教育部

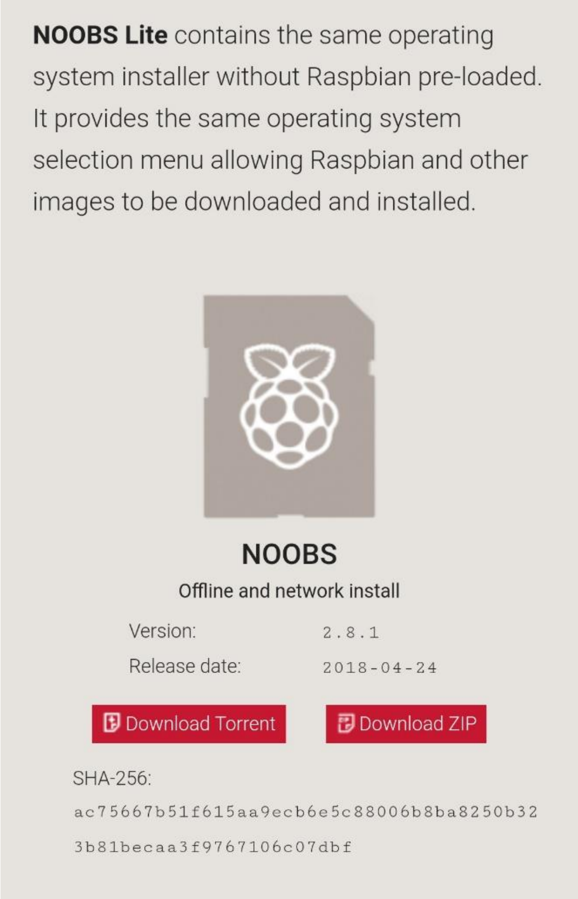
智慧電子整合性人才培育計畫平臺課程

實驗模組名稱：	行人偵測系統
開發學生：	沈思鎧
開發教師：	陳鵬升 教授
學校系所：	國立中正大學資訊工程學系
聯絡電話：	05-2720411 ext.33102
聯絡地址：	62102 嘉義縣民雄鄉大學路 168 號
實驗平台：	Raspberry Pi

<參考: 教育部 智慧電子整合性人才培育計畫平臺課程，"行人偵測系統"，郭俊因教授、劉千瑋同學>

階段一：安裝套件

- 安裝 Linux 作業系統
  到官網 https://www.raspberrypi.org/downloads/noobs/下載壓縮檔。



NOOBS Lite contains the same operating system installer without Raspbian pre-loaded. It provides the same operating system selection menu allowing Raspbian and other images to be downloaded and installed.

NOOBS

Offline and network install

Version:	2.8.1
Release date:	2018-04-24

SHA-256:

ac75667b51f615aa9ecb6e5c88006b8ba8250b32
3b81becaa3f9767106c07dbf

之後把 SD 卡格式化，先用 SD formatter 確認 SD 卡上的所有 partition 被刪除，再用 FAT32 Format (guiformat.exe) 格式化 SD 卡。(https://www.raspberrypi.org/documentation/installation/sdxc_formatting.md#)，

格式好後把下載好的壓縮檔解壓縮，把裡面的檔案全部複製到 SD 卡裡，再把 SD 卡插到 raspberrypi 上，電源螢幕鍵盤滑鼠接上後，就會出現安裝畫面。

- 安裝好作業系統後，進行系統更新及升級，並重新開機：

```
$ sudo rpi-update
$ sudo apt-get update
$ sudo apt-get upgrade
```

- 重新啟動 Raspberry Pi。

- 重新啟動系統後，安裝一些需要的編譯工具：

```
$ sudo apt-get install build-essential git cmake pkg-config
```

- 安裝影像 I/O 套件，包含 JPEG, PNG, TIFF 等所需套件，這個套件可以載入各種不同的影像檔案格式，如：JPEG, PNG, TIFF 等。

```
$ sudo apt-get install libjpeg8-dev libtiff5-dev libjasper-dev
libpng12-dev
```

註：安裝時出現以下錯誤：安裝 libtiff5-dev 前，需先安裝 libjpeg-dev

```
The following packages have unmet dependencies:
 libtiff5-dev : Depends: libjpeg-dev
E: Unable to correct problems, you have held broken packages.
```

- 安裝 video I/O 所需套件，使用 OpenCV 載入 video 檔案：

```
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

- 安裝 GTK 開發 library，這個 library 用在建立使用者介面 (Graphical User Interfaces, GUIs)，並可以編譯 OpenCV 的 highgui 子模組，才能顯示影像在畫面上。

```
$ sudo apt-get install libgtk2.0-dev
```

- 各種不同的 OpenCV 如矩陣運作等的最佳化之套件：

```
$ sudo apt-get install libatlas-base-dev gfortran
```

- **安裝 OpenCV:**

Cache opencv about software

```
$ apt-cache search opencv
```

Install opencv

//若 cache 版本非 2.1 請用 cache 到的版本

```
$ apt-get install libcv2.1 libcvaux2.1 libhighgui2.1
```

Install opencv dev

```
$ apt-get install libcv-dev libcvaux-dev libhighgui-dev

$ sudo apt-get install libopencv-dev
```

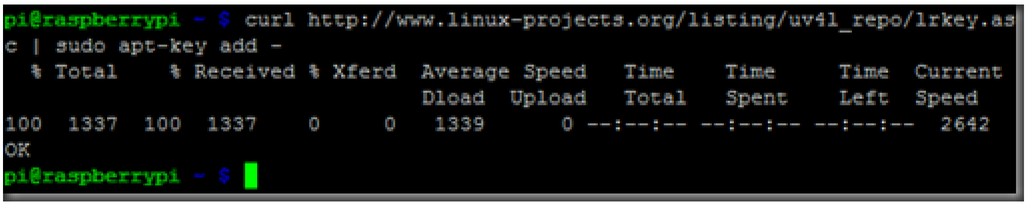
參考網址: http://atceiling.blogspot.tw/2017/02/raspberry-pi-opencv.html

階段二: 使用 USB Camera

除了官方版的 camera module, Raspberry Pi 也可以使用 USB Camera (或者叫 Web Cam)
以下使用 UV4L 及 v4l2-utils 來控制 USB amera.
安裝 UV4L

命令較長, 附上文字部份.
下載 UV4L

```
$curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc | sudo apt-key add -
```

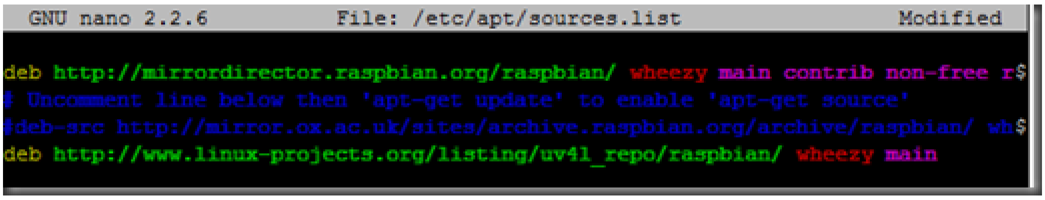


```
pi@raspberrypi ~ $ curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.as
c | sudo apt-key add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1337  100  1337    0     0   1339      0 --:--:-- --:--:-- --:--:--  2642
OK
pi@raspberrypi ~ $ 
```

打開 /etc/apt/sources.list

```
$ sudo nano /etc/apt/sources.list
```

在 /etc/apt/sources.list 這個檔案中, 加入以下文字
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ wheezy main



```
  GNU nano 2.2.6            File: /etc/apt/sources.list                 Modified

deb http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free r$
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://mirror.ox.ac.uk/sites/archive.raspbian.org/archive/raspbian/ wh$
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ wheezy main
```

然後更新及安裝

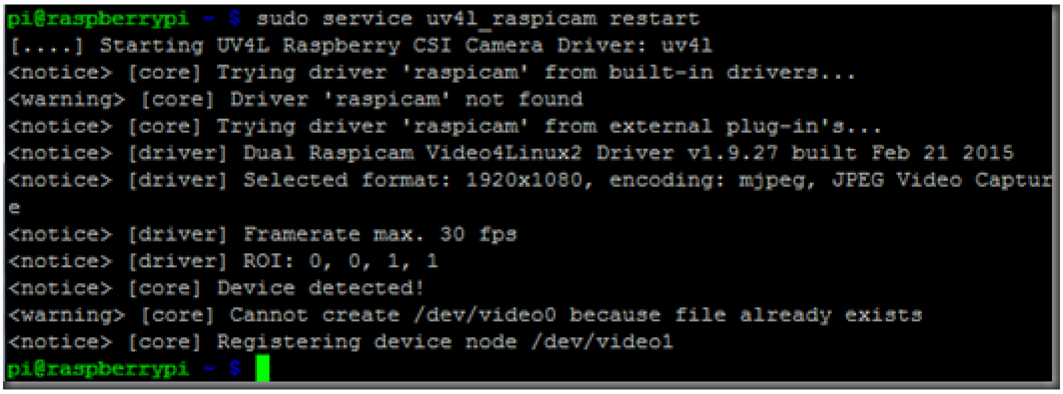
```
$ sudo apt-get update
$ sudo apt-get install uv4l uv4l-raspicam
```

如果要開機就載入, 要安裝額外的套件

```
$ sudo apt-get install uv4l-raspicam-extras
```

套件中已經包含了啟動的 script

```
$ sudo service uv4l_raspicam restart
```



```
pi@raspberrypi ~ $ sudo service uv4l_raspicam restart
[....] Starting UV4L Raspberry CSI Camera Driver: uv4l
<notice> [core] Trying driver 'raspicam' from built-in drivers...
<warning> [core] Driver 'raspicam' not found
<notice> [core] Trying driver 'raspicam' from external plug-in's...
<notice> [driver] Dual Raspicam Video4Linux2 Driver v1.9.27 built Feb 21 2015
<notice> [driver] Selected format: 1920x1080, encoding: mjpeg, JPEG Video Captur
e
<notice> [driver] Framerate max. 30 fps
<notice> [driver] ROI: 0, 0, 1, 1
<notice> [core] Device detected!
<warning> [core] Cannot create /dev/video0 because file already exists
<notice> [core] Registering device node /dev/video1
pi@raspberrypi ~ $
```

系統中有兩個 camera. 一個是連接到板子上的 CSI 介面的 camera, 一個是 USB camera

由於 driver 是新加入的, 也許會有一些問題. 先更新一下。

$ sudo rpi-update

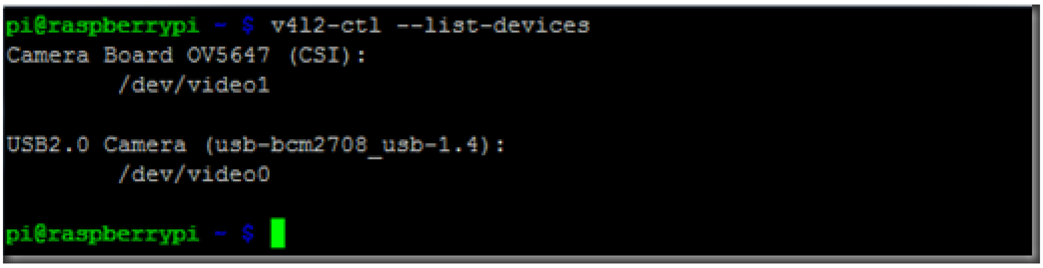
安裝 v4l2-utils

注意, v4l2 是 video for Linux version 2 的縮寫. 所以第 3 個字母是 L 的小寫, 而不是數字的 1。在新版的 raspbian 的 image 檔中, 已經加入了 v4l2-utils 這些工具. 可以試試看下以下的命令, 看看系統的回應. 如果是沒有安裝, 會出現 command not found 的錯誤訊息，如果是 command not found, 可以用以下命令安裝

$ sudo apt-get install v4l-utils

注意, 這邊是 v4l-utils, 不是 v4l2-utils

插上 usb camera 後, 重新開機, 下 v4l2-ctl 命令, 可以找到現在連接的 camera, 有兩個. CSI 介面 以及 USB 介面.



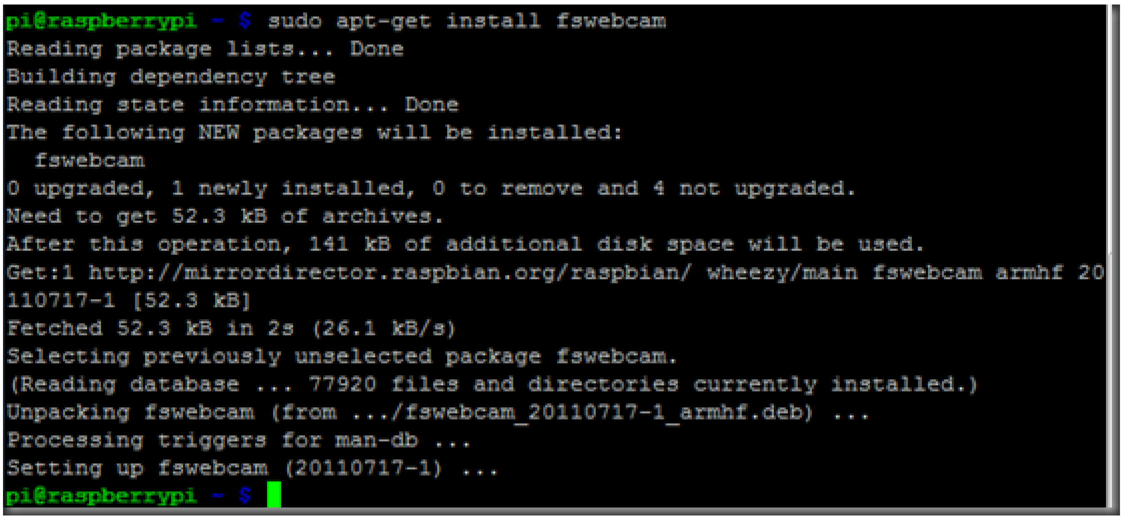
```
pi@raspberrypi ~ $ v4l2-ctl --list-devices
Camera Board OV5647 (CSI):
        /dev/video1

USB2.0 Camera (usb-bcm2708_usb-1.4):
        /dev/video0

pi@raspberrypi ~ $
```

安裝 fswebcam

由於 raspivid, raspistill 只能用在官方的 camera module 上. 我們需要其他的軟體來使用 USB camera. 這裡先使用 fswebcam.

先安裝

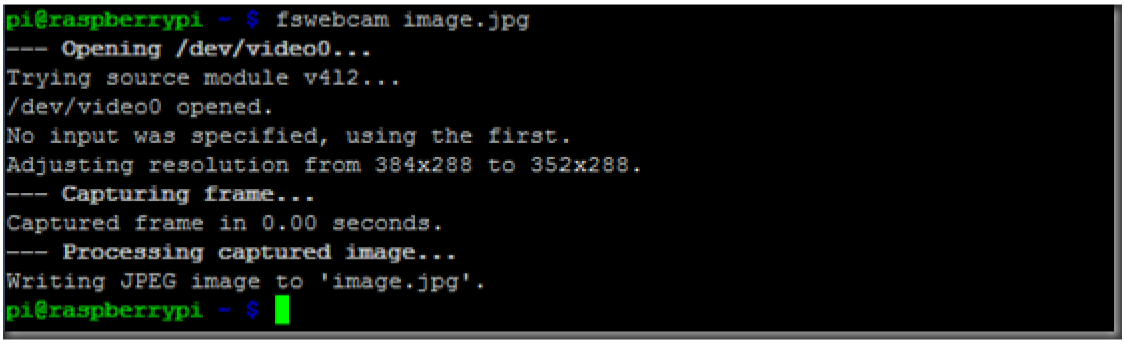
$ sudo apt-get install fswebcam



```
pi@raspberrypi ~ $ sudo apt-get install fswebcam
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  fswebcam
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 52.3 kB of archives.
After this operation, 141 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main fswebcam armhf 20
110717-1 [52.3 kB]
Fetched 52.3 kB in 2s (26.1 kB/s)
Selecting previously unselected package fswebcam.
(Reading database ... 77920 files and directories currently installed.)
Unpacking fswebcam (from .../fswebcam_20110717-1_armhf.deb) ...
Processing triggers for man-db ...
Setting up fswebcam (20110717-1) ...
pi@raspberrypi ~ $
```

直接拍照

$ fswebcam image.jpg



