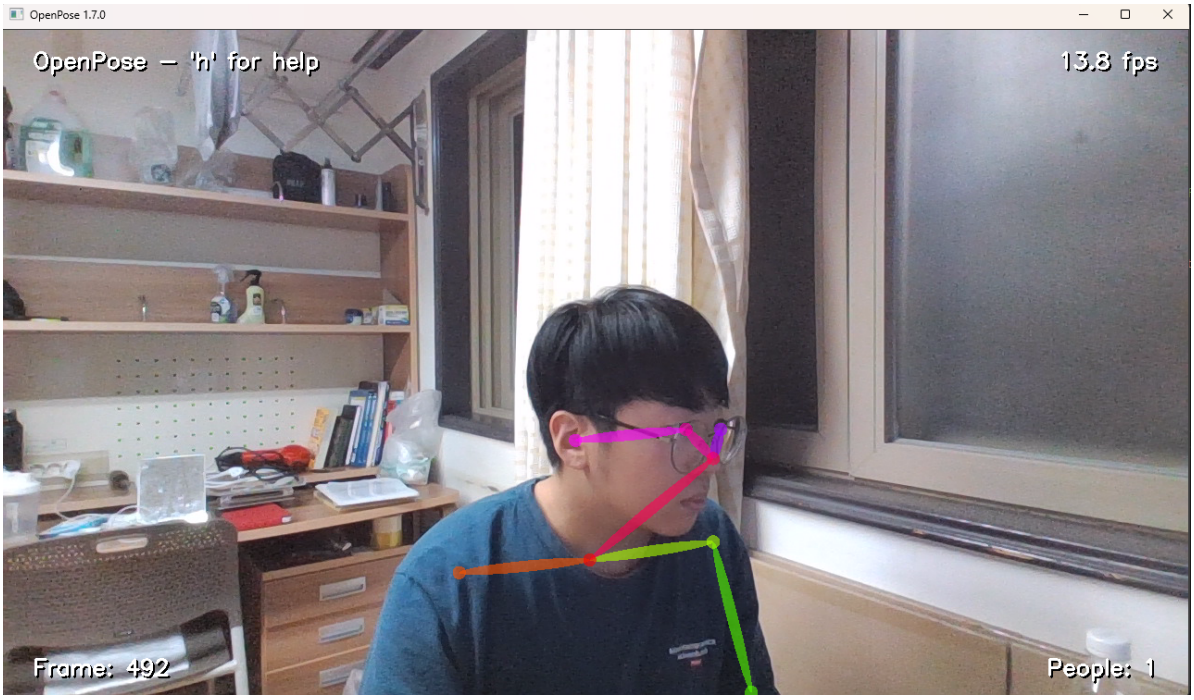


Change Debug to Release.

Clci Build Solution(Ctrl + Shift + B) in the Build tab.

After the build is complete

Run the build/examples/tutorial_api_python/**openpose_python.py** file to see if it works.



If the above file does not run

```
try:
    # Import Openpose (Windows/Ubuntu/OSX)
    #dir_path = os.path.dirname(os.path.realpath(__file__))

    # openpose/build/examples/tutorial_api_python 폴더의 절대 경로 설정
    dir_path = "C:/Users/tedya/source/repos/DLIP/openpose/build/examples/tutorial_api_python"
    try:
        # Windows Import
        if platform == "win32":
            # Change these variables to point to the correct folder (Release/x64 etc.)
            sys.path.append(dir_path + '/../../python/openpose/Release')
            os.environ['PATH'] = os.environ['PATH'] + ';' + dir_path + '/../../x64/Release;' + dir_path + '/../../bin;'
            import pyopenpose as op
        else:
            # Change these variables to point to the correct folder (Release/x64 etc.)
            sys.path.append('C:/Users/tedya/source/repos/DLIP/openpose/build/python')
            # If you run `make install` (default path is `/usr/local/python` for Ubuntu), you can also access the OpenPose/python module from there.
            # sys.path.append('/usr/local/python')
            from openpose import pyopenpose as op
    except ImportError as e:
        print('Error: OpenPose library could not be found. Did you enable `BUILD_PYTHON` in CMake and have this Python script in the right folder?')
        raise e

    # Flags
    parser = argparse.ArgumentParser()
    parser.add_argument("--hand", action="store_true", help="Enable hand keypoint detection.")
    args = parser.parse_known_args()

    # Custom Params (refer to include/openpose/flags.hpp for more parameters)
```

Set the path in the red box to your own

build/examples/tutorial_api_python and build/python respectively

By executing the above, Openpose can be executed in python.

Code

Module Import

```
import sys
import cv2
import os
import math
import numpy as np
from sys import platform
import argparse
from matplotlib.pyplot import hsv
from matplotlib import pyplot as plt
```

Set Openpose Parts

```
BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
               "LShoulder": 5, "LElbow": 6, "LWrist": 7, "MidHip": 8, "RHip":
9,
               "RKnee": 10, "RAnkle": 11, "LHip": 12, "LKnee": 13, "LAnkle":
14,
               "REye": 15, "LEye": 16, "REar": 17, "LEar": 18, "LBigToe": 19,
               "LSmallToe": 20, "LHeel": 21, "RBigToe": 22, "RSmallToe": 23,
               "RHeel": 24, "Background": 25 }

POSE_PAIRS = [ ["Nose", "Neck"], ["Neck", "RShoulder"], ["RShoulder", "RElbow"],
               ["RElbow", "RWrist"], ["Neck", "LShoulder"], ["LShoulder",
"LElbow"],
               ["LElbow", "LWrist"], ["Neck", "MidHip"], ["MidHip", "RHip"],
["RHip", "RKnee"],
               ["RKnee", "RAnkle"], ["MidHip", "LHip"], ["LHip", "LKnee"],
["LKnee", "LAnkle"],
               ["RBigToe", "RHeel"], ["LBigToe", "LHeel"]
               ]
```

Initialize Variables

```
frame_count = 0

neck = [0,0,0]
rhip = [0,0,0]
rknee = [0,0,0]
rankle = [0,0,0]
lhip = [0,0,0]
lknee = [0,0,0]
lankle = [0,0,0]
ltoe = [0,0,0]
lheel = [0,0,0]
rtoe = [0,0,0]
rheel = [0,0,0]

rtoe_max = 0
rheel_max = 0
lheel_max = 0
ground_right = 0
ground_left = 0
ground_toe = 0
ground = 0

HJFradian = 0
RKFradian = 0
RDFradian = 0

rankle_pred = [0,0,0]
lankle_pred = [0,0,0]
rheel_pred = [0,0,0]
lheel_pred = [0,0,0]


# initial list
```

```
HJFdegree = []
RKFdegree = []
RDFdegree = []
Frame_num = []
initial_contact = []
loading = []
mid = []
terminal = []
preswing = []
initial_swing = []
loading_swing = []
mid_swing = []
terminal_swing = []
contact = []

initial_flag = False
loading_flag = False
mid_flag = False
terminal_flag = False

preswing_flag = False
loading_swing_flag = False
mid_swing_flag = False
terminal_swing_flag = False
initial_contact_flag = True
mid_swing_flag2 = False
Start_flag = False
Begin_flag = False

# 발꿈치 좌표 저장 리스트
rheel_y_coords = []
lheel_y_coords = []
```

