

# 主題：防範未然 --- 跌倒偵測系統

科別：資訊科

作者：楊子慕、葉品和

指導老師：朱麗芬、黃培書老師

## 摘要

隨著科技發展，人口快速膨脹，現在各國老年人的數量在急劇上升中，因此老年人的照顧是近來備受重視的課題。而政府長期照護單位的人手不足已成為迫在眉睫的問題，我們希望能舒緩這個問題，使得老年人能得到更完善的照顧。

本研究將常見的網路攝影機和樹梅派主機板(Raspberry-Pi 3)結合。獲取受照顧者影像後，傳入樹梅派主機板中處理圖片和定位人形。再將座標化後的人物外框座標新建檔案儲存，再呼叫C語言子程式來做物件歸類、分析、過濾和判斷跌倒，以達到及時提醒的目標。

## 壹、研究動機

有親人在政府的長期照護部門工作，時常提到現在長期照護的使用人數太多，而照顧人手卻不足，長期下來會造成照顧者與被照顧者的糾紛，再加上市面上可靠的裝備不多，容易發生居家意外，嚴重者可能導致家庭的分崩離析和悲劇的發生。希望可以有一個結合網路攝影機與單晶片微電腦的資訊系統可以彌補人手不足的問題，期望能在有跌倒事件發生時，提出警示。

因而發想：是否能夠針對家中被照顧者跌倒的問題，用一個自動系統偵測跌倒事件，以減少使用的人力。現在市面上常見的設備大多數都是被照顧者主動呼叫，沒辦法自動偵測，而許多偵測跌倒之系統使用可攜式裝備，有可能會造成生活上的不便。因此本研究主要討論使用 OpenCV、Raspberry Pi 及常見網路攝影機的結合，以不干涉被照顧者生活活動為基準來達到解決此問題的目的。

## 貳、研究目的

- 一、分析不同偵測設備的優缺點
- 二、透過Python撰寫OpenCV程式，持續讀取、分離前景背景並對物件做定位
- 三、透過C語言撰寫判斷跌倒演算法

#### 四、分析不同參數給演算法的影響與結果

### 參、研究設備及器材

#### 一、單晶片微電腦( Raspberry Pi 3 主機)



#### 二、網路攝影機(羅技 c310)



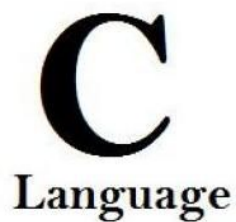
#### 三、桌上型電腦



#### 四、Python



#### 五、C語言



#### 六、VNC Viewer



## 肆、研究過程及方法

### 一、市面常見偵測使用設備探討

#### (一) 穿戴式與固定式裝備之比較

在做我們的研究之前，我們做了一些市面上常見研究使用設備的比較(見表一)，以了解該使用哪一類型的設備。

	穿戴式裝備	固定式裝備
判斷依據	◆ 裝置所收到的受測者動作訊號	◆ 接受到的圖像、數據訊息進行分析
優點	◆ 頭部、胸部、腰部的配戴裝置因為位置不常晃動，可以達到100%準確偵測  ◆ 可以結合其他的醫療儀器做使用，例如頭部配戴裝置可以結合助聽器等	◆ 不需穿戴任何物品在身上，十分方便即可隨時監控  ◆ 方便實測與調整
缺點	◆ 佩戴不便，舒適度不佳，可能影響生活。  ◆ 在沒有配戴的時候也無法進行判斷。	◆ 監測範圍有限  ◆ 在某些位置會有隱私上的疑慮  ◆ 較容易受到外在環境的影響

表一、穿戴式裝備與固定式裝備的比較

因穿戴式裝置硬體設備需要另外製作，技術發展也已經十分完備，我們選擇往固定式裝備方面研究。

#### (二) 常用單晶片微電腦(Raspberry Pi、Arduino)之比較

選用單晶片微電腦的原因是方便設備攜帶，如果是輕便的微電腦的話，只要有電源的地方都可以安裝使用。目前最常使用的有Raspberry Pi和Arduino兩種，而Raspberry Pi 是一台以Linux為基礎的單晶片微電腦，有內建的作業系統。相較於Arduino，Raspberry Pi 多了一些USB插座、網路線插座等。編寫程式的部分因為直接在內建系統上編寫，不需像Arduino一般須先傳入再執行，測試上方便許多。而