```
pi@raspberrypi ~ $ fswebcam image.jpg
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image.jpg'.
pi@raspberrypi ~ $
```

拍出來的照片有時候會壞掉. 官方網站上說是有些 web camera 不穩定.

以下說明建立 script 來拍攝照片.

首先建立 webcam 目錄

$ cd /home/pi

$ mkdir web

$ nano webcam.sh

在 webcam.sh 中, 加入以下命令

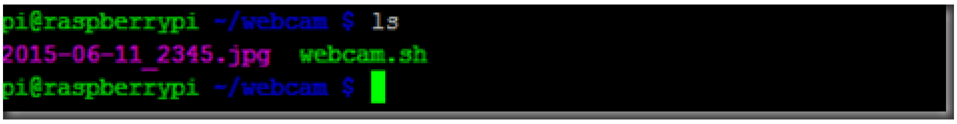
```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
fswebcam --no-banner /home/pi/web/$DATE.jpg
```

存檔 (按 control + X 跳出後, 在提示儲存的地方按 Y)

把它加上可以執行的屬性，之後執行。

$ chmod +x webcam.sh

$ ./webcam.sh



```
pi@raspberrypi ~/webcam $ ls
2015-06-11_2345.jpg  webcam.sh
pi@raspberrypi ~/webcam $ 
```

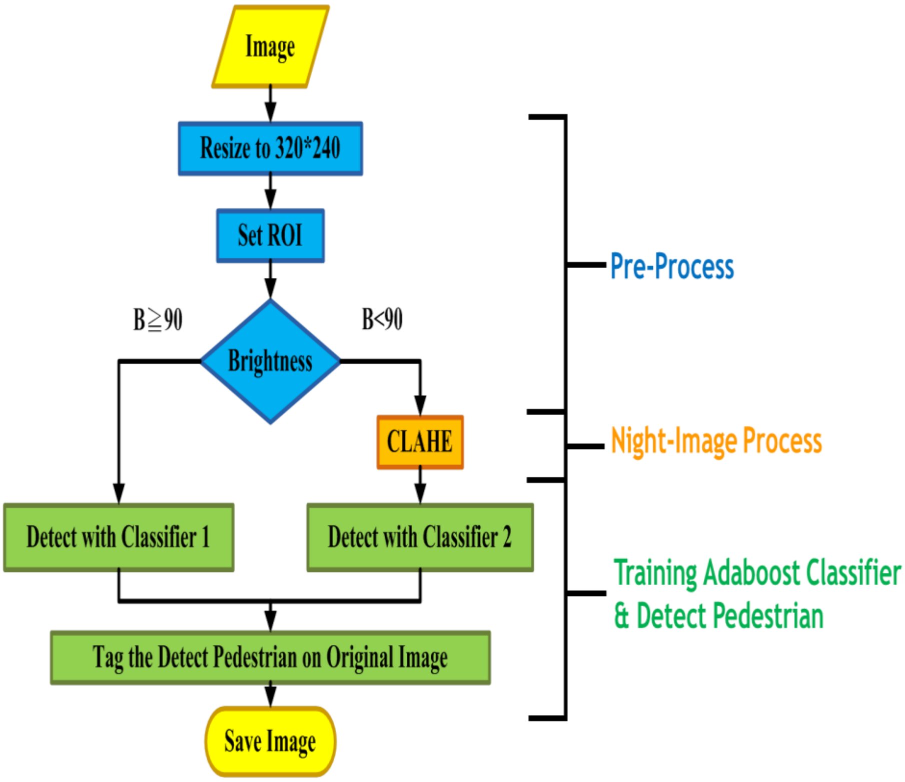
可以看到拍攝了一張照片

參考網址: http://nickinwork.blogspot.tw/2015/06/rpi-usb-camera.html

本實驗會用到許多 OpenCV 的函式，在開始寫程式之前可以先了解Ｏpen CV 中影像處理相關函式、變數的基本結構，並且將所有所需的函式載入寫在一個統一的檔案,如:myOpenCV.h:　之後只要載入這個檔就可使用:

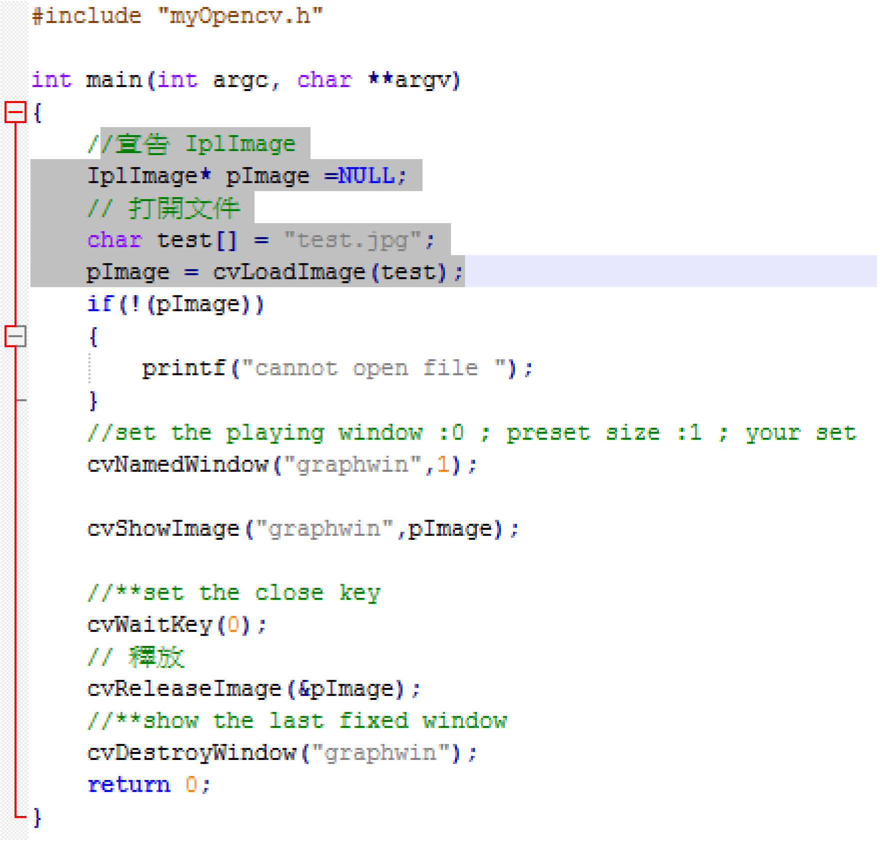
```
#include <cv.h>
//#include <cxore.h>
#include <highgui.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>
#include <ml.h>
#include <iostream>
using namespace std;
```

行人偵測系統的系統架構



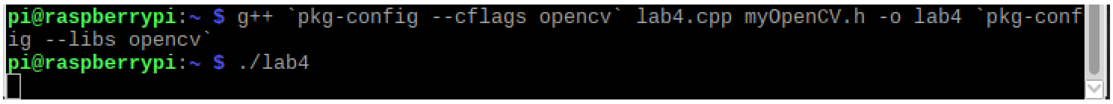
圖片處理:

- Step1: 開啟影像



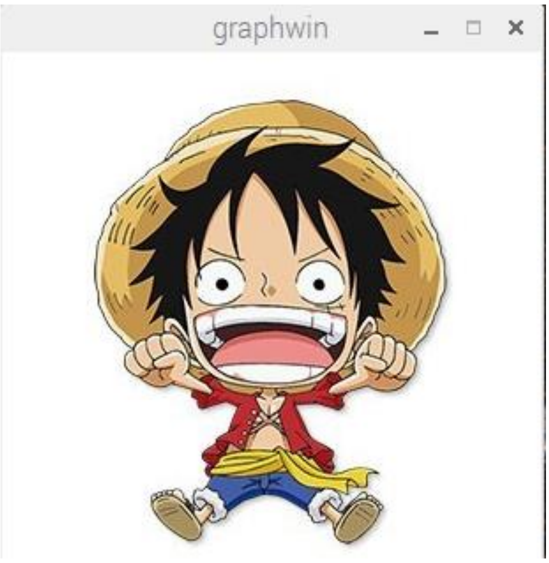
```
#include "myOpencv.h"

int main(int argc, char **argv)
{
    //宣告 IplImage
    IplImage* pImage =NULL;
    // 打開文件
    char test[] = "test.jpg";
    pImage = cvLoadImage(test);
    if(!(pImage))
    {
        printf("cannot open file ");
    }
    //set the playing window :0 ; preset size :1 ; your set
    cvNamedWindow("graphwin",1);

    cvShowImage("graphwin",pImage);

    //**set the close key
    cvWaitKey(0);
    // 釋放
    cvReleaseImage(&pImage);
    //**show the last fixed window
    cvDestroyWindow("graphwin");
    return 0;
}
```

編譯檔案:



```
pi@raspberrypi:~ $ g++ `pkg-config --cflags opencv` lab4.cpp myOpenCV.h -o lab4 `pkg-conf
ig --libs opencv`
pi@raspberrypi:~ $ ./lab4
```

完成出現影像:

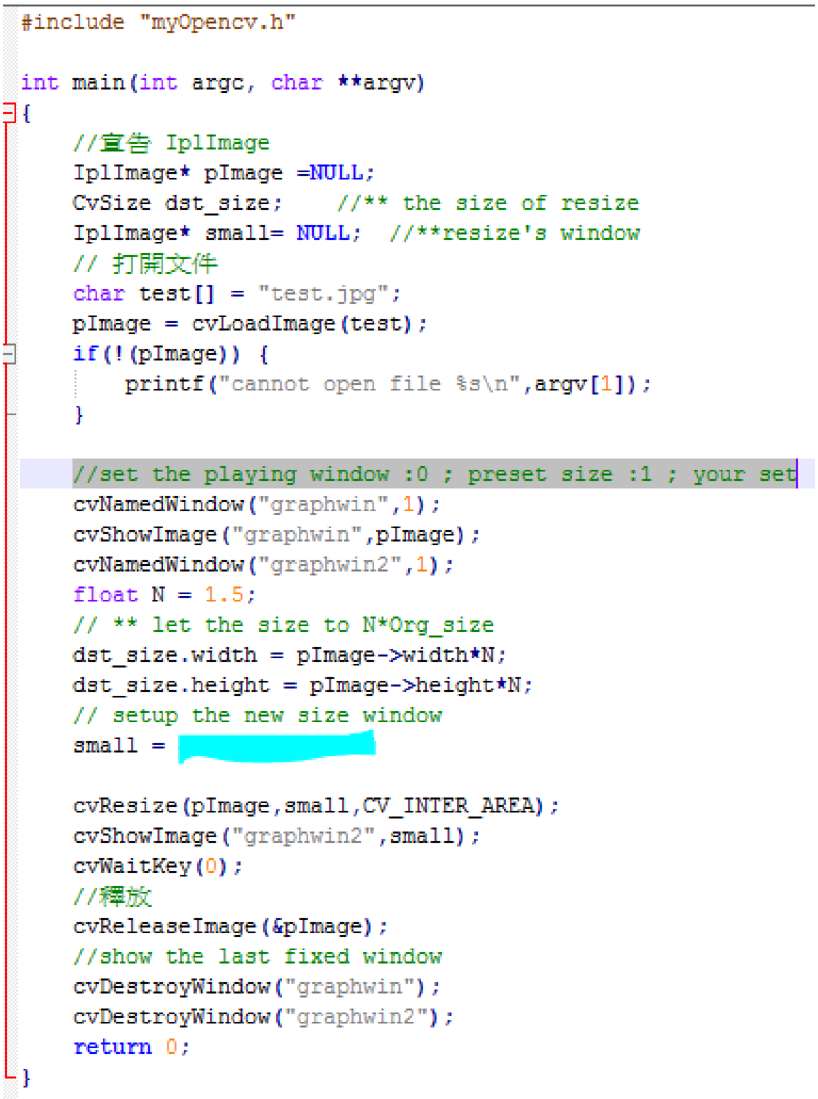


- Step2: 縮放尺寸

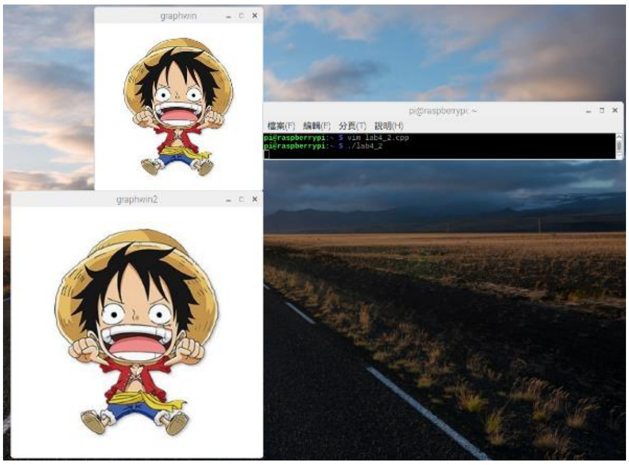
1. pImage：在Step1取得的影像。

2. small= cvCreateImage(cvSize(320,240),pImage->depth,pImage->nChannels);
設定縮小影像，宣告一個變數small，並利用cvCreateImage初始化存放空間，將大小設定成欲縮小的值，其餘設定與pImage相同。

3. cvResize(pImage,small,CV_INTER_AREA);
呼叫改變尺寸函式，輸入影像為 pImage，會依照輸出影像 small 的尺寸做縮放，使用 bilinear 演算法(CV_INTER_AREA)。



```
#include "myOpencv.h"

int main(int argc, char **argv)
{
    //宣告 IplImage
    IplImage* pImage =NULL;
    CvSize dst_size;    //** the size of resize
    IplImage* small= NULL;  //**resize's window
    // 打開文件
    char test[] = "test.jpg";
    pImage = cvLoadImage(test);
    if(!(pImage)) {
        printf("cannot open file %s\n",argv[1]);
    }

    //set the playing window :0 ; preset size :1 ; your set
    cvNamedWindow("graphwin",1);
    cvShowImage("graphwin",pImage);
    cvNamedWindow("graphwin2",1);
    float N = 1.5;
    // ** let the size to N*Org_size
    dst_size.width = pImage->width*N;
    dst_size.height = pImage->height*N;
    // setup the new size window
    small = [redacted]

