工業物聯網

利彥儒 n96104103 廖沁旋 n96104080

一、 過程(PY. CODE)

1-1 跌倒判斷系統

```python
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util
from enum import Enum
import math
from waring import
class BodyPart(Enum):
    NOSE = 0,
    LEFT_EYE = 1,
    RIGHT_EYE = 2,
    LEFT_EAR = 3,
    RIGHT_EAR = 4,
    LEFT_SHOULDER = 5,
    RIGHT_SHOULDER = 6,
    LEFT_ELBOW = 7,
    RIGHT_ELBOW = 8,
    LEFT_WRIST = 9,
    RIGHT_WRIST = 10,
    LEFT_HIP = 11,
    RIGHT_HIP = 12,
    LEFT_KNEE = 13,
    RIGHT_KNEE = 14,
    LEFT_ANKLE = 15,
    RIGHT_ANKLE = 16,
class Position:
    def __init__(self):
        self.x = 0
        self.y = 0
class KeyPoint:
    def __init__(self):
        self.bodyPart = BodyPart.NOSE
        self.position = Position()
        self.score = 0.0

class Person:
    def __init__(self):
        self.keyPoints = []
        self.score = 0.0

class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(640,480),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()
```

```
        # Variable to control when the camera is stopped
            self.stopped = False

        def start(self):
        # Start the thread that reads frames from the video stream
            Thread(target=self.update,args=()).start()
            return self

        def update(self):
            # Keep looping indefinitely until the thread is stopped
            while True:
                # If the camera is stopped, stop the thread
                if self.stopped:
                    # Close camera resources
                    self.stream.release()
                    return

                # Otherwise, grab the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()
```

```
        def read(self):
        # Return the most recent frame
            return self.frame

        def stop(self):
        # Indicate that the camera and thread should be stopped
            self.stopped = True

    # Define and parse input arguments
    def sigmoid(x):
            return 1. / (1. + math.exp(-x))

    AL_FRAME=20
    COUNTER=0
    min_threshold = float(0.1)
    GRAPH_NAME = 'posenet_mobilenet_v1_100_257x257_multi_kpt_stripped.tflite'
    min_conf_threshold = float(0.4)
    resW=640
    resH=480
    imW, imH = int(resW), int(resH)

    pkg = importlib.util.find_spec('tflite_runtime')
    if pkg:
        from tflite_runtime.interpreter import Interpreter
    else:
        from tensorflow.lite.python.interpreter import Interpreter

    # Get path to current working directory
    CWD_PATH = os.getcwd()

    # Path to .tflite file, which contains the model that is used for object detection
    PATH_TO_CKPT = os.path.join(CWD_PATH,GRAPH_NAME)

    # If using Edge TPU, use special load_delegate argument

    interpreter = Interpreter(model_path=PATH_TO_CKPT)

    interpreter.allocate_tensors()
```

```
    # Get model details
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    height = input_details[0]['shape'][1]
    width = input_details[0]['shape'][2]

    floating_model = (input_details[0]['dtype'] == np.float32)
    input_mean = 127.5
    input_std = 127.5
    state=0

    # Initialize frame rate calculation
    frame_rate_calc = 1
    freq = cv2.getTickFrequency()

    # Initialize video stream
    videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
    time.sleep(1)
    fr=0
    state=0
    foot=0
    head=0
    #for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):
    while True:
        fr+=1
        maxh=0
        maxf=0
        # Start timer (for calculating frame rate)
        t1 = cv2.getTickCount()

        # Grab frame from video stream
        frame1 = videostream.read()

        # Acquire frame and resize to expected shape [1xHxWx3]
        frame = frame1.copy()
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_resized = cv2.resize(frame_rgb, (width, height))
        input_data = np.expand_dims(frame_resized, axis=0)

        # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
        if input_details[0]['dtype'] == type(np.float32(1.0)):
                input_data = (np.float32(input_data) - input_mean) / input_std
```

```
        # Perform the actual detection by running the model with the image as input
        interpreter.set_tensor(input_details[0]['index'],input_data)
        interpreter.invoke()
        heat_maps = interpreter.get_tensor(output_details[0]['index'])
        offset_maps = interpreter.get_tensor(output_details[1]['index'])
        h_pose=len(heat_maps[0])
        w_pose=len(heat_maps[0][0])
        num_key_points = len(heat_maps[0][0][0])
        # Loop over all detections and draw detection box if confidence is above minimum threshold
        key_point_positions = [[0] * 2 for i in range(num_key_points)]
        for key_point in range(num_key_points):
            max_val = heat_maps[0][0][0][key_point]
            max_row = 0
            max_col = 0
            for row in range(h_pose):
                for col in range(w_pose):
                    heat_maps[0][row][col][key_point] = sigmoid(heat_maps[0][row][col][key_point])
                    if heat_maps[0][row][col][key_point] > max_val:
```

```
            for col in range(w_pose):
                heat_maps[0][row][col][key_point] = sigmoid(heat_maps[0][row][col][key_point])
                if heat_maps[0][row][col][key_point] > max_val:
                    max_val = heat_maps[0][row][col][key_point]
                    max_row = row
                    max_col = col
        key_point_positions[key_point] = [max_row, max_col]
        #print(key_point_positions[key_point])
    x_coords = [0] * num_key_points
    y_coords = [0] * num_key_points
    confidenceScores = [0] * num_key_points
    for i, position in enumerate(key_point_positions):
            position_y = int(key_point_positions[i][0])
            position_x = int(key_point_positions[i][1])

            y_coords[i] = (position[0] / float(h_pose - 1) * imH +
                           offset_maps[0][position_y][position_x][i])
            x_coords[i] = (position[1] / float(w_pose - 1) * imW +
                           offset_maps[0][position_y][position_x][i + num_key_points])
            #print('(x,y):',x_coords[i],y_coords[i])
            confidenceScores[i] = heat_maps[0][position_y][position_x][i]
            #print("confidenceScores[", i, "] = ", confidenceScores[i])
            x=int(x_coords[i])
            y=int(y_coords[i])
            #print('x=',x,'y=',y,'confidence=%.2f' %confidenceScores[i])

            if(confidenceScores[i]>0.4 and i<5):
                if(confidenceScores[i]>maxh):
                    maxh=confidenceScores[i]
                    head=int(y_coords[i])
            elif(confidenceScores[i]>0.4 and i>10):
                if(confidenceScores[i]>maxf):
                    maxf=confidenceScores[i]
                    foot=int(y_coords[i])
            if(confidenceScores[i]>0.4):
                cv2.circle(frame, (x,y), 5, (0, 255, 0), cv2.FILLED)
    if(head-foot<50):
            cv2.putText(frame,'FALL',(30,200),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),2,cv2.LINE_AA)
            if(state==0):
                waring_message();
                print("send waring message!")
                state=1
    else:
            cv2.putText(frame,'OK',(30,200),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)
            state=0
    cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

    # All the results have been drawn on the frame, so it's time to display it.
    cv2.imshow('Object detector', frame)

    # Calculate framerate
    t2 = cv2.getTickCount()
    time1 = (t2-t1)/freq
    frame_rate_calc= 1/time1
    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break
# Clean up
cv2.destroyAllWindows()
videostream.stop()