Arduino也有除非關掉電源否則無法停止程式執行等缺點，在編寫上有諸多不便，因而選擇使用Raspberry Pi晶片。

## 二、資料蒐集

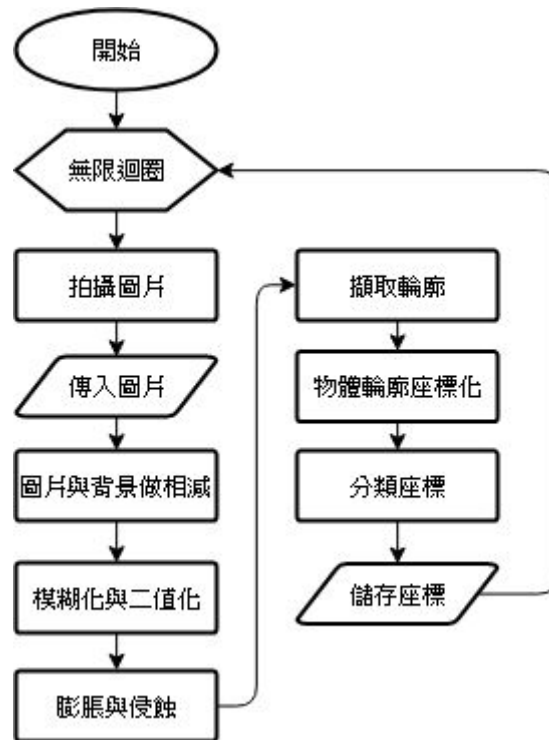
下表為本研究使用到的OpenCV函式，整理成函式庫以利於了解各項函式的功能。

編號	函式名稱	功能	備註
註一	absdiff	用以計算兩組數組之間差的絕對值	1. 必須相同數據類型 2. 圖片必須經過灰階處理
註二	BackgroundSubtractorMOG	訓練出一個基於像素樣本統計訊息的背景表示方法的模型，可對複數動態物體進行建模	
註三	filter2D	用在去除雜訊、影像銳化或提取感興趣的視覺特徵	
註四	medianBlur	將濾波範圍內的所有像素重新排序，並以中值替換當前的像素值	
註五	threshold	將某個強度(閾值)當作分割的標準，強度超過閾值的像素當作前景，反之則為背景	1. 圖片必須經過灰階處理
註六	dilate	擴大物體的邊界，能使原本分開的物體連接起來	1. 若物體擴張超出圖片，則會變形
註七	erode	消融物體的邊界，能使原本連接的物體分開或消失	
註八	findContours	找出物體的輪廓	
註九	boundingRect	得到包覆輪廓的最小正矩形	
註十	rectangle	在圖片上畫出指定座標的方框	

表二、OpenCV函式庫

本研究主要儲存檔案、執行指令等都是在Raspberry pi 3上，為了方便操作，我們使用了VNC Viewer進行遠端遙控，藉由桌上型電腦控制Raspberry pi 3。此部分的

程式使用Python編寫，負責拍攝圖片到將圖片中的物體座標數據化，以便之後資料分析的進行。流程圖及各步驟說明如下：



圖一、Python程式流程圖

#### (一) 拍攝圖片及傳入圖片

```
cap = cv2.VideoCapture(0)
count = 1

ret, f = cap.read()
cv2.imwrite("/home/pi/LALA0.jpg", f)

while(True):

    ret, frame = cap.read()
    cv2.imwrite("/home/pi/LALA1.jpg", frame)

    imgFirst = cv2.imread("/home/pi/LALA0.jpg", 0)
    imgtwo = cv2.imread("/home/pi/LALA1.jpg", 1)

    imgSecond = cv2.cvtColor(imgtwo, cv2.COLOR_BGR2GRAY)
```

圖二、拍攝、傳輸圖片程式圖

將網路攝影機連接至Raspberry pi 3，以每0.4秒一張的頻率拍攝圖片，將第一張圖片單獨儲存，而後的每一張圖片則覆蓋前一張圖片並儲存。以0.4秒一張拍攝的原因是因為設備效能不足，我們減少了另外再計算跌倒時間的步驟。若掉落的時間超過0.4秒，也就是超過了一張圖片間隔的時間，因為可能為跌倒事件以及可能受到的傷

害十分低，我們直接將這種可能性消除。另外每秒拍攝影像數少也可以提高本程式的運作效能。

圖像中的色彩是由兩個參數組成的：明度和彩度。若直接以彩色圖像進行圖片處理，除了物體的明、暗以外，還必須將彩度列入考慮。增加判斷的難度，也不會增加精準度，因此直接輸入灰階圖像。

(二) 背景相減法

```
result = cv2.absdiff(imgFirst,imgSecond)
```

圖三、背景相減法函式圖

以第一張圖片作為背景，使用absdiff函式(註一)將每一張圖片減去背景圖片，得出目標圖片與背景圖片之差的絕對值(見表三)，利用此方法將前景，也就是畫面中要偵測的物體提取出來，變成較清楚的色塊。

