# 多執行緒處理模式應用於影像監控系統效益之研究

# Study of Multi-Thread Mode of Image Surveillance System Efficiency

吳明川
國立台北科技大學
機電整合研究所
mcwu@ntut.edu.tw

李益榮
國立台北科技大學
機電整合研究所
t7408014@ntut.edu.tw

**摘要**

目前多數的數位影像監視系統，其程式不具有同步性，故在多事件偵測或偵測行程(Process)的權重較重時經常導致其它行程執行延遲。再者多事件偵測的影像處理是非常耗時，故如何充份發揮 CPU 效能，進而增加程式處理速度並降低處理時間是個新課題。因此，讓影像式監控系統的偵測事件具有同步性及發揮 CPU 使用效能，是本研究主要目的。

本文，以實驗室為例利用 PTZ IP 攝影機進行影像式安全偵測，其偵測模式分為警戒模式及監控模式，將偵測事件模組化並且利用多執行緒處理模式(Multi-Thread mode)建立程式架構。最後，利用多執行緒處理模式及非多執行緒處理模式進行性能比較，得到實驗結果確實為多執行緒處理模式是具有同步性及提升 CPU 使用效能。

**關鍵詞**：影像監控系統，程式效能，多執行緒模式，PTZ IP 攝影機

**Abstract**

In the present day, most of the digital motoring systems are not synchronous with software program. When during multiple events detection, the software always cause delay or postponement even overload. On the other hand, events multiple detection of software program always takes times, so how to improve computer's CPU performance also increase software program speed in order to save times will be a new subject. So, how to let the multiple event detection be synchronous with software program and improve computer's CPU effective utilization in image surveillance system will be this research main purpose. As laboratory as a sample, we use PTZ IP camera to do image security detect, detect mode will divide into security mode and monitor mode. We take result images to build a multi-thread program system.

At the end, we use multi-thread mode and non multi-thread mode to make an efficient compare, the multi-thread program system are better performance also can synchronous with software program and increase computer's CPU performance.

**Keywords**: Image Surveillance System，Procedure Efficiency，Multi-Thread mode，PTZ IP Camera

## 1. 前言

### 1.1 研究動機與目的

隨著社會的進步，人們對於安全的要求日益增加，而現今的影像監控系統程式架構欲不具有同步性，故在多事件偵測或偵測行程的權重較重時經常導致其它行程執行延遲。另一方面，由於偵測的項目多且複雜，故所考量的是程式是否具有可延展性。再者現今 CPU 發展到雙核心或者更多核心數，以往的影像監控系統是以單核心程式架構利用 CPU 資源，其架構應用在現今的多核心 CPU，是不能充份發揮多核心效能，故更有效利用 CPU 資源可增加程式處理速度並降低處理時間是個新課題。因此，讓影像式監控系統的偵測事件具有同步性、可延展性及提升 CPU 使用效率為本研究主要目的。

### 1.2 文獻探討

M. Valera[1]說明整個監控系統發展過程，其中點出第三代網路監控系統的研究重點之一為多攝影機的技術，其延伸的問題是多事件偵測架構。針對多事件偵測 Yi-Tsung Chien[2]提出一個具有可延展性與組織性架構，其將偵測事件模組化使獨立多個事件監控可以合作增加安全性與可靠性，在 Levente Kovacs[3]提

出可延伸性的模組化架構，其執行方式是當經常事件發生時，偵測物件的模組才執行，減少運算時的負擔並加快反應速度。

參考文獻[2][3]，都提出多事件偵測建構的方式，均有良好效果，但是美中不足的是其沒有探討同步性及提升 CPU 使用效率，如能讓程式具備上述兩項對於影像監控系統性能可達到加分效果。

## 1.3 相關背景知識介紹

### 1.3.1 傳統行程與多執行緒之間關係

行程與程式主要的不同有二點，第一點為程式是被放在外部的儲存裝置如磁碟上，而行程則被放在記憶體中，第二點為程式在儲存裝置中是靜態的，而行程在記憶體中是動態的，它會隨著一些事件的發生而產生相對的改變。所以簡單來說行程其實是在執行中的程式。

以一個單 CPU 的系統來說，隨時只能有一個行程在執行。其他行程則必須等待 CPU 空閒下來，然後再經由排程器選出，才能取得 CPU 的使用權。當 CPU 的使用權由一個行程轉到另一個行程時需進行內文切換。內文切換動作所花的時間對系統而言是額外的負擔。執行緒降低內文切換所花的時間。

傳統行程與多執行緒比較方面，傳統行程屬於重量級行程，可看成一個執行緒在執行行程，而執行緒是屬於輕量級行程，是使用 CPU 資源的基本單位，其組成包括程式計數器、一組暫存器和一個堆疊空間，與其他的執行緒共用同一個位址空間。

使用多執行緒取代傳統行程的優點為資源共享容易、節省記憶空間、快速內文切換及平行處理。本研究利用上述優點讓程式達到同步化及提升 CPU 使用效能。

### 1.3.2 執行緒

執行緒是行程中的子程式，這些子程式共享行程內的資源，同時也可以分配到CPU的使用權。而在多執行緒程式方面，其必須靠主執行緒去啟動其它執行緒的進行。

執行緒在程式的狀態包含起始狀態、可執行狀態、正執行狀態、等待狀態及死亡狀態，如圖1所示。執行緒在起始狀態被建立，進入可執行狀態等待排程，當排程排到時執行緒進入執行狀態，其依照向CPU取得CPU Time在時間內執行，當分配時間結束後，執行緒會交出CPU使用權，其狀態會變回可執行狀態或者等待狀態，而等待狀態當某事件發生時會再進入可執行狀態，當執行緒任務完成後，即進入死亡狀態，進入死亡狀態的執行緒不能被重新啟動。

