中華民國第 54 屆中小學科學展覽會

作品說明書

高中組　生活與應用科學科

佳作

040803

凡走過必留下痕跡－利用 Kinect 進行物件追蹤

學校名稱：國立新竹女子高級中學

作者：	指導老師：
高二 劉芸瑄 高二 曾聖茗 高二 林雅瑄	徐以誠 康雅蘭

關鍵詞：物件追蹤、Kinect、影像處理

壹、摘要：

我們利用物體的色彩特徵而發展出物體的追蹤程式。為了提升物體追蹤的精確度和速度，我們比較了多種色彩空間的特性，發現 HSV 色彩空間受光線影響較小且色彩區辨力較強，因此我們將數位系統初得的 RGB 色彩空間數據轉為 HSV 色彩空間資訊並將之呈現在程式畫面上，至於雜訊的處理，我們採用了模糊、侵蝕、擴張的方法來消除。微軟新一代體感遊戲的輸入介面 Kinect，除了提供傳統的 2D 彩色視訊外，還提供了深度資訊，藉由深度資訊的輔助，我們的物體追蹤程式可更為穩定，藉由 WPF 與 EmguCV 函式庫，程式的處理速度非常的快，可以達到真正的即時物體追蹤。

貳、研究動機與目的：

以往進行物體追蹤都需要使用昂貴的器材或較繁瑣的步驟才能得到較準確的數據，因此我們希望找到一個可簡便操作且成本較低的方法來進行。當我們看見 Xbox 推出的體感遊戲機 Kinect 可以偵測並追蹤人體動作，便想是否也可以應用在物體上，因其具有易操作、較為便宜等優點，且具有紅外線感測器及 RGB 彩色攝影機能取得深度和彩色影像資訊以提升物體位置與時間關係的精確度，進而達到與昂貴器材相近的追蹤效果。

參、器材及軟體：

一、多光源光箱。
二、色溫計。
三、筆記型電腦。
四、Microsoft Kinect for Windows。
五、數位相機 SONY DSC HX-200V。
六、數位攝影機 SONY HDR-CX380。

肆、研究過程：

一、物體追蹤簡介

在科學實驗或是專題研究中，常有紀錄運動物體之時間與位置的需求，例如蝴蝶飛舞的軌跡、模型飛機的飛行路線、小昆蟲的移動路徑及單擺擺盪運動的觀察等。目前的作法，通常是以攝影機將物體的運動過程攝錄下來，然後利用視訊處理軟體（如威力導演、繪聲繪影等）將視訊中的每格畫面分別存成影像檔，最後再以影像軟體（如 PhotoImpact、PhotoShop 等）開啟每張影像來進行逐格分析。採用這樣的處理方式，在取得物體的位置座標時，會耗費大量時間與人力，以現今的錄影設備每秒 30 格畫面來算，短短 1 分鐘的視訊，就必須分析 $30 \times 60 = 1800$ 張影像。

其實，可以不必那麼辛苦，根據參考資料[8]，我們可以利用物體追蹤的技術來自動完成上述工作。要實踐在視訊中的物體追蹤，可分為物體的偵測及追蹤等兩個步

驟，均為電腦視覺應用的核心技術，已在各種領域發現其應用範疇，例如在監控系統中，可用來圖形比對及追蹤各式入侵之人或物；在駕駛輔助系統，藉由視訊與感測信號之結合，可達到輔助駕駛人的各種感官死角，以增進行車安全；使用者介面的設計上，可作為數位藝術創作及遊戲設計的互動式技術等。

1. 物體偵測（Object Detection）

物體追蹤就是使用影像處理技術，讓電腦程式在畫面中自動偵測出想要偵測的物件，例如車輛、建築物、人類、球、號誌等，現有的物體偵測方法有點偵測器（Point detectors）、影像分割（Segmentation）、背景模型（Background Modeling）、監督式分類器（Supervised Classifiers）等四類型[8]，每種方法均有其精良的理論與設計，其共通點就是必須選擇適合該物體偵測的的特徵，以達到更佳的準確度，例如用於人臉偵測的 Haar-like features 就是非常有名且成功的人臉偵測特徵。

2. 物體追蹤（Object Tracking）

物體追蹤的目標就是藉由分析視訊的逐格畫面，找出目標物隨位置與時間變化而產生的軌跡，現有非常多的物體追蹤方法，主要可分為點追蹤（Point Tracking）、核心追蹤（Kernel Tracking）、輪廓追蹤（Silhouette Tracking）等三類[8]，但是沒有一種方法可以明顯地優於其他的方法，須依照使用的目的與環境而定，再選擇出最適合的方法。

要在視訊中穩健地追蹤物體是有其難度的，主要的原因有下列幾點：

- 畫面中呈現的是真實世界 3D 物體在 2D 影像的投影，外形會有所不同。
- 因為遮掩或是物體本身就沒有固定的外形。
- 畫面中的雜訊干擾。
- 外界光線的改變。
- 必須能夠在下一格畫面的到來前，即時的處理每格畫面的追蹤工作。

二、選擇適合偵測物體的特徵

現今的攝影機是以每秒 30 格畫面的速度進行紀錄，為了達到即時追蹤的目的，必須在很短的 33ms 時間內完成每一格畫面的追蹤工作，所以必須選擇可以快速計算的特徵，我們打算採用物體的「色彩」當作特徵來發展物體追蹤程式，主要的原因有：

- 不受遮掩及外形改變的影響。
- 處理的速度最快，只要把整個畫面掃描一次，就可以取出目標色彩。

但是根據參考資料[4]的說明，物體的色彩外觀，受到光源種類的影響很大，只要外界光源有些許變化，物體的色彩就會有很大的改變。參考資料[4]同時提到了色彩有許多表示方式，常用的色彩空間有 RGB、YCrCb、HSV、CIELab 等，那一種色彩表示法較能對抗外界的光源變化呢？

1. 不同光源下拍攝同樣的場景

我們在不同的光源下拍攝同樣的場景（A 光源的色溫為 6400K、B 光源的色溫為 5750K、C 光源的色溫為 5300K、D 光源的色溫為 3600K），然後以程式分析

物體顏色在不同的色彩空間分佈的情形，藉以尋找何種色彩最適合當作偵測物體時的特徵。

圖 1-1 A 光源

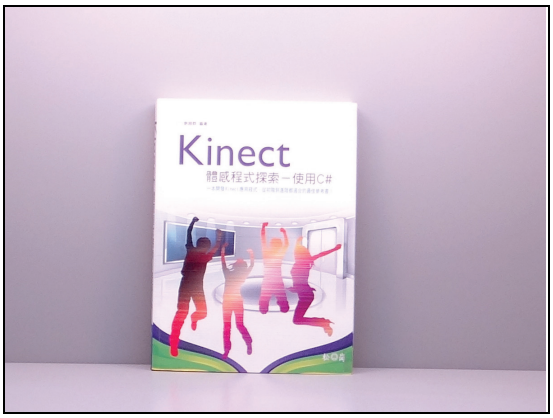


圖 1-2 B 光源

圖 1-3 C 光源

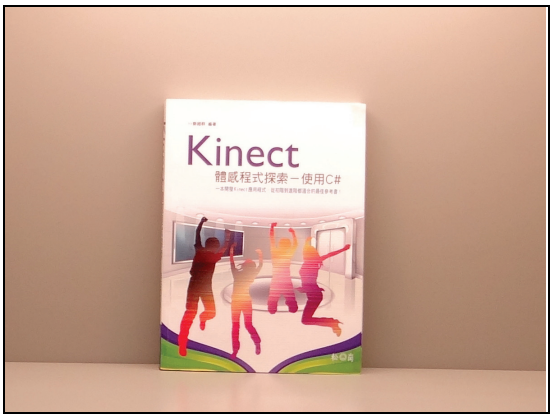


圖 1-4 D 光源

2. 色彩空間的轉換

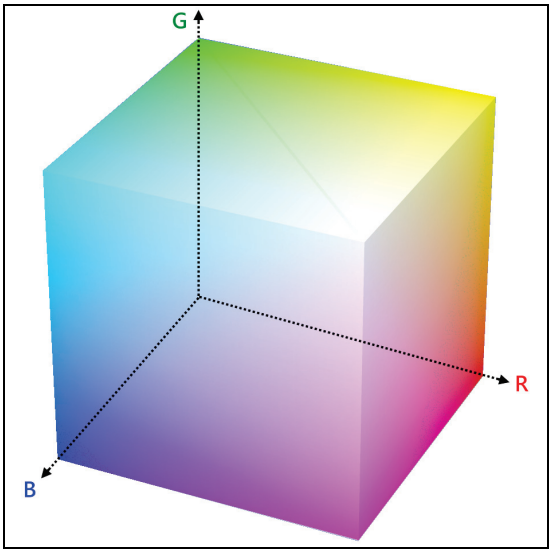


圖 2-1 RGB 色彩空間

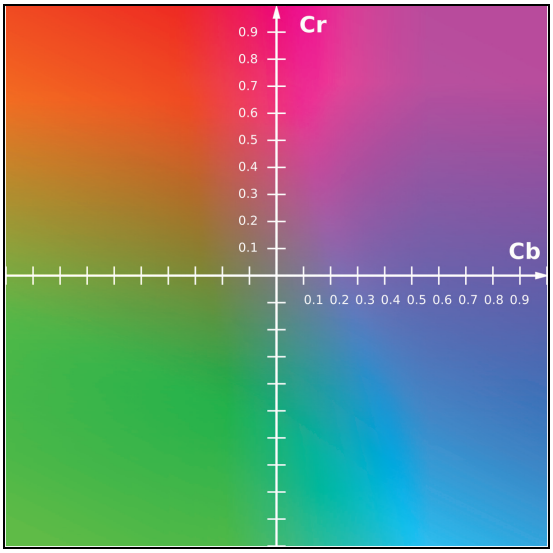


圖 2-2 YCrCb 色彩空間

RGB 色彩空間（圖 2-1）廣泛的運用在數位系統（掃描器、數位相機、攝影機等）上，此色彩空間是建立在以紅色（R）、綠色（G）、藍色（B）為三軸的立體直角座標系統上，所以每一個色彩均可用立體座標的三個分量來表示。