另一種方法是OpenCV所提供的BackgroundSubtractorMOG函式(註二)，此函式會使用最先輸入的自定義數量圖片拿來當模型，訓練出一份背景的資料以便提取前景圖像。使用這個函式的最大優點是能夠隨著環境的變化改變背景的模式，所以環境光源的變化和一些變動的物體不會影響到此函式提取出的前景。但是因為網路攝影機的效能不足，此函式隨著環境變化背景模型的功能一直無法流暢運轉，因此最後仍然是使用圖片相減法。

背景圖	原圖	背景相減法結果圖
		

表三、背景相減法結果比較圖

(三) 模糊化與二值化

```

rows,cols = result.shape[:2]
kernel_5x5=np.ones((15,15),np.float32)/225.0
img2 = cv2.filter2D(result, -1, kernel_5x5)

median = cv2.medianBlur(img2,5)

ret, result = cv2.threshold(median, 50, 255, cv2.THRESH_BINARY)




```

圖四、模糊化與二值化程式圖

畫面中會有一些因為環境因素而出現的前景，例如窗外的風吹草動、影子方向變動等。他們被判斷出來的圖形多半破裂、不成形或較小。一個像素黑白數值是由0(黑)到255(白)，相減後的差異大的數值會高，若差異的色塊面積小，只要將其與周圍低差異的區域做模糊，數值平均化，再執行二值化(註五)，就可以將其刪除，而此方法不會影響到較大的色塊，因此選用此方法消除雜質。

我們使用了兩種模糊，分別是filter2D函式(註三)及medianBlur函式(註四)，前者是簡單模糊，括號中的數字np.ones函式括弧中的數值(15,15)越大，執行的結果則模糊程度越高。後者是平滑濾波中的中值濾波，可以有效地去除雜訊。

相對於簡單模糊，medianBlur除了檢查像素間幾何上的靠近程度以外，還會將顏色的彩度以及明度列入處理條件，另外也會保留圖片的邊緣資訊，我們和簡單模糊搭配使用，以達到最好的模糊效果。

原圖	簡單模糊	medianBlur
		

表四、模糊執行結果比較表



圖五、二值化執行結果

#### (四) 膨脹與侵蝕

```
matrix = np.ones((5,5),np.uint8)
eff = cv2.dilate(result, matrix, iterations = 20)
erosion=cv2.erode(eff, matrix, iterations = 18)
```

圖六、膨脹與侵蝕程式圖

因為相減完的前景圖片可能因為背景光線、顏色等變得十分分裂。分離的影像經過dilate函式(註六)膨脹重疊後，會導致色塊結合為一。而erode函式(註七)運作模式是以一整個色塊為單位做侵蝕的動作。利用這兩個函式的運作模式，就可以將分離的物體形狀融合為一，而又不影響到物體的大小。先將所有畫面中的差異色塊分別膨脹，使大部分的小碎塊融合。再將融合完的圖形侵蝕回原本的大小。這樣就可以解決物件太過分散而將一個大物體判斷成四、五個小物體的錯誤(見表五)。

為了顯示此函式的效果，我們使用了有分裂的圖片來做膨脹和侵蝕來展示效果。

呈分裂的物體圖像	膨脹結果圖	侵蝕結果圖 (最終結果)
		

表五、膨脹與侵蝕比較表

#### (五) 輪廓

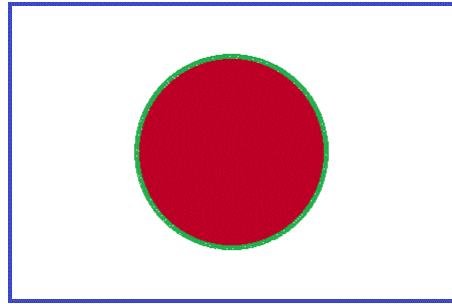
```
(cnts, _) = cv2.findContours(erosion.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

圖七、輪廓程式圖

將上一步驟的所得出的圖形碎塊進行尋找輪廓的動作，執行findContours函式(註八)，找出一個碎塊的邊緣輪廓，以利接下來的座標化進行。



取輪廓的時候，可以選擇要使用的模式。取得輪廓的方式有四種，分別可以取出不同的階層數。階層數是一個物體內最大的輪廓內的其他小輪廓。例如日本國旗的圖像，藍色線就會是最大輪廓，裡面綠色線就會是首階層。



圖八、階層示意圖

儲存輪廓和階層的方式有四種，分別是RETR\_EXTERNAL (取得最外層輪廓)、RETR\_LIST (取得所有輪廓)、RETR\_CCOMP (取得所有輪廓，並儲存成兩階層)、RETR\_TREE (取得所有輪廓，並儲存成全階層)，而我們只需要取得最外圍的輪廓，內部的階層會影響偵測，所以使用了第一種 (只取得最外層輪廓) 的方式。

儲存輪廓點的方式有兩種，分別是CHAIN\_APPROX\_NONE (儲存所有輪廓點) 及CV\_CHAIN\_APPROX\_SIMPLE (只儲存對角的四個頂點)，我們使用的是後者。

#### (六) 物體座標化