Input User Name and Foot Size

```
name = input("What is your name?")

foot_size = input("What is your foot size?")
```

Output Text setting

```
text_name = name + " Walking Analysis.txt"

analysis_text = open(text_name, 'w')
```

Define Function

```
def find_indexes(lst,ground):
    index = 0
    while index < len(lst):
        if lst[index] > ground:
            return index
        index += 1

def find_flag_index(lst, ground ,start_index):
    index = start_index + 1
```

```
    while index < len(lst):
        if lst[index] < ground - 10:
            return index
        index += 1

def find_indexes_repeated(lst, ground):
    result = []
    index = find_indexes(lst,ground)
    while index is not None:
        result.append(index)
        flag_index = find_flag_index(lst,ground ,index)
        if flag_index is None:
            break
        index = find_indexes(lst[flag_index+1:],ground)
        if index is not None:
            index += flag_index + 1
    return result

def print_index_differences(lst):
    for i in range(len(lst) - 1):
        diff = lst[i+1] - lst[i]
        print(f"Difference between index {i} and index {i+1}: {diff}")

# 관절 연결을 그리는 함수
def draw_body_connections(image, person):
    # 선 그리기
    for pair in POSE_PAIRS:
        partA = pair[0]
        partB = pair[1]

        if partA in BODY_PARTS.keys() and partB in BODY_PARTS.keys():
            indexA = BODY_PARTS[partA]
            indexB = BODY_PARTS[partB]

            confidence1 = person[indexA][2]
            confidence2 = person[indexB][2]

            if confidence1 > 0.5 and confidence2 > 0.5:
                x1, y1, _ = person[indexA]
                x2, y2, _ = person[indexB]
                cv2.line(image, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255,
0), 2)
```

Load Openpose Model

```
try:
    # Import Openpose (Windows/Ubuntu/OSX)
    #dir_path = os.path.dirname(os.path.realpath(__file__))

    # openpose/build/examples/tutorial_api_python 폴더의 절대 경로 설정
    dir_path =
"C:/Users/tedya/source/repos/DLIP/openpose/build/examples/tutorial_api_python"
    try:
        # Windows Import
        if platform == "win32":
            # Change these variables to point to the correct folder (Release/x64
etc.)
```

```
            sys.path.append(dir_path + '/../../python/openpose/Release')
            os.environ['PATH'] = os.environ['PATH'] + ';' + dir_path +
'/../../x64/Release;' + dir_path + '/../../bin;'
            import pyopenpose as op
        else:
            # Change these variables to point to the correct folder (Release/x64
etc.)

 sys.path.append('C:/Users/tedya/source/repos/DLIP/openpose/build/python')
            # If you run `make install` (default path is `/usr/local/python` for
Ubuntu), you can also access the OpenPose/python module from there. This will
install OpenPose and the python library at your desired installation path. Ensure
that this is in your python path in order to use it.
            # sys.path.append('/usr/local/python')
            from openpose import pyopenpose as op
    except ImportError as e:
        print('Error: OpenPose library could not be found. Did you enable
`BUILD_PYTHON` in CMake and have this Python script in the right folder?')
        raise e

    # Flags
    parser = argparse.ArgumentParser()
    parser.add_argument("--hand", action="store_true", help="Enable hand
keypoint detection.")
    args = parser.parse_known_args()

    # Custom Params (refer to include/openpose/flags.hpp for more parameters)
    params = dict()

    # 모델 폴더 절대 경로 설정
    params["model_folder"] = "C:/Users/tedya/source/repos/DLIP/openpose/models"
    params["body"] = 1  # 1은 관절 검출을 활성화하는 옵션입니다.
    params["disable_multi_thread"] = True  # 멀티 스레드 비활성화

    # Starting OpenPose
    opWrapper = op.WrapperPython()
    opWrapper.configure(params)
    opWrapper.start()
```

Video Input and Output setting

```
    # Set up video
    # Change name as your video name
    cap = cv2.VideoCapture('walking04.mp4')
    # Set up Webcam
    #cap = cv2.VideoCapture(0)
    # 저장할 동영상 파일 경로 및 설정
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    # 자를 동영상의 속성 설정
    output_width = width
    output_height = height
    output_path = name + '_Walking_analysis.avi'
    fourcc = cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')  # 동영상 코덱 설정
('MJPG')
    fps = cap.get(cv2.CAP_PROP_FPS)  # 원본 동영상의 FPS 가져오기
```

```
    output = cv2.VideoWriter(output_path, fourcc, fps, (output_width,
output_height))
```

Visualize the Result

From here, you have to put it in while loop.

```
while True:
```

Read Video

```
        ret, frame = cap.read()

        frame_count = frame_count + 1
        if not ret:
            break

        # image size
        Height, Width, _ = frame.shape

        # Fps
        fps = cap.get(cv2.CAP_PROP_FPS)
```

Datum

```
        # Create new datum
        datum = op.Datum()
        datum.cvInputData = frame

        # Process frame
        opWrapper.emplaceAndPop(op.VectorDatum([datum]))

        # 관절 및 신체 파츠 그리기
        keypoints = datum.poseKeypoints
```

Save previous value

```
        rankle_pred = rankle
        lankle_pred = lankle
        rheel_pred = rheel
        lheel_pred = lheel
```

Draw circle and line to the Parts

```
        if keypoints is not None:
            for person in keypoints:
                neck = person[BODY_PARTS["Neck"]]
                rhip = person[BODY_PARTS["RHip"]]
                rknee = person[BODY_PARTS["RKnee"]]
                rankle = person[BODY_PARTS["RAnkle"]]
                lhip = person[BODY_PARTS["LHip"]]
                lknee = person[BODY_PARTS["LKnee"]]
                lankle = person[BODY_PARTS["LAnkle"]]
                lheel = person[BODY_PARTS["LHeel"]]
```

```
                rtoe = person[BODY_PARTS["RBigToe"]]
                rheel = person[BODY_PARTS["RHeel"]]
                ltoe = person[BODY_PARTS["LBigToe"]]

                # Check if toe keypoints are missing
                if np.all(rtoe == 0):
                    rtoe = prev_rtoe
                else:
                    prev_rtoe = rtoe

                if np.all(ltoe == 0):
                    ltoe = prev_ltoe
                else:
                    prev_ltoe = ltoe

                # Check if heel keypoints are missing
                if np.all(rheel == 0):
                    rheel = prev_rheel
                else:
                    prev_rheel = rheel

                if np.all(lheel == 0):
                    lheel = prev_lheel
                else:
                    prev_lheel = lheel

                # 관절 그리기
                for idx, point in enumerate(person):
                    x, y, confidence = point
                    if confidence > 0.5:
                        cv2.circle(frame, (int(x), int(y)), 2, (0, 255, 255),
-1)
                        cv2.putText(frame, str(idx), (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)

                draw_body_connections(frame, person)
                rheel_y_coords.append(rheel[1])
                lheel_y_coords.append(lheel[1])
```