    cvResize(pImage,small,CV_INTER_AREA);
    cvShowImage("graphwin2",small);
    cvWaitKey(0);
    //釋放
    cvReleaseImage(&pImage);
    //show the last fixed window
    cvDestroyWindow("graphwin");
    cvDestroyWindow("graphwin2");
    return 0;
}
```



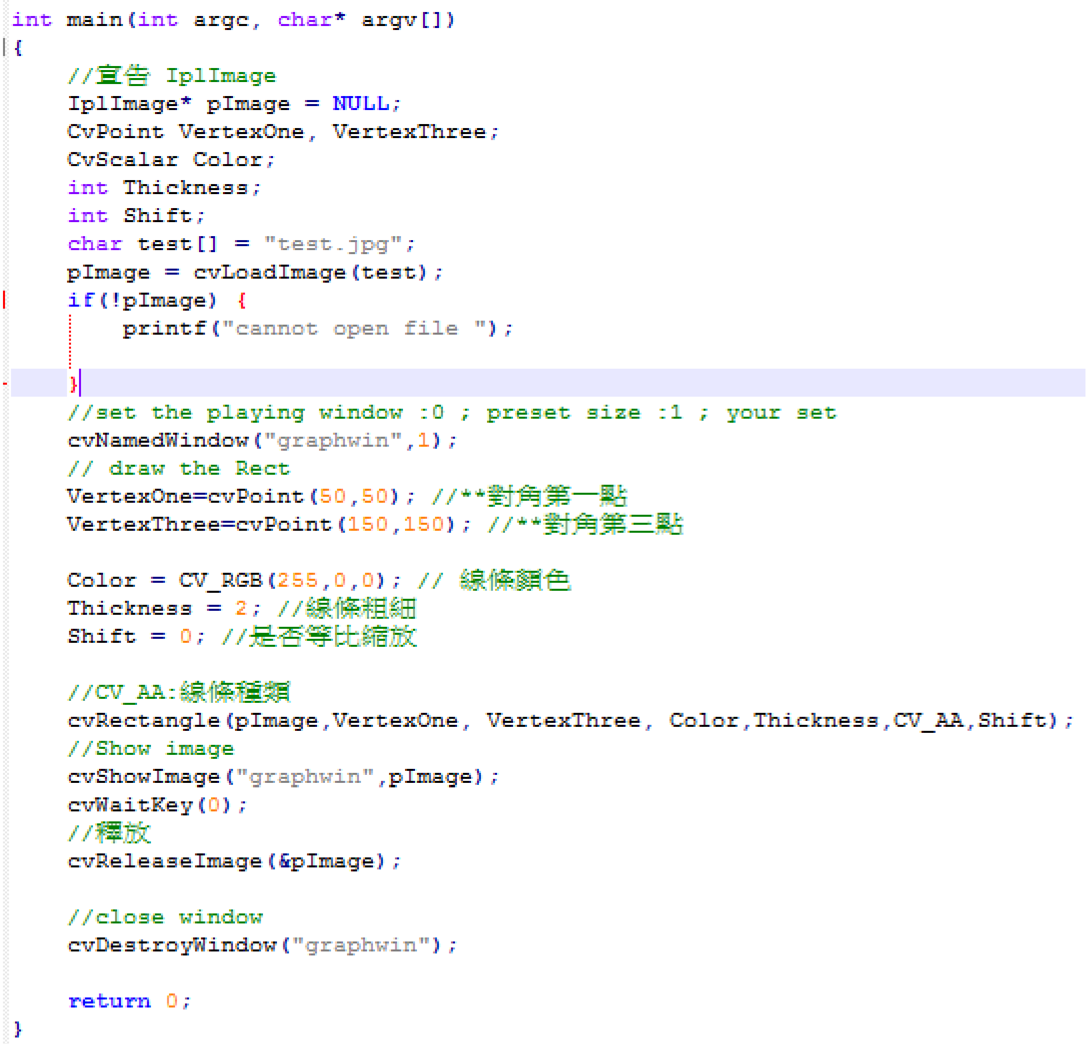
```
pi@raspberrypi:~ $ vim lab4_2.cpp
pi@raspberrypi:~ $ ./lab4_2
```

Step 3: 圈選感區域 (Set ROI) (ROI => Region of Interest)

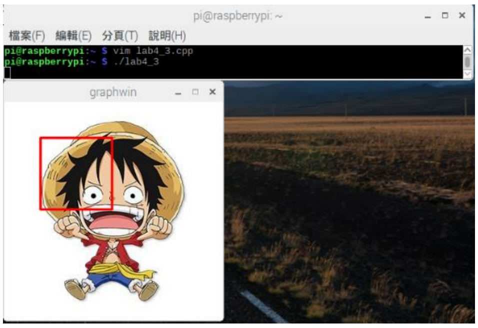
1. cvRect為初始化矩形函數，設定感興趣區域大小，型態為矩形，參數依序為左上角x座標、y座標，寬，高。

2. cvSetImageROI(pImage,ROI);(注:本次並未使用)
   影像設定感興趣區域。

3. cvResetImageROI(pImage);(注:本次並未使用)
   感興趣區域內運算結束，取消感興趣區域設定



```
int main(int argc, char* argv[])
{
    //宣告 IplImage
    IplImage* pImage = NULL;
    CvPoint VertexOne, VertexThree;
    CvScalar Color;
    int Thickness;
    int Shift;
    char test[] = "test.jpg";
    pImage = cvLoadImage(test);
    if(!pImage) {
        printf("cannot open file ");

    }
    //set the playing window :0 ; preset size :1 ; your set
    cvNamedWindow("graphwin",1);
    // draw the Rect
    VertexOne=cvPoint(50,50); //**對角第一點
    VertexThree=cvPoint(150,150); //**對角第三點

    Color = CV_RGB(255,0,0); // 線條顏色
    Thickness = 2; //線條粗細
    Shift = 0; //是否等比縮放

    //CV_AA:線條種類
    cvRectangle(pImage,VertexOne, VertexThree, Color,Thickness,CV_AA,Shift);
    //Show image
    cvShowImage("graphwin",pImage);
    cvWaitKey(0);
    //釋放
    cvReleaseImage(&pImage);

    //close window
    cvDestroyWindow("graphwin");

    return 0;
}
```



```
pi@raspberrypi:~ $ vim lab4_3.cpp
pi@raspberrypi:~ $ ./lab4_3
```

影像前置處理 **(Pre-process)**

- **Step 1:**取得影像

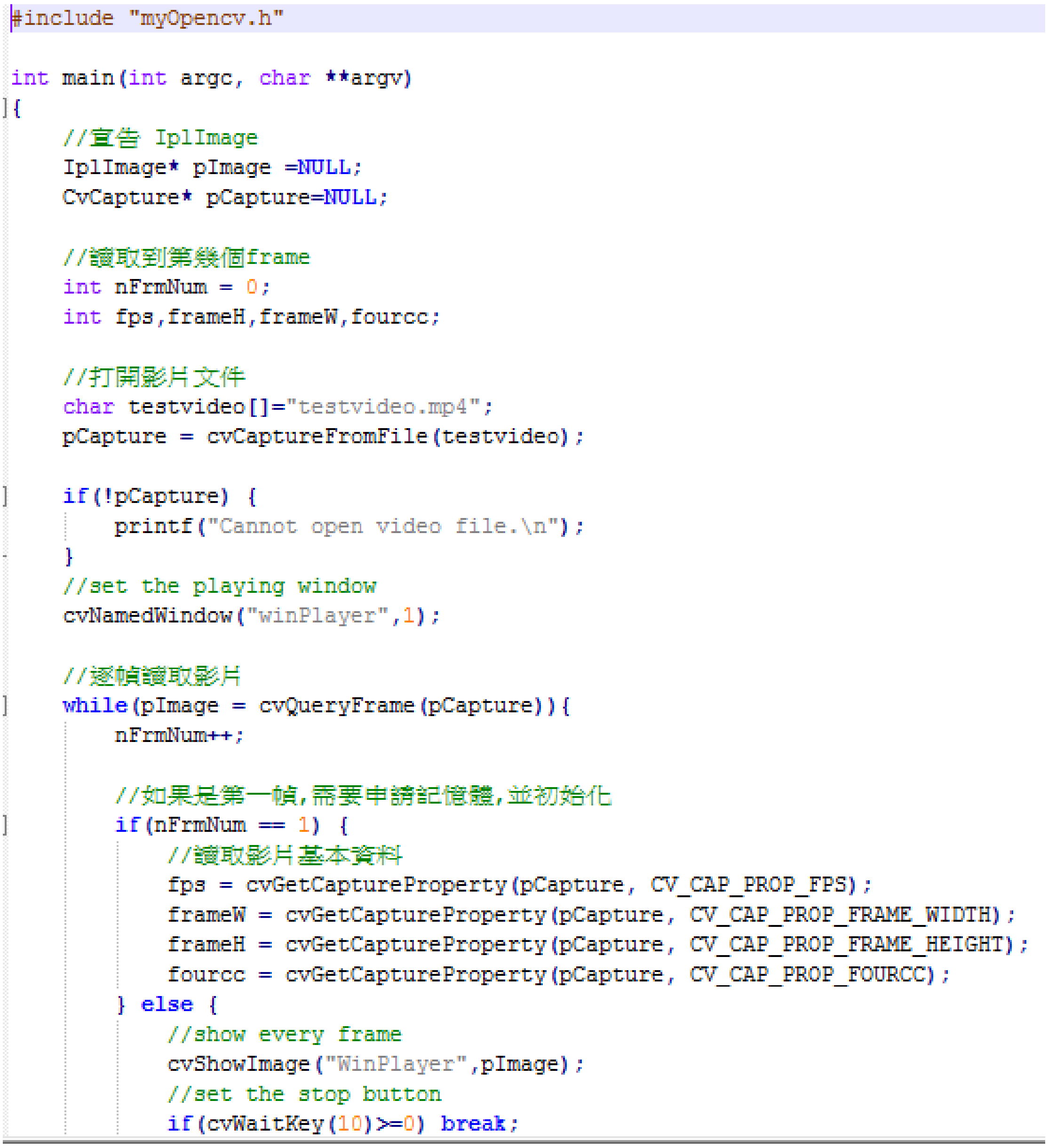
1. 宣告**cvCapture* pCapture;**
cvCapture 是視頻獲取結構，就是用來當作獲取視頻函式的參數。

2. **pCapture = cvCaptureFromFile(filename);**
呼叫從檔案讀取影像的函式，pCapture為1.宣告接收影像檔資料的變數，filename是要開啟的影像檔名，opencv支援.avi。

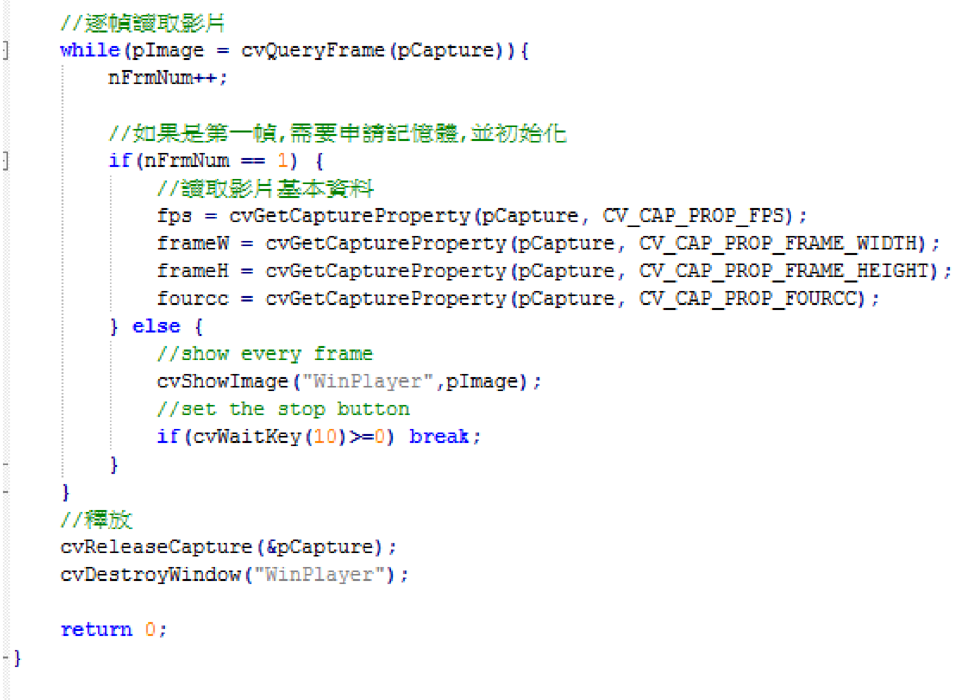
3. **IplImage*pImage = cvQueryFrame(pCapture);**
從pCapture取出影像中的幀(frame)存至pImage，接下來都將對pImage做處理，若函式回傳值為NULL表示最後一張frame。

4. **cvGetCaptureProperty**函式可以回傳影像裡的基本資料，再處理影像以及將影像寫入新視頻文件時需要這些資料，可透過此函式取得下列資料：

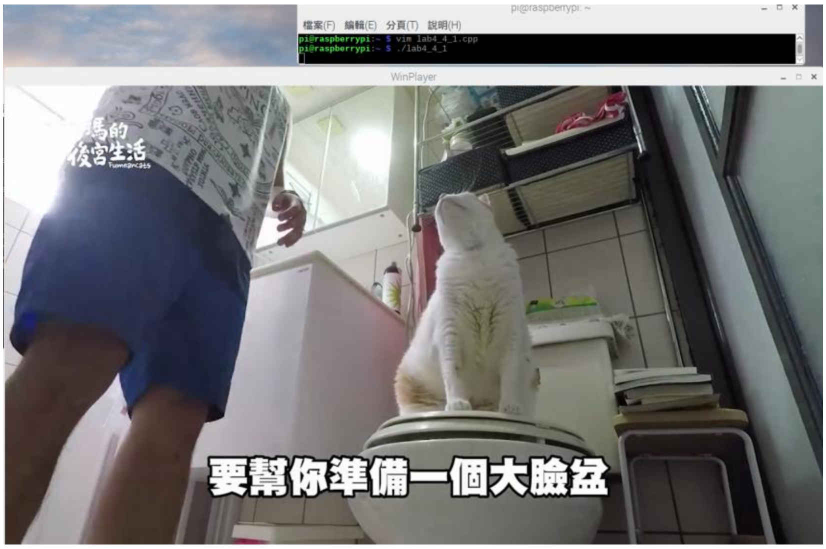
CV_CAP_PROP_POS_MSEC	影片目前位置，為毫秒數或者視頻獲取時間戳
CV_CAP_PROP_POS_FREMES	將被下一步解壓/獲取的幀索引，以0為起點
CV_CAP_PROP_POS_AVI_RATIO	視頻文件的相對位置(0: 影片的開始，1: 影片的結尾)
CV_CAP_PROP_FRAME_WIDTH	視頻流中的幀寬度
CV_CAP_PROP_FRAME_HEIGHT	視頻流中的幀高度
CV_CAP_PROP_FPS	幀率
CV_CAP_PROP_FOURCC	表示codec的四個字元
CV_CAP_PROP_FRAME_COUNT	視頻文件中幀的總數



```
#include "myOpencv.h"

int main(int argc, char **argv)
{
    //宣告 IplImage
    IplImage* pImage =NULL;
    CvCapture* pCapture=NULL;

    //讀取到第幾個frame
    int nFrmNum = 0;
    int fps,frameH,frameW,fourcc;

    //打開影片文件
    char testvideo[]="testvideo.mp4";
    pCapture = cvCaptureFromFile(testvideo);

    if(!pCapture) {
        printf("Cannot open video file.\n");
    }
    //set the playing window
    cvNamedWindow("winPlayer",1);

    //逐幀讀取影片
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            //讀取影片基本資料
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
        } else {
            //show every frame
            cvShowImage("WinPlayer",pImage);
            //set the stop button
            if(cvWaitKey(10)>=0) break;
```



```
    //逐幀讀取影片
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            //讀取影片基本資料
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
        } else {
            //show every frame
            cvShowImage("WinPlayer",pImage);
            //set the stop button
            if(cvWaitKey(10)>=0) break;
        }
    }
    //釋放
    cvReleaseCapture(&pCapture);
    cvDestroyWindow("WinPlayer");