YCrCb 色彩空間（圖 2-2）為國際無線電咨詢委員會所發展出來數位視訊標準中

的色彩座標系統，Y 代表亮度，而 Cr 與 Cb 代表彩度，因為其對於亮度和彩度有很高的分離性，所以目前也廣泛被應用在影像處理中，YCrCb 與 RGB 間的關係可用下式來轉換：

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ 0.5000 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.5000 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

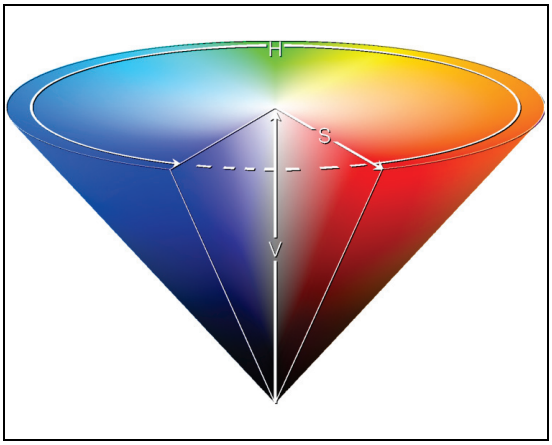


圖 2-3 HSV 色彩空間

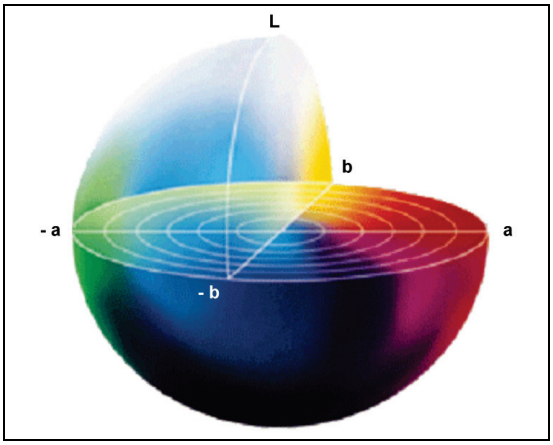


圖 2-4 CIELab 色彩空間

HSV 色彩空間（圖 2-3）是一種將 RGB 色彩模型中的點在圓柱坐標系中的表示法，H 表示色相、S 表示飽和度、V 表示明度，藝術家比較偏好使用 HSV 色彩模型而不選擇三原色 RGB 色彩模型，因為 HSV 類似於人類感覺顏色的方式，具有較強的感知度。從 RGB 到 HSV 的轉換如下：

$$V = \max(R,G,B) \qquad S = [V - \min(R,G,B)]/V$$

$$H = \begin{cases} (G-B)\times 60/S & \text{,假如 } V = R \\ 180 + (B-R)\times 60/S & \text{,假如 } V = G \\ 240 + (R-G)\times 60/S & \text{,假如 } V = B \end{cases}$$

由於人類眼睛對於不同色彩的寬容度缺乏一致性，所以國際照明委員會（CIE）於 1976 年發展出比較可以呈現視覺上均勻色彩空間 CIELab（圖 2-4），CIELab 是以對立色理論為基礎，以三組可區分不同色彩訊號的 L（明度）、a（紅與綠）、b（黃與藍）為座標軸，分別命名為 L^*、a^*、b^*，進而產生 CIELab 色彩空間。RGB 到 Lab 的轉換方式如下：

$$[X \quad Y \quad Z] = [R \quad G \quad B] \begin{bmatrix} 0.1618 & 0.0834 & 0.0076 \\ 0.1402 & 0.2805 & 0.0467 \\ 0.0707 & 0.0283 & 0.3727 \end{bmatrix}$$

$$L = 116 f\left(\frac{Y}{Y_{white}}\right) - 16$$

$$a = 500\left[f\left(\frac{X}{X_{white}}\right) - f\left(\frac{Y}{Y_{white}}\right)\right]$$

$$b = 200\left[f\left(\frac{Y}{Y_{white}}\right) - f\left(\frac{Z}{Z_{white}}\right)\right]$$

$$f(\alpha) = \begin{cases} \sqrt[3]{\alpha} & \alpha > 0.008856 \\ 7.787\alpha + \frac{16}{116} & \alpha \le 0.008856 \end{cases}$$

3. 色彩空間的比較

我們將書本的相同區域在不同光源拍攝時的影像取出，使用程式分析在不同色彩空間的分佈情形，並計算 4 個區域的標準差，有較小標準差者，表示該色彩頻道受外界光源變化的影響較小，實驗結果如表格 1 所示，色彩頻道 G、Y、H、a 較能對抗外界的光源變化。

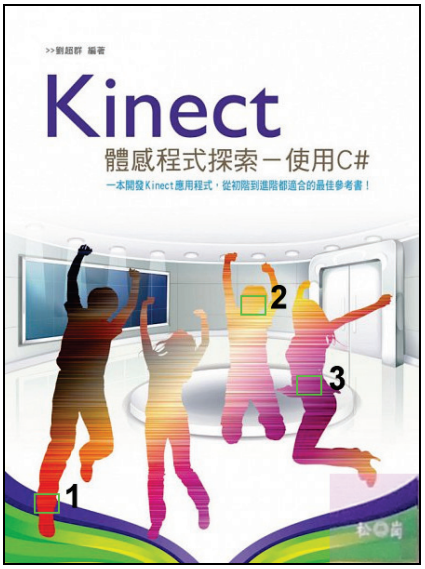


圖 3 各影像中取出做比對的區域

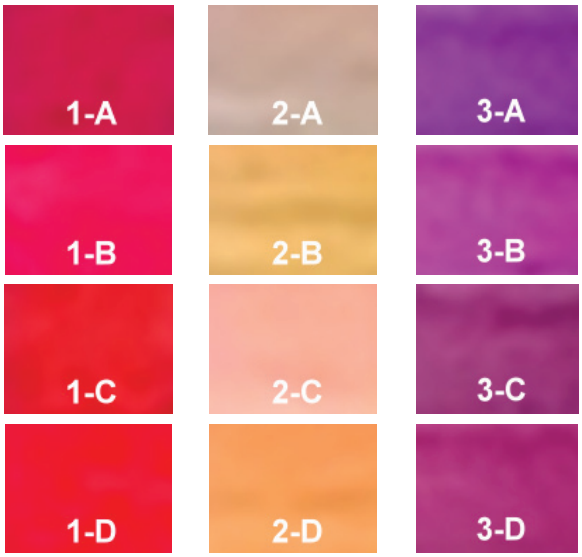


圖 4 各編號區域在不同光源下的色彩影像

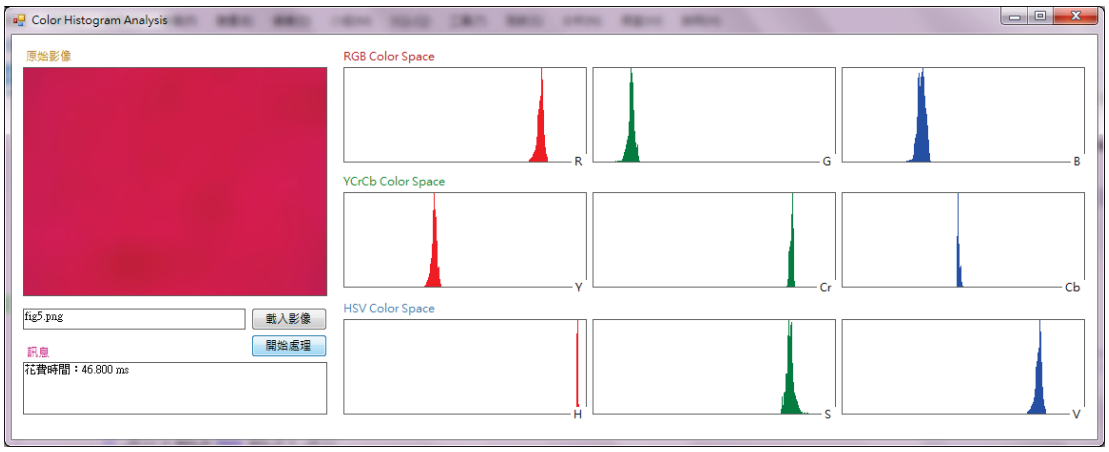


圖 5 分析影像在不同色彩空間的分佈情形

項目 \ 色彩頻道	R	G	B	Y	Cr	Cb	H	S	V	L	a	b
編號 1 區域	21.2	8.3	17.7	7.0	11.9	10.2	4.7	10.5	21.2	10.3	6.4	12.0
編號 2 區域	22.8	8.5	31.2	9.2	13.2	16.6	6.6	42.8	22.8	7.9	8.6	16.8
編號 3 區域	16.0	4.4	19.3	6.5	8.7	9.1	9.2	9.7	12.0	7.0	4.8	10.4
平均標準差	20.0	**7.1**	22.7	**7.6**	11.3	12.0	**6.8**	21.0	18.7	8.4	**6.6**	13.1

表格 1 各色彩頻道因外界光源變化而影響色彩變化的標準差

除了能夠對抗外界的光源變化，也得要有良好的色彩區別能力才行，我們準備了一些測試影像，用肉眼就可以看出與其他影像在色彩上的不同。由圖 6-1 看來，測試影像與前 4 張影像在 Y 頻道的分佈很接近，沒有區別的能力；圖 6-2 中，測試影像的 a 頻道分佈與前 4 張影像很接近，區別能力不佳；圖 6-3 中，測