Find Maximum value

```
        if lheel_max < lheel[1]:
            lheel_max = lheel[1]

        if rheel_max < rheel[1]:
            rheel_max = rheel[1]

        if rtoe_max < rtoe[1]:
            rtoe_max = rtoe[1]
```

Set Ground value

```
        if frame_count == 20:
            ground_toe = rtoe_max
            ground = rheel_max
            ground_right = rheel_max
            ground_left = lheel_max
            ground = rheel_max
            Start_flag = True
```

Start analysis

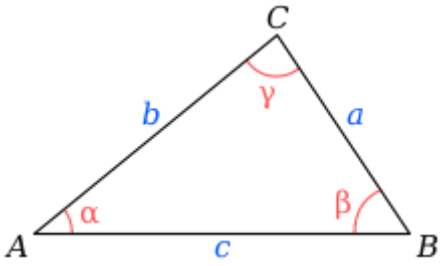
From here, it is a code for analyzing gait. At this time, only one flag is set to have a value of True. Therefore, the analysis is carried out sequentially from the top. It also measures the frame.

```
if Start_flag == True:

            if rheel[1] >=ground_right and rheel_pred[1] < ground_right and
rheel[0] > lheel[0]:
                initial_flag = True

        # Initial contact
        if initial_flag == True:
            if initial_contact_flag == True:
                if rheel[1] >= ground_right and rheel_pred[1] < ground_right :
                    initial_contact.append(frame_count)
                    initial_contact_flag = False
                    loading_flag = True
                    contact.append(rheel[0])
                    contact.append(rheel[1])
                    contact.append(rtoe[0])
                    contact.append(rtoe[1])

            # Mid Stance1
            if loading_flag == True:
                if ltoe[1] < ground_left and rheel[1] >= ground_right and
lheel[0] < rheel[0] :
                    loading.append(frame_count)
                    loading_flag = False
                    mid_flag = True

            # Terminal Stance1
            if mid_flag == True:
                if lheel[0] >= rheel[0] and rtoe[1] >= ground_right and lheel[1]
< ground_left:
                    mid.append(frame_count)
                    mid_flag = False
                    terminal_flag = True
            # Preswing
            if terminal_flag == True:
                if lheel[1] >= ground_left and rtoe[1] >= ground_right:
                    terminal.append(frame_count)
                    terminal_flag = False
                    preswing_flag = True

            #오른발 스윙

            #Initial Swing
            #왼발 닿는 순간 초기화
```

```
            if preswing_flag == True:

                if lheel[1] >= ground_left and lheel_pred[1] < ground_left :
                    preswing_flag = False
                    loading_swing_flag = True

            #Mid Swing
            if loading_swing_flag == True:

                if rtoe[1] < ground_right and lheel[1] >= ground_left and
rheel[0] < lheel[0]:
                    preswing.append(frame_count)
                    loading_flag = False
                    mid_swing_flag = True

            #Terminal Swing
            if mid_swing_flag == True:
                if rheel[0] >= lheel[0] and ltoe[1] >= ground_left and rheel[1]
< ground_right:
                    initial_swing.append(frame_count)
                    mid_swing_flag = False
                    mid_swing_flag2 = True

            if mid_swing_flag2 == True:
                if 3>(rheel[1]-rtoe[1])>-3:
                    mid_swing.append(frame_count)
                    mid_swing_flag2 = False
                    terminal_swing_flag = True
                    initial_contact_flag = True

        Frame_num.append(frame_count)
```

Calculate Flexion

We use law of cosines to calculate flexion



$$Cos\alpha = (a^2 + b^2 - c^2)/2bc$$

```
#Hip Joint Flexion
        Len1 = math.sqrt(math.pow(neck[0] - rhip[0], 2) + math.pow(neck[1] -
rhip[1], 2))
        Len2= math.sqrt(math.pow(rhip[0] - rknee[0], 2) + math.pow(rhip[1] -
rknee[1], 2))
        Len3 = math.sqrt(math.pow(rknee[0] - neck[0], 2) + math.pow(rknee[1] -
neck[1], 2))
        if Len1 == 0 or Len2 == 0 or Len3 == 0:
            HJFradian = HJFradian
        else:
```

```
            cos_angle = (Len1 * Len1 + Len2 * Len2 - Len3 * Len3) / (2 * Len1 *
Len2 +1.e-09)
            if -1 <= cos_angle <= 1:
                HJFradian = math.acos(cos_angle)
            else:
                HJFradian = 0
        HJFdegree.append(HJFradian / np.pi * 180)
        #Right Knee Flexion
        Len1= math.sqrt(math.pow(rhip[0] - rknee[0], 2) + math.pow(rhip[1] -
rknee[1], 2))
        Len2 = math.sqrt(math.pow(rknee[0] - rankle[0], 2) + math.pow(rknee[1] -
rankle[1], 2))
        Len3 = math.sqrt(math.pow(rankle[0] - rhip[0], 2) + math.pow(rankle[1] -
rhip[1], 2))
        if Len1 == 0 or Len2 == 0 or Len3 == 0:
            RKFradian = RKFradian
        else:
            cos_angle = (Len1 * Len1 + Len2 * Len2 - Len3 * Len3) / (2 * Len1 *
Len2 +1.e-09)
            if -1 <= cos_angle <= 1:
                RKFradian = math.acos(cos_angle)
            else:
                RKFradian = 0
        RKFdegree.append(RKFradian / np.pi * 180)
        #Right Dorsi Flexion
        Len1= math.sqrt(math.pow(rankle[0] - rheel[0], 2) + math.pow(rankle[1] -
rheel[0], 2))
        Len2 = math.sqrt(math.pow(rheel[0] - rtoe[0], 2) + math.pow(rheel[1] -
rtoe[0], 2))
        Len3 = math.sqrt(math.pow(rtoe[0] - rankle[0], 2) + math.pow(rtoe[1] -
rankle[0], 2))
        if Len1 == 0 or Len2 == 0 or Len3 == 0:
            RDradian = RDFradian
        else:
            cos_angle = (Len1 * Len1 + Len2 * Len2 - Len3 * Len3) / (2 * Len1 *
Len2 +1.e-09)
            if -1 <= cos_angle <= 1:
                RDFradian = math.acos(cos_angle)
            else:
                RDFradian = 0
        RDFdegree.append(RDFradian / np.pi * 180)
```

Put Frame and Flexion to Image

```
        # 텍스트 그리기
        cv2.putText(frame, f"Frame : {frame_count}", (20,30),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.putText(frame, f"Hip joint : {HJFdegree[-1]:.2f}", (20,280),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.putText(frame, f"Right knee : {RKFdegree[-1]:.2f}", (20,310),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.putText(frame, f"Right dorsi : {RDFdegree[-1]:.2f}", (20,340),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.imshow("OpenPose Webcam", frame)
        # 프레임 저장
        output.write(frame)
```

Set Stop key

```
        # Exit if 'q' is pressed
        if cv2.waitKey(1) == ord('q'):
            break
```

