FCU ePaper

逢甲大學學生報告 ePaper

報告題名：

智慧安控系統

Intelligent security control system

作者：蔡欣穎、劉祉傘、侯采汶、董子嫣

系級：自控三甲

學號：D0952564、D0952253、D0952581、D0988323

開課老師：梁詩婷 老師

課程名稱：智慧辨識檢測與應用

開課系所：資電綜班

開課學年：一百一十一學年度 第一學期

摘要

本專題研究主要目的為辨識出人體的各處關節點與姿勢，並判斷偵測中的人是否做出翻牆動作進而觸發警報，以達到安控的目的。

使用者可以利用外接鏡頭並擺放至需要偵測的場所位置，並依照需求調整觸發警報之牆位高度以及斜率，此系統會藉由偵測腰部節點是否超過偵測線以判斷畫面中人物是否正在進行翻牆的動作，如此一來可以避免經過牆前的行人以及偶然在牆下蹲下撿東西或綁鞋帶的人誤觸警報。

另外，此系統將會即時偵測並計算出現在畫面中的人數以及觸發警報之人數，並顯示在畫面上告知使用者，使用者可以藉由人物身上的紅框快速辨別觸發警報的人物。系統在每次有人觸發危險警報時，會在電腦中建檔將觸發警報時之螢幕畫面儲存為照片，並以當時之時間日期命名資料夾，讓使用者後續得以方便快速的查找偵測紀錄。在本專題研究中，此系統已能準確偵測人體並達到以上功能。

關鍵字：人體模型偵測、人體關節點捕捉、翻牆偵測、即時監控

Abstract

The main purpose of this independent study is to identify the joints and postures of the human body, and determine whether the detected person makes a movement over the wall to trigger an alarm to achieve the purpose of security control.

Users can use external lenses and place them in the place that needs to be detected, and adjust the height and slope of the wall that triggers the alarm according to their needs, the system can detect whether the waist node exceeds the detection line to determine whether the character in the picture is moving over the wall, so as to avoid pedestrians passing in front of the wall and people who accidentally squat under the wall to pick up things or tie their shoes by mistake.

In addition, the system will detect and calculate the number of people appearing in the screen and the number of people who trigger the alarm in real time, and display it on the screen to inform the user, and the user can quickly identify the person who triggered the alarm through the red frame on the character. Every time someone triggers a danger alarm, the system will create a file in the computer to save the screen when the alarm is triggered as a photo, and name the folder with the time and date at that time, so that users can easily and quickly find the detection record in the future. In this independent study, this system has been able to accurately detect the human body and achieve the above functions.

Keyword：Mannequin detection, human joint point capture, wall detection, real-time monitoring

目錄

圖目錄

第1章 研究動機與背景

至今為止曾經在新聞上看過許多關於校園中有學生在上課期間翻牆出校園或歹徒擅闖民宅，翻牆進入他人住宅行竊、行兇的社會案件，不論何者皆存有安全上很大的疑慮，然而目前以單純的人力或勸導還是無法根治這項問題，於是我們希望能以智慧辨識做出一個安控系統，以補足人力無法確實做到的外圍安全控管。

第2章 文獻探討

2.1 TensorFlow

TensorFlow 是一套由 Google 所發展的開放原始碼機器學習函式庫，其以流程圖的概念呈現整個資料分析流程，在流程圖中的每一個節點都代表一個運算，連接不同節點的連線則代表資料的傳遞，程式設計者可以運用各種不同的運算節點(不同的演算法)，組合成適用於各種問題的分析系統，運用 CPU 或 GPU 進行運算。

2.2 NumPy

NumPy 是 Python 語言的一個擴充程式庫。特色是它可以支援多維的陣列或矩陣，裡頭包含強大的數學函示庫，因為是平行運算，所以當資料量很大的話，可以比單純使用 list，還要快上很多。