試影像的 G 頻道分佈與前 4 張影像無明顯差異，也沒什麼區別能力。扣除了 Y、a、G，HSV 色彩空間中的 H（色相）應該是最好的選擇。

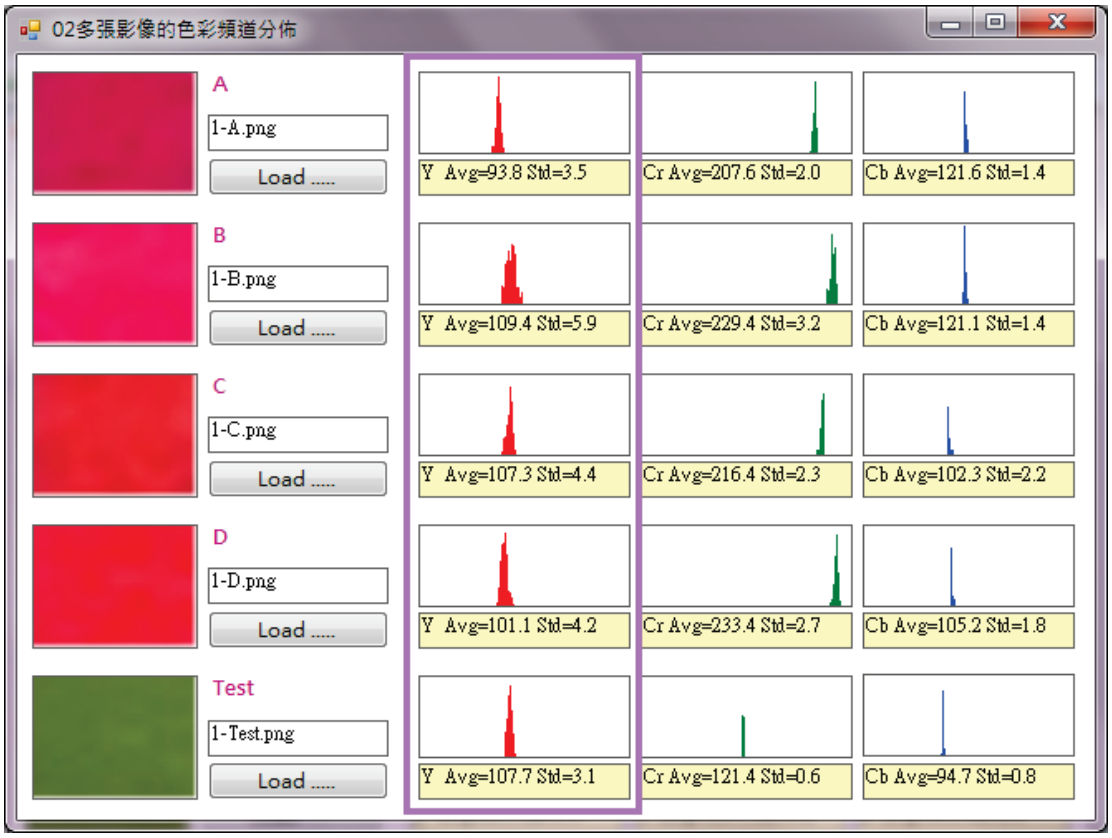


圖 6-1 測試影像的 Y 值分佈與前 4 張影像無明顯差異

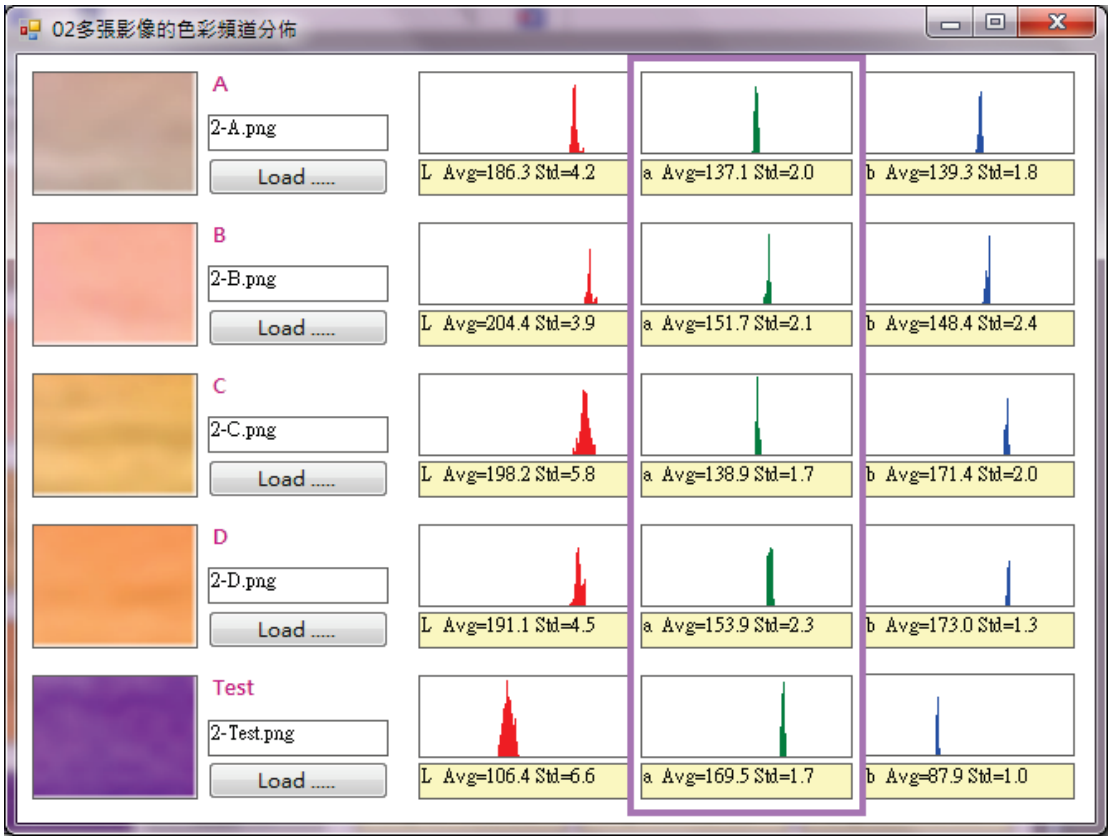


圖 6-2 測試影像的 a 值分佈與前 4 張影像很接近

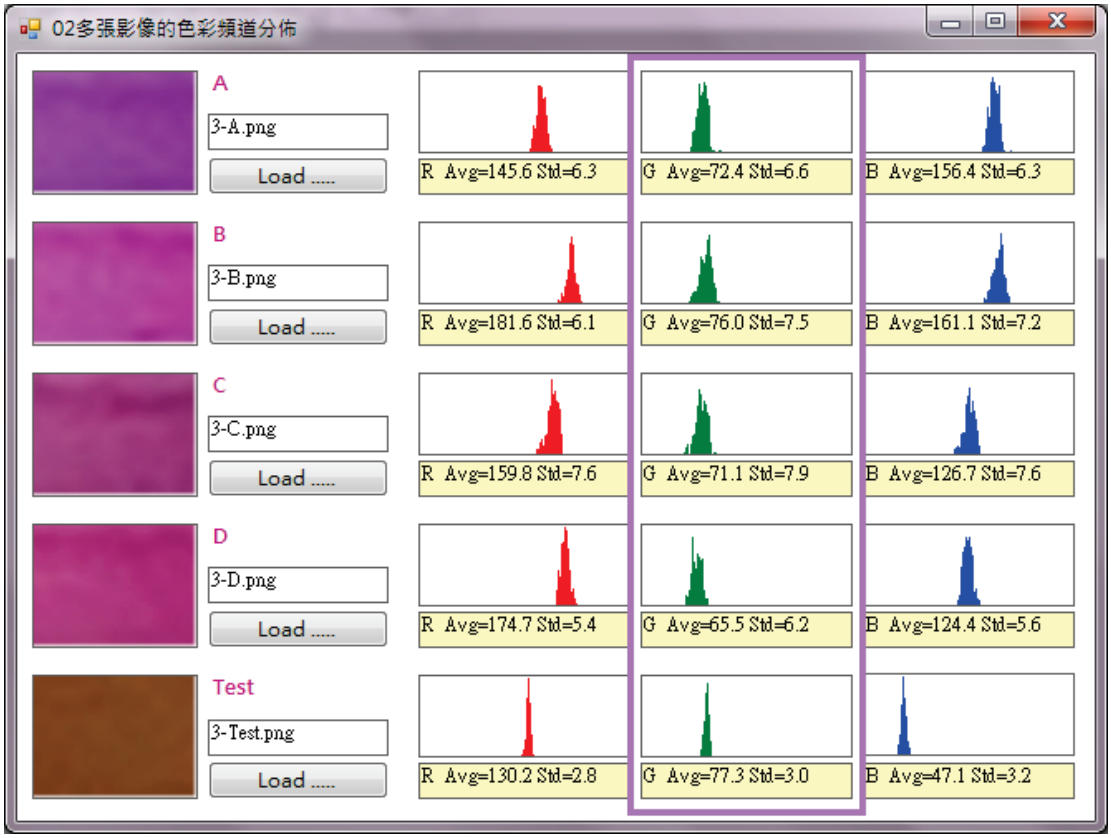


圖 6-3 測試影像的 G 值分佈與前 4 張影像無明顯差異

三、畫面中物體的偵測

1. 依使用者選擇的色彩偵測物體

初步設計的程式如圖 7-1 所示，載入影像後，利用滑鼠點選想要偵測的物體（黃色球）後，依色彩篩選出來的物體如右邊黑色遮罩影像中的白色區塊（黃圈處）。

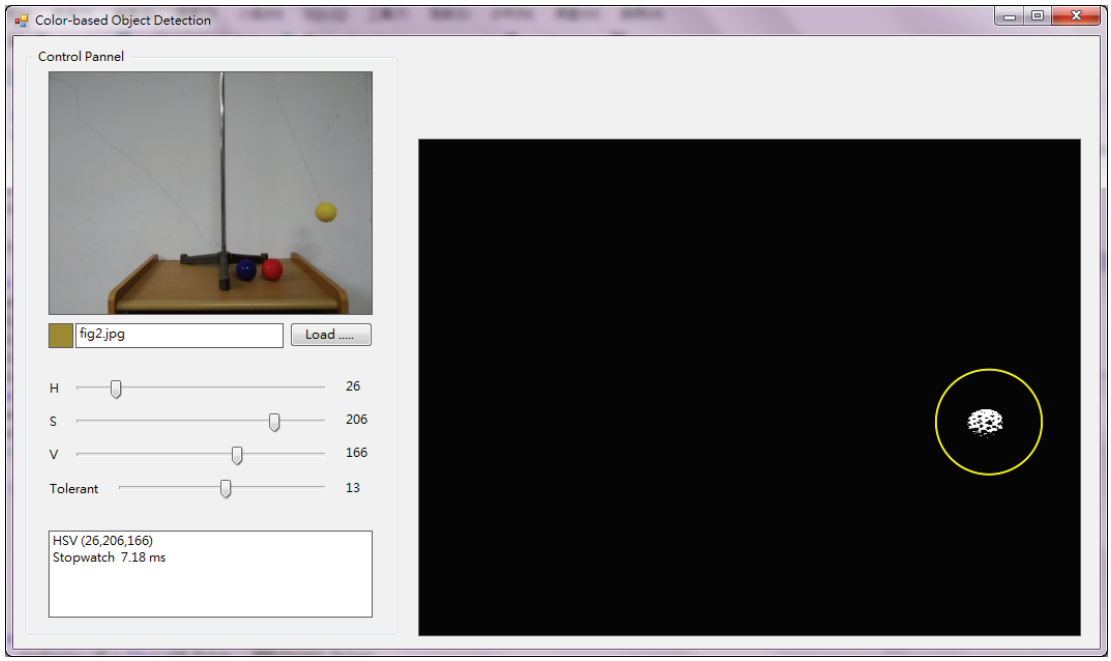


圖 7-1 依使用者點選的色彩進行偵測

2. 雜訊的消除

由圖 7-1 看出，偵測出來相對應黃球的白色區塊中有一些破洞，依據參考資料[3]的建議，可以使用模糊（Blur）、侵蝕（Erode）、擴張（Dilate）來處理，圖 7-2 使用了模糊、侵蝕及擴張的運算，把破洞都補齊了。

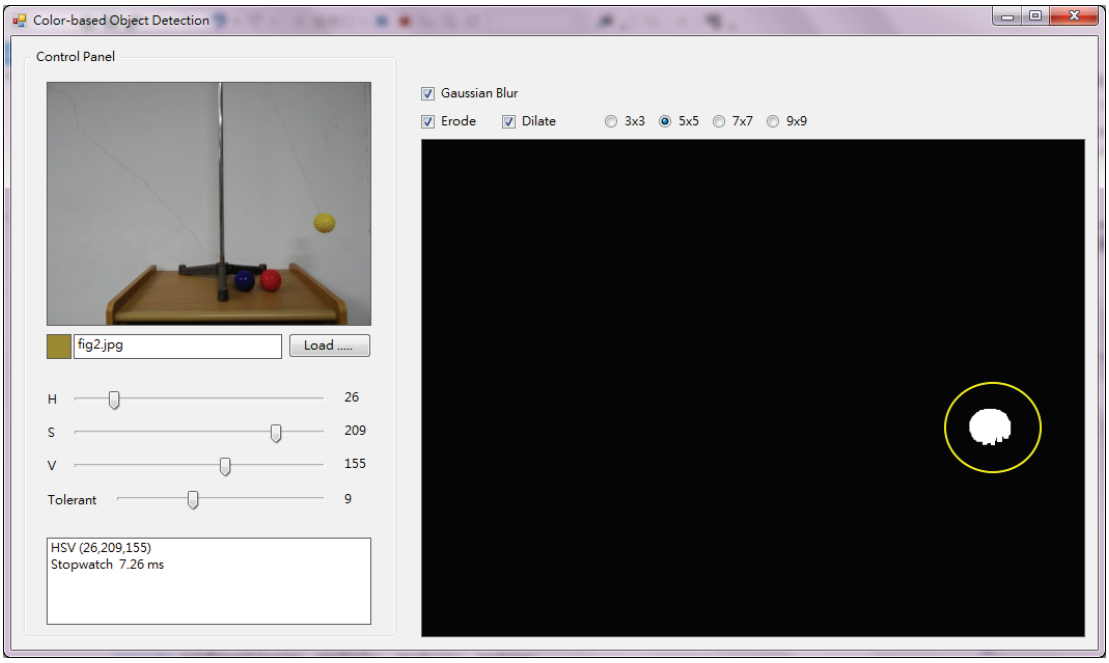


圖 7-2 利用模糊、侵蝕、擴張的運算將物體的破洞補齊