Put Name on the output Text

```
    analysis_text.write(f"{name}\n")
```

Writing gait advice in output Text

```
# initial contact
    if HJFdegree[initial_contact[0]] < 140:
        analysis_text.write("When your right heel touches the ground, stretch
your hip more!\n")
    if HJFdegree[initial_contact[0]] > 150:
        analysis_text.write("When your right heel touches the ground, bend your
hip more!\n")
    if RKFdegree[initial_contact[0]] < 167:
        analysis_text.write("When your right heel touches the ground, stretch
your knee more!\n")
    if RKFdegree[initial_contact[0]] > 177:
        analysis_text.write("When your right heel touches the ground, bend your
knee more!\n")

    # Loading response
    if HJFdegree[loading[0]] < 145:
        analysis_text.write("When your left heel lifted from the ground, stretch
your hip more!\n")
    if HJFdegree[loading[0]] > 155:
        analysis_text.write("When your left heel lifted from the ground, bend
your hip more!\n")
    if RKFdegree[loading[0]] < 160:
        analysis_text.write("When your left heel lifted from the ground, stretch
your knee more!\n")
    if RKFdegree[loading[0]] > 170:
        analysis_text.write("When your left heel lifted from the ground, bend
your knee more!\n")

    # Mid stance
    if HJFdegree[mid[0]] < 169:
        analysis_text.write("When your left foot is ahead of your right foot,
stretch your hip more!\n")
    if HJFdegree[mid[0]] > 179:
        analysis_text.write("When your left foot is ahead of your right foot,
bend your hip more!\n")
    if RKFdegree[mid[0]] < 163:
        analysis_text.write("When your left foot is ahead of your right foot,
stretch your knee more!\n")
    if RKFdegree[mid[0]] > 173:
        analysis_text.write("When your left foot is ahead of your right foot,
bend your knee more!\n")

    # Terminal stance
    if HJFdegree[terminal[0]] < 169:
```

```
        analysis_text.write("When your left heel touches the ground, bend your
hip more!\n")
    if HJFdegree[terminal[0]] > 179:
        analysis_text.write("When your left heel touches the ground, stretch
your hip more!\n")
    if RKFdegree[terminal[0]] < 167:
        analysis_text.write("When your left heel touches the ground, stretch
your knee more!\n")
    if RKFdegree[terminal[0]] > 177:
        analysis_text.write("When your left heel touches the ground, bend your
knee more!\n")

    # preswing
    if HJFdegree[preswing[0]] < 175:
        analysis_text.write("When your right heel is lifted from the ground,
stretch your hip more!\n")
    if HJFdegree[preswing[0]] > 150:
        analysis_text.write("When your right heel is lifted from the ground,
bend your hip more!\n")
    if RKFdegree[preswing[0]] < 140:
        analysis_text.write("When your right heel is lifted from the ground,
stretch your knee more!\n")
    if RKFdegree[preswing[0]] > 150:
        analysis_text.write("When your right heel is lifted from the ground,
bend your knee more!\n")


    # Initial swing
    if HJFdegree[initial_swing[0]] < 155:
        analysis_text.write("When your right foot is ahead of your left foot,
stretch your hip more!\n")
    if HJFdegree[initial_swing[0]] > 165:
        analysis_text.write("When your right foot is ahead of your left foot,
bend your hip more!\n")
    if RKFdegree[initial_swing[0]] < 111:
        analysis_text.write("When your right foot is ahead of your left foot,
stretch your knee more!\n")
    if RKFdegree[initial_swing[0]] > 121:
        analysis_text.write("When your right foot is ahead of your left foot,
bend your knee more!\n")

    # Mid swing
    if HJFdegree[mid_swing[0]] < 138:
        analysis_text.write("When your right and left feet are parallel, stretch
your hip more!\n")
    if HJFdegree[mid_swing[0]] > 148:
        analysis_text.write("When your right and left feet are parallel, bend
your hip more!\n")
    if RKFdegree[mid_swing[0]] < 140:
        analysis_text.write("When your right and left feet are parallel, stretch
your knee more!\n")
    if RKFdegree[mid_swing[0]] > 150:
        analysis_text.write("When your right and left feet are parallel, bend
your knee more!\n")
```

Close Text

```
    analysis_text.close()
```

Draw Plot

```
    # Draw plot
    plt.figure()
    plt.plot(Frame_num, HJFdegree, label='Hip Joint Flexion')
    plt.plot(Frame_num, RKFdegree, label='Right Knee Flexion')
    plt.plot(Frame_num, RDFdegree, label='Right Dorsi Flexion')
```

Draw Line for each steps

```
 #loading response
    plt.axvline(initial_contact[0], linewidth=2, color='r')
    plt.text(initial_contact[0]+0.2, 15, 'initial contact', color='b',
fontsize=12)
    plt.hlines(120, initial_contact[0], loading[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text((initial_contact[0]+1), 121, 'loading response', color='b',
fontsize=12)

    #mid stance
    plt.axvline(loading[0], linewidth=2, color='r')
    plt.hlines(105, loading[0], mid[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text((loading[0]+1), 106, 'mid stance', color='b', fontsize=12)

    # terminal stance
    plt.axvline(mid[0], linewidth=2, color='r')
    plt.hlines(90, mid[0], terminal[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text((mid[0]+1), 91, 'terminal stance', color='b', fontsize=12)

    #preswing
    plt.axvline(terminal[0], linewidth=2, color='r')
    plt.hlines(75, terminal[0], preswing[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text(terminal[0]+1, 76, 'preswing', color='b', fontsize=12)

    #initial swing
    plt.axvline(preswing[0], linewidth=2, color='r')
    plt.hlines(60, preswing[0], initial_swing[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text(preswing[0]+1, 61, 'initial swing', color='b', fontsize=12)

    # mid swing
    plt.axvline(initial_swing[0], linewidth=2, color='r')
    plt.hlines(45, initial_swing[0], mid_swing[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text(initial_swing[0]+1, 46, 'mid swing', color='b', fontsize=12)

    #terminal swing
    plt.axvline(mid_swing[0], linewidth=2, color='r')
    plt.hlines(30, mid_swing[0], initial_contact[1], linewidth=1, color='r',
linestyle='dashed')
    plt.text(mid_swing[0]+1, 31, 'terminal swing', color='b', fontsize=12)
```

```
    plt.axvline(initial_contact[1], linewidth=2, color='r')
```

Add Plot detail

```
    plt.xlabel('Frame Number')
    plt.ylabel('Degree')
    plt.title(name + ' Joint Flexion')
    plt.legend(loc='upper right')
    plt.grid(True)
    plt.show()
```