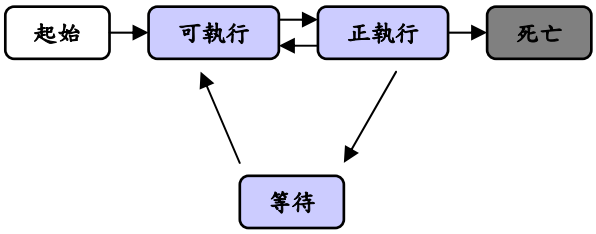



圖 1 執行緒生命週期示意圖

### 1.3.3 多執行緒處理模式

常常可以聽到很多作業系統都會聲稱自己擁有「多工」的處理模式，其實這些多工或是多處理器的運作基礎都是「多執行緒」。執行緒可以想像成程式的某個功能，其執行方面是先取得 CPU Time 的使用權如圖 2 所示，得到使用權後在時間內執行該執行緒程式內容，所得到時間結束後就會把使用權還給 CPU。

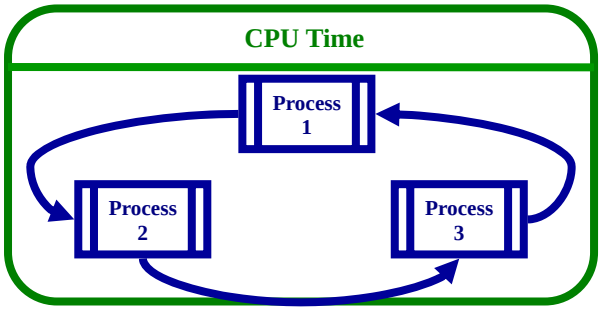



圖 2 三個行程輪流使用 CPU Time 示意圖

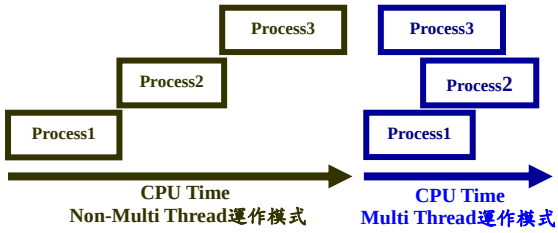



圖 3 多執行緒處理模式與非多執行緒處理模式比較圖

由圖 3 可知多執行緒處理模式程式處理時間比非多執行緒處理模式少。典型的例子為網頁瀏覽器，它大致可分為三個行程，分別為接收資料、顯示圖片、顯示文字，如果網路瀏覽器是非多執行緒處理模式，當開一個網頁可能會讓使用者等待時間過久，當然現在網頁瀏覽器是多執行緒處理模式，所以當點選 IE 時文字跟圖片是交錯的出現等待的時間會比前者少。

由上述的網頁瀏覽器例子可看出，程式執行具有同步性。另一方面，在圖 3 中的多執行緒模式行程重疊部份會使 CPU 使用率增加。本研究利用多執行緒建構程式架構，使程式執行達到同步並且讓 CPU 使用率提升。

## 2. 研究方法

### 2.1 程式開發工具與實驗設備

本研究所選擇使用的程式開發工具為 Borland C++ Builder 6.0，對於使用者介面與網路元件的支援程度高，可以節省許多開發時間。在實驗設備方面，本研究電腦規格使用 CPU 為 AMD Athlon64X2 5000+雙核心處理器，而記憶體為 DDR2 800 2GB。攝影機規格則使用 Sony SNC-RZ30N PTZ IP 攝影機，如圖 4 所示，其具有水平迴轉(Pan)、仰俯(Tilt)、光學倍率縮放(Zoom)等功能，可以根據需求對物件進行追蹤及放大檢視的效果。另外，IP 攝影機具有網路應用功能，如圖 5 所示可藉由網路通訊協定與硬體控制代碼，由遠端控制 IP 攝影機的動作與監視影像。