3. 物體偵測與效能分析

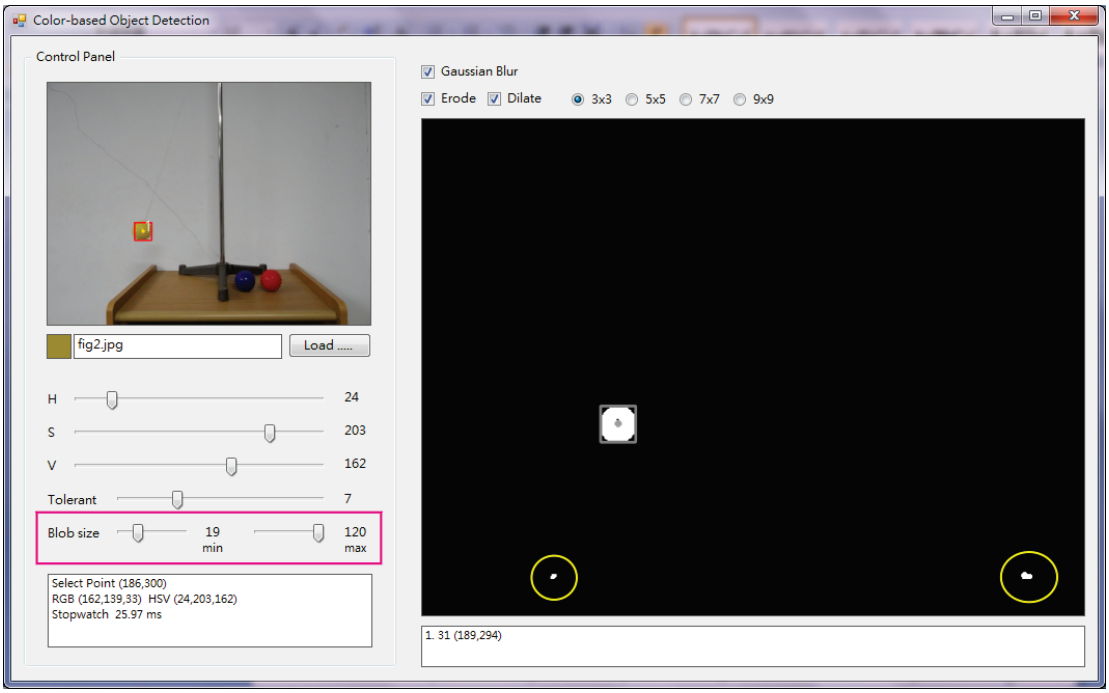


圖 7-3 利用連通物件標示的方法偵測出物體

電腦無法像人類的眼睛與大腦，可以很快的識別出物體所在的位置，我們利用影像處理中連通區域標記[3]的方法在遮罩影像中偵測出物體，並將其加上方框及中心點，同時在原始彩色影像中加上紅框、中心點與編號。圖 7-3 下方的黃圈處有兩個較小的白色區域，應該是目前選擇的色彩與兩邊桌角的顏色相近，

所以同時也被偵測出來，我們增加了兩個捲軸，可由使用者設定想要偵測物體的大小範圍，這樣會更有彈性。

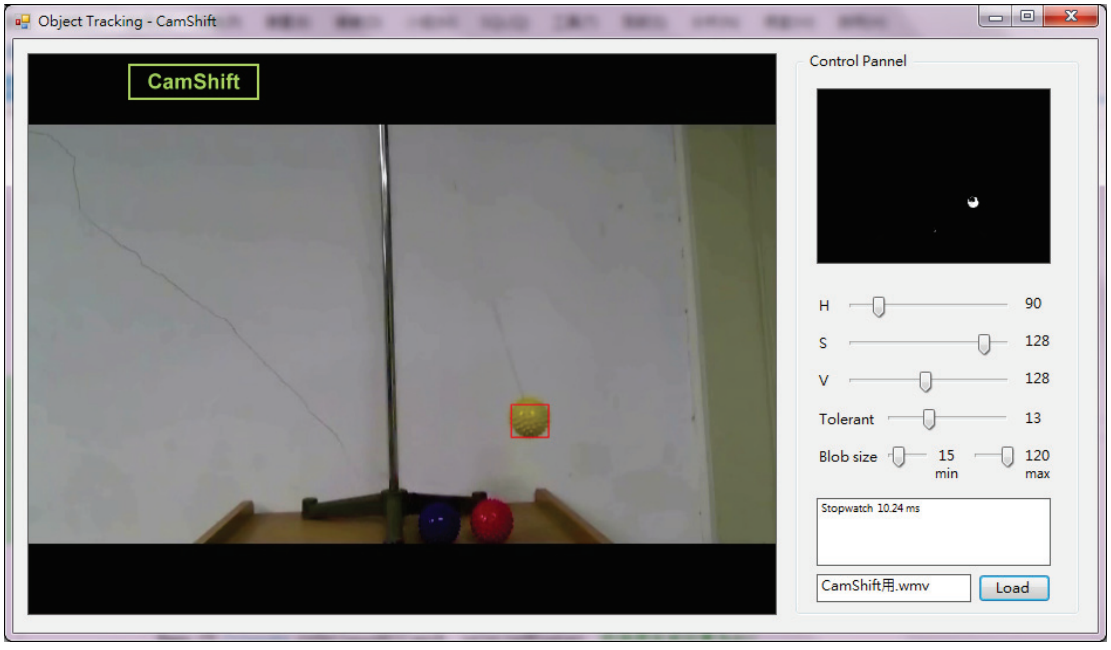
到目前為止，要依使用者選定的色彩來偵測出影像中的物體，需要經過 RGB 到 HSV 的色彩空間轉換，還有模糊、侵蝕、擴張的消除雜訊處理，以及連通物件的尋找，這一連串的指令是否可以在 33ms 的時間內，即時完成一幅 640×480 影像的處理呢？本來是採用微軟視窗系統中最常用使用的影像處理指令 GetPixel 與 SetPixel，但是執行結果非常令人洩氣，每幅平均處理時間約為 100ms，經過網路上的搜尋，許多網頁都提到，若是先將影像放置到陣列（記憶體）中，處理後再拷貝回原影像，可以大幅提升效能。我們利用迴圈重複處理 1000 次，在 Intel i5 M560 @2.67GHz 的筆電上測試，平均的處理時間為 11.06ms，因此發現改用直接記憶體存取的方法後，程式的執行速度快到超乎想像。