    return 0;
}
```



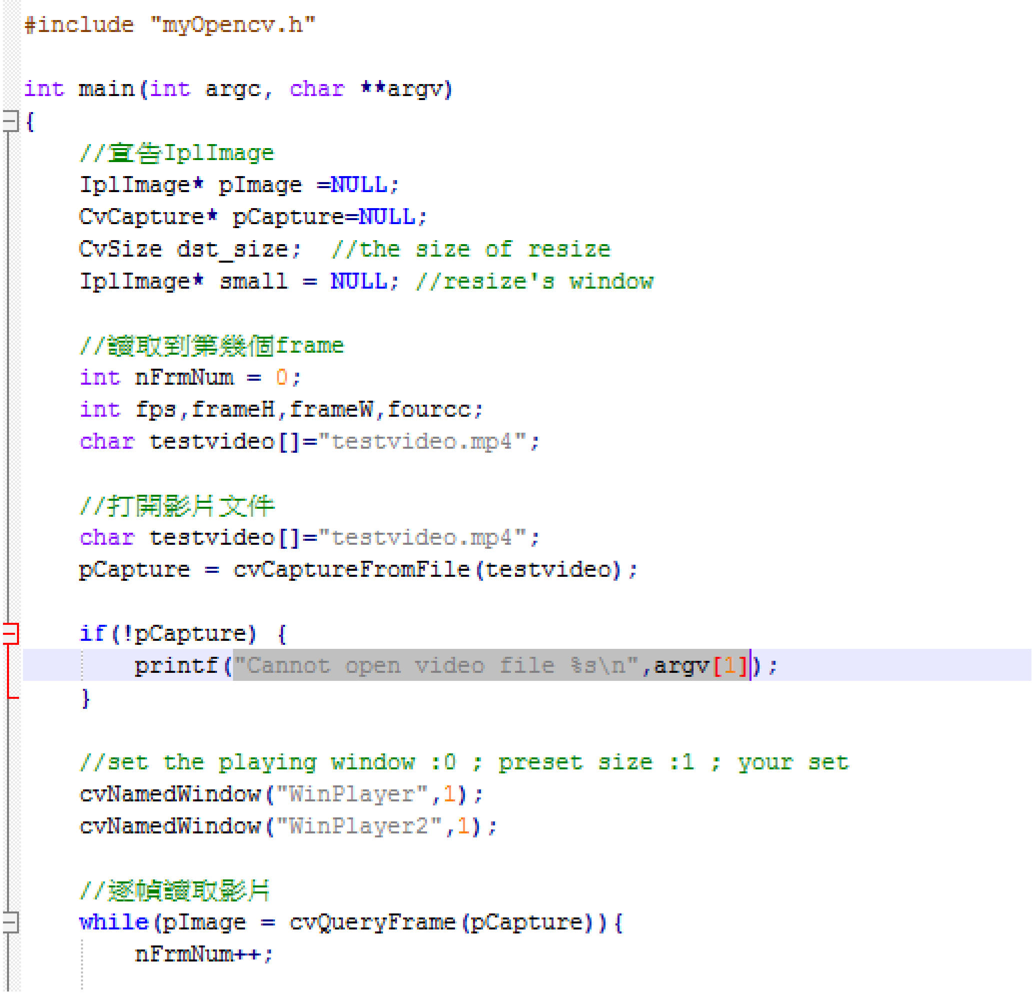
```
pi@raspberrypi:~ $ vim lab4_4_1.cpp
pi@raspberrypi:~ $ ./lab4_4_1
```

- **Step 2:** 縮小尺寸 **(Resize)**

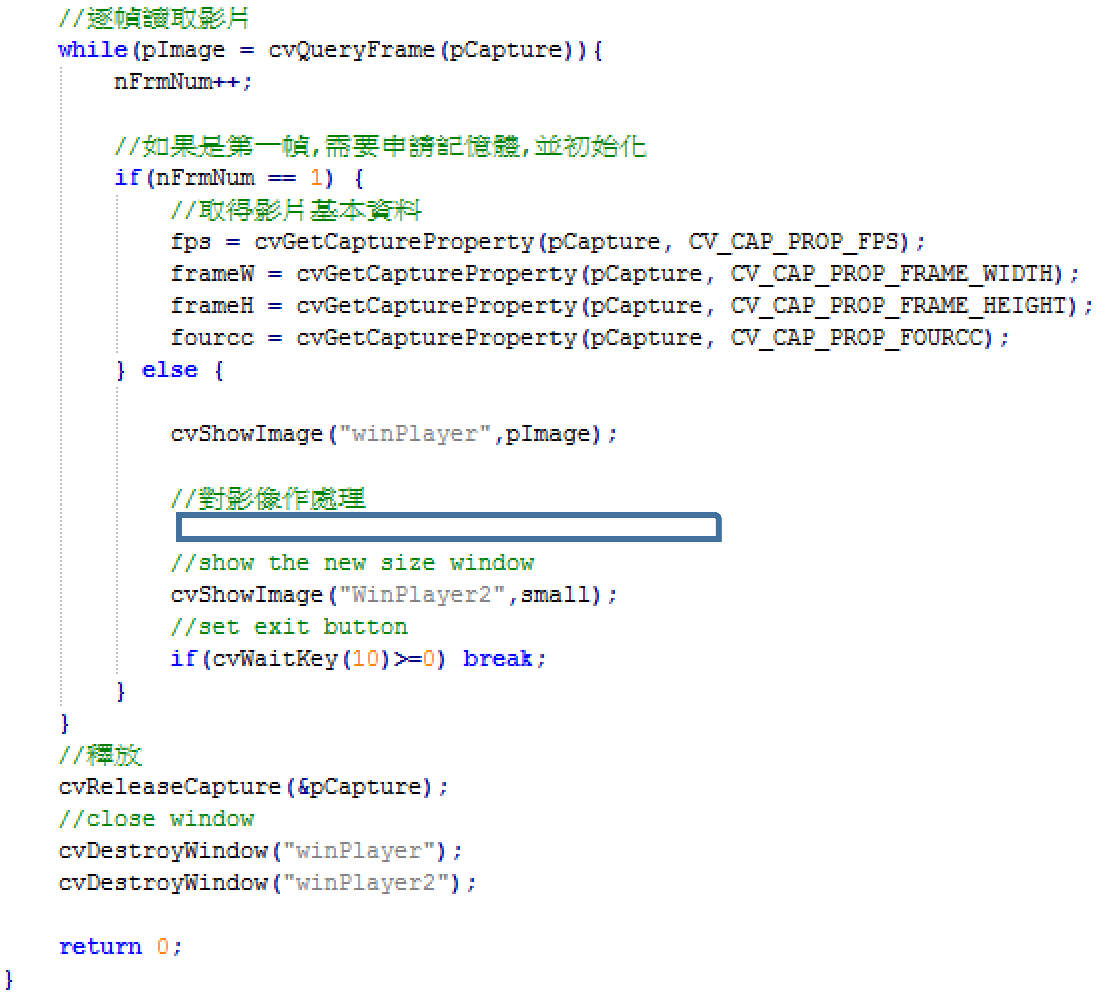
1. **pImage**：在Step1取得的影像。

2. **small= cvCreateImage(cvSize(320,240),pImage->depth,pImage->nChannels);**
設定縮小影像，宣告一個變數small，並利用cvCreateImage初始化存放空間，將大小設定成欲縮小的值，其餘設定與pImage相同。

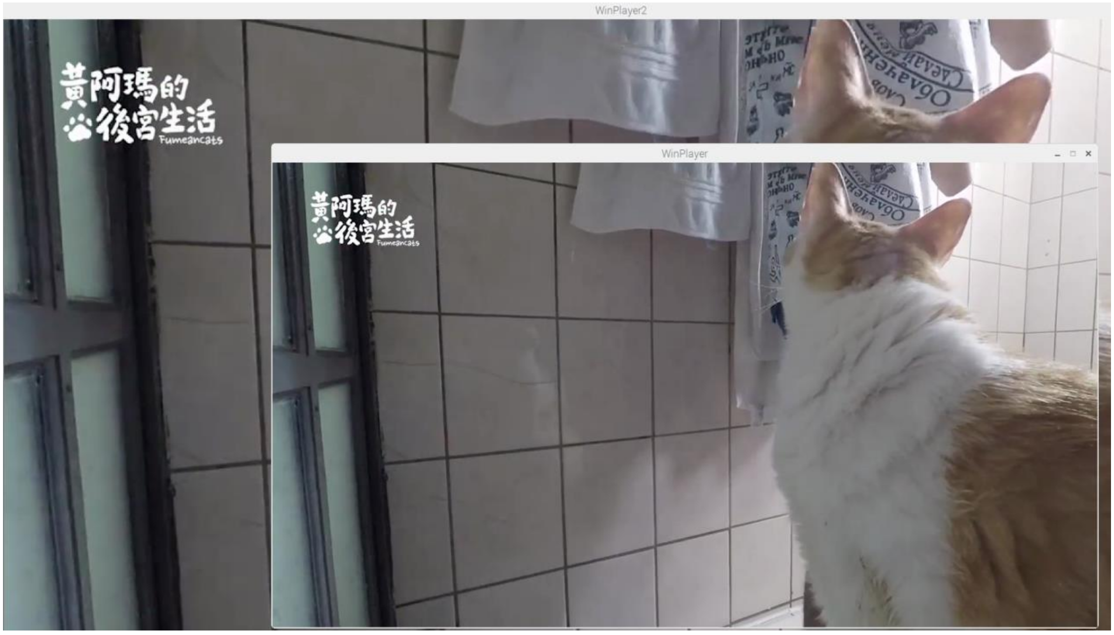
3. **cvResize(pImage,small,CV_INTER_AREA);**
呼叫改變尺寸函式，輸入影像為 pImage，會依照輸出影像 small 的尺寸做縮放，使用 bilinear 演算法(**CV_INTER_AREA**)。



```
#include "myOpencv.h"

int main(int argc, char **argv)
{
    //宣告IplImage
    IplImage* pImage =NULL;
    CvCapture* pCapture=NULL;
    CvSize dst_size;  //the size of resize
    IplImage* small = NULL; //resize's window

    //讀取到第幾個frame
    int nFrmNum = 0;
    int fps,frameH,frameW,fourcc;
    char testvideo[]="testvideo.mp4";

    //打開影片文件
    char testvideo[]="testvideo.mp4";
    pCapture = cvCaptureFromFile(testvideo);

    if(!pCapture) {
        printf("Cannot open video file %s\n",argv[1]);
    }

    //set the playing window :0 ; preset size :1 ; your set
    cvNamedWindow("WinPlayer",1);
    cvNamedWindow("WinPlayer2",1);

    //逐幀讀取影片
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;
```



```
    //逐幀讀取影片
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            //取得影片基本資料
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
        } else {

            cvShowImage("winPlayer",pImage);

            //對影像作處理

            //show the new size window
            cvShowImage("WinPlayer2",small);
            //set exit button
            if(cvWaitKey(10)>=0) break;
        }
    }
    //釋放
    cvReleaseCapture(&pCapture);
    //close window
    cvDestroyWindow("winPlayer");
    cvDestroyWindow("winPlayer2");

    return 0;
}
```



- **Step 3:** 圈選感興趣區域 **(Set ROI)**

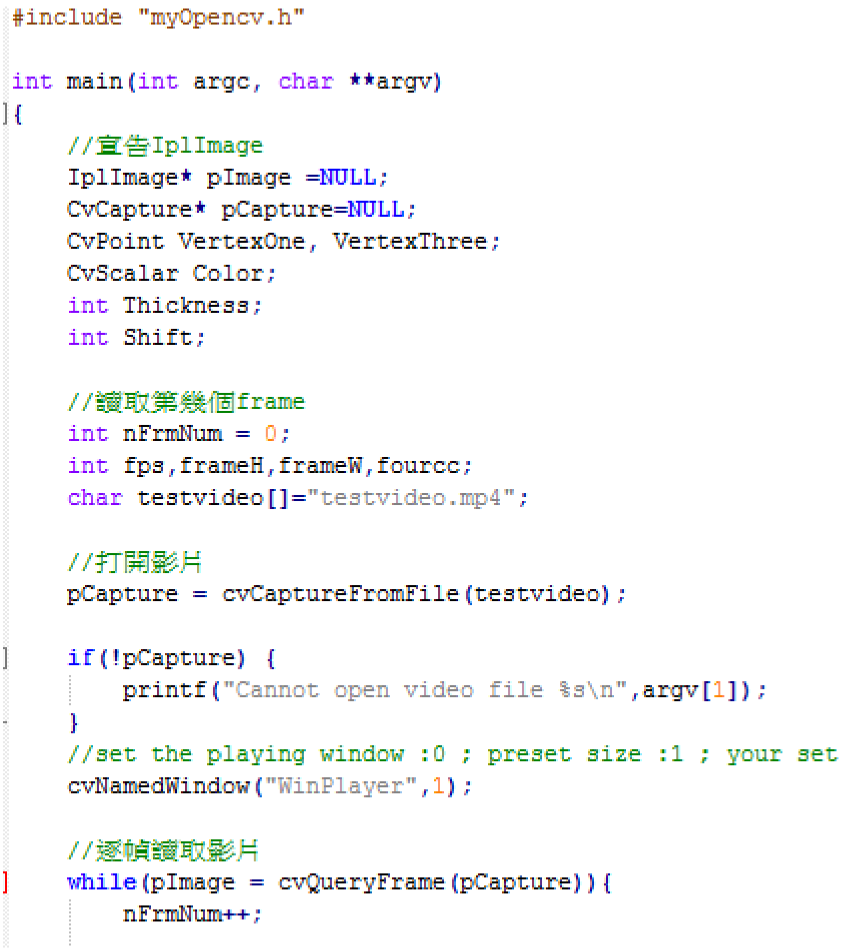
1. **cvRect**為初始化矩形函數，設定感興趣區域大小，型態為矩形，參數依序為左上角x座標、y座標，寬，高。

2. **cvSetImageROI(pImage,ROI);(**注**:**本次並未使用**)**

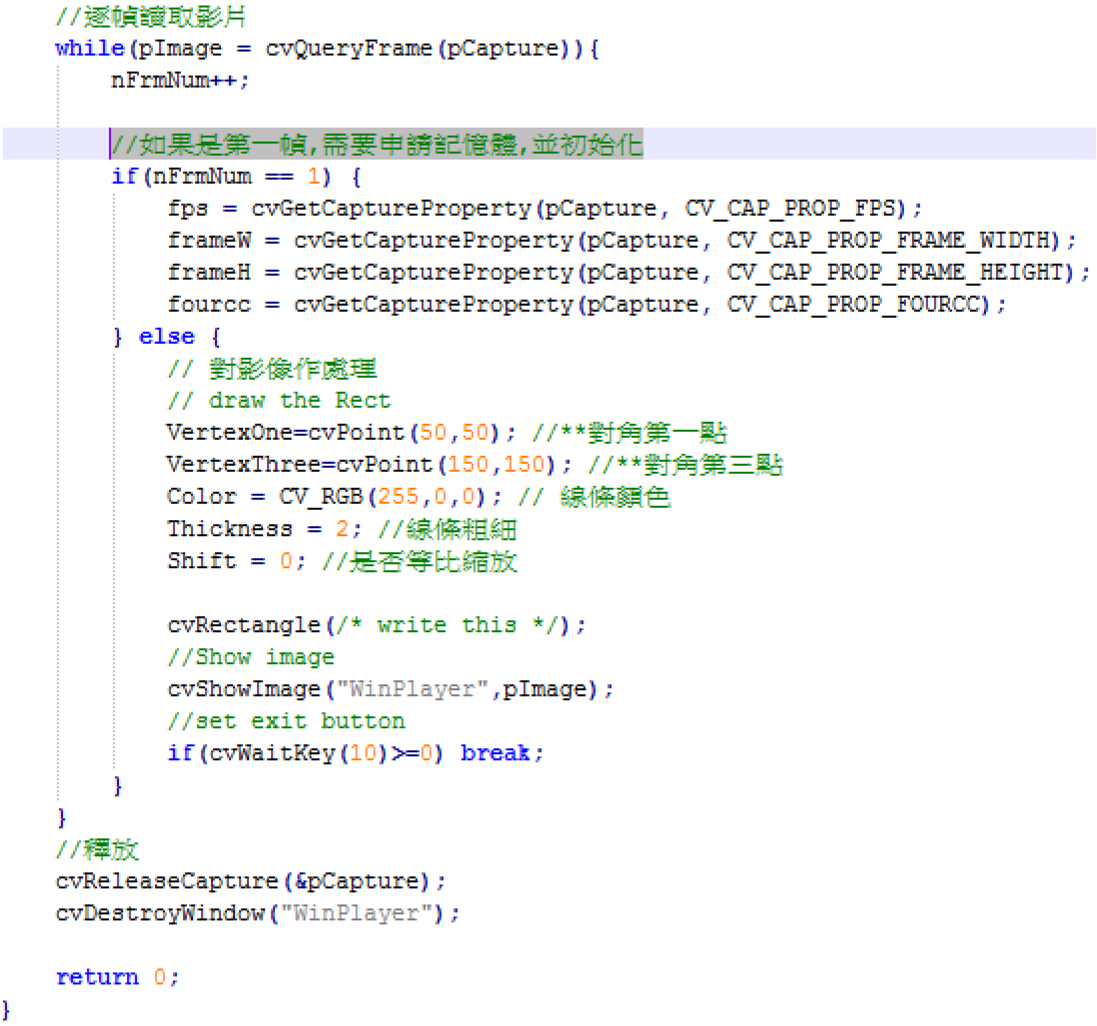
   影像設定感興趣區域。

3. **cvResetImageROI(pImage);(**注**:**本次並未使用**)**

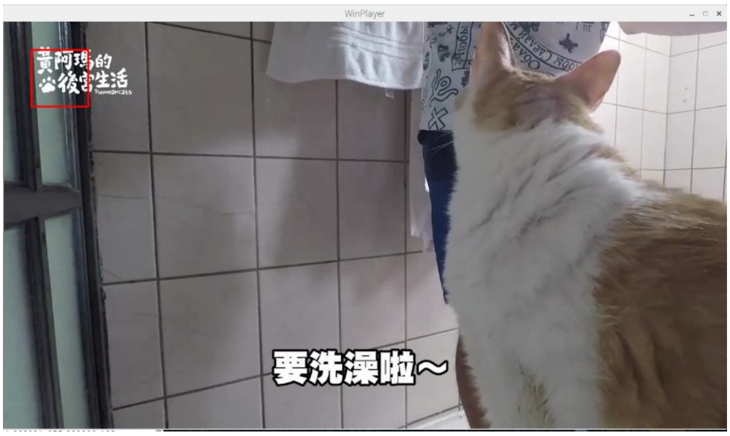
感興趣區域內運算結束，取消感興趣區域設定