```
for c in cnts:
    x, y, w, h = cv2.boundingRect(c)
    cv2.rectangle(clone, (x, y), (x + w, y + h), (0, 255, 0), 2)

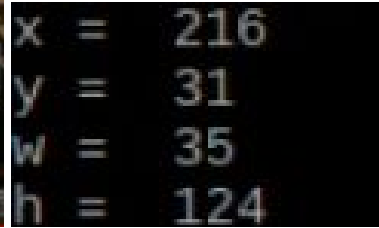
    spot[idx][0] = x
    spot[idx][1] = y
    spot[idx][2] = w
    spot[idx][3] = h
    idx = idx + 1
```

圖九、座標化與畫外框程式圖

使用boundingRect函數(註九)，以上步驟抓取出來的輪廓為基準，定位出物件輪廓的最大四邊形外框，取出頂點座標存入變數x (x座標)、y (y座標)、w (寬)、h (高)，再利用rectangle函數(註十)將四點座標以方形的方式畫出，顯示在原圖上，以利偵錯。



圖十、座標方向示意圖



圖十一、座標畫結果

此系統的圓點位在圖片的左上角，x軸向右為正，y軸向下為正。綠色框線的左上角為座標 (x, y)。

### (七) 分類與儲存

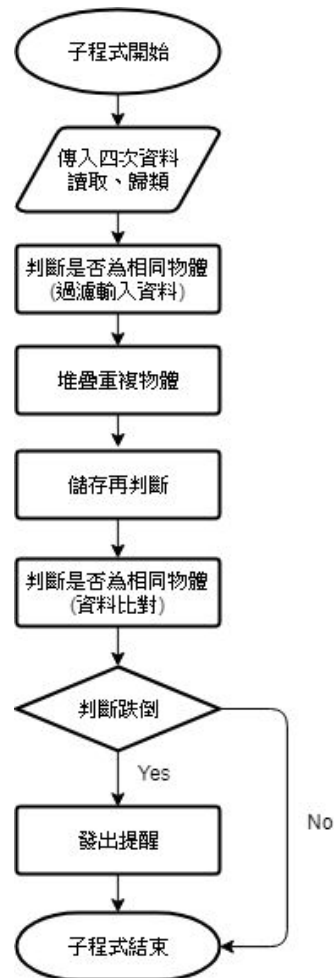
```
if count == 1:
    print "=====111111"
    fo = open("foo.txt1", "w")
    for i in range(idx):
        for j in range(4):
            fo.write('{:03d} '.format(spot[i][j]))
            print spot[i][j]
    fo.close()
    count += 1
elif count == 2:
    print "=====222222"
    fo = open("foo.txt2", "w")
    for i in range(idx):
        for j in range(4):
            fo.write('{:03d} '.format(spot[i][j]))
            print spot[i][j]
    fo.close()
    count += 1
elif count == 3:
    print "=====333333"
    fo = open("foo.txt3", "w")
    for i in range(idx):
        for j in range(4):
            fo.write('{:03d} '.format(spot[i][j]))
            print spot[i][j]
    fo.close()
    count += 1
elif count == 4:
    print "=====444444"
    fo = open("foo.txt4", "w")
    for i in range(idx):
        for j in range(4):
            fo.write('{:03d} '.format(spot[i][j]))
            print spot[i][j]
    fo.close()
    count = 1
output = subprocess.check_output(['/home/pi/detect.exe'])
```

圖十二、分類儲存程式圖

這時候，將照片裡的x、y、w、h數值用連續的方式存成四個檔案，第五張照片的數值將會覆蓋第一個檔案，以此類推。

## 三、資料分析

每儲存四次座標後，會呼叫一次C語言程式做一次資料分析。經過資料分類、去除重複資料後，會進行判斷跌倒的工作。整個資料分析共分為六個步驟，見以下流程圖：



圖十三 C語言程式流程圖

## (二) 讀取數據及歸類

```

for(z = 0; z < 4; z++){
    char a[2];
    char place[20];

    add = z+1;
    sprintf(place, "%s%d", "/home/pi/foo.txt", add);

    if ((fp = fopen(place, "r+")) == NULL){
        printf("can not open file");
    }
    else{
        for(i = 0; i < 10; i++){
            for(j = 0; j < 4; j++){
                if(fscanf(fp, "%d", &save[z][i][j]) == EOF)
                    break;
                printf("%d\n", save[z][i][j]);
            }
        }
        fclose(fp);
        printf("\n");
    }
}
  
```

圖十四、讀取、歸類數據程式圖

將上一步驟儲存的所有數據讀入save[z][i][j]陣列裡，第一項 (z)是用來區分四個檔案。第二項 (i) 是為了區分同一個檔案中不同碎塊的x、y、w、h。第三項 (j)分別讀入x、y、w、h的數值。

### (三) 判斷是否為相同物體