四、視訊中物體的追蹤

物體的追蹤程式種類繁多，著名的有 Optical flow、MeanShift、CamShift、SURF 等[8]，我們在網路找到了 Optical flow[9]、CamShift[10]與 MeanShift[9]等三種物體追蹤程式。同樣地，我們以擺盪中的黃球追蹤來進行測試，令人訝異的，追蹤的效能並未如預期中的好，我們以 CamShift 為例，利用 VB.net 及 EmguCV 函式庫[7]實做了物體追蹤程式，藉以瞭解其原因所在，另外，為了使得測試的環境一致，以錄影機拍攝了一段黃球擺盪的影片當作測試的視訊。

1. CamShift 物體追蹤程式

CamShift[8][10]演算法是 Continuously Adaptive Mean Shift algorithm 的簡稱，由其名稱就可以知道它是 MeanSift 的改良方法，於 1998 年，首次由 Gary R.Bradski 等人提出和應用在人臉的追蹤上，獲得不錯的效果，由於是利用顏色機率分佈的直方圖及可以動態調整搜尋範圍的大小來進行物體的追蹤，使得它的運行效率也比較高。



利用 CamShift 來追蹤擺盪中的黃球，當擺盪幅度較大時，幾乎是無法追蹤的上

黃球的位置，此點可由圖 8 中的第 5 張圖片看出。只有像圖 9，當擺動幅度較小時，勉強可以追蹤。探究其原因，可能是因為黃球在擺幅較大時，移動的速度較快，在下一幅的畫面中位置，已超過程式能夠搜尋的範圍，所以跟丟了。

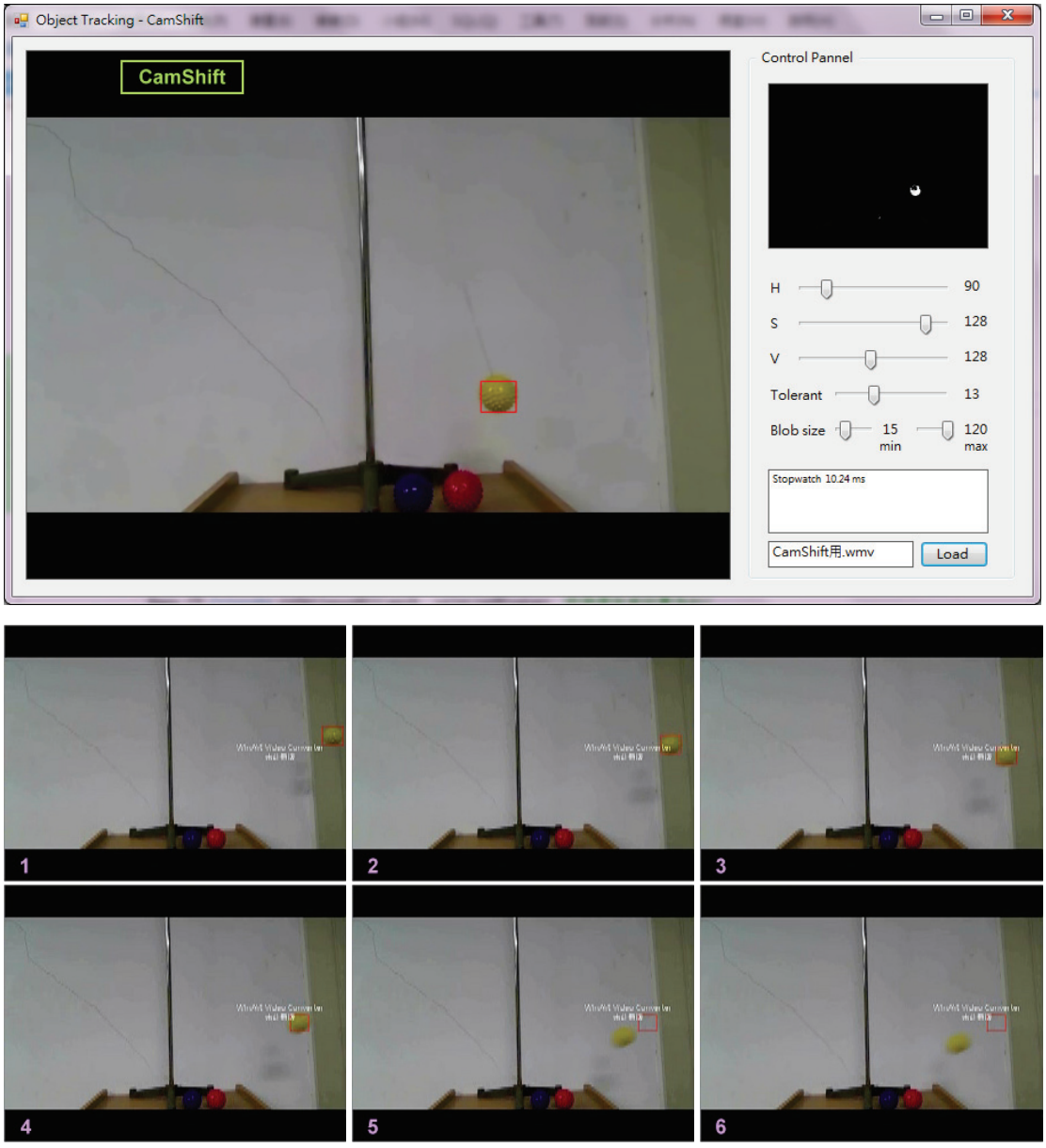


圖 8 利用 CamShift 來追蹤大幅度擺盪中的黃球

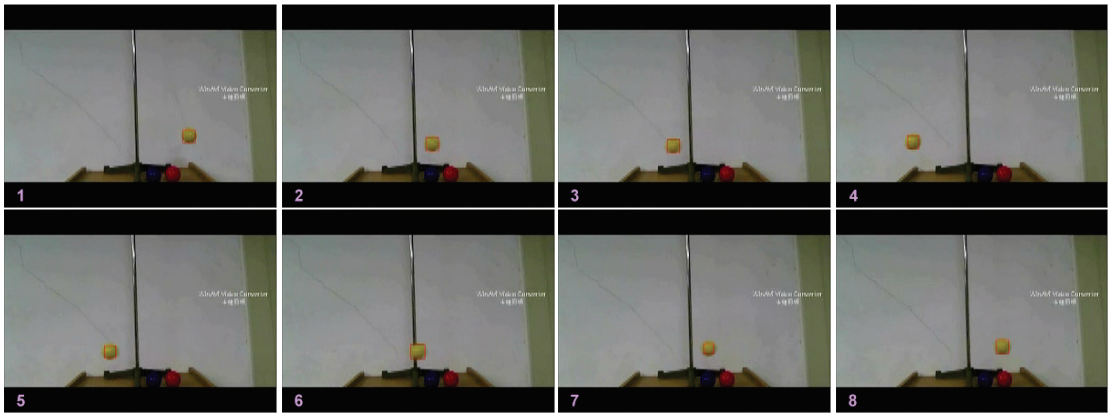


圖 9 在小擺幅的擺盪中 CamShift 物體追蹤的表現尚稱良好

2. 我們的方法

對於大幅度快速移動的物體，似乎追蹤程式都無法勝任，所以我們改用每幅畫面都執行物體的偵測來追蹤物體，還好利用顏色偵測物體的速度夠快，所以感覺起來就像是在視訊中追蹤物體。