```
#include "myOpencv.h"

int main(int argc, char **argv)
{
    //宣告IplImage
    IplImage* pImage =NULL;
    CvCapture* pCapture=NULL;
    CvPoint VertexOne, VertexThree;
    CvScalar Color;
    int Thickness;
    int Shift;

    //讀取第幾個frame
    int nFrmNum = 0;
    int fps,frameH,frameW,fourcc;
    char testvideo[]="testvideo.mp4";

    //打開影片
    pCapture = cvCaptureFromFile(testvideo);

    if(!pCapture) {
        printf("Cannot open video file %s\n",argv[1]);
    }
    //set the playing window :0 ; preset size :1 ; your set
    cvNamedWindow("WinPlayer",1);

    //逐幀讀取影片
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;
```



```
    //逐幀讀取影片
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
        } else {
            // 對影像作處理
            // draw the Rect
            VertexOne=cvPoint(50,50); //**對角第一點
            VertexThree=cvPoint(150,150); //**對角第三點
            Color = CV_RGB(255,0,0); // 線條顏色
            Thickness = 2; //線條粗細
            Shift = 0; //是否等比縮放

            cvRectangle(/* write this */);
            //Show image
            cvShowImage("WinPlayer",pImage);
            //set exit button
            if(cvWaitKey(10)>=0) break;
        }
    }
    //釋放
    cvReleaseCapture(&pCapture);
    cvDestroyWindow("WinPlayer");

    return 0;
}
```



夜間影像處理 **(Night-image process)**

當夜間影像光照度不足時，影像像素值會集中偏暗，此時像素值過於相近，使特徵值時辨識度不高，本實驗使用 Contrast Limited AdaptiveHistogram Equalization(CLAHE)方法，增強對比度，CLAHE 演算法是利用限制直方圖的高度來限制局部對比度的增強幅度，進而抑制雜訊及局部對比度過於增強的狀況。

- **Step 1:** 建立函式，宣告與初始化參數

1. 宣告void CLAHE(IplImage* img)

2. **IplImage* v**將影像轉為灰階圖v，宣告並初始化v為單通道灰階圖，呼叫轉換函式將彩色圖img轉為灰階圖v

3. 宣告直方圖矩陣、剪切後直方圖矩陣、累進機率矩陣、受限值、受限值参數、迴圈所需變數。

- **Step 2:** 計算影像直方圖 將影像中出現過的像素值數目總合計算出來

使用雙迴圈走遍所有影像中的像素，將對應index的直方圖矩陣加一，計算出此張影像中每個像素值(0~255)的個數。

- **Step 3:** 計算受限值

限制直方圖高度，將高於受限值的個數都截斷，可以限制局部對比度過高。

受限值計算公式： β=$MN(1+\alpha 100(s_{max}-1))$

M: 總像素數目、N: 可能出現像素值、s_{max}: 為直方圖最大斜率、

α: 截斷係數，α=[1,100]

- **Step 4:** 截斷像素

走遍直方圖，將高於受限值的數目截斷，並計算全部截斷數目。

- **Step5:** 像素量重新分配

1. 將截斷的數目平均分配回直方圖中，降低過強的局部對比度

2. 計算平均截斷數目m=$ex256$

3. 截斷後數目小於受限值且超過m的，將m個像素填回直方圖

4. 截斷後數目小於受限值且不超過m的，將數目填滿至受限值

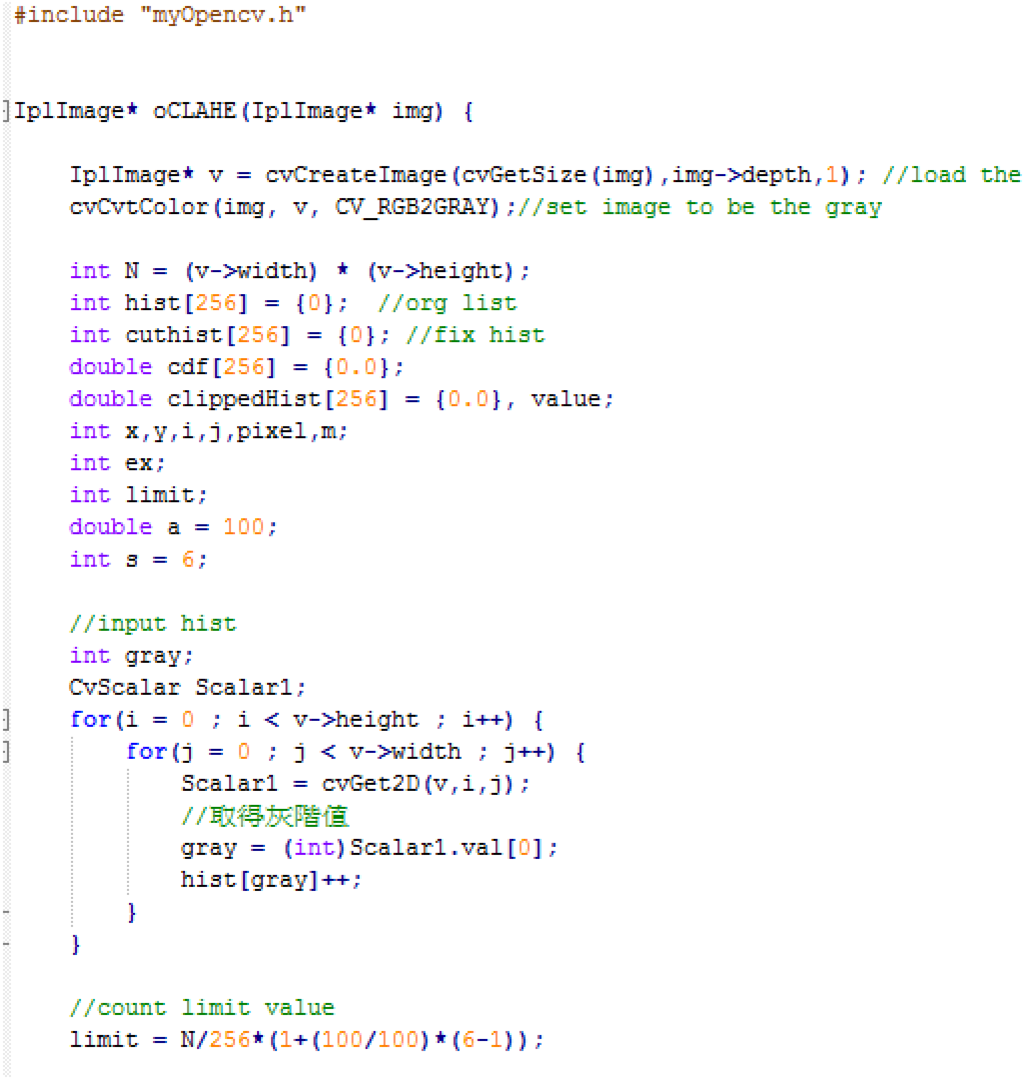
5. 若截斷像素數目尚未分配完，依序將尚未達到受限值的直方圖加一，直到截斷像素分配完

- **Step 6:** 均衡化值方圖

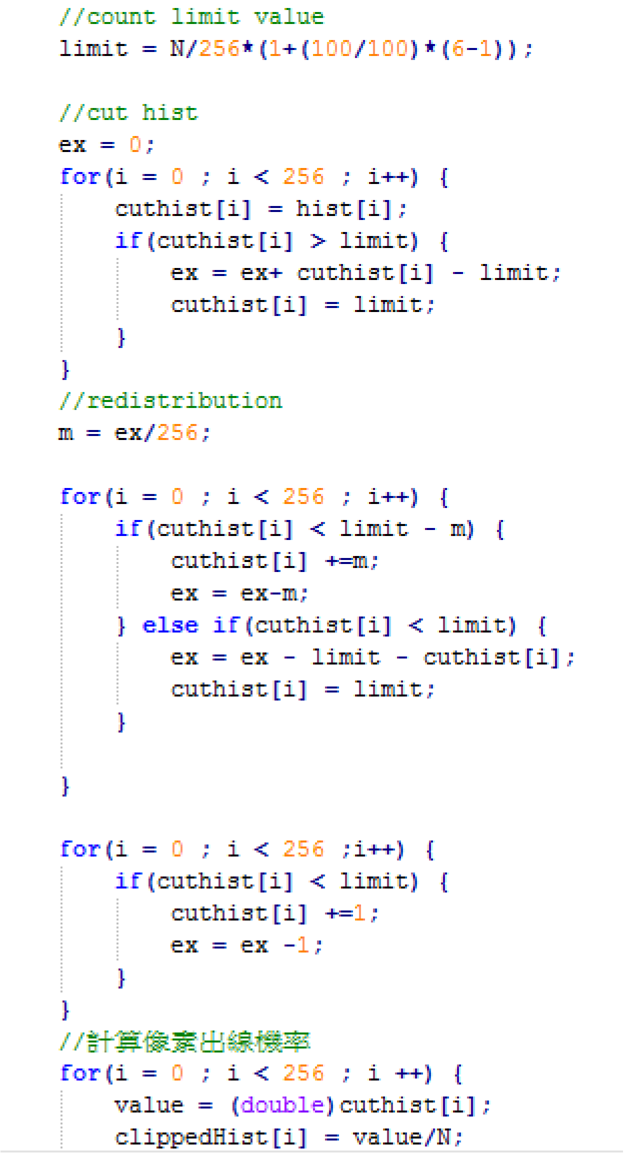
1. 走遍直方圖將像素數目除以總像數數目，計算直方圖機率
2. 將機率累加，計算直方圖累進機率
3. 根據均衡化公式，計算出新像素值

- **Step 7:** 設定視窗與運用

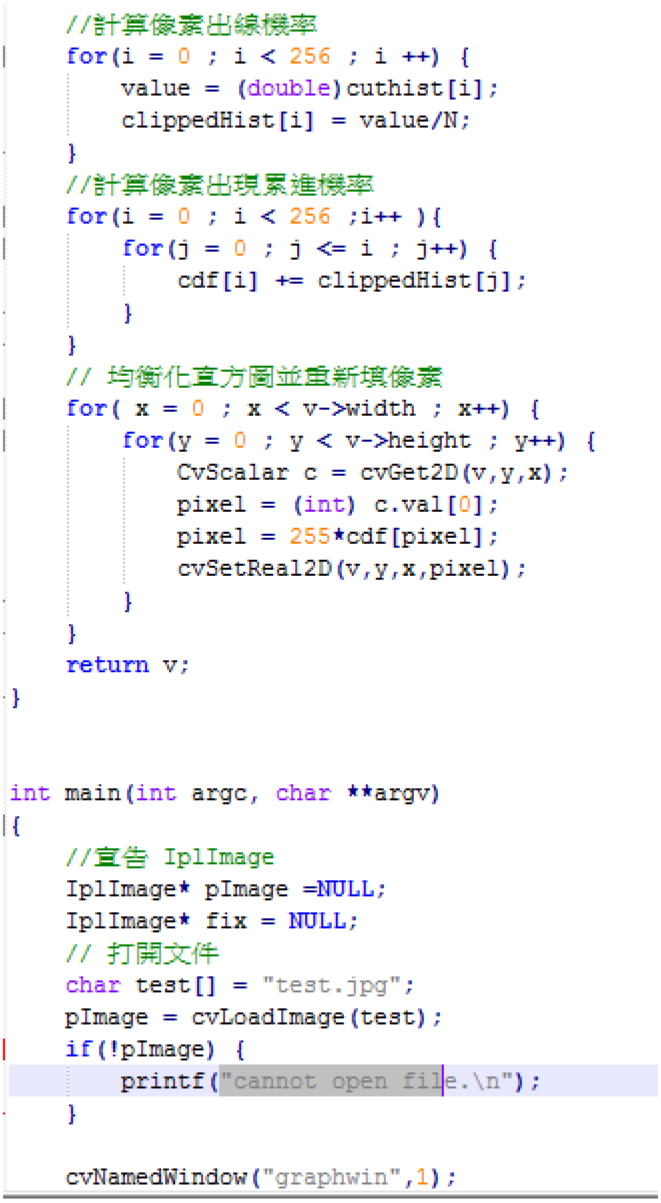
1. 運用上述方法來完成程式運行
2. 將一張彩色圖片轉成灰階



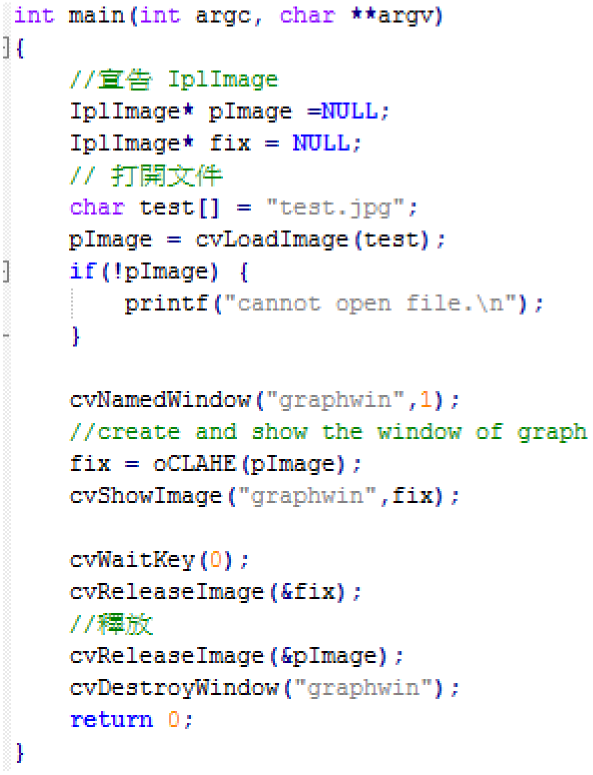
```
#include "myOpencv.h"


IplImage* oCLAHE(IplImage* img) {

    IplImage* v = cvCreateImage(cvGetSize(img),img->depth,1); //load the
    cvCvtColor(img, v, CV_RGB2GRAY);//set image to be the gray

    int N = (v->width) * (v->height);
    int hist[256] = {0};  //org list
    int cuthist[256] = {0}; //fix hist
    double cdf[256] = {0.0};
    double clippedHist[256] = {0.0}, value;
    int x,y,i,j,pixel,m;
    int ex;
    int limit;
    double a = 100;
    int s = 6;

    //input hist
    int gray;
    CvScalar Scalar1;
    for(i = 0 ; i < v->height ; i++) {
        for(j = 0 ; j < v->width ; j++) {
            Scalar1 = cvGet2D(v,i,j);
            //取得灰階值
            gray = (int)Scalar1.val[0];
            hist[gray]++;
        }
    }

    //count limit value
    limit = N/256*(1+(100/100)*(6-1));
```



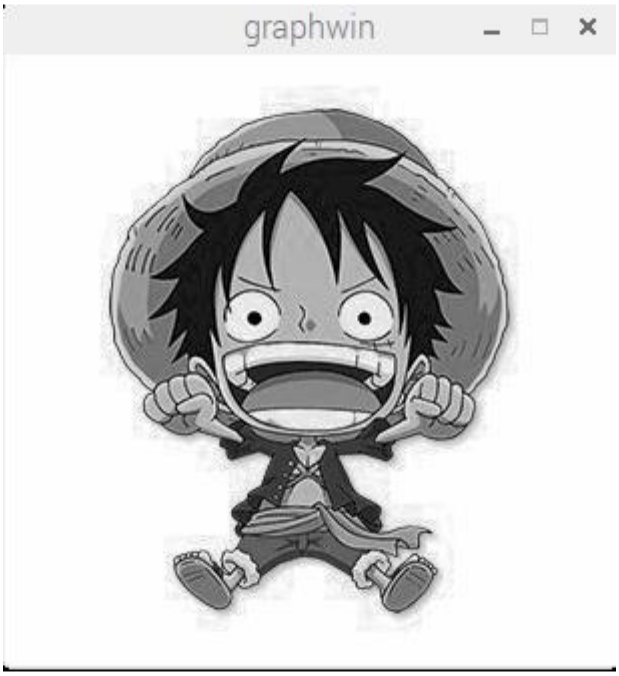
```
//count limit value
limit = N/256*(1+(100/100)*(6-1));

//cut hist
ex = 0;
for(i = 0 ; i < 256 ; i++) {
    cuthist[i] = hist[i];
    if(cuthist[i] > limit) {
        ex = ex+ cuthist[i] - limit;
        cuthist[i] = limit;
    }
}
//redistribution
m = ex/256;

for(i = 0 ; i < 256 ; i++) {
    if(cuthist[i] < limit - m) {
        cuthist[i] +=m;
        ex = ex-m;
    } else if(cuthist[i] < limit) {
        ex = ex - limit - cuthist[i];
        cuthist[i] = limit;
    }