NumPy 是參考 CPython 創造出來的，因為 Python 在執行數學運算上，直譯器的程式碼會比編譯器來的慢許多，因此特別為 NumPy 引入多維矩陣與陣列的數學函式，使的它的速度可以跟編譯器一樣快。

智慧安控系統

2.3 OpenCV

OpenCV 全名是 Open Source Computer Vision Library，是當今最知名、也最被廣泛採用的影像處理函式庫，它是由 Intel 發起並參與開發，以 BSD 授權條款發行，可在商業和研究領域中免費使用，目前是非營利的基金組織 OpenCV.org 進行維護。

OpenCV 在影像處理方面應用廣泛，可以讀取儲存圖片、視訊、矩陣運算、統計、影像處理等，可用在物體追蹤、人臉辨識、紋理分析、動態視訊的影像處理等。目前 OpenCV 模組如下表所列（圖一）。

模組	模組功能
core	資料類型,資料結構,記憶體管理
Highgui	讀寫圖檔 螢幕輸入處理 簡單的UI功能
Imgproc	圖形篩選(filtering) 幾何圖形轉變 形狀分析
Calib3d	照相機校準(Camera calibration) 多視角3D重建
Feature2d	特徵萃取 描敘 比對
Video	影像物件追蹤與移動分析
Objdetect	用階梯式(cascade)與漸層色階分佈(histogram-of-gradient) 歸類法做物件辨識
ML	用於影像處理的統計模式與歸類演算法
Flann	全名是Fast Library for Approximate Nearest Neighbors是用於高維度資料的快速搜尋
GPU	以選擇的平行運算法在GPU快速執行
Stitching	影像接合處理(stitching)的方法：彎曲 混合 集體調整(bundle adjustment)
Viz	opencv主要都是處理二維的圖像 這一模組就是透過使用VTK三維的功能作三維圖像處理

圖一 OpenCV 可用模組

OpenCV（開放原始碼之電腦視覺化）包含多種即時電腦視覺功能的函式庫。因此，OpenCV 在影像處理方面應用非常廣泛，舉凡即時人臉偵測識別、物體識別、動作識別、物體追蹤、動態視訊…等抓取影像後進行識別的工作，搭配機器學習、深度學習相關的函式庫，使電腦視覺（Computer Vision）與人工智慧（AI）應用逐漸普遍且廣泛應用在生活中及工作當中，例如：汽車自動停車、輔助駕駛、車道偏移偵測等技術都和物件偵測息息相關。

智慧安控系統

2.4 OpenCV Python API

OpenCV-Python 是 OpenCV 的 Python API。它結合了 OpenCV C++ API 和 Python 語言的最佳品質。

Python 是由 Guido van Rossum 創立的一種通用編程語言，它的簡潔性和代碼可讀性使其在短時間內變得非常流行。它使程序員可以用更少的代碼行來表達他的想法，而不會降低可讀性。

與 C/ C++等其他語言相比，Python 比較慢。但是 Python 的另一個重要特性是它可以用 C/ C++輕鬆擴展。這個特性幫助我們可以用 C/ C++ 編寫計算密集型代碼，並為其創建一個 Python 包裝器，以便我們可以將這些包裝器用作 Python 模塊。這給了我們兩個優勢：首先，我們的代碼與原始的 C/ C++代碼一樣快（因為它在後台實際工作運行的是 C++代碼），其次，用 Python 編寫代碼非常容易。這就是 OpenCV-Python 的工作原理，它是以原始 C++實現的 Python 包裝器為基礎的。而 Python 中眾多的 library 之一 NumPy 的支持使這項任務變得更加容易。NumPy 是一個高度優化的數值操作庫。它給出了一個 MATLAB 風格的語法。所有的 OpenCV 數組結構都可以被轉換為 NumPy 數組或從 NumPy 數組轉換成別的數組結構。所以無論你在 NumPy 中做什麼操作，你都可以把它和 OpenCV 結合起來，這在實際運用中非常有用。除此之外，我們還可以使用其他一些庫，比如支持 NumPy 的 SciPy、Matplotlib。因此，OpenCV-Python 是一種設計用來快速解決計算機視覺問題的工具。