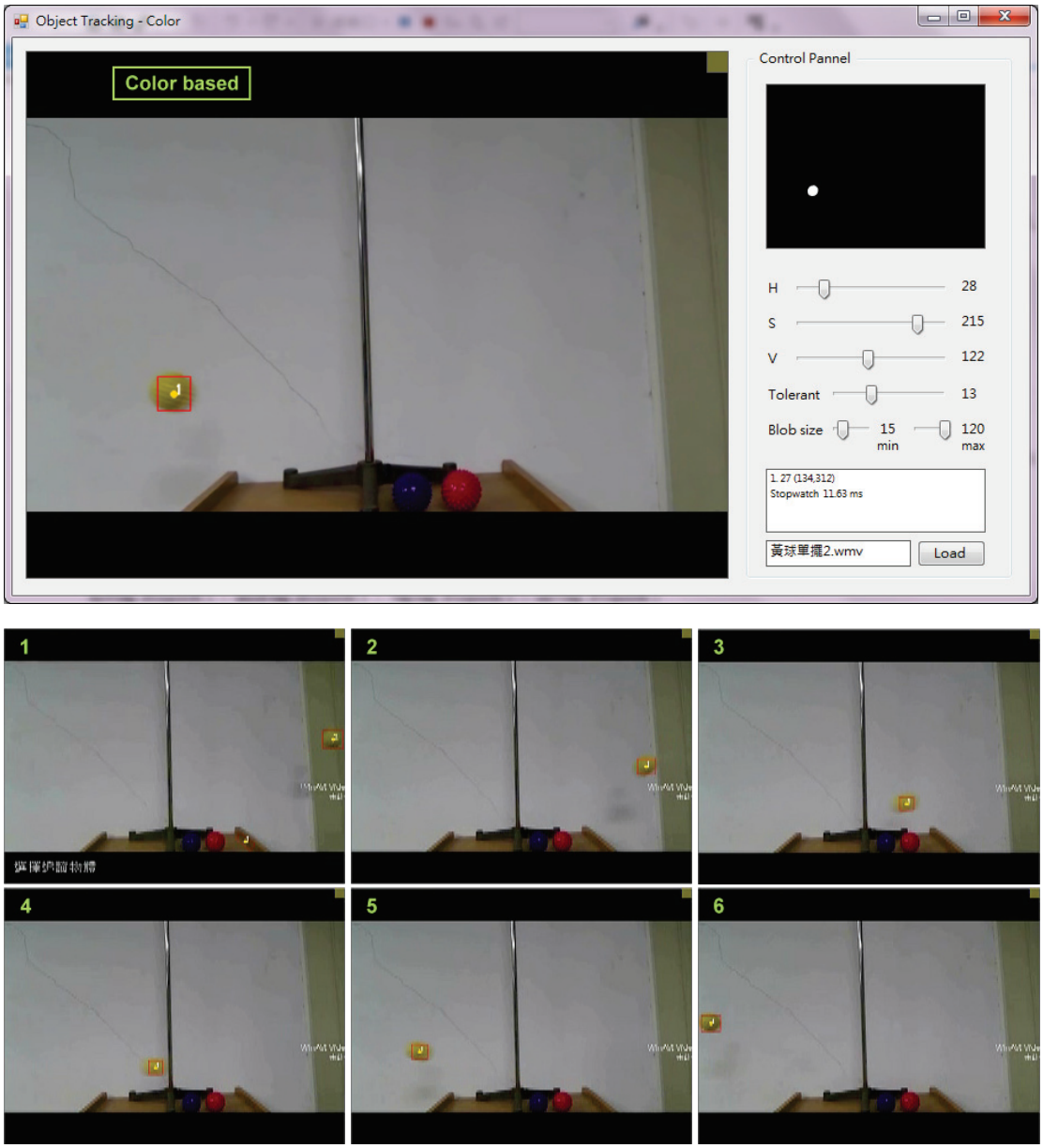


圖 10 利用顏色偵測每幅畫面中的物體感覺就像物體追蹤

五、微軟的新產品 Kinect

Kinect 是由微軟開發，應用於 Xbox 主機的周邊設備，2010 年 11 月 4 日於美國上市，台灣則在 2010 年 11 月 20 日販售，它讓玩家不再需控制器，而是改用語音指令及手勢來當作系統介面。它能夠捕捉玩家全身上下的動作，用身體來進行遊戲，帶給玩家「免控制器的遊戲與娛樂體驗」。是微軟研究院近期重要的研究成果之一[1][2][6]。

圖 11 Microsoft Kinect 硬體結構

Kinect 有三個鏡頭，中間的鏡頭是 RGB 彩色攝影機，用來拍攝彩色視訊。左右兩邊鏡頭則分別為紅外線發射器和紅外線 CMOS 攝影機所構成的 3D 結構光深度感應器，用來擷取深度數據（物體到攝影機的距離）。彩色攝影機的最高解析度為 1280×960，紅外線攝影機的最高解析度則為 640×480。Kinect 也內建陣列式麥克風（Microphone Array），由四個麥克風同時收音，比對後消除雜音，並透過其採集聲音，進行語音識別和聲源定位。

Kinect 上市後造成了很大的轟動，也給競爭對手任天堂（Nintendo）及新力（Sony）形成了很大的壓力。本來微軟只提供協力開發廠商開發工具，用以開發專屬於 Xbox 系統的應用，但在網路上眾多軟體愛好者（包括許多駭客）共同努力下，網路上逐漸出現能夠解譯來自 Kinect 與 Xbox 間的免費驅動軟體，爾後利用 Kinect 來開發的相關應用程式如雨後春筍般的蓬勃發展。微軟認清了這個事實，2011 年 6 月也釋出了供開發、測試、教學用途的 Kinect for Windows SDK，使得 Kinect 的用途可以更為廣範。

六、加入深度（3D）資訊的物體追蹤

1. Kinect 的初步體驗

項 目	用途說明	命名空間	相關類別
Windows Forms	微軟在.NET 平台上最基礎與常見的圖形用戶介面。	System.Drawing	Bitmap Image Graphics
WPF	微軟.NET 平台 3.0 及以後版本的新一代使用者介面技術，支援視訊加速及向量繪圖。	System.Windows.Media.Imaging	BitmapSource BitmapImage WriteableBitmap
Kinect	微軟 Xbox 體感遊戲的使用者介面，目前釋出的開發工具為 Kinect for Windows SDK 1.8。	Microsoft.Kinect	ColorImageFrame DepthImageFrame SkeletonFrame
EmguCV	由 Intel 發起與開發的跨平台電腦視覺庫 OpenCV 的.NET 版本。	Emgu.CV	Image<TColor, TDepth> IImage

表格 2 與本專題開發有關的各類型函式庫

依照微軟的建議[6]，開發 Kinect 相關的應用程式最好在 WPF 的平台上，因為 WPF 平台可以加速視訊的處理。我們整合了參考資料[1][2]中的 C#範例程式，以 VB.Net 撰寫了可同時顯示彩色攝影機與紅外線攝影機傳送過來的串流程

式，執行畫面如圖 12 所示。

另外，我們在不同平台間影像資料轉換時，發生了不少的困擾，似乎沒有一種通用的影像資料型態可以所有的平台流通，都必須花費額外的時間做轉換（表格 2）。例如由 Kinect 傳送過來的 ColorImageFram，必須先使用 CopyPixelDataTo 指令將其拷貝至陣列中，然後再用 WritePixels 指令由陣列寫入 WriteableBitmap，如此才能由 WPF 中的 Image Box 顯示。

圖 12 可同時顯示彩色及深度視訊的程式執行畫面

2. 彩色與深度視訊的校正重合

仔細觀察彩色視訊及深度視訊的畫面，似乎有位移並沒有完全地重合，像是圖 13 中黃色及青色箭頭指示處的 Kinect 包裝盒及電腦椅，我們先將深度影像做雙色處理，再設定成半透明後與彩色影像重疊，這樣可以更清楚的印證兩張影像並沒有重合。

圖 13 彩色與深度視訊畫面的比較

若是能夠將深度影像與彩色影像重合，那麼 Kinect 的功能將可大幅提升，除了一般攝影機有的 2D 資訊外，更多了 3D 的深度訊息。是否可以利用轉換函數，將深度影像校正，與彩色影像重合呢？

尋找兩組資料間的最佳轉換方法為多項式迴歸分析法，設(x,y)為彩色視訊畫面中某一像素的座標，而(X,Y)為該像素在深度視訊畫面中相對應位置的座標，多項式迴歸分析法就是在尋找兩個座標間的最佳轉換函數，以二階多項式為例：

$$x = f_x(X,Y) = a_0 + a_1X + a_2Y + a_3X^2 + a_4XY + a_5Y^2$$
$$y = f_y(X,Y) = b_0 + b_1X + b_2Y + b_3X^2 + b_4XY + b_5Y^2$$

可以使用矩陣的形態來表示

$$\begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} 1 & X & Y & X^2 & XY & Y^2 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ \vdots & \vdots \\ a_5 & b_5 \end{bmatrix}$$

當我們尋找到夠多的對應點時，將彩色視訊畫面中的像素座標建立成矩陣T，而將深度視訊中對應點的座標建成矩陣S，再利用多項式迴歸分析法求得最佳轉換函數之係數所構成的矩陣M。

$$T = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}_{n\times 2}$$

$$S = \begin{bmatrix} 1 & X_1 & Y_1 & X_1^2 & X_1Y_1 & Y_1^2 \\ 1 & X_2 & Y_2 & X_2^2 & X_2Y_2 & Y_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_n & Y_n & X_n^2 & X_nY_n & Y_n^2 \end{bmatrix}_{n\times 6}$$

$$M = \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ \vdots & \vdots \\ a_5 & b_5 \end{bmatrix}_{6\times 3}$$

$$T \approx SM$$

$$M = \left(S^T S\right)^{-1} S^T T$$

圖 14 彩色與深度視訊畫面上選定的對應點

座標＼編號	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	107	152	167	250	256	286	311	322	405	438	493	554	561	122	461
y	252	244	207	318	186	245	172	215	246	212	242	254	318	71	92
X	71	114	152	233	234	258	310	324	404	456	516	584	582	100	486
Y	257	248	207	332	184	250	168	216	253	213	247	263	331	89	83

表格 3 對應點的座標、(*x*,*y*)為彩色影像座標、(X,Y)為深度影像座標

我們在彩色與深度視訊畫面上選定了 15 個對應點，分別記錄下每個對應點的座標，經過 Matlab 軟體的實作，所求得的 2 階最佳校正矩陣為

$$M = \begin{bmatrix} -7.8915816 & 30.9455239 \\ 0.8337053 & -0.0139751 \\ 0.4436073 & 0.8396435 \\ -0.0001528 & 0.0000007 \\ 0.0005198 & 0.0000393 \\ -0.0010228 & 0.0000803 \end{bmatrix}_{6\times 2}$$

3. 修正後的程式

經過校正函數校正彩色視訊後的程式如圖 15-1 所示，最左邊為原始的彩色視訊畫面，最右邊為校正後的結果，我們將其重疊起來，由圖 15-2 看來，校正的效果還不錯。

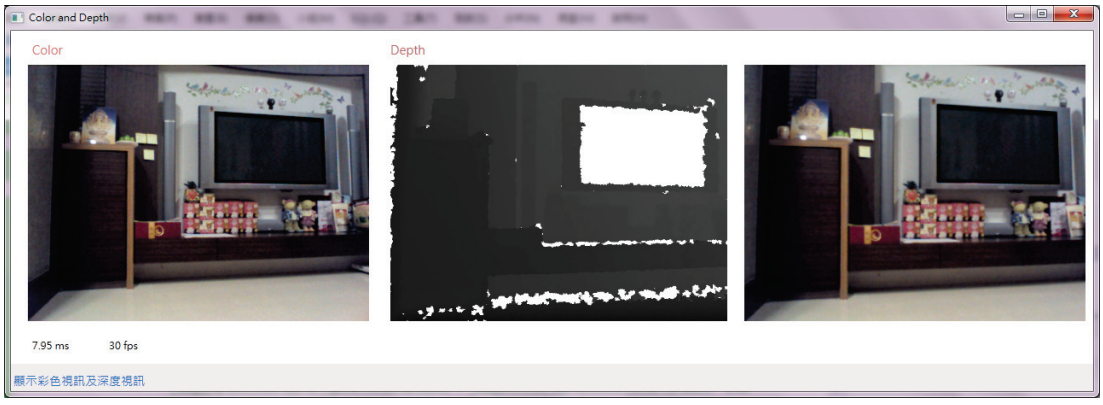


圖 15-1 最右邊為校正後的彩色視訊畫面

圖 15-2 校正後的彩色與深度視訊畫面的比較

伍、研究結果：

一、2D 視訊中的物體追蹤

採用 VB.Net 與 Windows Forms 視窗程式的架構，搭配一般的視訊攝影機，來開發以色彩為基礎的物體追蹤程式。圖 16 中，我們以黃色彩球單擺為例，除了追蹤黃球目前的位置外，並以其座標變化繪出其偏離垂直軸距離及時間的圖形，以高中物理課程的單擺運動來看，完全符合理論所述。

