學號：408410054
姓名：蕭子祥
系級：資工三
信箱：900509@gmail.com

Final Project 詳細實驗步驟與報告

實驗名稱：

Body Detection - am I right while working out ?

實驗目的：

如何利用 raspberry pi 3 和 OpenCV 套件，即時檢查運動時的姿勢正不正確。

實驗步驟：

Part 1 安裝 Raspberry Pi 3 OS

安裝作業系統到官網 https://www.raspberrypi.com/software/operating-systems/ 下載安裝檔。

下載 Raspberry Pi OS (Legacy) with desktop

安裝好作業系統後，進行系統更新及升級：

```
sudo apt-get update
sudo apt-get upgrade
```

Part 2 安裝 OpenCV 及其所需要的套件：

```
sudo apt update
sudo apt upgrade
sudo pip3 install -U numpy
sudo apt install ffmpeg python3-opencv python3-pip
sudo apt install libxcb-shm0 \
    libcdio-paranoia-dev \
    libsdl2-2.0-0 libxv1 \
    libtheora0 libva-drm2 \
    libva-x11-2 \
    libvdpau1 \
    libharfbuzz0b \
    libbluray2 \
    libatlas-base-dev \
    libhdf5-103 \
    libgtk-3-0 \
    libdc1394-22 \
    libopenexr23
sudo pip3 install mediapipe-rpi3
```

Part 3 在板子上及個人電腦安裝 VNC 及其所需要的套件：

PS：如果用 HDMI 線 直接連到 螢幕就可以挑過這一步

```
# 在 Pi 上安裝 VNC 伺服器
sudo apt-get install tightvncserver

# 在個人電腦安裝 VNC 用戶端
sudo apt-get install vncviewer gtkvncviewer

#在 Pi 上啟動 vncserver
vncserver

#設定密碼
You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n

# 之後我們就可以透過 vncviewer 或是 gtkvncviewer 之類的軟體和 Pi 連線了。假設 Pi 的 IP 為 192.168.1.2。
vncviewer 192.168.1.2:5901
```

Part 4 寫偵測的程式：

```
mkdir Final
cd Final
```

並加入下面三個檔案：

1. gym_small_gui.py

```
from tkinter import *
import cv2
from PIL import Image, ImageTk
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

import squat
import dumbbell

camera = cv2.VideoCapture(0)  # 攝像頭
x = "initial"
global button_click
button_click = False

def video_loop():
    success, img = camera.read()  # 從攝像頭讀取照片
    if success and button_click == False:
        img = cv2.flip(img,1)
        cv2image = cv2.cvtColor(img, cv2.COLOR_BGR2RGBA)  # 轉換顏色從BGR到RGBA
        current_image = Image.fromarray(cv2image)  # 將圖像轉換成Image對象
        imgtk = ImageTk.PhotoImage(current_image)
        panel.imgtk = imgtk
        panel.config(image=imgtk)
    elif success and button_click == "squat":
        squat.do_squat(lbl_1, panel, camera, root)
    elif success and button_click == "dumbbell":
        dumbbell.do_dumbbell(lbl_1, panel, camera, root)
    root.after(1, video_loop)

def prepare_squat():
    global button_click
    button_click = "squat"

def prepare_dumbbell():
    global button_click
    button_click = "dumbbell"

global root
root = Tk()
root.title("opencv + tkinter")
#root.protocol('WM_DELETE_WINDOW', detector)

panel = Label(root)  # initialize image panel
panel.pack(padx=10, pady=10)
root.config(cursor="arrow")
btn = Button(root, text="舉啞鈴(dumbbell)", command=prepare_dumbbell)
btn.pack(fill="both", expand=True, padx=10, pady=10)
btn2 = Button(root, text="深蹲(squat)", command=prepare_squat)
btn2.pack(fill="both", expand=True, padx=10, pady=10)
global lbl_1
lbl_1 = Label(root, text = x)
lbl_1.pack(padx=10, pady=10)

video_loop()

root.mainloop()
# 當一切都完成後，關閉攝像頭並釋放所佔資源
camera.release()
cv2.destroyAllWindows()
```