```
for(f = 0; f < 3; f++){  
    for(a = 0; a < ab; a++){  
        for(b = 0; b < ab; b++){  
            times = 0;  
  
            xa = save[f][a][0]*1.0;  
            ya = save[f][a][1]*1.0;  
            wa = save[f][a][2]*1.0;  
            ha = save[f][a][3]*1.0;  
  
            xb = save[f+1][b][0]*1.0;  
            yb = save[f+1][b][1]*1.0;  
            wb = save[f+1][b][2]*1.0;  
            hb = save[f+1][b][3]*1.0;
```

圖十五、判斷相同物體程式圖(片段)

為了避免程式執行過多不必要的次數，先判斷此次讀取的資料內不同輪廓的數量最大值，存入變數，這樣就可以以此變數控制程式執行的次數。

以便後續的處理，一次將兩組x、y、w、h儲存成浮點數的資料結構，將所有物體的座標值分別做處理，變化差距在閾值的範圍內都認定為同一物體，以下四點為四種數據的組合，應用來判斷是否為相同物體之情形。

#### 1.長寬比例

```
if ( (wa/ha)/(wb/hb) <= 1.1 && (wa/ha)/(wb/hb) >= 0.9 )  
    times++;
```

圖十六、判斷相同物體程式圖(長寬比例)

根據函式輸出的長、寬數據相除後得到的長寬比例，因為人體站立時的tan值固定，所以能夠判斷人進行相對於網路攝影機遠近的移動。

#### 2.長寬比例(倒數)

```
else if ((wa/ha)/(hb/wb) <= 1.1 && (wa/ha)/(hb/wb) >= 0.9)  
    times++;
```

圖十七、判斷相同物體程式圖(長寬比例倒數)

利用長寬比例的倒數，可以判斷由站立變為躺臥的移動。

### 3.寬

```
else if ((wa/wb) <= 1.1 && (wa/wb) >= 0.9)
    times++;
```

圖十八、判斷相同物體程式圖(寬)

若兩數據的寬大小相近，則可以判斷人在站立、蹲坐、跳躍等不同動作之間的變換。

### 4.長

```
else if ((ha/hb) <= 1.1 && (ha/hb) >= 0.9)
    times++;
```

圖十九、判斷相同物體程式圖(長)

長可以判斷人左右走動的動態。將判斷為true的數據存入object陣列，其他數據則捨去。

```
if (times == 1){
    for(c = 0; c < 4; c++){
        object[obj][c] = save[f+1][b][c];
    }
    obj++;
}
```

圖二十、判斷相同物體程式圖(片段)

將判斷為true的數據存入object陣列，其他數據則捨去。

## (四) 堆疊重複物體

```

for(d = 0; d < 20; d++){
    if (object[d][0] == 0 && object[d][1] == 0 && object[d][2] == 0 && object[d][3] == 0)
        continue;

    for(e = d+1; e < 20; e++){
        obj_h = object[e][3] - object[d][3];
        if (obj_h <= 0)
            obj_h = obj_h*-1;
        obj_x = object[e][0] - object[d][0];
        if (obj_x <= 0)
            obj_x = obj_x*-1;
        obj_w = object[e][2] - object[d][2];
        if (obj_w <= 0)
            obj_w = obj_w*-1;
        obj_y = object[e][1] - object[d][1];
        if (obj_y <= 0)
            obj_y = obj_y*-1;

        if(obj_x <= 20 && obj_y <= 20 && obj_w <= 20 && obj_h <= 20){
            for(g = 0; g < 4; g++){
                object[d][g] = object[e][g];
                object[e][g] = 0;
                counter[d]++;
            }
        }
    }
}

for(killswitch = 0; killswitch < 20; killswitch++){
    if (counter[killswitch] < 2){
        for(ks = 0; ks < 4; ks++){
            object[killswitch][ks] = 0;
        }
    }
}

for(k = 1; k < 20; k++){
    if(object[k][0] == 0 && object[k][3] == 0){
        for(l = k; l < 19; l++){
            for(m = 1; m < 4; m++){
                object[l][m] = object[l+1][m];
            }
        }
    }
}

```

圖二十一、堆疊重複物體程式圖

object陣列是儲存所有的座標數據，圖片裡重複出現的物體會被重複儲存。例如甲物件在四張圖片都有數據傳入，每一張圖片裡的甲物件的座標都會分別被存入並占用一個空間。在者裡我們將各個數據做交叉比對，差距低於閾值的數據判斷為同一個物件，並將該物件較先前的資料歸零，只留下該物件最後一張有出現之圖片的座標。最後再重新整理陣列，將已被歸零的部分刪除。

#### (五) 儲存與再判斷