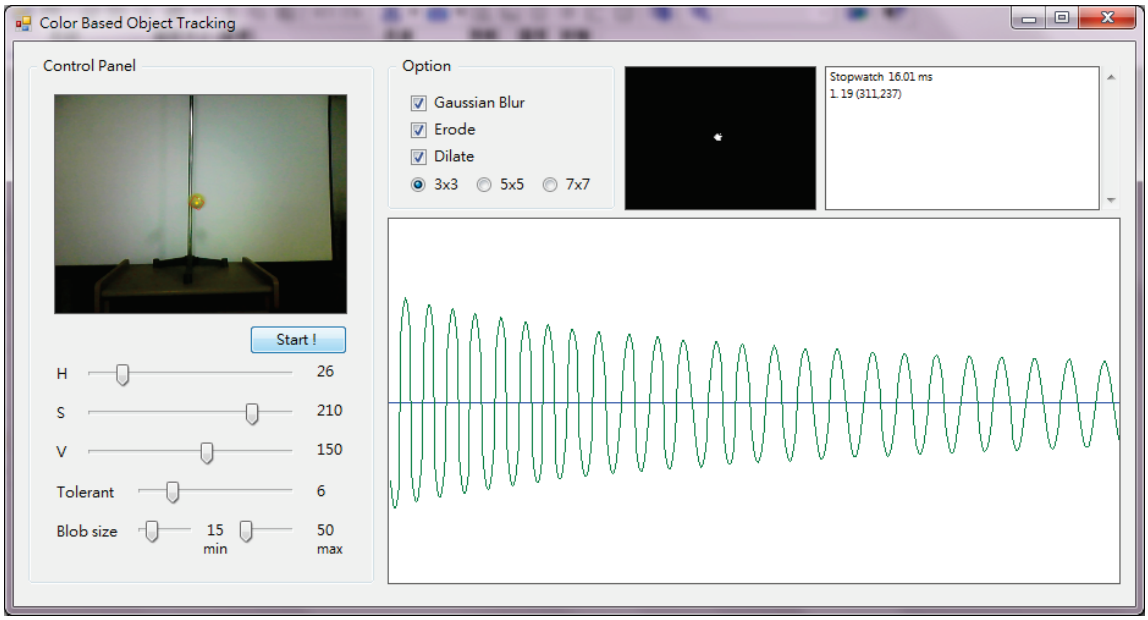


圖 16 2D 視訊攝影機的物體追蹤

二、3D 視訊中的物體追蹤

Kinect 提供了物體在三度空間中的座標，除了色彩外還提供了深度資訊，使得在偵測物體時更為穩健，較不受畫面中相似色彩的干擾，由於 Kinect 有彩色及紅外線攝影機，所以兩者所拍攝的畫面並沒有重合，在使用深度資訊時，必須先加以校正。

圖 17 為以 WPF 平台與 VB.Net 所開發的物體追蹤程式，加上深度資訊後可以很明確的偵測出黃色彩球的位置，因為需處理彩色及深度視訊串流，所以每一畫面約需處理 25ms，雖然多花了些時間，但還是符合即時追蹤的要求。

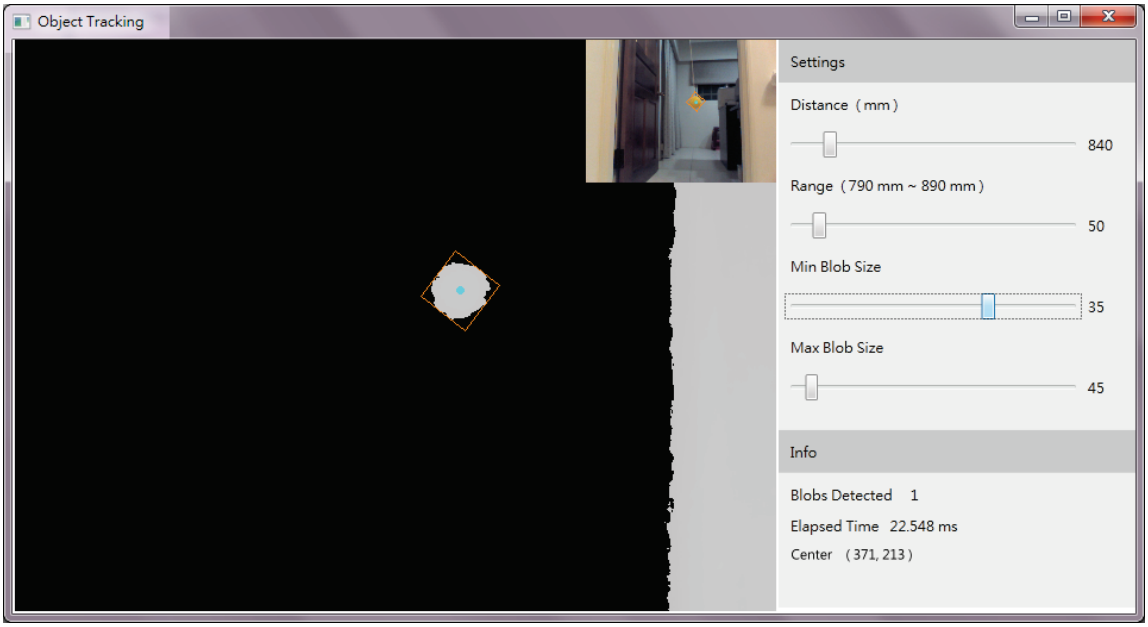


圖 17 Microsoft Kinect 的物體追蹤

陸、討論及結論：

物體追蹤可以應用的領域非常的廣泛，例如蝴蝶飛舞的軌跡、模型飛機的飛行路線、物體飛行軌跡、小昆蟲的移動路徑、監控入侵之人或物等，若是運用到中小學的實驗室，則可以做運動學物理量的即時測量與顯示，符合眼見為憑的實驗精神。

相對於以攝影機將物體的運動過程攝錄下來，然後利用視訊處理軟體（如威力導演、繪聲繪影等）將視訊中的每格畫面分別存成影像檔，最後再進行逐格分析的方法，物體追蹤程式可以節省非常多的時間與人力。

微軟的體感遊戲機Kinect，由於其價格便宜，功能強大，若僅限於遊戲操作實在有點大材小用，若能經過精心的校正彩色攝影機與紅外線攝影機的畫面，使其對應點能夠重合的話，那麼Kinect就會變成一台3D攝影機，追蹤三度空間中物體的運動軌跡，將是可行的，期盼在未來的研究中能夠繼續往這個方向發展，試想，若是能夠繪出蝴蝶在三度空間中曼妙的飛行舞姿，那是多麼美麗的軌跡啊！

柒、參考資料：

一、余濤著, “Kinect 應用開發實戰,” 上奇資訊, 2013。

二、劉超群著, “Kinect 體感程式探索－使用 C#,” 松崗資產管理, 2013。

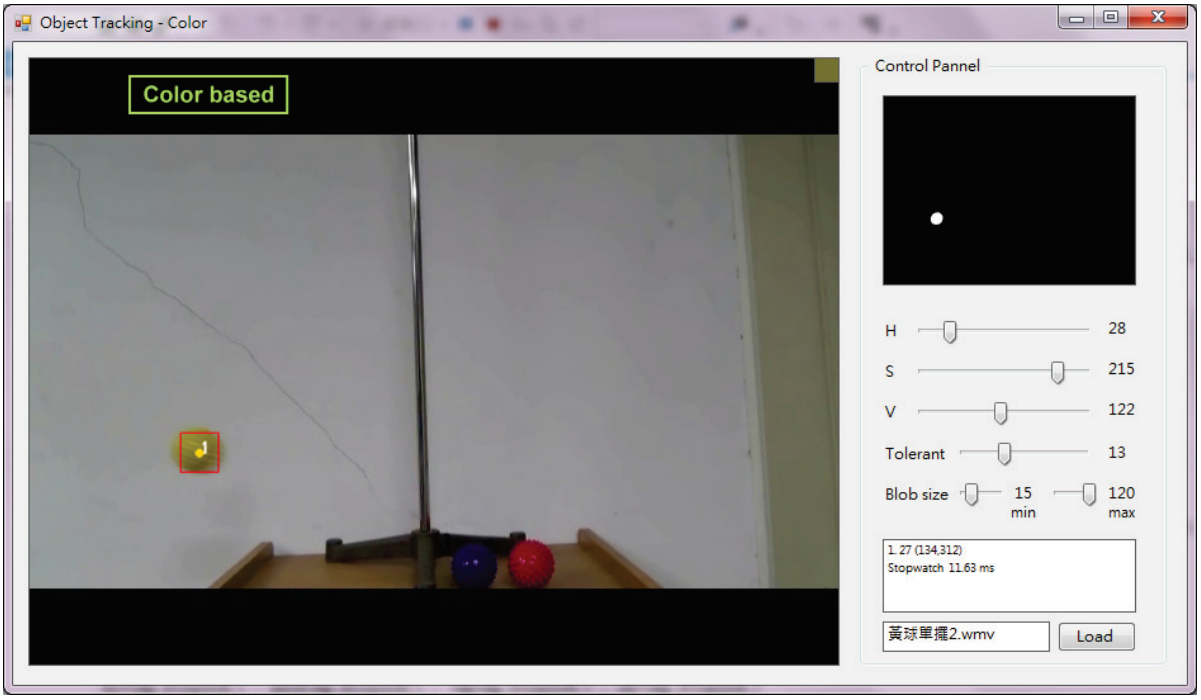
三、繆紹綱譯, “數位影像處理,” 培生出版社, 2009。

四、胡國瑞、孫沛立、徐道義、陳鴻興、黃日鋒、詹文鑫、羅梅君著, “顯示色彩工程學,” 全華圖書, 2009。

五、資訊教育研究室著, “Visual Basic 2012 從零開始,” 博碩文化, 2013。

六、http://www.microsoft.com/en-us/kinectforwindows/, Kinect for Windows | Voice, Movement & Gesture Recognition Technology。
七、http://www.emgu.com/wiki/index.php/Main_Page, Emgu CV : OpenCV in .NET。
八、Alper Yilmaz, Omar Javed, Mubarak Shah, “Object Tracking : A Survey,” ACM Journal of Computing Surveys, Vol. 38, No. 4, Article No. 13。
九、http://yy-programer.blogspot.tw/, yy’s Program。
十、http://blog.zack.tw/, 查克的部落格。