}

for(i = 0 ; i < 256 ;i++) {
    if(cuthist[i] < limit) {
        cuthist[i] +=1;
        ex = ex -1;
    }
}
//計算像素出線機率
for(i = 0 ; i < 256 ; i ++) {
    value = (double)cuthist[i];
    clippedHist[i] = value/N;
```



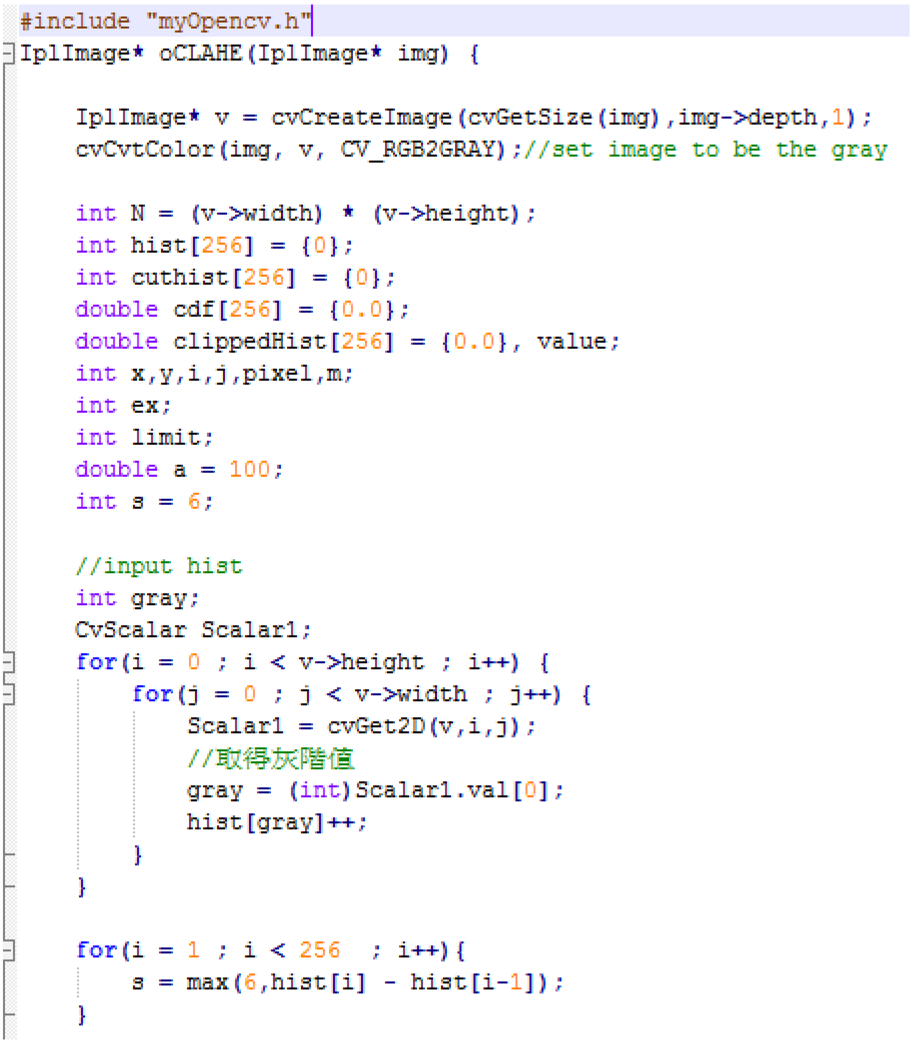
```
    //計算像素出線機率
    for(i = 0 ; i < 256 ; i ++) {
        value = (double)cuthist[i];
        clippedHist[i] = value/N;
    }
    //計算像素出現累進機率
    for(i = 0 ; i < 256 ;i++ ){
        for(j = 0 ; j <= i ; j++) {
            cdf[i] += clippedHist[j];
        }
    }
    // 均衡化直方圖並重新填像素
    for( x = 0 ; x < v->width ; x++) {
        for(y = 0 ; y < v->height ; y++) {
            CvScalar c = cvGet2D(v,y,x);
            pixel = (int) c.val[0];
            pixel = 255*cdf[pixel];
            cvSetReal2D(v,y,x,pixel);
        }
    }
    return v;
}


int main(int argc, char **argv)
{
    //宣告 IplImage
    IplImage* pImage =NULL;
    IplImage* fix = NULL;
    // 打開文件
    char test[] = "test.jpg";
    pImage = cvLoadImage(test);
    if(!pImage) {
        printf("cannot open file.\n");
    }

    cvNamedWindow("graphwin",1);
```



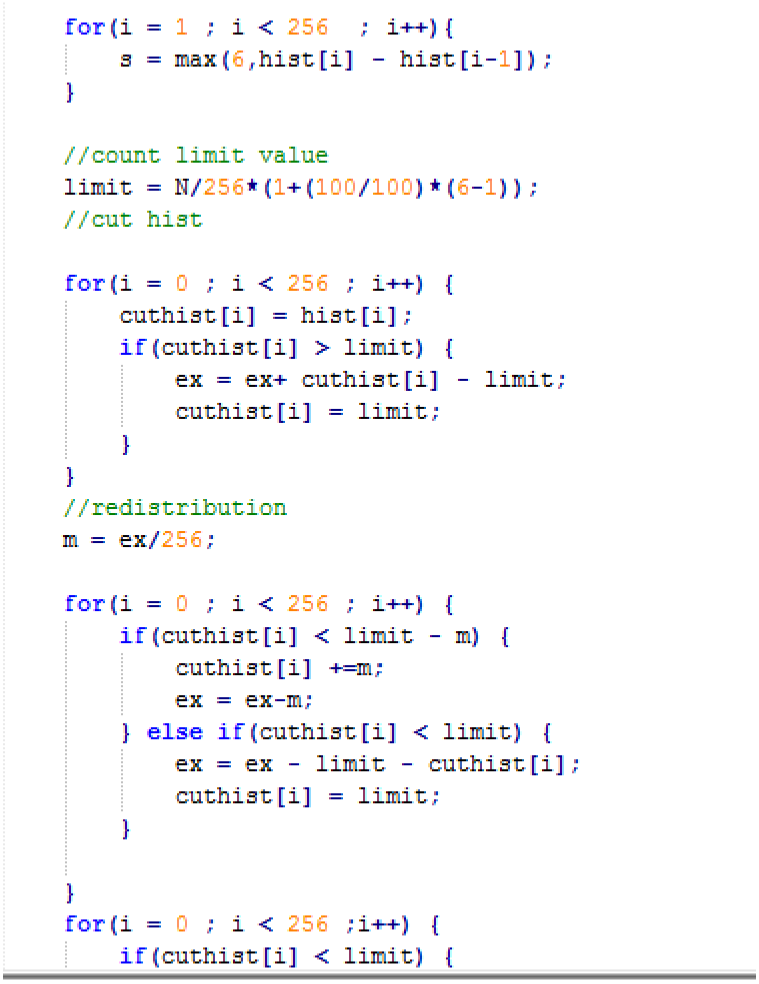
```
int main(int argc, char **argv)
{
    //宣告 IplImage
    IplImage* pImage =NULL;
    IplImage* fix = NULL;
    // 打開文件
    char test[] = "test.jpg";
    pImage = cvLoadImage(test);
    if(!pImage) {
        printf("cannot open file.\n");
    }

    cvNamedWindow("graphwin",1);
    //create and show the window of graph
    fix = oCLAHE(pImage);
    cvShowImage("graphwin",fix);

    cvWaitKey(0);
    cvReleaseImage(&fix);
    //釋放
    cvReleaseImage(&pImage);
    cvDestroyWindow("graphwin");
    return 0;
}
```



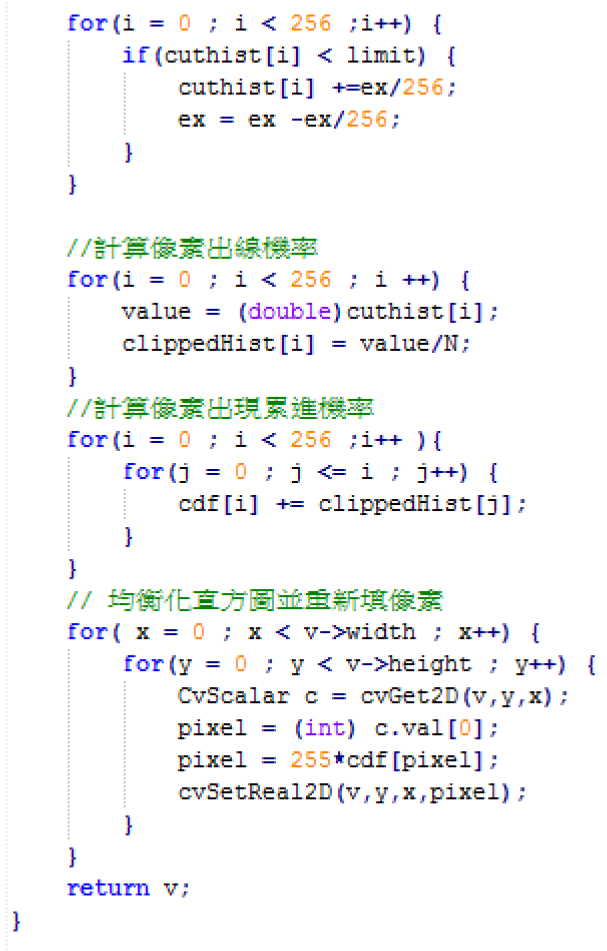
3. 將彩色影片轉灰階



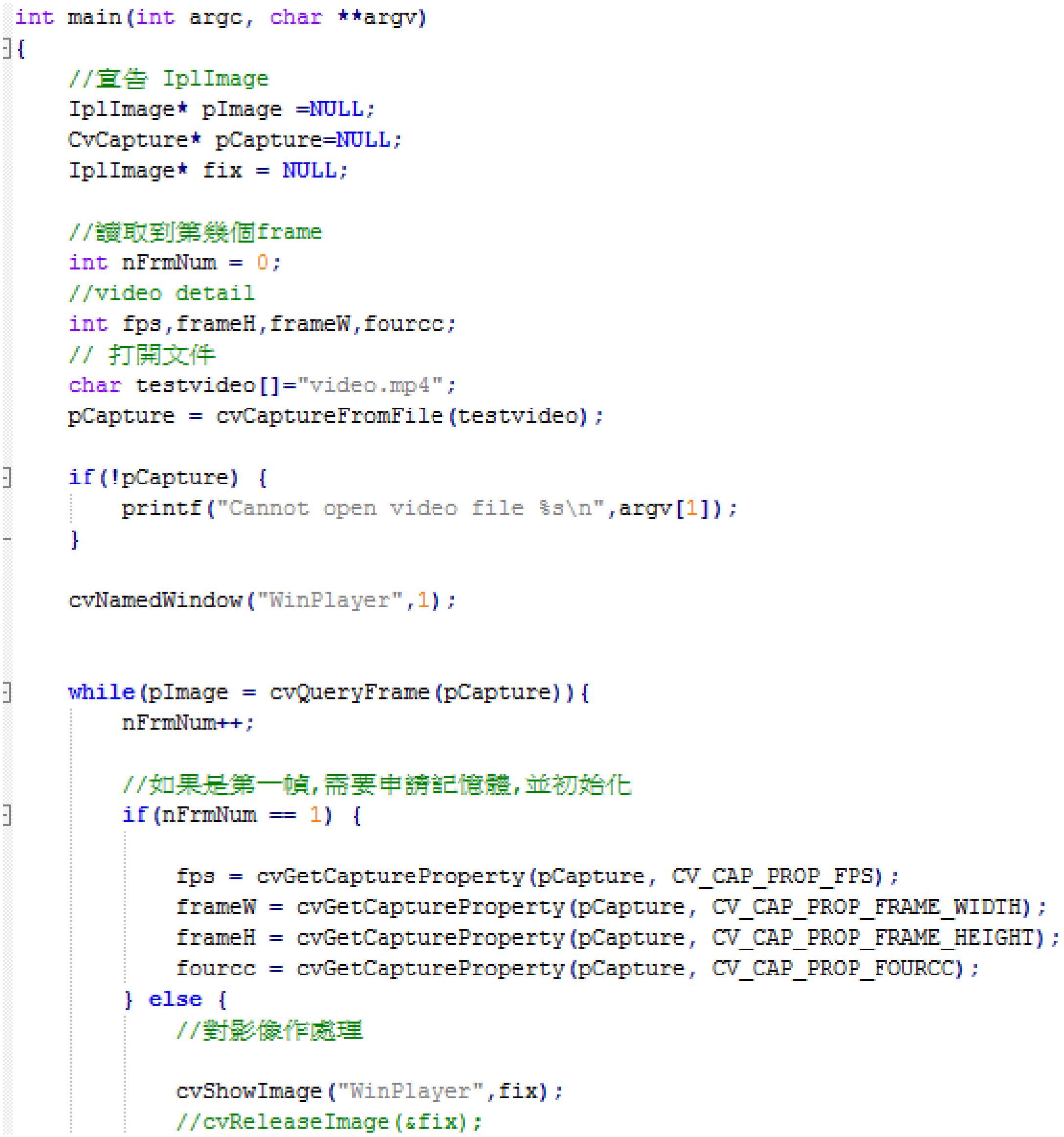
```
#include "myOpencv.h"
IplImage* oCLAHE(IplImage* img) {

    IplImage* v = cvCreateImage(cvGetSize(img),img->depth,1);
    cvCvtColor(img, v, CV_RGB2GRAY);//set image to be the gray

    int N = (v->width) * (v->height);
    int hist[256] = {0};
    int cuthist[256] = {0};
    double cdf[256] = {0.0};
    double clippedHist[256] = {0.0}, value;
    int x,y,i,j,pixel,m;
    int ex;
    int limit;
    double a = 100;
    int s = 6;

    //input hist
    int gray;
    CvScalar Scalar1;
    for(i = 0 ; i < v->height ; i++) {
        for(j = 0 ; j < v->width ; j++) {
            Scalar1 = cvGet2D(v,i,j);
            //取得灰階值
            gray = (int)Scalar1.val[0];
            hist[gray]++;
        }
    }

    for(i = 1 ; i < 256  ; i++){
        s = max(6,hist[i] - hist[i-1]);
    }
```



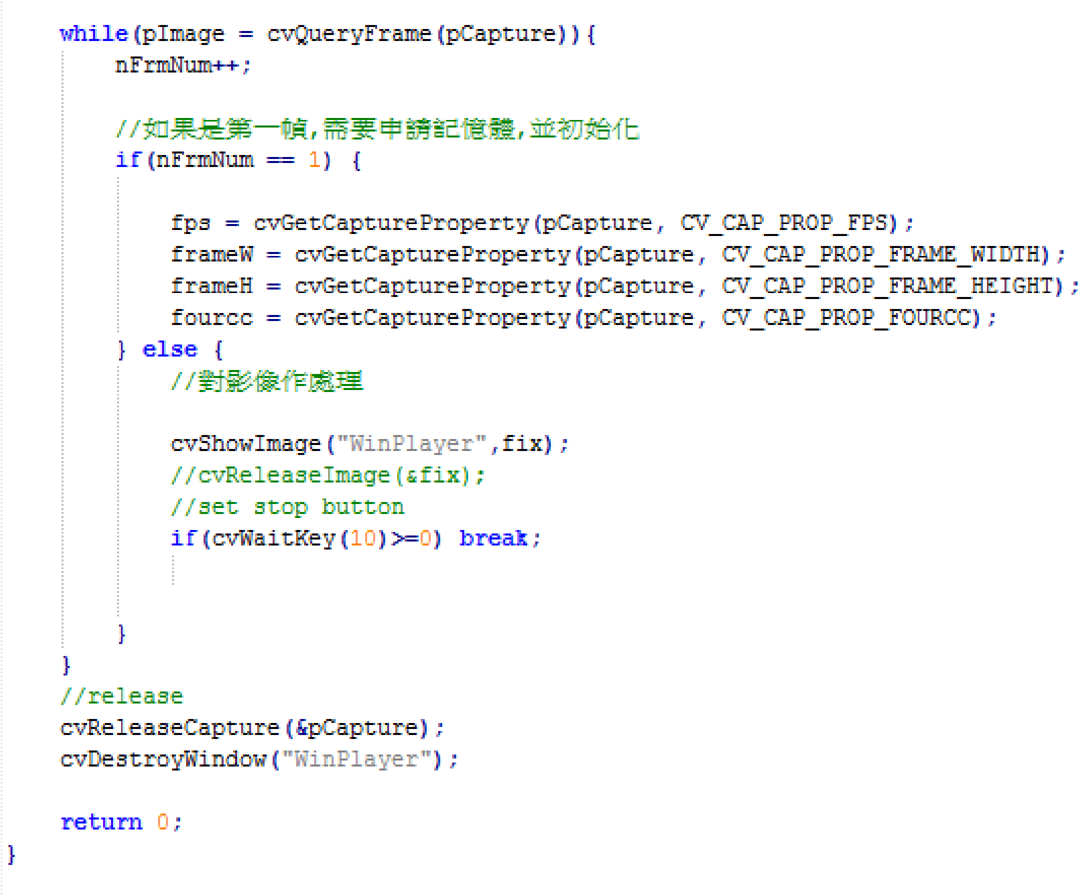
```
for(i = 1 ; i < 256  ; i++){
    s = max(6,hist[i] - hist[i-1]);
}