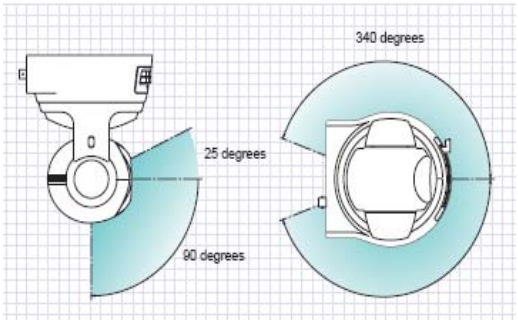



**圖 4　PTZ IP 攝影機水平與仰俯之迴轉角度示意圖**

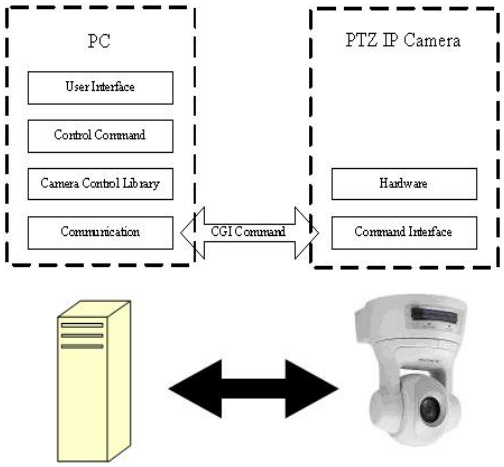



**圖 5　PC 程式與 PTZ IP 攝影機之控制運作示意圖**

### 2.2 研究架構

本研究建立自動化實驗室影像式安全偵測，其程式架構利用多執行緒處理模式撰寫，偵測模式分為警戒模式及監控模式，將所偵測到事件錄影並發出警告，如圖 6 中所示。

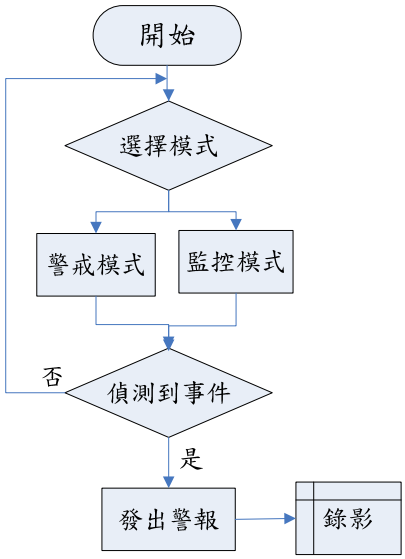



**圖 6 PTZ IP 攝影機偵測系統流程圖**

### 2.3 警戒模式

模式偵測功能有移動物體追蹤及可疑區域的標示。其偵測目的為在冗長監視畫面中可以快速得到有問題區域(ROI)並作回報，使監控人員可以第一時間作出事件處理。

在移動物體追蹤分成兩個主要部份，一個為前景偵測部份，另一個將所得到前景萃取需要特徵進行追蹤方法部份。

#### 2.3.1 前景偵測方法比較

前景偵測方法，常見的有高斯混合模型(Modified Mixture of Gaussians[4])前景偵測，其作法是將每一點像素都建立高斯分佈，能避免光線或者自然界微小變動而得到較好的前景，但其缺點是運算量過大，另一偵測法為固定背景相減法，雖然運算量較輕但是容易受到光線變動將背景像素誤判成前景。本研究採取時間平均法，其原理是以連續時間均值與連續時間中值法來建立參考背景，可達到微光源變化下的背景建立。

#### 2.3.2 追蹤方法比較

追蹤方法，常見的有平均位移演算法(Mean Shift Algorithm[5])，以及其延伸的連續可適性質量中心偏移演算法(Continuously Adaptive Mean Shift Algorithm; CamShift[6])，

這二種方法是利用顏色的資訊進行目標追蹤，在目標顏色單純及背景顏色單純下，有良好的追蹤效果，但是在實務中所要監控的環境及追蹤目標顏色複雜故強健性不足。本研究採用的是輪廓搜尋法(Find Counters)，作法將所得前景二值化並利用連通法標示出物件，其作法不會受到顏色的所干擾。

### 2.3.3 移動物體追蹤及可疑區域的標示模組

本功能流程如圖 7 所示，首先利用 PTZ 擷取影像，採用時間均值法建立背景模型，其建立的方法為先建立連續時間相素的平均值，如式(1)所示，再考量背景光源的微變量成份，由建立時間均值影像的 n 張影像，計算其與時間平均值影像的 n 張差值影像，如式(2)所示。同樣，計算其 n 張差值影像的平均值，如式(3)所示。最後，由時間均值影像加與減去其平均差值影像，即分別可得上邊界影像與下邊界影像，如式(4)、(5)所示。即可建立時間平均背景影像以及其平均光線微變動範圍的背景模型。

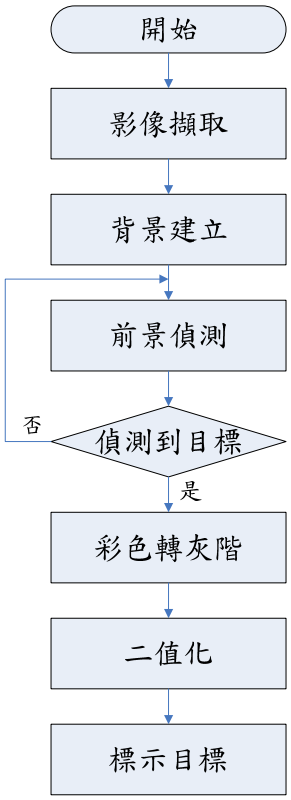



**圖 7 移動物體追蹤與可疑區域標示模組流程圖**

$$\overline{Bg}_{img} = \frac{\sum_{n=1}^{N} frame(n)}{N} \quad (1)$$

$$\overline{Bg}_{img}^{Dif}(n) = \left| frame(n) - \bar{Bg}_{img} \right|, n = 1 \sim N \quad (2)$$

$$\overline{Bg}_{img}^{Dif} = \frac{\sum_{n=1}^{N} Bg_{img}^{Dif}(n)}{N} \quad (3)$$

$$Bg_{img}^{Up} = \overline{Bg}_{img} + \overline{Bg}_{img}^{Dif} \quad (4)$$

$$Bg_{img}^{Down} = \overline{Bg}_{img} - \overline{Bg}_{img}^{Dif} \quad (5)$$

由記錄背景光影的變動範圍，求得影像變動的上下邊界的平均值，可依其雙曲面閥值，相對一般單平面閥值，更能偵測出完整的前景區域，如圖 8 所示。

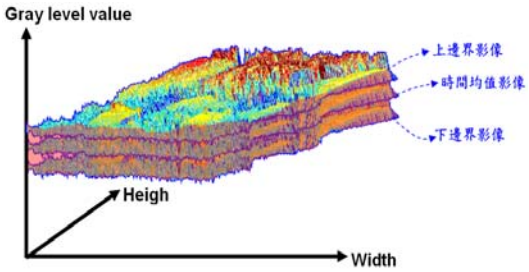


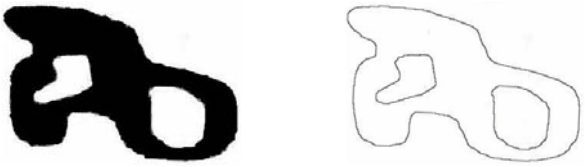

**圖 8 時間均值法所建立背景模型示意圖**

$$\left| frame(n) - Bg_{img} \right| = Fg_{img}, n = 1 \sim N \quad (6)$$

$$Fg_{img}(x,y) = \begin{cases} 255 if \ \geq Bg_{img}^{Up}(x,y) \\ 255 if \leq Bg_{img}^{Down}(x,y) \\ 0 \quad otherwise \end{cases}, x = 1 \sim m, y = 1 \sim n \quad (7)$$