第3章研究方法

3.1 模型介紹

MoveNet.MultiPose 是在 RGB 圖像上運行並判定圖像中人物的人體關節位置的捲積神經網絡模型，這款模型與 MoveNet.SinglePose 模型之間的主要區別在於，該模型能夠實現同時偵測多人，並且仍能實現即時偵測功能。

該模型使用帶有特徵金字塔網絡解碼器的 MobileNetV2 圖像特徵提取器（步幅為 4），其次是具有自定義後處理邏輯的 CenterNet 預測頭。該模型使用深度乘數為 1.5。

3.1.1 輸入要求

輸入格式需為影片或圖像，表示為 int32（對於 TF.js）或 uint8（對於 TF Lite），故在使用上，需先調整影片或圖像的長寬為 32 的倍數，方可進行運算。

3.1.2 輸出數據

該模型會產生[1, 6, 56]的矩陣：

- 「1」：表示批次維度，它始終等於 1。
- 「6」：表示可同時偵測的最多人數。
- 「56」：前 51 個數值表示為 17 個關節點皆有 3 個數值，17 個關節點依序為鼻子、左眼、右眼、左耳、右耳、左肩膀、右肩膀、左手肘、右手肘、左手腕、右手腕、左腰、右腰、左膝蓋、右膝蓋、左腳踝、右腳踝，3 個數值包含 Y 座標、X 座標、準確率，[y_0, x_0, s_0, y_1, x_1, s_1, …, y_16, x_16, s_16]，剩下的 5 個數值代表所有關節點中最小 Y 座標、最小 X 座標、最大 Y 座標、最大 X 座標、準確率[ymin, xmin, ymax, xmax, score]，X 座標、Y 座標、準確率的值皆介於 0.0~1.0。

3.2 前置處理

首先，先載入模型、導入影片或開啟鏡頭，調整輸入影像的大小，再依照模型輸出結果，繪製出人體關節點以及關節連接線。

3.2.1 載入模型

可寫入該模型在 TensorFlow 上的網址，如圖四所示，亦可將模型下載至本機，並將模型的路徑寫入，如圖二所示，如果選擇寫入網址，電腦需連接網路方可執行，故我們選擇下載模型，並將路徑寫為本機路徑，如圖三所示。

```
model = hub.load('https://tfhub.dev/google/movenet/multipose/lightning/1')
movenet = model.signatures['serving_default']
```

圖二 載入網址

```
model = hub.load('C:\\Users\\User\\PycharmProjects\\pythonProject\\model')
movenet = model.signatures['serving_default']
```

圖三 輸入本機

3.2.2 輸入影片或鏡頭

利用 Opencv 的 VideoCapture 即可輸入影片，如圖四所示，如要使用鏡頭輸入，只需更改參數，依照參數指定要使用哪一隻鏡頭（0 代表第一隻，1 代表第二隻），通常電腦鏡頭預設為 0，如圖五所示。

```
cap = cv2.VideoCapture('C:\\Users\\User\\Desktop\\database\\01.mp4')
```

圖四 輸入為影片

```
cap = cv2.VideoCapture(0)
```

圖五 輸入為鏡頭

3.2.3 改變輸入影像大小

如同 4.1.1 所述，故我們將輸入影片更改為 384X640 的大小，並將每一幀影像命名為 img，如圖六所示。

```
# Resize image
img = frame.copy()
img = tf.image.resize_with_pad(tf.expand_dims(img, axis=0), 384, 640)
input_img = tf.cast(img, dtype=tf.int32)
```

圖六 更改影像大小

3.2.4 數據資料預處理

我們將常使用的數據另外儲存為新矩陣為 keypoints_with_scores[6, 17, 3]，「6」代表第幾個人，「17」代表第幾個關節，「3」則代表 Y 座標、X 座標、準確率，如圖七所示。

```
# Detection section
results = movenet(input_img)
keypoints_with_scores = results['output_0'].numpy()[:, :, :51].reshape((6, 17, 3))
```