//count limit value
limit = N/256*(1+(100/100)*(6-1));
//cut hist

for(i = 0 ; i < 256 ; i++) {
    cuthist[i] = hist[i];
    if(cuthist[i] > limit) {
        ex = ex+ cuthist[i] - limit;
        cuthist[i] = limit;
    }
}
//redistribution
m = ex/256;

for(i = 0 ; i < 256 ; i++) {
    if(cuthist[i] < limit - m) {
        cuthist[i] +=m;
        ex = ex-m;
    } else if(cuthist[i] < limit) {
        ex = ex - limit - cuthist[i];
        cuthist[i] = limit;
    }

}
for(i = 0 ; i < 256 ;i++) {
    if(cuthist[i] < limit) {
```



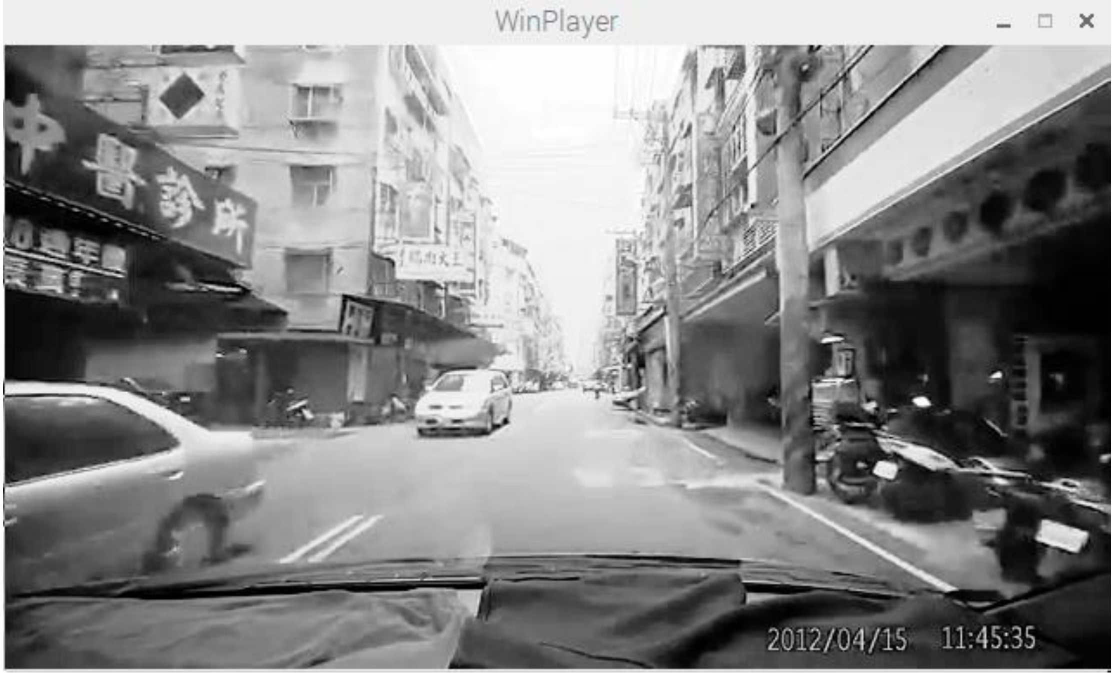
```
    for(i = 0 ; i < 256 ;i++) {
        if(cuthist[i] < limit) {
            cuthist[i] +=ex/256;
            ex = ex -ex/256;
        }
    }

    //計算像素出線機率
    for(i = 0 ; i < 256 ; i ++) {
        value = (double)cuthist[i];
        clippedHist[i] = value/N;
    }
    //計算像素出現累進機率
    for(i = 0 ; i < 256 ;i++ ){
        for(j = 0 ; j <= i ; j++) {
            cdf[i] += clippedHist[j];
        }
    }
    // 均衡化直方圖並重新填像素
    for( x = 0 ; x < v->width ; x++) {
        for(y = 0 ; y < v->height ; y++) {
            CvScalar c = cvGet2D(v,y,x);
            pixel = (int) c.val[0];
            pixel = 255*cdf[pixel];
            cvSetReal2D(v,y,x,pixel);
        }
    }
    return v;
}
```



```
int main(int argc, char **argv)
{
    //宣告 IplImage
    IplImage* pImage =NULL;
    CvCapture* pCapture=NULL;
    IplImage* fix = NULL;

    //讀取到第幾個frame
    int nFrmNum = 0;
    //video detail
    int fps,frameH,frameW,fourcc;
    // 打開文件
    char testvideo[]="video.mp4";
    pCapture = cvCaptureFromFile(testvideo);

    if(!pCapture) {
        printf("Cannot open video file %s\n",argv[1]);
    }

    cvNamedWindow("WinPlayer",1);


    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {

            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
        } else {
            //對影像作處理

            cvShowImage("WinPlayer",fix);
            //cvReleaseImage(&fix);
```



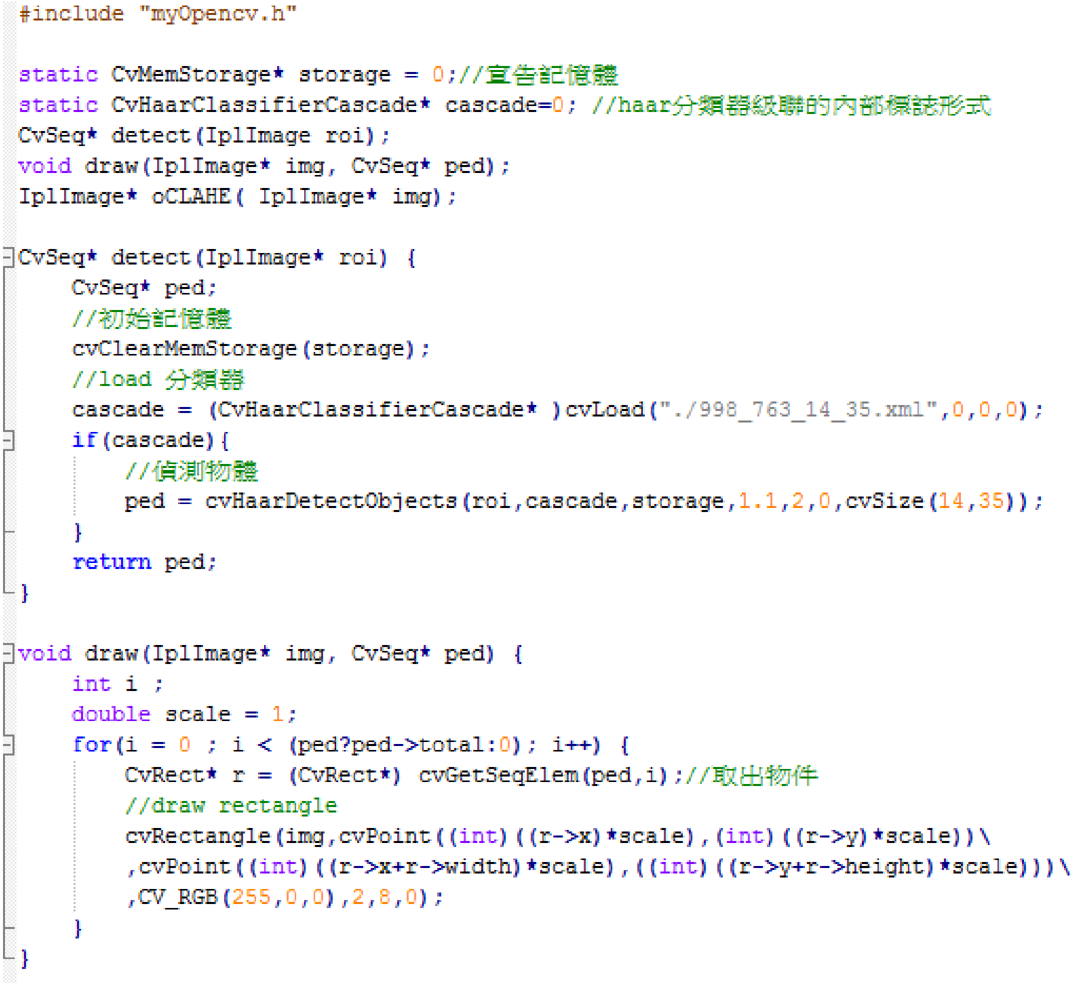
```
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {

            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
        } else {
            //對影像作處理

            cvShowImage("WinPlayer",fix);
            //cvReleaseImage(&fix);
            //set stop button
            if(cvWaitKey(10)>=0) break;


        }
    }
    //release
    cvReleaseCapture(&pCapture);
    cvDestroyWindow("WinPlayer");

    return 0;
}
```



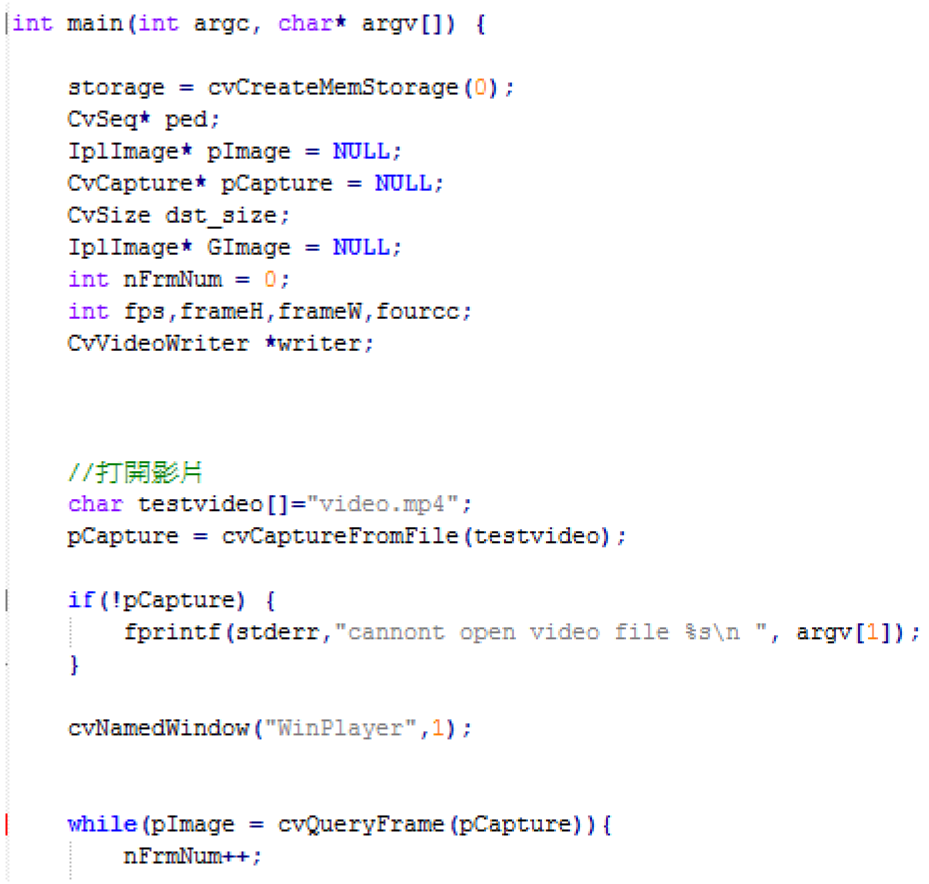
行人偵測系統

- **Step 1:** 影片人形

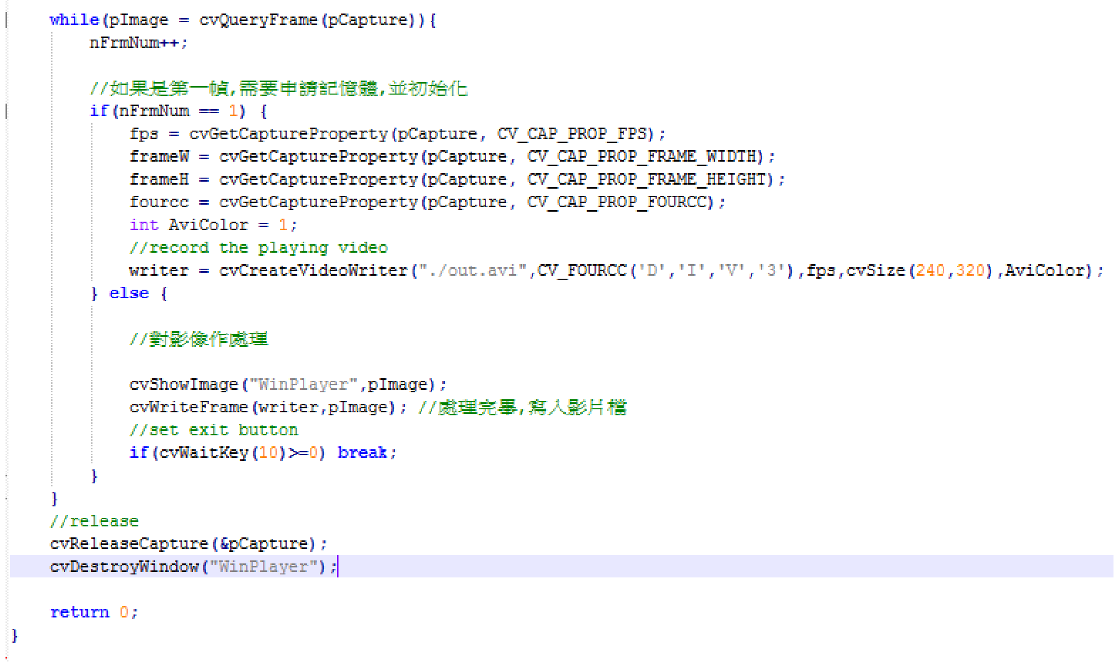
1. 使用取樣文件:998_763_14_35.xml
2. 使用draw()來框出分析的人形
3. 使用取樣文件來辨識人形並放置到彩色影片
4. 最後利用cvCreateVideoWriter記錄下分析的影片



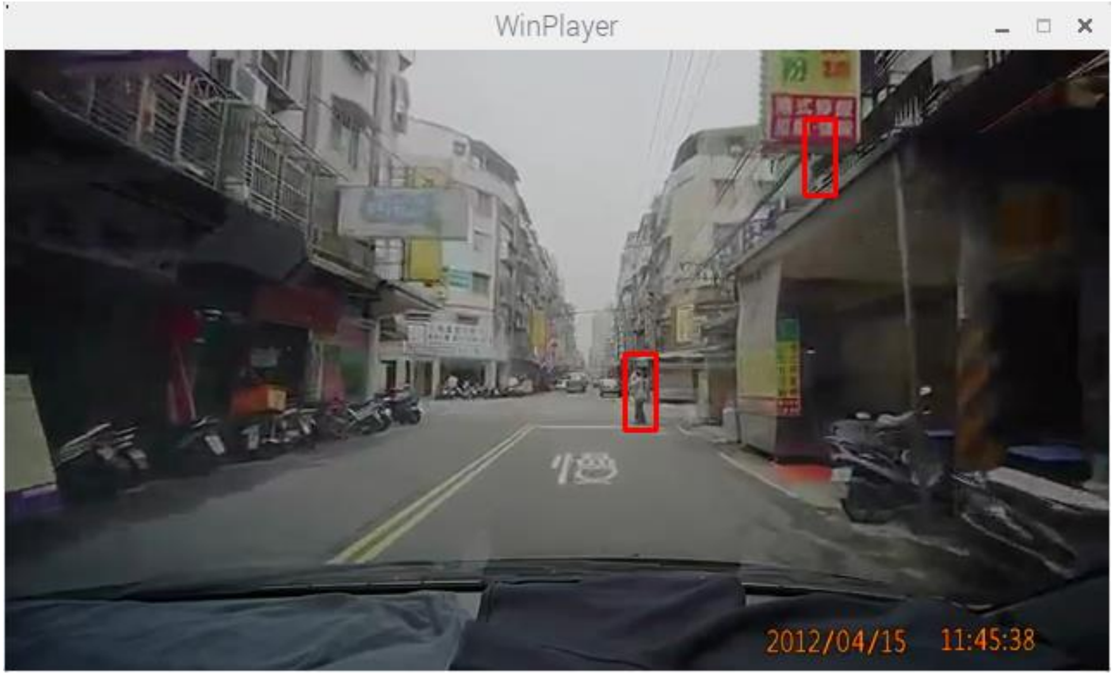
```
#include "myOpencv.h"

static CvMemStorage* storage = 0;//宣告記憶體
static CvHaarClassifierCascade* cascade=0; //haar分類器級聯的內部標誌形式
CvSeq* detect(IplImage roi);
void draw(IplImage* img, CvSeq* ped);
IplImage* oCLAHE( IplImage* img);