前景偵測方式是使用背景相減法偵測前景影像，如式(6)所示。所得相減的影像先轉成灰階影像，利用式(7)判斷各點像素點是否超出其容許變動範圍內，超出變動範圍的像素強度設為 255 反之設為 0，進而得到二值化影像。

輪廓搜尋法演算原理是掏空內部點，如果原圖中有一點為黑，且它的 8 個相鄰點都是黑色時(此時該點是內部點)，則將該點刪除，如圖 9 所示。最後將得到二值化影像利用上述的輪廓搜尋法找出其前景輪廓並且標示。



(a)原影像　　(b)輪廓提取後影像

圖 9 輪廓搜尋法示意圖

### 2.4 監控模式

模式功能有禁止區移動偵測及實驗室工作區膚色偵測。禁止區移動偵測目的為實驗室裡常存放具有危險性藥品或者高速運轉機具，針對需警戒區域進行管制。實驗室工作區膚色偵測目的為實驗室常發生未依規定穿著實驗衣及手套，導致實驗者手部或者身體燒灼傷，故本功能針對實驗室工作區上設置偵測區，在偵測區進行膚色的偵測。

#### 2.4.1 禁止區移動偵測模組

本功能流程如圖 10 所示，首先利用 PTZ 擷取影像，在程式介面上選取偵測的區域，區域影像運用式(8)作移動偵測，如圖 11 所示，所得前景利用式(9)轉換成二值影像，最後將二值影像內的白點標示。

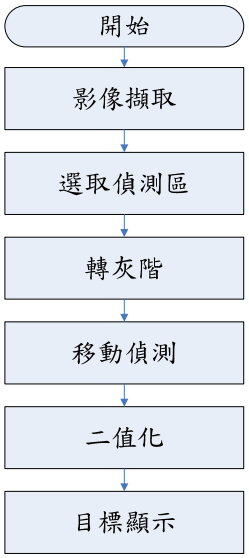



圖 10 禁止區移動偵測模組流程圖

$$\left| frame(n) - frame(n-1) \right| = Fg_{img}, n = 1 \sim N \tag{8}$$

$$f(x,y) = \begin{cases} 255 & if \quad f(x,y) \le \phi \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

#### 2.4.2 實驗室工作區膚色偵測模組

膚色偵測在 P. S. Hiremath[7]提供在色彩空間 YCrCb 膚色範圍值，如圖 11 所示，所得到膚色範圍完整，但是其缺點為背景會有少數相近的色彩落入此範圍內，而產生干擾像素點。本研究改良是先將移動成份取出，再去偵測其膚色，消除背景所產生干擾。

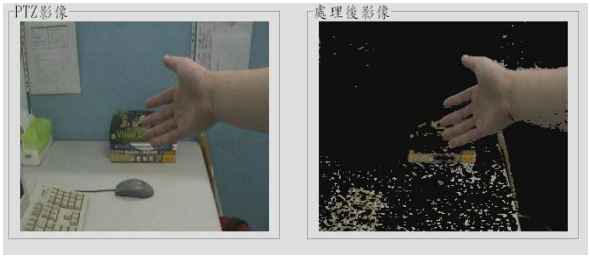



圖 11 利用[7]提供 YCrCb 範圍值所得到膚色影像

本功能流程如圖 12 所示，首先利用 PTZ 擷取影像，在程式介面上選取偵測的區域，利用式(8)得到移動像素影像，將此影像利用式(10)作色彩空間轉換成 YCrCb 影像，最後用式(11)篩選出膚色並且標示目標。

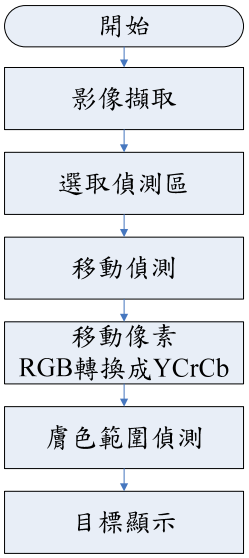



圖 12 實驗室工作區膚色偵測模組流程圖

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112.000 \\ 112.000 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{10}$$

$$M_{img}(x,y)=\begin{cases} M_{img}(x,y) & if \quad 133 \le Cr \le 173 \\ 0 & \& \; 77 \le Cb \le 127 \\ & otherwise \end{cases}$$
$$, x=1 \sim m, y=1 \sim n \qquad (11)$$

### 2.5 多執行緒處理模式

本研究多執行緒程式架構是將偵測項目分別模組化，並將各模組由以往非多執行緒用多執行緒取代。多執行緒比較非多執行緒的優點為資源共享容易、節省記憶體空間、快速的內文切換、平行處理等，其中利用其平行處理的優點用於多核心處理器達到同步多工效果，另三項可達到加快行程處理時間。

#### 2.5.1 多執行緒處理模式建立方法

偵測執行流程如圖 13 所示，首先建立執行緒並將所要偵測模組放入所建立的執行緒中，依照執行緒的優先權向 CPU 取得 CPU Time 的使用權，然後執行緒在時間內執行，而當獲得處理時間結束後執行緒會交出 CPU 使用權並將其狀態轉變成暫停狀態，等待下次的獲得 CPU Time 使用權，最後依照先前定義判斷執行緒是否繼續執行。