圖七 預處理數據

如同 4.1.2 所述，該模型輸出的座標值介於 0.0~1.0，是依照畫面中的比例所定義，故我們將其改為更方便計算的座標，圖八是將右腰的關節點分別依照 X 座標、Y 座標，乘以影像寬度及長度

```
hip_x[i] = keypoints_with_scores[i][12][1] * width
hip_y[i] = keypoints_with_scores[i][12][0] * height
```

圖八 預處理數據

3.3 狀態分析

我們將依照偵測到的人的關節點進行狀態判定。

3.3.1 判定為人的條件式

輸出數據中的準確率是代表該關節的準確率，也就是說，如果準確率過低，它就不是人的關節點，故我們寫了一個條件式，判斷是否為人，本專題使用右腰及右肩的準確率進行判斷，當兩個關節點的準確率都達到 0.4 以上才會視為人，並且將畫面中的人數加一，如圖九所示，且每一幀影像都會重新判斷六個人的準確率，如圖十所示。

```
people_now = people_now + 1  # 畫面中的人數
```

圖九 畫面中人數計算

```
for i in range(0, 6):
    # 正確率達0.4再將其視為"人"
    if keypoints_with_scores[i][12][2] > confidence_scores and keypoints_with_scores[i][6][2] > confidence_scores:
        hip_x[i] = keypoints_with_scores[i][12][1] * width
        hip_y[i] = keypoints_with_scores[i][12][0] * height
        hip_accuracy[i] = keypoints_with_scores[i][12][2]
        hip[i] = [hip_x[i], hip_y[i]]
```

圖十 判斷為人的條件式

3.3.2 判定狀態的條件式

我們主要使用兩點決定斜率的方式進行人物狀態判定，先將牆面高度計算為 y=ax+b 的函數，如圖十一所示，先帶入人的 X 座標取得該牆的 Y 座標（tmp_y[i]），再與人的右腰的 Y 座標（hip_y[i]）做比較，考量到象限問題，所以如果 hip_y[i] < -tmp_y[i]，會判定為有人在翻牆，其餘則否，如圖十二所示。

```
# y=ax+b，計算斜率跟常數
def calculate_yt(a, b, c, d):
    a = np.array(a)
    b = np.array(b)
    c = np.array(c)
    d = np.array(d)
    b = -b
    d = -d
    slope = (d - b) / (c - a)
    number_b = b - slope * a
    return slope, number_b
```

圖十一 計算斜率

```
# 計算y(x)值，判斷狀態
[slope1, number_b1] = calculate_yt(line_x1, line_y1, line_x2, line_y2)
tmp_y[i] = (slope1 * hip_x[i]) + number_b1
if hip_y[i] < -tmp_y[i]:
    number_total = number_total + 1  # 翻牆的人數
```

圖十二 判斷斜率狀態

3.3.3 排除重複寫入錯誤

經過多次檢查，我們發現該模型會將一個人的座標重複寫入，比如說，畫面中只有兩個人，卻有三個人的數據，而其中有兩個人的 X 座標、Y 座標卻是一樣的，故我們寫了一個條件式，如果兩個人 X 座標相差±5，會視為同一個人，並將其所有的值歸零，如圖十三所示。

```
# 找出並清除重複寫入的人的x, y, accuracy(誤差範圍正負5)
for p in range(i, 0, -1):
    print('p=', p, ',i=', i)
    if int(hip_x[p - 1] - 5) < int(hip_x[i]) < int(hip_x[p - 1] + 5):
        hip_x[i] = 0
        hip_y[i] = 0
        hip_accuracy[i] = 0
```