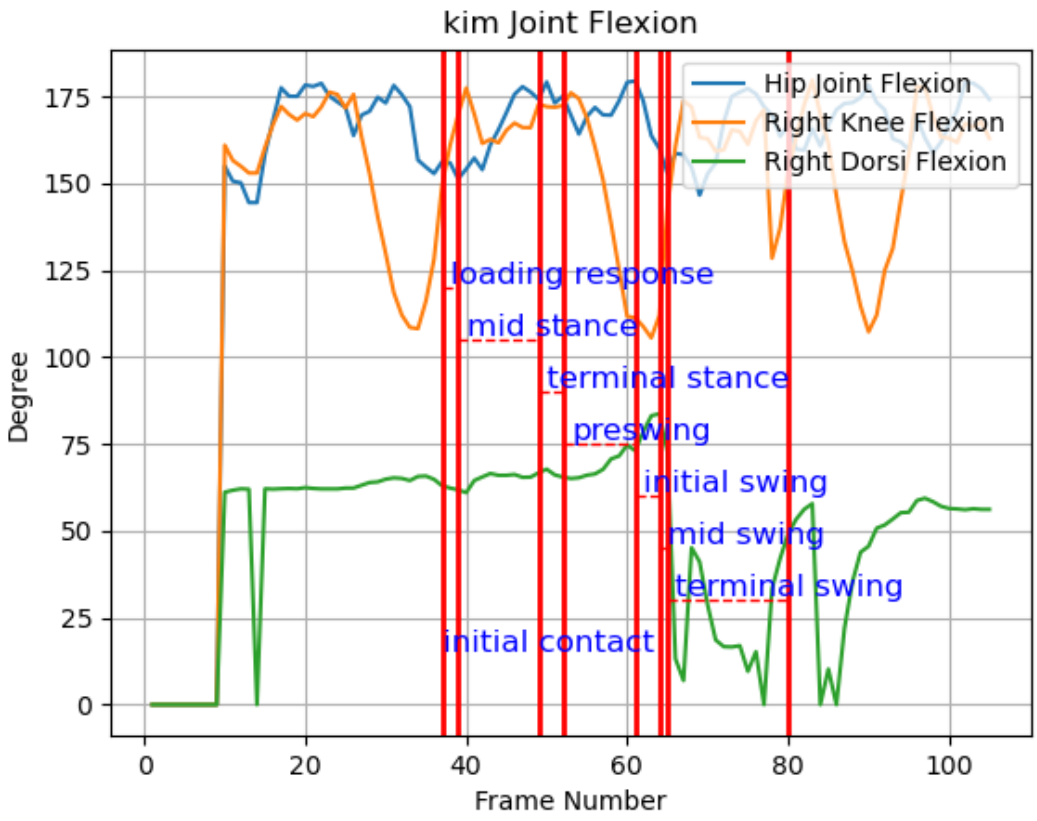
Release

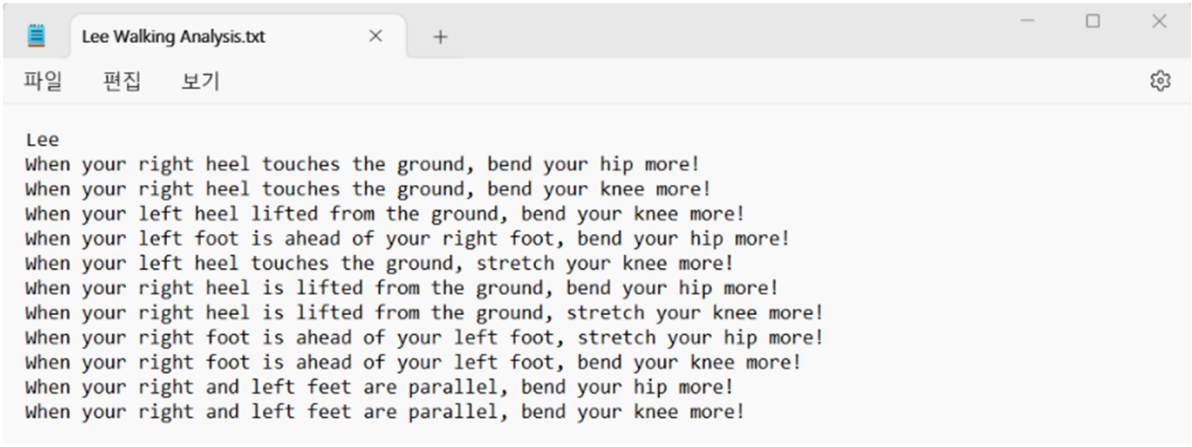
```
    # Release webcam and close windows
    output.release()
    cap.release()
    cv2.destroyAllWindows()
```

Now end the while loop

Results and Analysis

Results





Discussion

Our project has some similarities to the expected results, but some are not.

- Similar parts:
    1. The hip and knee angles came out quite similar to the expected results. However, there were parts that slightly deviated from the angle value required for each cycle set based on the paper.
- Different parts:
    1. The ankle angle did not meet expectations. Openpose was less accurate in tracking foot points.
    2. There was a problem that a camera with a high frame was required. Video taken with a cell phone has a low frame and often has a disconnection phenomenon, so the results were not properly derived.
- Another parts:
    1. We analyzed the gait analysis only in the sagittal plane. However, since the actual walking analysis is measured not only in the sagittal plane but also in the coronary plane and the horizontal plane, it is judged that the result value was somewhat insufficient.