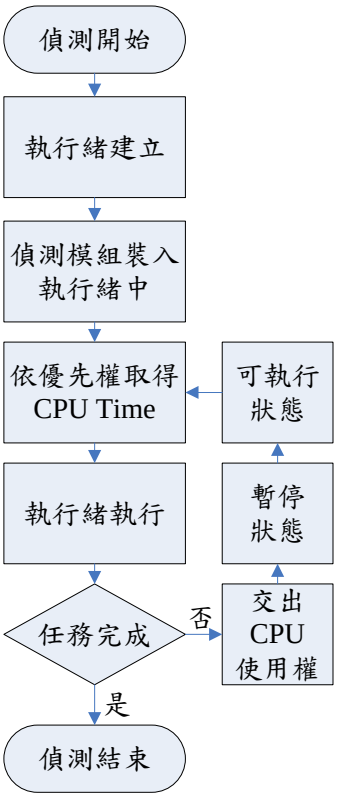



**圖 13 執行緒執行流程圖**

#### 2.5.2 優先權設定

優先權設定方面，本研究是採用程式開發工具 Borland C++ Builder 6.0 所提供的 7 個優先權，由高至低優先權分別為 tpTimeCritical、tpHighest、tpHigher、tpNormal、tpLower、tpLowest、tpIdle 等，各個執行緒可依照程式權重的高低手動調整優先權，來決定各偵測事件獲得 CPU Time 多寡。本研究建議最佳優先權設定在第二或第三的優先權，如果優先權設定太高容易使程式資源被某行程獨佔，設定太低會使行程等待執行時間變長。

#### 2.5.3 多執行緒處理模式與非多執行緒處理模式性能比較方法

本研究主要目的是要使程式執行具有同步性及提高 CPU 使用率用以增加程式速度。如圖 14 所示，利用此架構比較單核心 CPU 及雙核心 CPU 的多執行緒處理模式與非多執行緒處理模式的程式性能，用此判斷程式是否為上述兩項目的。

圖 14 中，行程 1 功用是管控行程 2 至行程 11 的執行，並設有計時器記錄程式執行時間，行程 2 至行程 11 功用為偵測事件程式，各行程設有計數器計算執行次數。

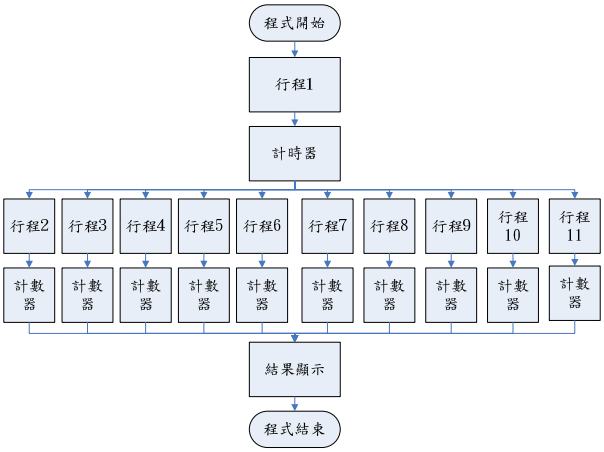



**圖 14 程式性能測試架構圖**

程式執行速度數據取得方式，當程式測試按鈕按下時，程式中計時器開始計時，測試時間為 60 秒，在測試時間內行程 1 至行程 11 的計數器計算程式進行行程次數，當測試時間結束時各行程執行次數加總取平均值，再將平均值除以執行時間得到程式執行速度。

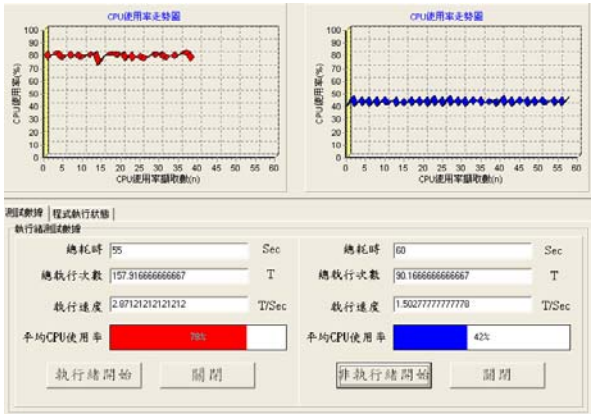



**圖 15 多執行緒處理模式與非多執行緒處理模式性能比較程式介面**

圖 15 中，為程式測試的顯示介面，其分別顯示程式執行時間、執行次數、執行速度及平均 CPU 使用率，用此比較多執行緒處理模式及非多執行緒處理模式的執行速度及 CPU 使用率。

本研究利用 API 函式取得程式執行時 CPU 使用率，其函式為 NtQuerySystemInformation。平均 CPU 使用率的數據取得方式，當測試開始鈕按下時，數據從執行到第 5 秒後開始擷取 CPU 使用率，擷取頻率為 1 次/秒，測試時間為 60 秒，將所得 CPU 使用率累加並求其平均值。