圖十三 清除重複數據

3.3.4 將人物框選

將 X 座標最大值設為 0，X 座標最小值設為 1000，考量到象限問題，故 Y 座標需將最大值設為 1000，最小值設為 0，再判斷是否為人的關節點，如果是，再找進行比較運算，找出人的 X 座標的最大值及最小值、Y 座標的最大值及最小值，如圖十四所示，再依照座標，利用 Opencv，畫出相對應的框框，如圖十五所示。

- cv2.rectangle（img, 左上座標, 右下座標, 顏色, 厚度）

```
# 找出人的高度及寬度
tmp_min_x = 1000
tmp_max_x = 0
tmp_min_y = 0
tmp_max_y = 1000
for t in range(0, 17):
    if keypoints_with_scores[i][t][2] > confidence_scores:
        if keypoints_with_scores[i][t][0] * height < tmp_max_y:
            tmp_max_y = keypoints_with_scores[i][t][0] * height
        if keypoints_with_scores[i][t][0] * height > tmp_min_y:
            tmp_min_y = keypoints_with_scores[i][t][0] * height
        if keypoints_with_scores[i][t][1] * width > tmp_max_x:
            tmp_max_x = keypoints_with_scores[i][t][1] * width
        if keypoints_with_scores[i][t][1] * width < tmp_min_x:
            tmp_min_x = keypoints_with_scores[i][t][1] * width
```

圖十四 找出座標的最大最小值

```
cv2.rectangle(frame, (int(min_x) - 10, int(min_y) - 20), (int(max_x) + 10, int(max_y) + 20),
              (36, 28, 237), 3)
```

圖十五 繪出人形框

3.4 輸出結果

程式執行完，會產生具有狀態判定的影片（名為 output），並且每輸入一個新影片，皆會產生一個新資料夾，並將其危險事件的照片存入。

3.4.1 結果影片

首先，要先產生一個空的影片，如圖十六所示，其中 20 代表影片幀數，程式執行時，會將每一幀照片寫入空的影片，進而累積成具有狀態判定的結果影片，並且在程式執行時，會顯示出每一幀影像的執行結果，如圖十七所示。

```
out = cv2.VideoWriter('output.mp4', fourcc, 20.0, (width, height))  # 產生空的影片
```

圖十六 產生空影片

```
out.write(frame)
cv2.imshow('Movenet_Multipose', frame)
```

圖十七 寫入影片

3.4.2 照片儲存至新資料夾

每次程式執行時，都會先創建新的資料夾，命名為 danger_images i，其中 i 會由 0 開始陸續加 1，程式碼如圖十八所示，範例結果如圖十九所示，當有危險事件發生時（number_total>0），會將照片存入該資料夾，並且以時間命名，方便後續查詢，程式碼如圖二十所示，範例結果如圖二十一所示。

```
# 將照片儲存在新資料夾
j = 1
while 1:
    images = f'danger_images' + str(j)
    if not os.path.exists(images):
        os.makedirs(images)
        break
    else:
        j = j + 1
```

圖十八 創建新資料夾　　圖十九 資料夾範例

```
if number_total > 0:
    stage_danger()
    localtime = time.localtime()
    result = time.strftime("%y-%m-%d_%I-%M-%S", localtime)
    fileNameTemp = ".//" + str(images) + "//" + result + ".jpg"
    filename = fileNameTemp
    cv2.imwrite(filename, frame)
else:
    stage_safety()
```