2. dumbbell.py

```
import cv2
import mediapipe as mp
import numpy as np
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
from tkinter import *
import cv2
from PIL import Image, ImageTk

def calculate_angle(a, b, c):
    a = np.array(a)  # First
    b = np.array(b)  # Mid
    c = np.array(c)  # End

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return float(angle)

## Setup mediapipe instance
def do_dumbbell(lbl_1, panel, camera, root):

    # Curl counter variables
    counter = 0
    stage = None
    arm_max_angle = -1
    arm_min_angle = 999
    idx = 0
    agle = None
    dif = 0
    notification = None
    message = ''
    msg = ""
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)
as pose:
        while camera.isOpened():
            ret, frame = camera.read()
            frame =  cv2.flip(frame,1)

            # Recolor image to RGB
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            # image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark

                # Get coordinates
                shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,

 landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
                elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
                        landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
                wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
                        landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]

                r_shoulder =
[landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,

 landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y]
                r_elbow = [landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,
                        landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].y]
                r_wrist = [landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].x,
                        landmarks[mp_pose.PoseLandmark.RIGHT_WRIST.value].y]
                r_hip = [landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x,
                        landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y]

                # print(r_shoulder)
                # Calculate angle
                hip_angle = calculate_angle(r_elbow, r_shoulder, r_hip)
                angle = calculate_angle(shoulder, elbow, wrist)
                r_angle = calculate_angle(r_shoulder, r_elbow, r_wrist)
                # print(r_angle)

                # Visualize angle
                cv2.putText(image, str(r_angle),
                            tuple(np.multiply(elbow, [640, 480]).astype(int)),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA

                            )

                # save angle
                if idx == 0:
                    agle = float(r_angle)
                idx += 1
                # print(idx)

                # determine the stage
                if idx % 10 == 0:
                    dif = r_angle - agle
                    agle = r_angle

                    if abs(dif) < 2:
                        continue
                    """
                    if hip_angle > 30 :
                        print("wrong")
                        message = "Wrong!!"
                        cv2.rectangle(image, (0, 0), (225, 73), (0, 0, 255), -1)

                        cv2.putText(image, str(message),
                        (10, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2,
cv2.LINE_AA)


                        continue
                    """

                    # print(dif)
                    # down stage
                    if dif > 0:
                        # if stage == None:
                        #     stage = "down"
                        # elif stage == "up":
                        #     stage = "down"
                        stage = "down"
                    # down stage
                    elif dif < 0:
                        if stage == "down":
                            # stage = "up"
                            counter += 1
                        elif stage == None:
                            # stage = "up"
                            counter += 1
                        stage = "up"

                # print(stage)
                if stage == 'up':
                    # reset max angle of this iteration
                    if arm_max_angle != -1:
                        arm_max_angle = -1
                    # find min angle
                    if r_angle <= arm_min_angle:
                        arm_min_angle = r_angle
                    # message
                    if arm_min_angle >= 30:
                        # print(arm_min_angle)
                        message = 'up'
                        # print('keep going up')
                    elif arm_min_angle < 30:
                        message = 'good'

                elif stage == 'down':
                    # reset min angle
                    if arm_min_angle != 999:
                        arm_min_angle = 999
                    # find max angle
                    if r_angle >= arm_max_angle:
                        arm_max_angle = r_angle
                    # message
                    if arm_max_angle <= 150:
                        # print('keep going down')
                        message = 'down'
                    else:
                        message = 'good'

                if hip_angle > 15 :
                        print("wrong")
                        message = "Wrong!!"
            except:
                pass

            # Render curl counter
            # Setup status box
            cv2.rectangle(image, (0, 0), (225, 73), (0, 0, 255), -1)

            cv2.putText(image, str(message),
                        (10, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2,
cv2.LINE_AA)
            # Stage data
            cv2.putText(image, 'STAGE', (65, 12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1,
cv2.LINE_AA)
            # cv2.putText(image, stage,
            #              (60, 60),
            #             cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2,
cv2.LINE_AA)

            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(245, 117, 66),
thickness=2, circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(245, 66, 230),
thickness=2, circle_radius=2)
                                    )

            # cv2.imshow('Output Feed', image)
            cv2image = cv2.cvtColor(image, cv2.COLOR_BGR2RGBA)  # 轉換顏色從BGR到
RGBA
            current_image = Image.fromarray(image)  # 將圖像轉換成Image對象
            imgtk = ImageTk.PhotoImage(current_image)
            panel.imgtk = imgtk
            panel.config(image=imgtk)
            lbl_1.config(text = message)
            root.update()

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

    # cap.release()
    # cv2.destroyAllWindows()
```