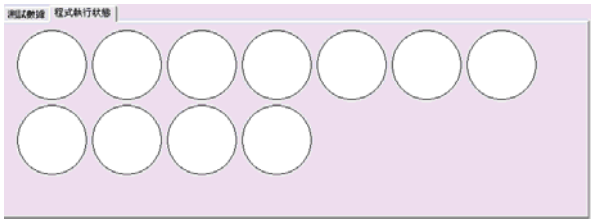



**圖 16 行程執行狀態顯示燈**

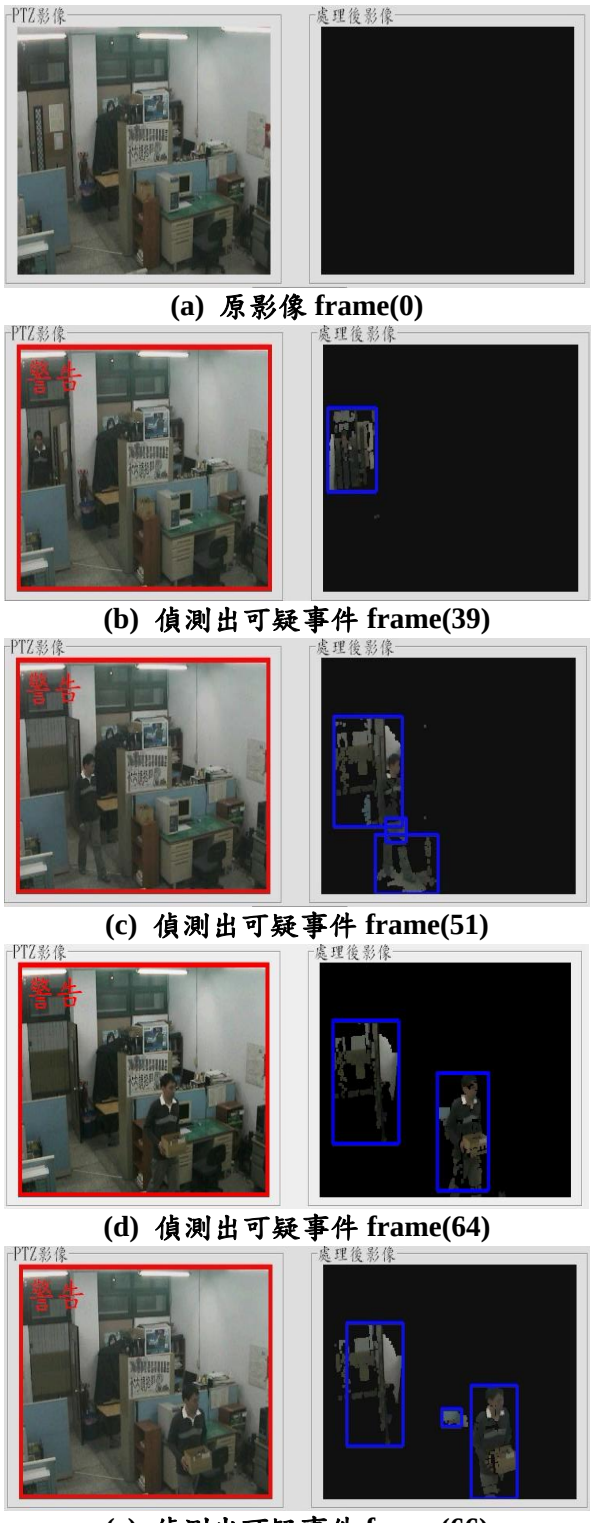
圖 16 中，顯示各行程執行狀態，燈亮表示執行中，反之為不在執行中，用此來觀看程式執行是否具有同步性。

## 3. 研究結果

本研究主要發展為多事件偵測時，讓偵測事件具有同步性並且提高 CPU 使用率幫助程式運算。借由實驗室安全監控的偵測事件來模擬多事件偵測，利用圖 14 的程式架構比較單核心 CPU 及雙核心 CPU 的多執行緒處理模式及非多執行緒處理模式程式性能。程式開發上有三項程式功能，分別為警戒模式事件偵測、監控模式事件偵測及多執行緒處理模式與非多執行緒模式性能比較，其分別結果如下:

### 3.1 警戒模式結果

圖 17(a)中利用時間均值法所建立背景模型，而圖 17(b)-(e)是利用前景偵測產生前景畫出其可疑區域。其中圖 17(c)可知，前景的部份有斷開的情況產生，後續會針對此問題去找出解決的方法。




**(a) 原影像 frame(0)**

**(b) 偵測出可疑事件 frame(39)**

**(c) 偵測出可疑事件 frame(51)**

**(d) 偵測出可疑事件 frame(64)**

**(e) 偵測出可疑事件 frame(66)**

**圖 17 移動物體追蹤及可疑區域模組範例**

### 3.2 監控模式結果

#### 3.2.1 禁止區移動偵測模組

圖 18 (a)中圈選所需管制的區域，如圖 18(b)所示，而圖 18(c)(d)爲利用移動偵測當有物體進入區域內時系統會發出警報。


**(a) 原影像 frame(0)**


**(b) 偵測區設定 frame(5)**


**(c) 可疑事件發生 frame(15)**


**(d) 可疑事件發生 frame(18)**

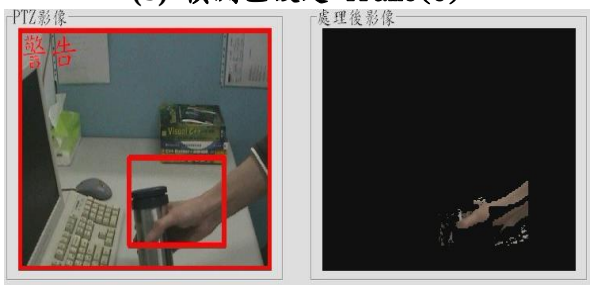
**圖 18 禁止區移動偵測模組範例**

#### 3.2.2 實驗室工作區膚色偵測模組

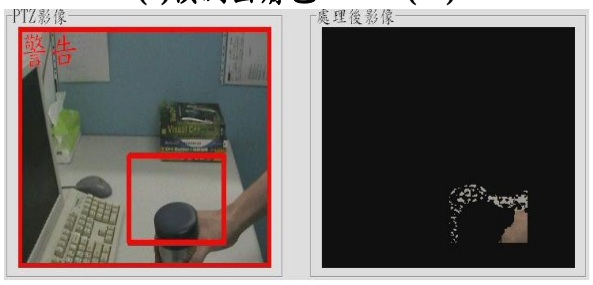
圖 19(a)中圈選所需進行偵測膚色區域，如圖 19(b)所示，而圖 19(c)(d)與圖 20(a)(b)為利用區域內移動成份進而將背景跟膚色相近像素濾除，再進行膚色成份抽取，當偵測出膚色時系統會發出警報。此方法在雖然避免固定背景的相似顏色雜訊，但是其所取得的膚色部份並不完整，後續會針對此問題去做探討。