圖二十 存入危險事件的照片　　圖二十一 危險事件照片範例

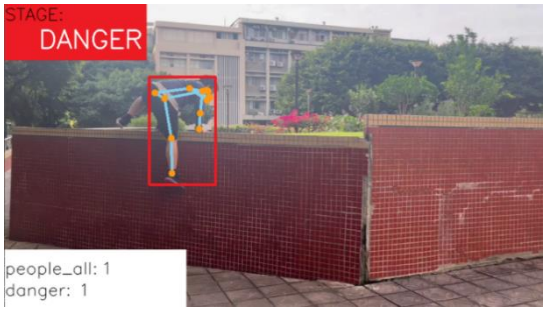
第4章 研究結果

- STAGE：為狀態表示，分為安全事件（SAFE）、危險事件（DANGER）兩種狀態。
- people_all：為畫面中的所有人數。

- danger：表示為畫面中正在翻牆的人數。

4.1 單人爬牆

圖二十二 白天_單人爬牆_安全事件



圖二十三 白天_單人爬牆_危險事件

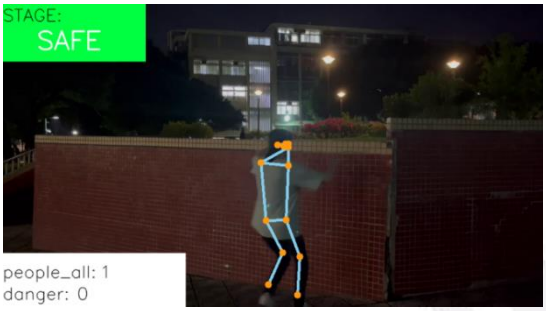


圖 二十四 夜晚_單人爬牆_安全事件

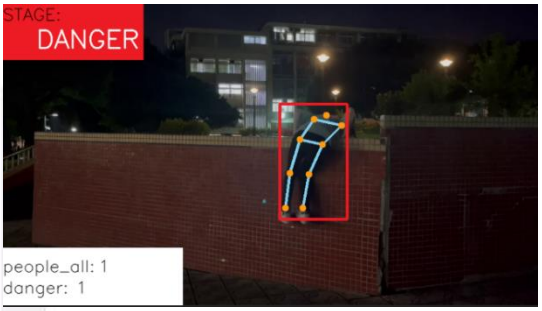


圖 二十五 夜晚_單人爬牆_危險事件

4.2 雙人爬牆

圖二十六 白天_雙人爬牆_安全事件

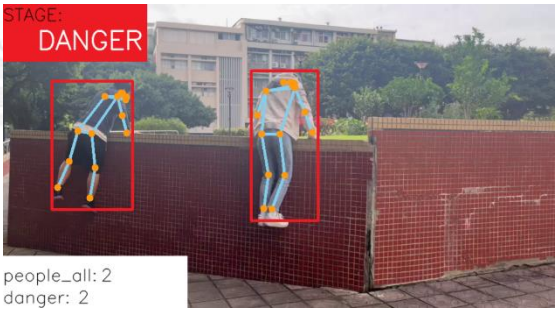
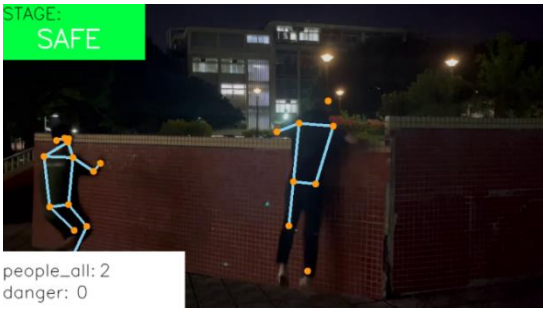
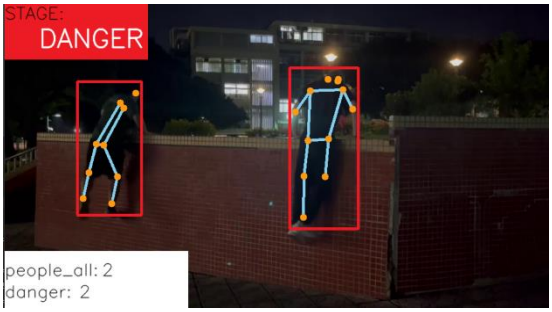