Appendix

Full Code

```
import sys
import cv2
import os
import math
import numpy as np
from sys import platform
import argparse
from matplotlib.pyplot import hsv
from matplotlib import pyplot as plt

BODY_PARTS = { "Nose": 0, "Neck": 1, "RShoulder": 2, "RElbow": 3, "RWrist": 4,
               "LShoulder": 5, "LElbow": 6, "LWrist": 7, "MidHip": 8, "RHip":
9,
               "RKnee": 10, "RAnkle": 11, "LHip": 12, "LKnee": 13, "LAnkle":
14,
               "REye": 15, "LEye": 16, "REar": 17, "LEar": 18, "LBigToe": 19,
```

```
                "LSmallToe": 20, "LHeel": 21, "RBigToe": 22, "RSmallToe": 23,
                "RHeel": 24, "Background": 25 }

POSE_PAIRS = [ ["Nose", "Neck"], ["Neck", "RShoulder"], ["RShoulder", "RElbow"],
                ["RElbow", "RWrist"], ["Neck", "LShoulder"], ["LShoulder",
"LElbow"],
                ["LElbow", "LWrist"], ["Neck", "MidHip"], ["MidHip", "RHip"],
["RHip", "RKnee"],
                ["RKnee", "RAnkle"], ["MidHip", "LHip"], ["LHip", "LKnee"],
["LKnee", "LAnkle"],
                ["RBigToe", "RHeel"], ["LBigToe", "LHeel"]
                ]


frame_count = 0

neck = [0,0,0]
rhip = [0,0,0]
rknee = [0,0,0]
rankle = [0,0,0]
lhip = [0,0,0]
lknee = [0,0,0]
lankle = [0,0,0]
ltoe = [0,0,0]
lheel = [0,0,0]
rtoe = [0,0,0]
rheel = [0,0,0]

rtoe_max = 0
rheel_max = 0
lheel_max = 0
ground_right = 0
ground_left = 0
ground_toe = 0
ground = 0

HJFradian = 0
RKFradian = 0
RDFradian = 0

rankle_pred = [0,0,0]
lankle_pred = [0,0,0]
rheel_pred = [0,0,0]
lheel_pred = [0,0,0]
rtoe_pred = [0,0,0]
ltoe_pred = [0,0,0]

prev_rheel = [0,0,0]
prev_lheel = [0,0,0]
prev_rtoe = [0,0,0]
prev_ltoe = [0,0,0]

# initial list
HJFdegree = []
RKFdegree = []
RDFdegree = []
Frame_num = []
initial_contact = []
```

```
loading = []
mid = []
terminal = []
preswing = []
initial_swing = []
loading_swing = []
mid_swing = []
terminal_swing = []
contact = []

initial_flag = False
loading_flag = False
mid_flag = False
terminal_flag = False

preswing_flag = False
loading_swing_flag = False
mid_swing_flag = False
terminal_swing_flag = False
initial_contact_flag = True
mid_swing_flag2 = False
Start_flag = False
Begin_flag = False

# 발꿈치 좌표 저장 리스트
rheel_y_coords = []
lheel_y_coords = []

name = input("What is your name?")

foot_size = input("What is your foot size?")

text_name = name + " Walking Analysis.txt"

analysis_text = open(text_name, 'w')


def find_indexes(lst,ground):
    index = 0
    while index < len(lst):
        if lst[index] > ground:
            return index
        index += 1

def find_flag_index(lst, ground ,start_index):
    index = start_index + 1
    while index < len(lst):
        if lst[index] < ground - 10:
            return index
        index += 1

def find_indexes_repeated(lst, ground):
    result = []
    index = find_indexes(lst,ground)
    while index is not None:
        result.append(index)
        flag_index = find_flag_index(lst,ground ,index)
        if flag_index is None:
```

```
            break
        index = find_indexes(lst[flag_index+1:],ground)
        if index is not None:
            index += flag_index + 1
    return result

def print_index_differences(lst):
    for i in range(len(lst) - 1):
        diff = lst[i+1] - lst[i]
        print(f"Difference between index {i} and index {i+1}: {diff}")

# 관절 연결을 그리는 함수
def draw_body_connections(image, person):

    # 선 그리기
    for pair in POSE_PAIRS:
        partA = pair[0]
        partB = pair[1]

        if partA in BODY_PARTS.keys() and partB in BODY_PARTS.keys():
            indexA = BODY_PARTS[partA]
            indexB = BODY_PARTS[partB]

            confidence1 = person[indexA][2]
            confidence2 = person[indexB][2]

            if confidence1 > 0.5 and confidence2 > 0.5:
                x1, y1, _ = person[indexA]
                x2, y2, _ = person[indexB]
                cv2.line(image, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255,
0), 2)


try:
    # Import Openpose (Windows/Ubuntu/OSX)
    #dir_path = os.path.dirname(os.path.realpath(__file__))

    # openpose/build/examples/tutorial_api_python 폴더의 절대 경로 설정
    dir_path =
"C:/Users/tedya/source/repos/DLIP/openpose/build/examples/tutorial_api_python"
    try:
        # Windows Import
        if platform == "win32":
            # Change these variables to point to the correct folder (Release/x64
etc.)
            sys.path.append(dir_path + '/../../python/openpose/Release')
            os.environ['PATH'] = os.environ['PATH'] + ';' + dir_path +
'/../../x64/Release;' + dir_path + '/../../bin;'
            import pyopenpose as op
        else:
            # Change these variables to point to the correct folder (Release/x64
etc.)

 sys.path.append('C:/Users/tedya/source/repos/DLIP/openpose/build/python')
            # If you run `make install` (default path is `/usr/local/python` for
Ubuntu), you can also access the OpenPose/python module from there. This will
install OpenPose and the python library at your desired installation path. Ensure
that this is in your python path in order to use it.
```

```
            # sys.path.append('/usr/local/python')
            from openpose import pyopenpose as op
    except ImportError as e:
        print('Error: OpenPose library could not be found. Did you enable
`BUILD_PYTHON` in CMake and have this Python script in the right folder?')
        raise e

    # Flags
    parser = argparse.ArgumentParser()
    parser.add_argument("--hand", action="store_true", help="Enable hand
keypoint detection.")
    args = parser.parse_known_args()

    # Custom Params (refer to include/openpose/flags.hpp for more parameters)
    params = dict()

    # 모델 폴더 절대 경로 설정
    params["model_folder"] = "C:/Users/tedya/source/repos/DLIP/openpose/models"
    params["body"] = 1  # 1은 관절 검출을 활성화하는 옵션입니다.
    params["disable_multi_thread"] = True  # 멀티 스레드 비활성화

    # Starting OpenPose
    opWrapper = op.WrapperPython()
    opWrapper.configure(params)
    opWrapper.start()

    # Set up webcam
    cap = cv2.VideoCapture('walking04.mp4')
    #cap = cv2.VideoCapture(0)
    # 저장할 동영상 파일 경로 및 설정
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    # 자를 동영상의 속성 설정
    output_width = width
    output_height = height
    output_path = name + '_Walking_analysis.avi'
    fourcc = cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')  # 동영상 코덱 설정
('MJPG')
    fps = cap.get(cv2.CAP_PROP_FPS)  # 원본 동영상의 FPS 가져오기
    output = cv2.VideoWriter(output_path, fourcc, fps, (output_width,
output_height))

    while True:
        # Read frame from webcam
        ret, frame = cap.read()

        frame_count = frame_count + 1
        if not ret:
            break

        # Create new datum
        datum = op.Datum()
        datum.cvInputData = frame

        # image size
        Height, Width, _ = frame.shape

        # Fps
```

```
        fps = cap.get(cv2.CAP_PROP_FPS)

        # Process frame
        opWrapper.emplaceAndPop(op.VectorDatum([datum]))

        # 관절 및 신체 파츠 그리기
        keypoints = datum.poseKeypoints


        rankle_pred = rankle
        lankle_pred = lankle
        rheel_pred = rheel
        lheel_pred = lheel
        ltoe_pred = ltoe
        rtoe_pred = rtoe

        if keypoints is not None:
            for person in keypoints:
                neck = person[BODY_PARTS["Neck"]]
                rhip = person[BODY_PARTS["RHip"]]
                rknee = person[BODY_PARTS["RKnee"]]