3. squat.py

```
import cv2
import mediapipe as mp
import numpy as np
import time
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
from tkinter import *
import cv2
from PIL import Image, ImageTk
import threading


def calculate_angle(a, b, c):
    a = np.array(a)  # First
    b = np.array(b)  # Mid
    c = np.array(c)  # End

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1],
a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return angle
```

```
# old 站立, new 蹲下
def lim_squat_2(old_blen, shoulder, hip):
    new_blen = get_body_len(shoulder, hip)
    # print(new_blen)
    # print(old_blen)
    if new_blen < old_blen*0.75:
        return 0 # 駝背
    return 1 # correct


# 計算軀幹與水平面的角度
def lim_squat_3(a, b):
    a = np.array(a)  # shoulder
    b = np.array(b)  # hip
    c = np.array([b[0]+0.5,b[1]]) # horizon
    angle = calculate_angle(a, b, -c)
    # print("軀幹與水平面: "angle)
    if angle < 35 or angle > 145: # 朝地
        return 0
    return 1 #朝前

def get_body_len(shoulder, hip):
    s = np.array(shoulder)
    h = np.array(hip)
    return ((s[0] - h[0])**2 + (s[1] - h[1])**2)**0.5


# Curl counter variables
def do_squat(lbl_1, panel, camera, root):
    counter = 0
    stage = None
    body_len = 1
    msg = ""
    stage = "up"

    ## Setup mediapipe instance
    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
        while camera.isOpened():
            ret, frame = camera.read()
            frame = cv2.flip(frame,1)

            # Recolor image to RGB
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image.flags.writeable = False

            # Make detection
            results = pose.process(image)

            # Recolor back to BGR
            image.flags.writeable = True
            # image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)



            # Extract landmarks
            try:
                landmarks = results.pose_landmarks.landmark
                # Get coordinates


                l_shoulder = 
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,

 landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
                r_shoulder = 
[landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].x,

 landmarks[mp_pose.PoseLandmark.RIGHT_SHOULDER.value].y]
                l_elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
                            landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
                r_elbow = [landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].x,
                            landmarks[mp_pose.PoseLandmark.RIGHT_ELBOW.value].y]
                l_hip = [landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,
                            landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y]
                r_hip = [landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x,
                            landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].y]
                l_knee = [landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,
                            landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y]
                r_knee = [landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x,
                            landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].y]
                l_knee = [landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,
                            landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y]
                r_knee = [landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x,
                            landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].y]
                l_ankle = [landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].x,
                            landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].y]
                r_ankle = [landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].x,
                            landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].y]


                # Calculate angle
                #angle = calculate_angle(l_shoulder, l_elbow, wrist)
                l_angle = calculate_angle(l_hip, l_knee, l_ankle)
                body_angle = calculate_angle(l_shoulder, l_hip, l_ankle)

                # print(landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x)
                # Visualize angle
                '''
                cv2.putText(image, str(angle),
                            tuple(np.multiply(elbow, [640, 480]).astype(int)),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, 
cv2.LINE_AA
                            )
                '''


                # Curl counter logic
                # l:(hip, knee, ankle), body:(shoulder, hip, ankle)
                if l_angle > 160 and body_angle > 160:
                    body_len = get_body_len(l_shoulder, l_hip)
                    stage = "up"
                if l_angle < 90 and stage == "up":  #蹲下時增加次數
                    stage = "down"
                    counter += 1
                    print(counter)
                    msg = ""
                    if lim_squat_2(body_len, l_shoulder, l_hip) == 0:
                            msg = "背一定要打直。"
                        # # 判斷身體跟水平面的夾角
                    if lim_squat_3(l_shoulder, l_hip) == 0 or
lim_squat_3(r_shoulder, r_hip) == 0:
                        msg = msg + "胸朝前方，不要朝地板。"
                    if msg == "":
                        msg = "做得不錯！"

            except:
                pass

            # Render curl counter
            # Setup status box
            cv2.rectangle(image, (0, 0), (225, 73), (0, 0, 255), -1)

            cv2.putText(image, str(counter),
                        (10, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2,
cv2.LINE_AA)

            # Stage data
            cv2.putText(image, 'STAGE', (65, 12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1,
cv2.LINE_AA)
            cv2.putText(image, stage,
                        (60, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2,
cv2.LINE_AA)

            # Render detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                        mp_drawing.DrawingSpec(color=(245, 117,
66), thickness=2, circle_radius=2),
                                        mp_drawing.DrawingSpec(color=(245, 66,
230), thickness=2, circle_radius=2)
                                        )

            if cv2.waitKey(10) & 0xFF == ord('q'):
                break
            cv2image = cv2.cvtColor(image, cv2.COLOR_BGR2RGBA)  # 轉換顏色從BGR到
RGBA
            current_image = Image.fromarray(image)  # 將圖像轉換成Image對象
            imgtk = ImageTk.PhotoImage(current_image)
            panel.imgtk = imgtk
            panel.config(image=imgtk)
            lbl_1.config(text = msg)
            root.update()
        # camera.release()
        # cv2.destroyAllWindows()
```

Part 5 執行：

```
python3 /home/pi/Final/gym_small_gui.py
```

即可在螢幕上呈現結果。