```

if ((ofp = fopen("object.txt", "r+")) == NULL){
    printf("can not open file");
}
else{
    for(i = 0; i < 2; i++){
        for(j = 0; j < 4; j++){
            fscanf(ofp, "%d", &objj[i][j]);
        }
    }
}

```

圖二十二、儲存再判斷程式圖

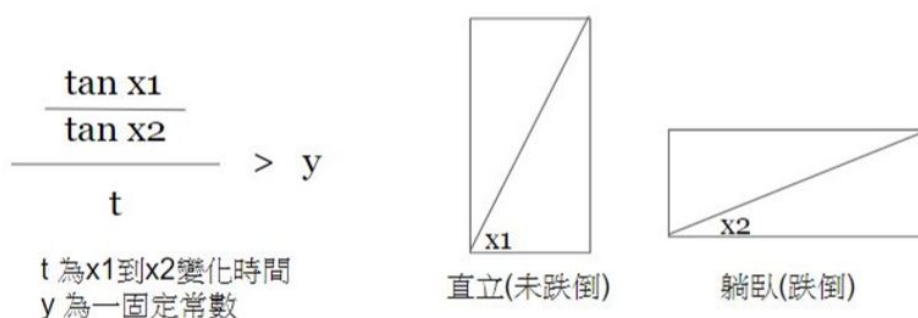
將上述整理完後object陣列中的數據存入另一個檔案裡，以利接下來的判斷進行，之後回到步驟 (三) 判斷是否為相同物體，此陣列裡的數據將直接做為判斷是否跌倒的依據。

## (六) 判斷跌倒