                rankle = person[BODY_PARTS["RAnkle"]]
                lhip = person[BODY_PARTS["LHip"]]
                lknee = person[BODY_PARTS["LKnee"]]
                lankle = person[BODY_PARTS["LAnkle"]]
                lheel = person[BODY_PARTS["LHeel"]]
                rtoe = person[BODY_PARTS["RBigToe"]]
                rheel = person[BODY_PARTS["RHeel"]]
                ltoe = person[BODY_PARTS["LBigToe"]]


                # Check if toe keypoints are missing
                if np.all(rtoe == 0):
                    rtoe = prev_rtoe
                else:
                    prev_rtoe = rtoe

                if np.all(ltoe == 0):
                    ltoe = prev_ltoe
                else:
                    prev_ltoe = ltoe

                # Check if heel keypoints are missing
                if np.all(rheel == 0):
                    rheel = prev_rheel
                else:
                    prev_rheel = rheel

                if np.all(lheel == 0):
                    lheel = prev_lheel
                else:
                    prev_lheel = lheel

                # 관절 그리기
                for idx, point in enumerate(person):
                    x, y, confidence = point
                    if confidence > 0.5:
```

```
                        cv2.circle(frame, (int(x), int(y)), 2, (0, 255, 255),
-1)
                        cv2.putText(frame, str(idx), (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)

                draw_body_connections(frame, person)
                rheel_y_coords.append(rheel[1])
                lheel_y_coords.append(lheel[1])


        if lheel_max < lheel[1]:
            lheel_max = lheel[1]

        if rheel_max < rheel[1]:
            rheel_max = rheel[1]

        if rtoe_max < rtoe[1]:
            rtoe_max = rtoe[1]

        if frame_count == 20:
            ground_toe = rtoe_max
            ground_right = rheel_max - 3
            ground_left = lheel_max - 3
            ground = rheel_max - 2
            Start_flag = True
        if Start_flag == True:

            if rheel[1] >=ground_right and rheel_pred[1] < ground_right and
rheel[0] > lheel[0]:
                initial_flag = True

        # Initial contact
        if initial_flag == True:
            if initial_contact_flag == True:
                if rheel[1] >= ground_right and rheel_pred[1] < ground_right :
                    initial_contact.append(frame_count)
                    initial_contact_flag = False
                    loading_flag = True
                    contact.append(rheel[0])
                    contact.append(rheel[1])
                    contact.append(rtoe[0])
                    contact.append(rtoe[1])

            # Mid Stance1
            if loading_flag == True:
                if ltoe[1] < ground_left and rheel[1] >= ground_right and
lheel[0] < rheel[0] :
                    loading.append(frame_count)
                    loading_flag = False
                    mid_flag = True

            # Terminal Stance1
            if mid_flag == True:
                if lheel[0] >= rheel[0] and rtoe[1] >= ground_right and lheel[1]
< ground_left:
                    mid.append(frame_count)
                    mid_flag = False
                    terminal_flag = True
```

```
            # Preswing
            if terminal_flag == True:
                if lheel[1] >= ground_left and rtoe[1] >= ground_right:
                    terminal.append(frame_count)
                    terminal_flag = False
                    preswing_flag = True

            #오른발 스윙

            #Initial Swing
            #왼발 닿는 순간 초기화
            if preswing_flag == True:

                if lheel[1] >= ground_left and lheel_pred[1] < ground_left :
                    preswing_flag = False
                    loading_swing_flag = True

            #Mid Swing
            if loading_swing_flag == True:

                if rtoe[1] < ground_right and lheel[1] >= ground_left and
rheel[0] < lheel[0]:
                    preswing.append(frame_count)
                    loading_flag = False
                    mid_swing_flag = True

            #Terminal Swing
            if mid_swing_flag == True:
                if rheel[0] >= lheel[0] and ltoe[1] >= ground_left and rheel[1]
< ground_right:
                    initial_swing.append(frame_count)
                    mid_swing_flag = False
                    mid_swing_flag2 = True

            if mid_swing_flag2 == True:
                if 3>(rheel[1]-rtoe[1])>-3:
                    mid_swing.append(frame_count)
                    mid_swing_flag2 = False
                    terminal_swing_flag = True
                    initial_contact_flag = True

        Frame_num.append(frame_count)

        #Hip Joint Flexion
        Len1 = math.sqrt(math.pow(neck[0] - rhip[0], 2) + math.pow(neck[1] -
rhip[1], 2))
        Len2= math.sqrt(math.pow(rhip[0] - rknee[0], 2) + math.pow(rhip[1] -
rknee[1], 2))
        Len3 = math.sqrt(math.pow(rknee[0] - neck[0], 2) + math.pow(rknee[1] -
neck[1], 2))
        if Len1 == 0 or Len2 == 0 or Len3 == 0:
            HJFradian = HJFradian
        else:
            cos_angle = (Len1 * Len1 + Len2 * Len2 - Len3 * Len3) / (2 * Len1 *
Len2 +1.e-09)
            if -1 <= cos_angle <= 1:
                HJFradian = math.acos(cos_angle)
            else:
```

```
                HJFradian = 0
        HJFdegree.append(HJFradian / np.pi * 180)
        #Right Knee Flexion
        Len1= math.sqrt(math.pow(rhip[0] - rknee[0], 2) + math.pow(rhip[1] -
rknee[1], 2))
        Len2 = math.sqrt(math.pow(rknee[0] - rankle[0], 2) + math.pow(rknee[1] -
rankle[1], 2))
        Len3 = math.sqrt(math.pow(rankle[0] - rhip[0], 2) + math.pow(rankle[1] -
rhip[1], 2))
        if Len1 == 0 or Len2 == 0 or Len3 == 0:
            RKFradian = RKFradian
        else:
            cos_angle = (Len1 * Len1 + Len2 * Len2 - Len3 * Len3) / (2 * Len1 *
Len2 +1.e-09)
            if -1 <= cos_angle <= 1:
                RKFradian = math.acos(cos_angle)
            else:
                RKFradian = 0
        RKFdegree.append(RKFradian / np.pi * 180)
        #Right Dorsi Flexion
        Len1= math.sqrt(math.pow(rankle[0] - rheel[0], 2) + math.pow(rankle[1] -
rheel[0], 2))
        Len2 = math.sqrt(math.pow(rheel[0] - rtoe[0], 2) + math.pow(rheel[1] -
rtoe[0], 2))
        Len3 = math.sqrt(math.pow(rtoe[0] - rankle[0], 2) + math.pow(rtoe[1] -
rankle[0], 2))
        if Len1 == 0 or Len2 == 0 or Len3 == 0:
            RDradian = RDFradian
        else:
            cos_angle = (Len1 * Len1 + Len2 * Len2 - Len3 * Len3) / (2 * Len1 *
Len2 +1.e-09)
            if -1 <= cos_angle <= 1:
                RDFradian = math.acos(cos_angle)
            else:
                RDFradian = 0
        RDFdegree.append(RDFradian / np.pi * 180)


        # 프레임 표시

        # 텍스트 그리기
        cv2.putText(frame, f"Frame : {frame_count}", (20,30),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.putText(frame, f"Hip joint : {HJFdegree[-1]:.2f}", (20,280),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.putText(frame, f"Right knee : {RKFdegree[-1]:.2f}", (20,310),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.putText(frame, f"Right dorsi : {RDFdegree[-1]:.2f}", (20,340),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,255),2)
        cv2.imshow("OpenPose Webcam", frame)
        # 프레임 저장
        output.write(frame)