**(a) 初始影像 frame(0)**


**(b) 偵測區設定 frame(3)**




**(c)偵測出膚色 frame(10)**




**(d) 偵測出膚色 frame(14)**

**圖 19 實驗室工作區膚色偵測模組範例一**


**(a) 未偵測出膚色 frame(14)**


**(b) 未偵測出膚色 frame(16)**

**(c)**


(c) **未偵測出膚色 frame(19)**

**圖 20 實驗室工作區膚色偵測模組範例二**

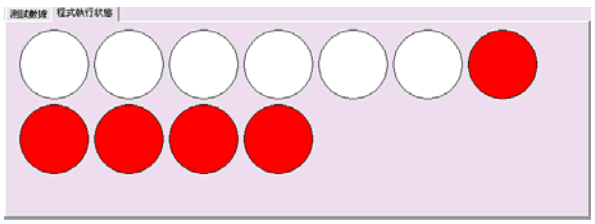
在圖 20(a)(b)(c)的右側畫面中會出非膚色像素雜訊點，這是因為在移動過程中會有少數背景像素點被誤認成移動像素，且這些像素的範圍與膚色的範圍值相近，解決方法為統計偵測出的像素點，並設閥值，大於此閥值才警報。

### 3.3 程式性能比較結果

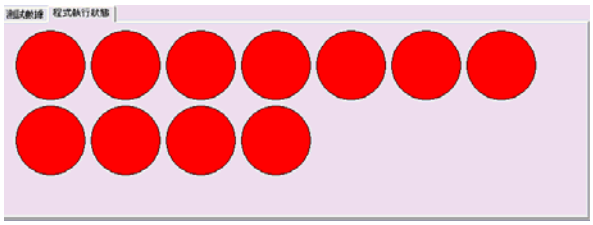
本項性能測試目的是在相同的系統設備與相同處理行程的條件下比較多執行緒處理模式與非多執行緒處理模式程式執行速度及 CPU 使用率，用此了解多執行緒處理模式是否具有同步性及在 CPU 使用率上是否有提升。其結果分別如下:

#### 3.3.1 同步性測試結果

圖 21 中，在同時執行 11 個行程時，非多執行緒處理模式程式僅在同一時間執行 5 個行程，尚有 6 個行程等待執行，所以當其中 1 個行程權重較重時會使得其他行程空轉造成偵測上的死角產生，而在多執行緒處理模式部份 11 個行程享有 CPU Time，讓偵測時達到同步化。



**(a) 非多執行緒處理模式**



**(b) 多執行緒處理模式**

**圖 21 行程執行狀態圖**

**表 1 雙核心程式在非多執行緒處理模式的程式執行性能比較**

| 比較項目 / CPU 規格 | CPU 使用率 (%) | 執行速度 (次/秒) |
|---|---|---|
| AMD Athlon64x2 | 40 | 1.657 |
| Intel Dual-core CPU E5200 | 37 | 2.53 |
| Intel Dual CPU E2180 | 41 | 2.119 |
| Intel Core2 Quad CPU Q6600 | 14 | 1.9237 |
| Intel Core( TM ) 2 Duo CPU P8400 | 38 | 1.758 |

**表 2 雙核心程式在多執行緒處理模式的程式執行性能比較**

| 比較項目 / CPU 規格 | CPU 使用率 (%) | 執行速度 (次/秒) |
|---|---|---|
| AMD Athlon64x2 | **78** | **3.2547** |
| Intel Dual-core CPU E5200 | **64** | **5.21** |
| Intel Dual CPU E2180 | **81** | **3.88** |
| Intel Core2 Quad CPU Q6600 | **32** | **3.88** |
| Intel Core( TM ) 2 Duo CPU P8400 | **73** | **3.458** |

#### 3.3.2 程式執行速度與 CPU 使用率比較結果

本測試利用圖 14 程式架構進行測試，將行程 2 至行程 11 分別放入警戒模式及監控模式偵測程式，分別在 5 個不同雙核心 CPU 電腦進行多執行緒處理模式及非多執行緒處理模式，各個 CPU 進行 10 次測試取其平均值得到實驗數據，即產生表 1 及表 2。

其中(次/秒)單位代表每秒執行偵測事件的次數，以本研究程式測試來說，每增加 1 單位即讓行程 2 至行程 11 每秒各增加 1 次的執行次數。

由表 1 及表 2 可知本研究建立多執行緒處理模式比傳統非多執行緒處理模式更適合建立在多事件偵測架構。以 CPU AMD Athlon64X2 實驗設備來說使用率提升 40%左右，而執行速度方面多執行緒比非執行緒提升至 1 倍。本研究開發測試架構在另四台雙核心電腦上測試，得到數據再一次驗證多執行緒程式架構在 CPU 及程式處理速度上均有所提升，強化數據客觀性。另一方面，可由表中看出 CPU 使用率在相同程式執行上略有不同，原因是出於各個 CPU 算運能力不同，不論 CPU 有所不同都證明多執行緒處理 CPU 使用率及程式處理速度均有所提升。

圖 21 及圖 22 中，合理化表 1 及表 2 得到實驗數據。在圖 21 中，非多執行緒處理模式僅會將程式指派給 1 顆 CPU 做運算，其 CPU 平均使用率均不會超出 50%，如同表 1 所示。在圖 22 中，執行緒會將程式分割成若干行程平行輸入給 CPU 做運算，所以在雙核心 CPU 上不論是 CPU 使用率或是程式執行速度都提升至 1 倍左右，如表 2 所示。