```
for(i = 0; i < 5; i++){  
    if (fall[i][3] != 0){  
        if ((fall[i][3]*1.0)/(fall[i][2]*1.0) > 0.4)  
            printf("save");  
        else  
            printf("be careful!!!");  
    }  
}
```

圖二十三、判斷跌倒程式圖

因為網路攝影機拍照效能的缺乏，我們無法在即時演算系統中實現計算跌倒發生及持續時間再用tan值除以跌倒時間的部分。我們將物體的長除以寬，計算出物體的tan值後直接與閾值做比對。透過調整閾值到最適合的參數來增加正確判斷跌倒的成功率。

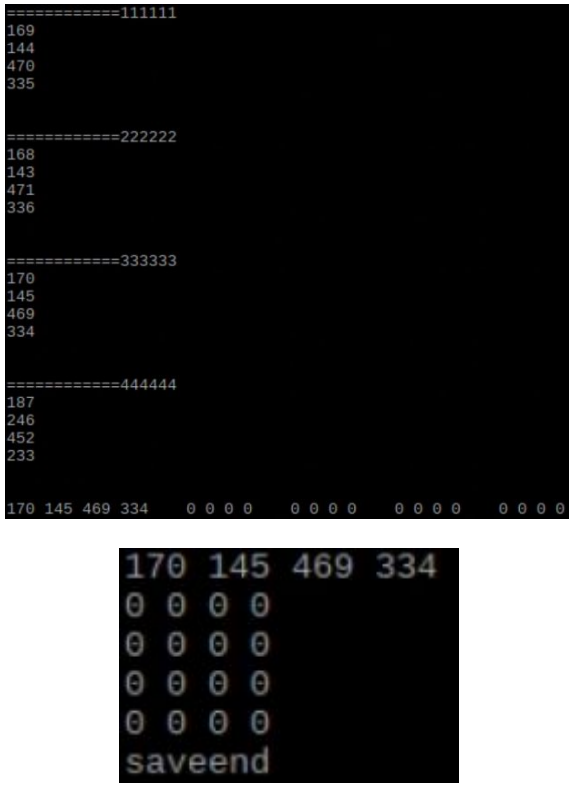


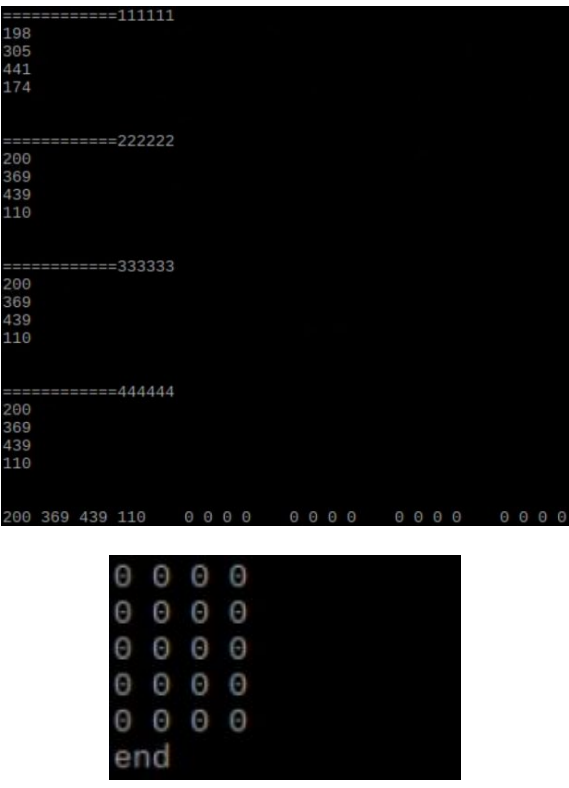
圖二十四、原演算法示意圖(取自小論文: 居家安全-跌倒偵測)

## 伍、研究結果

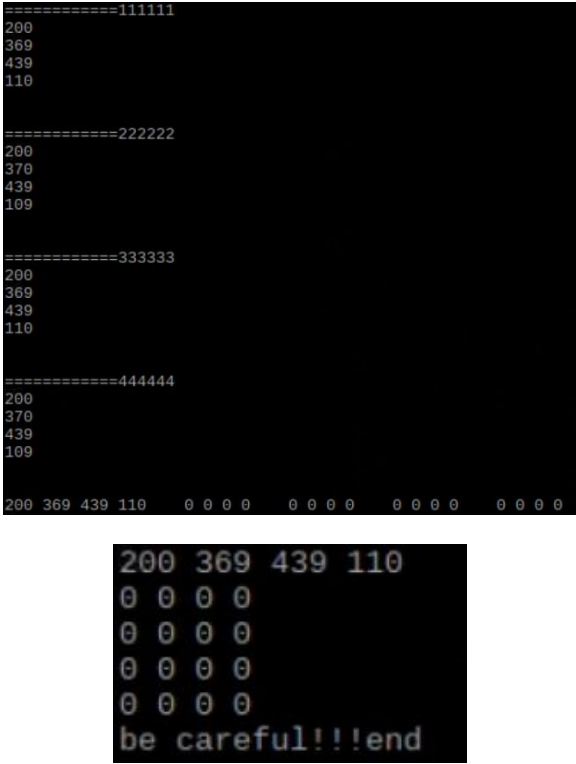
每執行四次Python程式 (獲取四次數據)，會執行一次C程式。輸出格式是，先輸出四張圖片中的物體座標，經過過濾後存入一個陣列中，此陣列中的數值將會與下一次執行C程式時的其他物體座標做相似度判斷，若在閾值範圍內，則接續到下一步驟進行判斷，(h/w < 0.4)判斷為跌倒。若顯示save則為非跌倒狀態、若顯示be careful則為跌倒狀態、若無顯示則代表物體在移動、變換形態或其他狀態。



<p>非跌倒狀態</p>	 <pre> =====111111 169 144 470 335  =====222222 168 143 471 336  =====333333 170 145 469 334  =====444444 187 246 452 233  170 145 469 334  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0 170 145 469 334 0 saveend </pre> <p>圖二十五、圖二十六、C語言執行結果 (一)</p>	<p>此為半坐狀態的執行結果。</p> <p>圖二十五等號後面的數字代表第目前是幾次讀取數據。由上到下分別是x、y、w、h的數據。</p> <p>圖二十五最下面一行為以上資料經過相似度過濾篩選後存入object的數據，由左到右分別為x、y、w、h。</p> <p>圖二十六為object陣列經過與下一次讀入資料相似度過濾篩選後所得的最終數據(fall陣列)由左到右分別為x、y、w、h。</p> <p>其<math>h/w = 334/469 = 0.712</math>(四捨五入) <math>&gt; 0.4</math>，判斷為非跌倒狀態。</p>
--------------	---	--

<p>移動、變換形態及其他狀態</p>	 <pre> =====111111 198 305 441 174  =====222222 200 369 439 110  =====333333 200 369 439 110  =====444444 200 369 439 110  200 369 439 110  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0 200 369 439 110 end </pre>	<p>此為移動狀態的執行結果。</p> <p>由圖中可見，圖二十八的object陣列相似比較結果輸出並未存入任何數值，意思是物體移動比率大，代表待測物目前處在移動或變換型態的狀態。</p>
---------------------	--	--



	圖二十七、圖二十八、C語言執行結果 (二)	
跌倒狀態	 <p>圖二十九、圖三十、C語言執行結果(三)</p>	<p>此為躺臥狀態，其在圖三十一的輸出結果中，<math>h/w = 110/439 = 0.251</math>(四捨五入) <math>&lt; 0.4</math>，tan值小於閾值，判斷為跌倒狀態，輸出be careful!!!的警告語。</p>

表六、C語言執行結果比較表

我們測試了受測者在各個不同動作階段系統判斷出來了十次 tan值並將之列成表六。可以由表中看出，十次的數據十分接近，可以得知框人形的圖片處理部分十分穩定，並且不同動作差距甚大，能有效的分別不同動作之間的差別。

	第一次	第二次	第三次	第四次	第五次	第六次	第七次	第八次	第九次	第十次