```

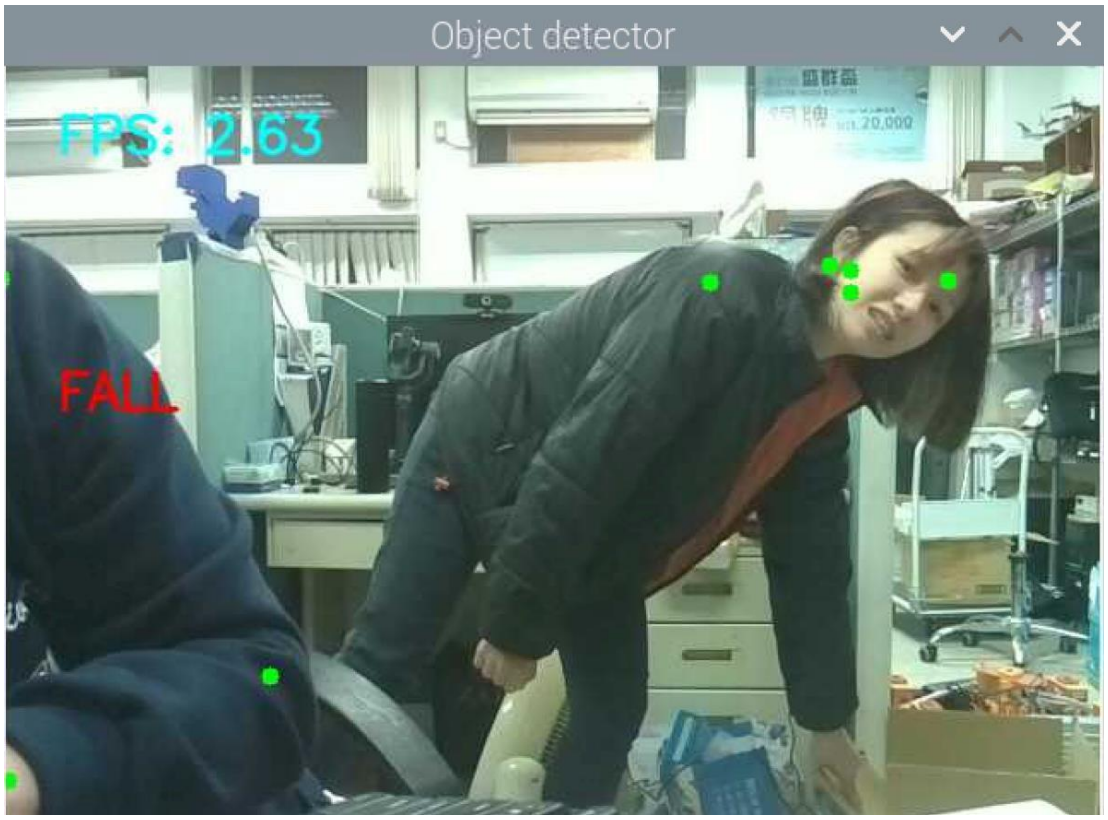
1-2 偵測到跌倒後發送郵件

```python
import smtplib
from email.mime.text import MIMEText

def waring_message():
    gmail_user = 'n96104103@gs.ncku.edu.tw'
    gmail_password = ''''''''''''''' your gmail password

    msg = MIMEText('someone fall !!!')
    msg['Subject'] = 'Test'
    msg['From'] = gmail_user
    msg['To'] = 'leo210370@gmail.com'

    server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
    server.ehlo()
    server.login(gmail_user, gmail_password)
    server.send_message(msg)
    server.quit()
    print('Email sent!')
    #https://accounts.google.com/DisplayUnlockCaptcha
```

二、 執行

在 cmd 中啟動我們偵測跌倒的系統

假裝跌倒



三、 結果

Cmd 中的訊息

信箱所收到的訊息

四、 心得

此次實驗利用 Tensorflow-Lite 框架於樹莓派上運行，以 python 為基礎影像辨識，協助判斷是否有跌倒的行為。如今，影像辨識的研究非常熱門，而將其應用在樹梅派上便可達成行動辨識。然而，在此次過程中，我們的 FPS 只有 2.65，因此有嚴重 delay 的問題，若能改善此部分的話，也許對於跌倒的偵測準確率會大幅提高。此外，由於我們使用的進度畫素不高，因此可能會影響機器抓取其特徵而造成偏差，因此若能更換成更加的鏡頭也可以讓準確度提高。