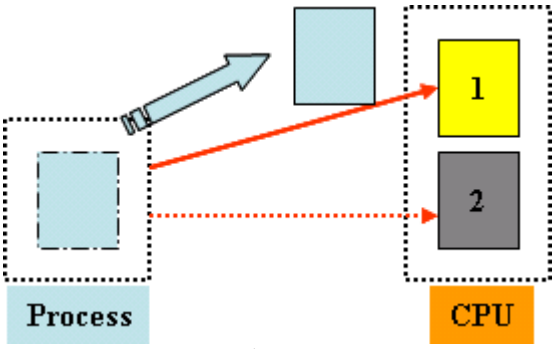



**圖 21 雙核心 CPU 處理器在非多執行緒處理模式程式執行時狀態示意圖**

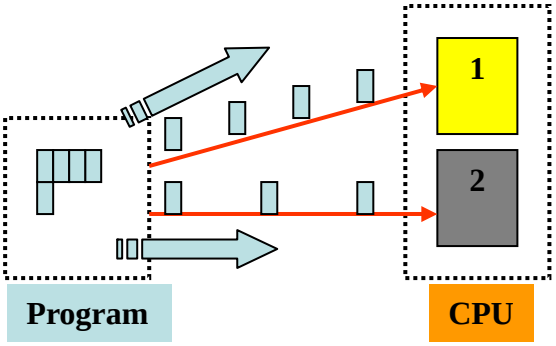



**圖 22 雙核心 CPU 處理器在多執行緒處理模式程式執行時狀態示意圖**

表 3 為利用上述程式測試方式，在另一台 PC 上所得的數據，其電腦規格為 CPU 是採用 AMD K8-3000+ 單核心處理器及記憶體為 DDR1 1.5G。由表 3 結果可知執行緒取代非執行緒的方式在效果上差不多，原因如圖 24 所示。圖 23 中，非多執行緒處理模式(淺藍色)和多執行緒處理模式(深藍色)，其目標 CPU 僅有 1 個，所以多執行緒處理模式的平行處理優點在單核心系統無法發揮，故得到的效果與非執行緒處理模式相同，如表 3 所示。

**表 3 單核心程式執行性能比較**

| 比較項目 / 處理模式 | CPU 使用率 (%) | 執行速度 (次/秒) |
|---|---|---|
| 非多執行緒處理模式 | 79 | 2.0785 |
| 多執行緒處理模式 | 80 | 2.1022 |

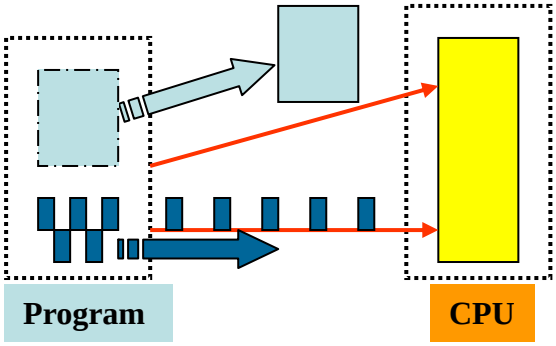



**圖 23 單核心 CPU 處理器在非多執行緒與多執行緒處理模式程式執行時狀態示意圖**

## 4. 結論與未來展望

### 4.1 結論

本研究提出一個比傳統的多事件偵測更適合的程式架構，將各偵測事件程式模組化，降低程式複雜度，讓程式具有延展性。其中針對程式同步性、提升 CPU 效能進行探討。借由實驗室的警戒模式及監控模式的偵測事件，進行多執行緒與非多執行緒的效能比較。結論如下：

(1) 利用多執行緒取代傳統非多執行緒，讓執行平行處理，在相同偵測事件下達到提升 1 倍左右的 CPU 效能。

(2) 利用多執行緒比非多執行緒較資源共享容易、節省記憶空間、快速內文切換等優點每秒各增加 1 倍左右的處理速度。

(3) 利用多執行緒分配 CPU Time 的特性，讓程式達到同步多工效果。

(4) 利用執行緒有優先權的特點，能針對行程權重決定給予優先權的高低。

(5) 將相同程式及測試方法，在另一台單核心處理器進行測試，可發現其多執行緒與非多執行緒效果差不多，原因是多執行緒平行處理的優點僅在多核心系統上可看出其優點。

(6) 實驗室安全事件偵測方面，初步建構功能有人物追蹤與問題區域標示、禁區警戒及實驗台膚色偵測等功能，其測試結果有不錯準確性及即時性。

### 4.2 未來展望

本研究針對第三代監控系統的多攝影機技術所延伸的多事件偵測提出更適合的偵測架構，而在未來要繼續探討的事項有下列幾點:

(1) 本研究首先以一台攝影機及數個偵測事件模擬多攝影機技術，故可能忽略了許多實際狀況才會發生的錯誤，此問題是後續進行探討。

(2) 影像監控系統中功能項目可分成四類，分別為物件偵測與識別、追蹤、行為分析及資料管理，其中本研究首先對物件偵測、追蹤部份功能建立，後續會針對行為分析及資料管理做探討，使本研究的實驗室影像監控系統功能更加完備。

## 參考文獻

[1] M. Valera, and S. A. Velastin, "Intelligent Distributed Surveillance Systems: A Review," ***IEE Proc.-Vis. Image Signal Process.*** , Vol. 152, No. 2, April 2005.

[2] Yi-Tsung Chien, Ting-Wu Ho, and Ching-Chun Hunag, "Framework for Manageable Multi-Event Detection，Alarming and Monitoring," ***CCL Technical Journal***, 6.25.2005.

[3] L. Kovacs, A. Utasi, Z. Szlavik, L. Havasi, and I. Petras, "Digital Video Event Detector Framework for Surveillance Applications," ***2009 Advanced Video and Signal Based Surveillance***, 2009.

[4] Humayun Shahid, Kamran Khan, and Waqas Ahmed Qazi, "Using Modified Mixture of Gaussians for Background Modeling in Video Surveillance," ***ICAST 2008 2nd International Conference on Advances in Spaces Technologies Islamabad***, Pakistan, 29th -30th November, 2008.

[5] Dorin Comaniciu, "Mean Shift: A Robust Approach Toward Feature Space Analysis," ***IEEE Transactions on Pattern Analysis and Machine Intelligence***, Vol. 24, No. 5, May 2009.

[6] John G. Allen, Richard Y. D. Xu, and Jesse S. Jin, "Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces," ***Inc. Australian Computer Society***, Vol. 36, 2004.

[7] P. S. Hiremath, and A. Danti, "Detection of Multiple Faces in an Image Using Skin Color Information and Lines of Separability Face Model," ***International Journal of Pattern Recognition and Artificial Intelligence***, Vol. 20, 2006.