站立	3.73	3.71	3.76	3.79	3.73	3.88	3.82	3.71	3.79	3.76
蹲	1.18	1.17	1.18	1.17	1.16	1.16	1.17	1.18	1.17	1.18
坐	1.11	1.12	1.12	1.10	1.13	1.12	1.11	1.15	1.13	1.10
躺	0.35	0.37	0.35	0.35	0.34	0.36	0.36	0.35	0.35	0.36

表七、不同動作階段之tan值比較表

## 陸、討論與結論

## 一、為何OpenCV的程式用Python而判斷卻在C語言裡編寫？

最初在執行的時候，因為安裝的失誤或版本的問題，導致一些C語言函式(例如imshow等)無法執行，改用Python後得以解決，雖然後來發現原本錯誤的原因，但還是決定在Python中繼續完成。而基本的過濾、判斷使用C語言編寫則是因為，相較於Python，我們對C語言的掌握度比較高。

## 二、其他物體的移動是否會導致判斷錯誤？

如果是風吹動樹葉、窗簾，一些物品輕微搖動、擺動，會因為模糊化的作用與背景同化，不會被判斷為物體。如果是較大物品的移動，則會經過C語言程式的過濾，把非連續移動的物體剔除，不會對判斷造成影響。

## 三、圖片的解析度對判斷準確率是否有影響？

因為是兩張相同解析度的圖片做背景相減法，所以並無太大的影響。另外因為是透過長寬比例的變化做判斷，比例的影響並不會受圖片大小單位變化的影響。再者，即時演算對效率要求高，若使用較高畫質的圖片，將會增加判斷時間，不太適合以效率為要求的即時演算系統。另外較高解析度反而因為畫面細膩，會判斷出更多的環境變化，反而造成判斷的不便利。因此低解析度反而在去除背景的方面更乾淨。較容易影響到系統執行的是設備對色彩明暗的感知度，若畫面明暗不夠明顯，在做以色彩差異為基準的背景相減法時就無法確保相減後得出的前景是否準確到能偵測的程度，會造成判斷失準等問題。

## 四、使用影像監測有什麼優缺點？

在應用上，若將使用真實圖像的跌倒判斷系統安裝進居家做長時間的偵測，將會造成隱私上的疑慮，另外一些較隱私的地方也無法安裝設備，會有監測上的漏洞。視線死角、待測物離設備太近等情況下也會造成偵測上的困難，無法達到最大範圍的安全保障。但反之，使用影像監測沒有設備限制，只要有監視器等的影像擷取設備，即可安裝系統做檢測，不會有特定設備的限制。

## 柒、未來展望

一、期待有機會使用更高階的設備，因為網路攝影機的性能限制使得我們能使用的功能受限。

二、未來可以結合紅外線熱感應儀，在一些隱私敏感的地點可以改用熱感應儀，一般情況也能輔助主要系統來增加精準度，用來去除一些非人的物體。

三、在應用方面，未來可以與其他系統結合，如自動撥打電話、使用App監測等措施，以達到及時補救的功效。

四、希望能將我們所使用的設備做成一個可方便安裝的裝置，以利實際在受照顧者居家中安裝。

## 捌、參考資料及其他

王進德(2017)。Raspberry Pi 入門與機器人實作應用。新北市：博碩文化股份有限公司

Jack Creasey(2015)。Raspberry Pi 專案實作 | 語音時鐘 x 動作偵測 x 網路電台 x 循跡機器人。台北市：碁峰資訊股份有限公司

Brain W. Kernighan·Dennis M. Ritchie(2012)。C語言程式設計。新北市：台灣培生教育出版股份有限公司

蔡明志(2011)。資料結構：使用C。台北市：碁峰資訊股份有限公司

OpenCV官網。2017年11月5號，取自<https://opencv.org/>

OpenCV教學| 阿洲的程式教學。2017年11月5號，取自[http://monkeycoding.com/?page\\_id=12](http://monkeycoding.com/?page_id=12)

Caroline Rougier, Alain St-Arnaud, Jacqueline Rousseau and Jean Meunier (2011). Video Surveillance for Fall Detection, Video Surveillance, Prof. Weiyao Lin (Ed.), ISBN: 978-953-307-436-8, InTech

演算法筆記-Image。2017年11月5號，取自<http://www.csie.ntnu.edu.tw/~u91029/Image.html>

(2017年11月5號)取自邱俊賓(2011年)。腕錶式跌倒偵測系統之開發研究 The Decelopment of Wrist-Watch Fall Detection System。碩士論文