圖 二十七 白天_雙人爬牆_危險事件

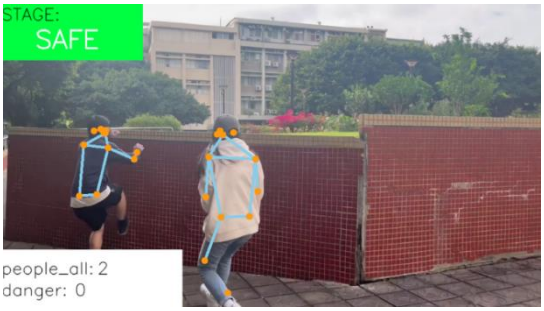


圖二十八 夜晚_雙人爬牆_安全事件

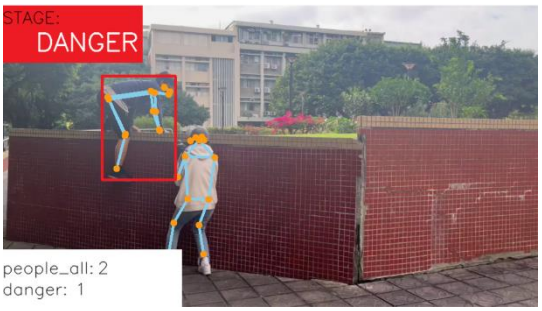


圖二十九 夜晚_雙人爬牆_危險事件

4.3 一爬一不爬

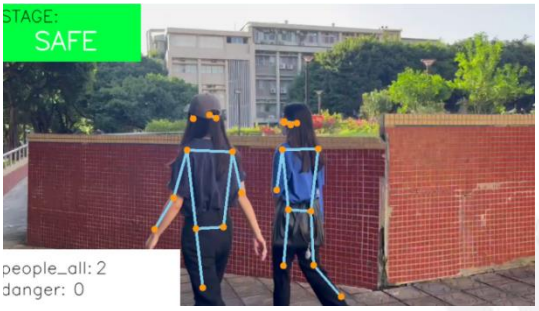


圖三十 白天_一爬一不爬_安全事件



圖三十一 白天_一爬一不爬_危險事件

4.4 不爬牆



圖三十二 不爬牆

第5章 未來展望

5.1 優化夜視情境精準度

夜間光線不足，無法將人體的輪廓以及關節顯示清楚，以至於系統無法準確偵測人體關節點，未來期望能增加畫面前處理的功能，讓系統在運用節點作條件判斷前，先對鏡頭畫面進行影像處理，以增加系統功能的準確度。

5.2 系統嵌入邊緣實現

未來希望能結合硬體裝置，將系統製作成能獨立的一項產品，讓使用者可以自由決定擺放的地點，且系統能不因環境改變而影響運作，進而提高此安控系統的實用性以及靈活度。

智慧安控系統

5.3 結合物聯網

現今生活中對與物聯網的使用非常發達，未來希望能嘗試製作成一個此系統專屬的應用程式，供使用者以更方便、更人性化的方式操作系統，並且不限制在任何裝置上使用系統的功能以及資料，讓此系統更加貼近使用者的生活。

第6章　參考文獻

[1] Tibame 小編（Ed.）.（2020, March 23）. 你的 AI 會看圖嗎? Open CV 介紹. 取自：https：//blog.tibame.com/?p=15141%E5%8D%81%E4%B8%80

[2] Alexander mordvintsev , & Abid rahman k. .（2022, December 29）. Introduction to OpenCV-Python Tutorials. OpenCV. 取自：https：//blog.tibame.com/?p=15141%E5%8D%81%E4%B8%80

[3] Wang, G. T.（2017, June 28）. TensorFlow 機器學習軟體工具入門教學與範例實作. GTW. 取自：https：//blog.gtwang.org/statistics/tensorflow-google-machine-learning-software-library-tutorial/

[4] Francois Bellett, Ard Oerlemans, Yu-Hui Chen, & Ronny Votel.（2017, September）. MoveNet.MultiPose. TensorFlow Hub. 取自：https：//tfhub.dev/google/lite-model/movenet/multipose/lightning/tflite/float16/1