        # Exit if 'q' is pressed
        if cv2.waitKey(1) == ord('q'):
            break

    print(ground_right)
```

```
    analysis_text.write(f"{name}\n")

    # initial contact
    if HJFdegree[initial_contact[0]] < 140:
        analysis_text.write("When your right heel touches the ground, stretch
your hip more!\n")
    if HJFdegree[initial_contact[0]] > 150:
        analysis_text.write("When your right heel touches the ground, bend your
hip more!\n")
    if RKFdegree[initial_contact[0]] < 167:
        analysis_text.write("When your right heel touches the ground, stretch
your knee more!\n")
    if RKFdegree[initial_contact[0]] > 177:
        analysis_text.write("When your right heel touches the ground, bend your
knee more!\n")

    # Loading response
    if HJFdegree[loading[0]] < 145:
        analysis_text.write("When your left heel lifted from the ground, stretch
your hip more!\n")
    if HJFdegree[loading[0]] > 155:
        analysis_text.write("When your left heel lifted from the ground, bend
your hip more!\n")
    if RKFdegree[loading[0]] < 160:
        analysis_text.write("When your left heel lifted from the ground, stretch
your knee more!\n")
    if RKFdegree[loading[0]] > 170:
        analysis_text.write("When your left heel lifted from the ground, bend
your knee more!\n")

    # Mid stance
    if HJFdegree[mid[0]] < 169:
        analysis_text.write("When your left foot is ahead of your right foot,
stretch your hip more!\n")
    if HJFdegree[mid[0]] > 179:
        analysis_text.write("When your left foot is ahead of your right foot,
bend your hip more!\n")
    if RKFdegree[mid[0]] < 163:
        analysis_text.write("When your left foot is ahead of your right foot,
stretch your knee more!\n")
    if RKFdegree[mid[0]] > 173:
        analysis_text.write("When your left foot is ahead of your right foot,
bend your knee more!\n")

    # Terminal stance
    if HJFdegree[terminal[0]] < 169:
        analysis_text.write("When your left heel touches the ground, bend your
hip more!\n")
    if HJFdegree[terminal[0]] > 179:
        analysis_text.write("When your left heel touches the ground, stretch
your hip more!\n")
    if RKFdegree[terminal[0]] < 167:
        analysis_text.write("When your left heel touches the ground, stretch
your knee more!\n")
    if RKFdegree[terminal[0]] > 177:
        analysis_text.write("When your left heel touches the ground, bend your
knee more!\n")
```

```
    # preswing
    if HJFdegree[preswing[0]] < 175:
        analysis_text.write("When your right heel is lifted from the ground,
stretch your hip more!\n")
    if HJFdegree[preswing[0]] > 150:
        analysis_text.write("When your right heel is lifted from the ground,
bend your hip more!\n")
    if RKFdegree[preswing[0]] < 140:
        analysis_text.write("When your right heel is lifted from the ground,
stretch your knee more!\n")
    if RKFdegree[preswing[0]] > 150:
        analysis_text.write("When your right heel is lifted from the ground,
bend your knee more!\n")


    # Initial swing
    if HJFdegree[initial_swing[0]] < 155:
        analysis_text.write("When your right foot is ahead of your left foot,
stretch your hip more!\n")
    if HJFdegree[initial_swing[0]] > 165:
        analysis_text.write("When your right foot is ahead of your left foot,
bend your hip more!\n")
    if RKFdegree[initial_swing[0]] < 111:
        analysis_text.write("When your right foot is ahead of your left foot,
stretch your knee more!\n")
    if RKFdegree[initial_swing[0]] > 121:
        analysis_text.write("When your right foot is ahead of your left foot,
bend your knee more!\n")

    # Mid swing
    if HJFdegree[mid_swing[0]] < 138:
        analysis_text.write("When your right and left feet are parallel, stretch
your hip more!\n")
    if HJFdegree[mid_swing[0]] > 148:
        analysis_text.write("When your right and left feet are parallel, bend
your hip more!\n")
    if RKFdegree[mid_swing[0]] < 140:
        analysis_text.write("When your right and left feet are parallel, stretch
your knee more!\n")
    if RKFdegree[mid_swing[0]] > 150:
        analysis_text.write("When your right and left feet are parallel, bend
your knee more!\n")

    result = find_indexes_repeated(rheel_y_coords,ground_right)
    print(result)
    print_index_differences(result)
    analysis_text.close()


    # # velocity
    # foot_size2 = math.sqrt(math.pow(contact[0]-
contact[2],2)+math.pow(contact[1]-contact[3],2))
    # real_width = float(foot_size) * Width / foot_size2 / 100
    # m = real_width * (contact[2]-contact[0]) / Width
    # v = m/(result[1] - result[0])*fps
    # print(v)
```

```
    # Draw plot
    plt.figure()
    plt.plot(Frame_num, HJFdegree, label='Hip Joint Flexion')
    plt.plot(Frame_num, RKFdegree, label='Right Knee Flexion')
    plt.plot(Frame_num, RDFdegree, label='Right Dorsi Flexion')

    #loading response
    plt.axvline(initial_contact[0], linewidth=2, color='r')
    plt.text(initial_contact[0]+0.2, 15, 'initial contact', color='b',
fontsize=12)
    plt.hlines(120, initial_contact[0], loading[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text((initial_contact[0]+1), 121, 'loading response', color='b',
fontsize=12)

    #mid stance
    plt.axvline(loading[0], linewidth=2, color='r')
    plt.hlines(105, loading[0], mid[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text((loading[0]+1), 106, 'mid stance', color='b', fontsize=12)

    # terminal stance
    plt.axvline(mid[0], linewidth=2, color='r')
    plt.hlines(90, mid[0], terminal[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text((mid[0]+1), 91, 'terminal stance', color='b', fontsize=12)

    #preswing
    plt.axvline(terminal[0], linewidth=2, color='r')
    plt.hlines(75, terminal[0], preswing[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text(terminal[0]+1, 76, 'preswing', color='b', fontsize=12)

    #initial swing
    plt.axvline(preswing[0], linewidth=2, color='r')
    plt.hlines(60, preswing[0], initial_swing[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text(preswing[0]+1, 61, 'initial swing', color='b', fontsize=12)

    # mid swing
    plt.axvline(initial_swing[0], linewidth=2, color='r')
    plt.hlines(45, initial_swing[0], mid_swing[0], linewidth=1, color='r',
linestyle='dashed')
    plt.text(initial_swing[0]+1, 46, 'mid swing', color='b', fontsize=12)

    #terminal swing
    plt.axvline(mid_swing[0], linewidth=2, color='r')
    plt.hlines(30, mid_swing[0], initial_contact[1], linewidth=1, color='r',
linestyle='dashed')
    plt.text(mid_swing[0]+1, 31, 'terminal swing', color='b', fontsize=12)
    plt.axvline(initial_contact[1], linewidth=2, color='r')


    plt.xlabel('Frame Number')
    plt.ylabel('Degree')
    plt.title(name + ' Joint Flexion')
    plt.legend(loc='upper right')
    plt.grid(True)
```

```
    plt.show()

    # Release webcam and close windows
    output.release()
    cap.release()
    cv2.destroyAllWindows()


except Exception as e:
    print(e)
    sys.exit(-1)
```

Reference

About the Openpose

- https://arxiv.org/pdf/1611.08050.pdf
- https://velog.io/@oneul1213/OpenPose-%EA%B0%9C%EC%9A%94-%EB%B0%8F-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0

About the Gait Analysis

- https://koreascience.kr/article/JAKO199609409672368.pdf
- https://blog.naver.com/choiyuwon/221383858281
- https://blog.naver.com/choiyuwon/220549212812