CvSeq* detect(IplImage* roi) {
    CvSeq* ped;
    //初始記憶體
    cvClearMemStorage(storage);
    //load 分類器
    cascade = (CvHaarClassifierCascade* )cvLoad("./998_763_14_35.xml",0,0,0);
    if(cascade){
        //偵測物體
        ped = cvHaarDetectObjects(roi,cascade,storage,1.1,2,0,cvSize(14,35));
    }
    return ped;
}

void draw(IplImage* img, CvSeq* ped) {
    int i ;
    double scale = 1;
    for(i = 0 ; i < (ped?ped->total:0); i++) {
        CvRect* r = (CvRect*) cvGetSeqElem(ped,i);//取出物件
        //draw rectangle
        cvRectangle(img,cvPoint((int)((r->x)*scale),(int)((r->y)*scale))\
        ,cvPoint((int)((r->x+r->width)*scale),((int)((r->y+r->height)*scale)))\
        ,CV_RGB(255,0,0),2,8,0);
    }
}
```



```
int main(int argc, char* argv[]) {

    storage = cvCreateMemStorage(0);
    CvSeq* ped;
    IplImage* pImage = NULL;
    CvCapture* pCapture = NULL;
    CvSize dst_size;
    IplImage* GImage = NULL;
    int nFrmNum = 0;
    int fps,frameH,frameW,fourcc;
    CvVideoWriter *writer;



    //打開影片
    char testvideo[]="video.mp4";
    pCapture = cvCaptureFromFile(testvideo);

    if(!pCapture) {
        fprintf(stderr,"cannont open video file %s\n ", argv[1]);
    }

    cvNamedWindow("WinPlayer",1);


    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;
```

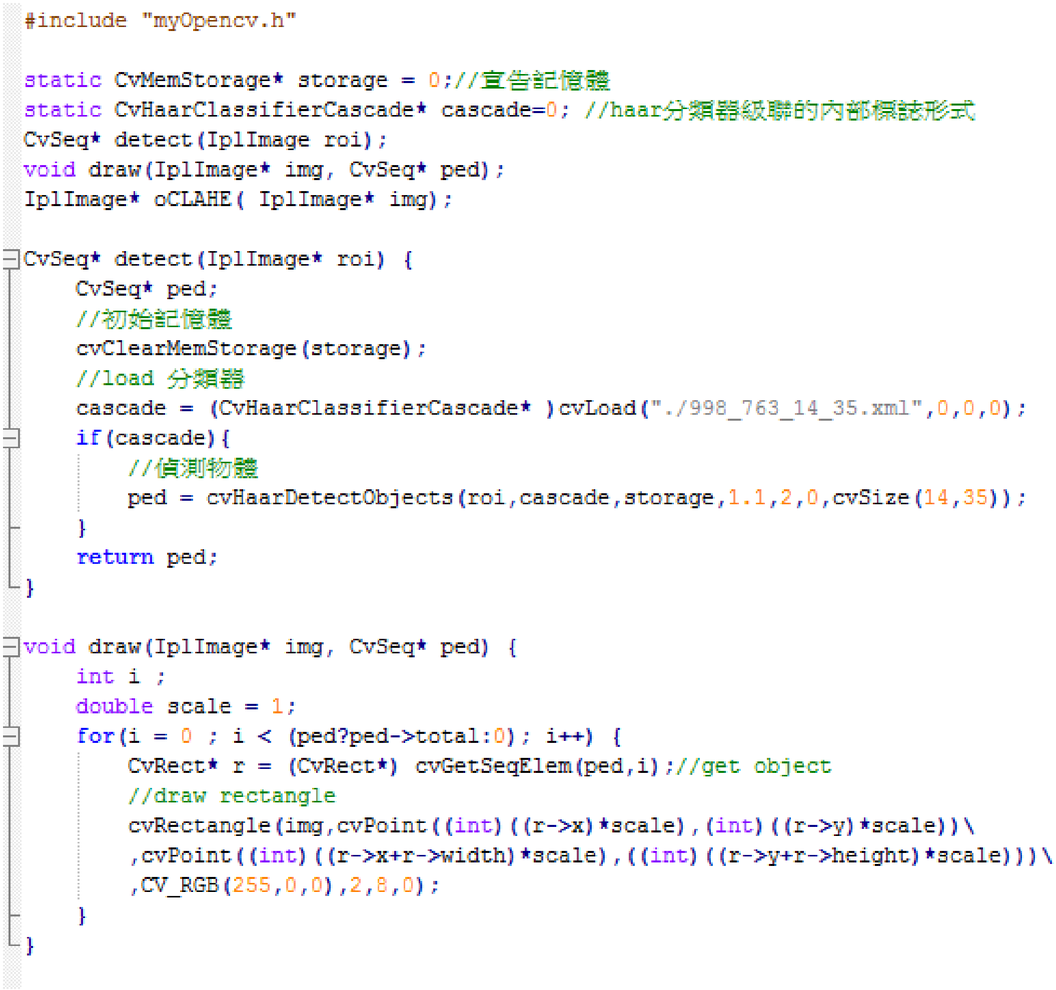


```
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
            int AviColor = 1;
            //record the playing video
            writer = cvCreateVideoWriter("./out.avi",CV_FOURCC('D','I','V','3'),fps,cvSize(240,320),AviColor);
        } else {

            //對影像作處理

            cvShowImage("WinPlayer",pImage);
            cvWriteFrame(writer,pImage); //處理完畢,寫入影片檔
            //set exit button
            if(cvWaitKey(10)>=0) break;
        }
    }
    //release
    cvReleaseCapture(&pCapture);
    cvDestroyWindow("WinPlayer");

    return 0;
}
```

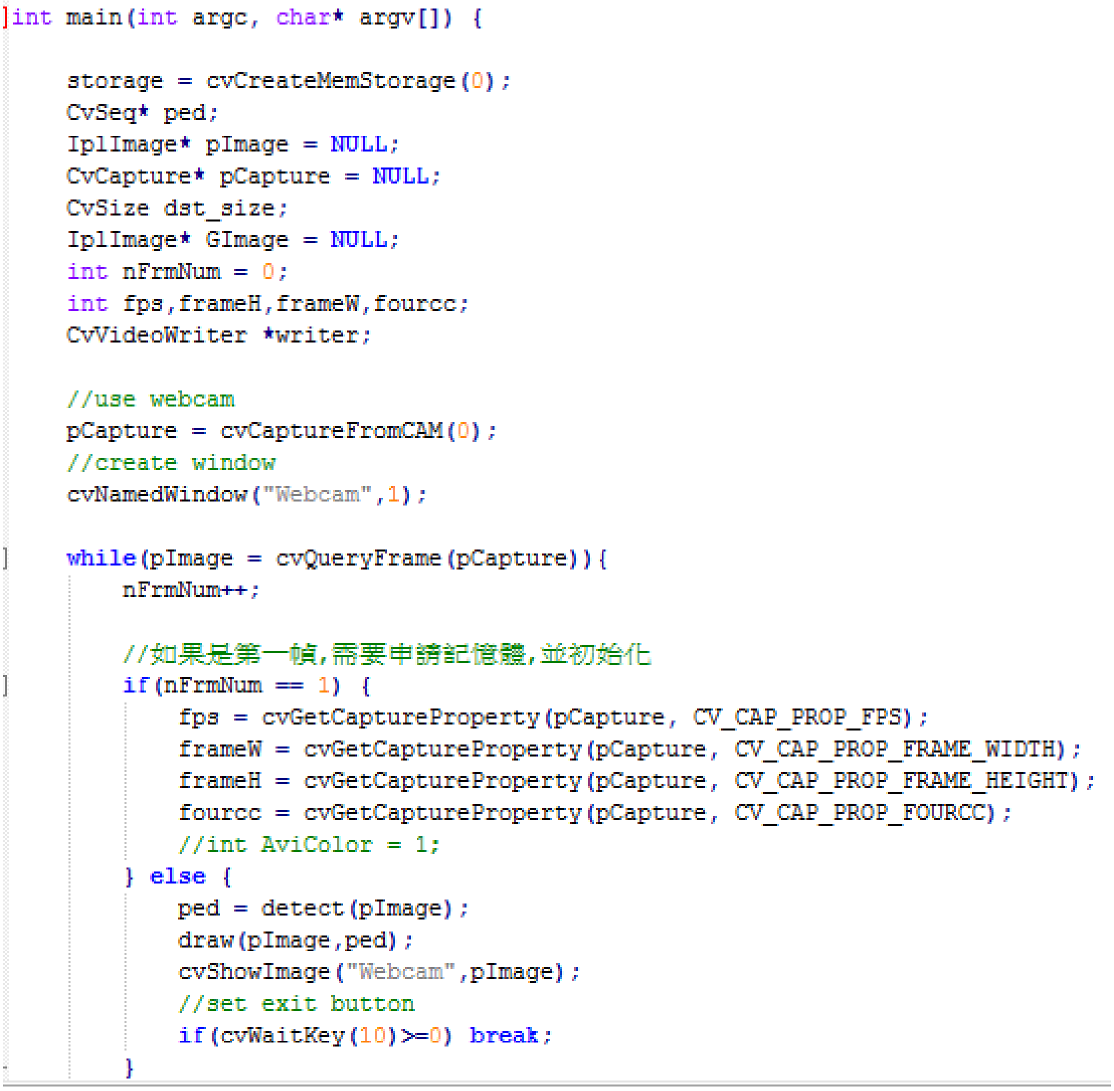


- **Step 2:** 使用 **webcam** 抓取人形

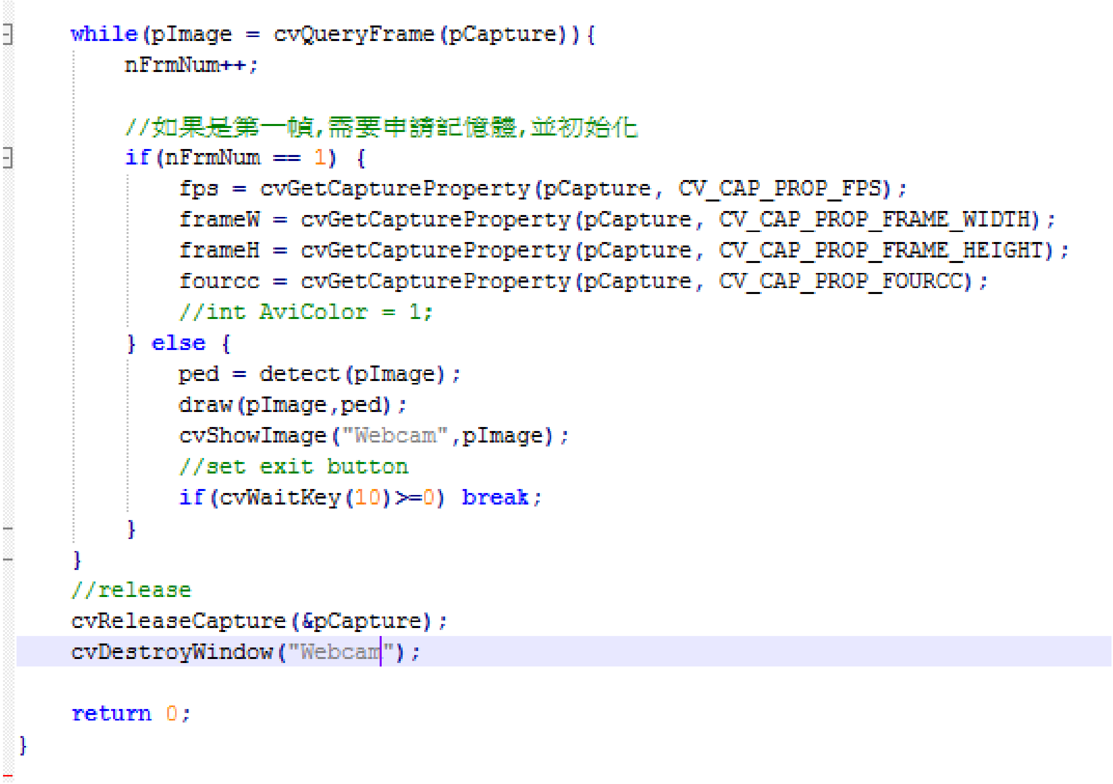
1. 使用webcam來擷取成影像
2. 使用取樣文件:998_763_14_35.xml
3. 使用draw()來框出分析的人形



```c
#include "myOpencv.h"

static CvMemStorage* storage = 0;//宣告記憶體
static CvHaarClassifierCascade* cascade=0; //haar分類器級聯的內部標誌形式
CvSeq* detect(IplImage roi);
void draw(IplImage* img, CvSeq* ped);
IplImage* oCLAHE( IplImage* img);

CvSeq* detect(IplImage* roi) {
    CvSeq* ped;
    //初始記憶體
    cvClearMemStorage(storage);
    //load 分類器
    cascade = (CvHaarClassifierCascade* )cvLoad("./998_763_14_35.xml",0,0,0);
    if(cascade){
        //偵測物體
        ped = cvHaarDetectObjects(roi,cascade,storage,1.1,2,0,cvSize(14,35));
    }
    return ped;
}

void draw(IplImage* img, CvSeq* ped) {
    int i ;
    double scale = 1;
    for(i = 0 ; i < (ped?ped->total:0); i++) {
        CvRect* r = (CvRect*) cvGetSeqElem(ped,i);//get object
        //draw rectangle
        cvRectangle(img,cvPoint((int)((r->x)*scale),(int)((r->y)*scale))\
        ,cvPoint(((int)((r->x+r->width)*scale),((int)((r->y+r->height)*scale)))\
        ,CV_RGB(255,0,0),2,8,0);
    }
}
```



```
int main(int argc, char* argv[]) {

    storage = cvCreateMemStorage(0);
    CvSeq* ped;
    IplImage* pImage = NULL;
    CvCapture* pCapture = NULL;
    CvSize dst_size;
    IplImage* GImage = NULL;
    int nFrmNum = 0;
    int fps,frameH,frameW,fourcc;
    CvVideoWriter *writer;

    //use webcam
    pCapture = cvCaptureFromCAM(0);
    //create window
    cvNamedWindow("Webcam",1);

    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
            //int AviColor = 1;
        } else {
            ped = detect(pImage);
            draw(pImage,ped);
            cvShowImage("Webcam",pImage);
            //set exit button
            if(cvWaitKey(10)>=0) break;
        }
```



```
    while(pImage = cvQueryFrame(pCapture)){
        nFrmNum++;

        //如果是第一幀,需要申請記憶體,並初始化
        if(nFrmNum == 1) {
            fps = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FPS);
            frameW = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_WIDTH);
            frameH = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FRAME_HEIGHT);
            fourcc = cvGetCaptureProperty(pCapture, CV_CAP_PROP_FOURCC);
            //int AviColor = 1;
        } else {
            ped = detect(pImage);
            draw(pImage,ped);
            cvShowImage("Webcam",pImage);
            //set exit button
            if(cvWaitKey(10)>=0) break;
        }
    }
    //release
    cvReleaseCapture(&pCapture);
    cvDestroyWindow("Webcam");

    return 0;
}
```