捌、附錄：2D 視訊中的物體追蹤程式




```
Imports System.Drawing
Imports System.Drawing.Imaging
Imports Emgu.CV
Imports Emgu.CV.CvEnum
Imports Emgu.CV.Structure
Imports System.Runtime.InteropServices


Public Class Form1

    Const maxValue = 255
    Const minValue = 0
    Const maskSize = 3 'Gaussian Blur

    Dim _Capture As Capture
    Dim captureInProgress As Boolean
    Dim ImageFrame, outImg As Image(Of Bgr, Byte)
    Dim hsvImg As Image(Of Hsv, Byte)
```

```
Dim contourImg As Image(Of Gray, Byte)
Dim minScalar, maxScalar As New MCvScalar()
Dim _font As New MCvFont(Emgu.CV.CvEnum.FONT.CV_FONT_HERSHEY_DUPLEX, 0.45, 0.45)
Dim stopColorWatch As New Stopwatch()

'------------------------------------------------------------------------------

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
    picSrc.Image = Nothing
    picMask.Image = Nothing
    txtInfo.Text = ""
    captureInProgress = False
    AddHandler Application.Idle, New EventHandler(AddressOf ProcessFrame)
End Sub

Private Sub ProcessFrame(ByVal sender As Object, ByVal e As EventArgs)
    If captureInProgress Then
        ImageFrame = _Capture.QueryFrame()
        'ImageFrame = ImageFrame.Resize(picSrc.Width, picSrc.Height,
            Emgu.CV.CvEnum.INTER.CV_INTER_LINEAR) '8ms
        picSrc.Image = ImageFrame
        ImageProcessing()
    End If
End Sub

'------------------------------------------------------------------------------

Private Sub ImageProcessing()
    stopColorWatch.Reset()
    stopColorWatch.Start()
    Dim maskImg, tmpImg As New Image(Of Gray, Byte)(ImageFrame.Width, ImageFrame.Height)
    If hsvImg IsNot Nothing Then hsvImg.Dispose()
    hsvImg = ImageFrame.SmoothGaussian(maskSize).Convert(Of Hsv, Byte)() '模糊
    minScalar.v1 = Math.Max(minValue, CInt(tbS.Value - 8 * tbTolerant.Value))
    maxScalar.v1 = Math.Min(maxValue, CInt(tbS.Value + 8 * tbTolerant.Value))
    minScalar.v2 = Math.Max(minValue, CInt(tbV.Value - 6 * tbTolerant.Value))
    maxScalar.v2 = Math.Min(maxValue, CInt(tbV.Value + 6 * tbTolerant.Value))
    If (CInt(tbH.Value) >= CInt(tbTolerant.Value)) And (CInt(tbH.Value) <= CInt(tbH.Maximum - tbTolerant.Value))
    Then
        minScalar.v0 = CInt(tbH.Value - tbTolerant.Value)
        maxScalar.v0 = CInt(tbH.Value + tbTolerant.Value)
        CvInvoke.cvInRangeS(hsvImg, minScalar, maxScalar, maskImg) '篩選
    Else
        If CInt(tbH.Value) < CInt(tbTolerant.Value) Then '0~Tolerant-1
            minScalar.v0 = tbH.Minimum
            maxScalar.v0 = CInt(tbH.Value + tbTolerant.Value)
            CvInvoke.cvInRangeS(hsvImg, minScalar, maxScalar, maskImg)
            minScalar.v0 = CInt(tbH.Value - tbTolerant.Value + 1 + tbH.Maximum)
            maxScalar.v0 = tbH.Maximum
            CvInvoke.cvInRangeS(hsvImg, minScalar, maxScalar, tmpImg)
            CvInvoke.cvOr(maskImg, tmpImg, maskImg, System.IntPtr.Zero)
        Else '179-Tolerant+1~179
            minScalar.v0 = CInt(tbH.Value - tbTolerant.Value)
            maxScalar.v0 = tbH.Maximum
            CvInvoke.cvInRangeS(hsvImg, minScalar, maxScalar, maskImg)
            minScalar.v0 = tbH.Minimum
            maxScalar.v0 = CInt(tbH.Value + tbTolerant.Value - tbH.Maximum - 1)
            CvInvoke.cvInRangeS(hsvImg, minScalar, maxScalar, tmpImg)
            CvInvoke.cvOr(maskImg, tmpImg, maskImg, System.IntPtr.Zero)
        End If
    End If
```

```
        Dim SElement As New StructuringElementEx(maskSize, maskSize, CInt(maskSize / 2), CInt(maskSize / 2),
            Emgu.CV.CvEnum.CV_ELEMENT_SHAPE.CV_SHAPE_RECT)
        CvInvoke.cvErode(maskImg, maskImg, SElement, 1) '侵蝕
        CvInvoke.cvDilate(maskImg, maskImg, SElement, 1) '擴張
        Dim areaIndex As Integer = 0
        contourImg = New Image(Of Gray, Byte)(maskImg.Width, maskImg.Height)
        CvInvoke.cvCopy(maskImg, contourImg, System.IntPtr.Zero)
        outImg = New Image(Of Bgr, Byte)(ImageFrame.Width, ImageFrame.Height)
        CvInvoke.cvCopy(ImageFrame, outImg, System.IntPtr.Zero)
        txtInfo.Text = ""
        Using storage As New MemStorage()
            Dim contours As Contour(Of System.Drawing.Point) =
                contourImg.FindContours(Emgu.CV.CvEnum.CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_SIMP
                LE, Emgu.CV.CvEnum.RETR_TYPE.CV_RETR_EXTERNAL, storage)
            While contours IsNot Nothing
                If (contours.Area > Math.Pow(tbMinSize.Value, 2)) AndAlso (contours.Area < Math.Pow(tbMaxSize.Value,
                    2)) Then
                    areaIndex += 1
                    Dim box As MCvBox2D = contours.GetMinAreaRect()
                    outImg.Draw(box, New Bgr(System.Drawing.Color.Red), 1)
                    'maskImg.Draw(box, New Gray(128), 1)
                    Dim blobCenter As PointF = box.center
                    Dim cr As New Emgu.CV.Structure.CircleF(blobCenter, 2)
                    outImg.Draw(cr, New Bgr(System.Drawing.Color.Gold), 2)
                    'maskImg.Draw(cr, New Gray(160), 2)
                    outImg.Draw(areaIndex, _font, Point.Round(box.center), New Bgr(255, 255, 255))
                    txtInfo.Text &= areaIndex & ". " & Math.Round(Math.Sqrt(contours.Area), 0) & " (" &
                        Math.Round(blobCenter.X, 0) & "," & Math.Round(blobCenter.Y, 0) & ")" & Chr(13) & Chr(10)
                End If
                contours = contours.HNext
            End While
        End Using
        picMask.Image = maskImg
        picSrc.Image = outImg
        maskImg.Dispose() : tmpImg.Dispose() : outImg.Dispose()
        stopColorWatch.Stop()
        txtInfo.Text &= "Stopwatch    " & String.Format("{0:n2} ms", stopColorWatch.Elapsed.TotalMilliseconds)
    End Sub

    '-------------------------------------------------------------------------

    Private Sub picSrc_MouseDown(sender As Object, e As MouseEventArgs) Handles picSrc.MouseDown
        Dim p As Point
        Dim tmpX, tmpY, imgWidth, imgHeight, boxWidth, boxHeight As Integer
        Dim hue, satuation, value, _R, _G, _B, tmpMax, tmpMin As Integer
        Dim tmpR, tmpG, tmpB As Double
        If picSrc.Image IsNot Nothing Then
            p.X = e.X : p.Y = e.Y '取得滑鼠敲擊點
            boxWidth = picSrc.Width : boxHeight = picSrc.Height
            imgWidth = ImageFrame.Width : imgHeight = ImageFrame.Height
            tmpX = CInt(Math.Round(p.X / boxWidth * imgWidth)) : tmpY = CInt(Math.Round(p.Y / boxHeight *
                imgHeight))
            tmpX = Math.Min(imgWidth - 1, tmpX) : tmpY = Math.Min(imgHeight - 1, tmpY)
            _B = ImageFrame(tmpY, tmpX).Blue
            _G = ImageFrame(tmpY, tmpX).Green
            _R = ImageFrame(tmpY, tmpX).Red
            picColor.BackColor = Color.FromArgb(_R, _G, _B)
            tmpMax = Math.Max(_R, Math.Max(_G, _B))
            tmpMin = Math.Min(_R, Math.Min(_G, _B))
            tmpR = (tmpMax - _R) / (tmpMax - tmpMin + 0.1)
            tmpG = (tmpMax - _G) / (tmpMax - tmpMin + 0.1)
```

```
            tmpB = (tmpMax - _B) / (tmpMax - tmpMin + 0.1)
            value = tmpMax
            If tmpMax = 0 Then
                satuation = 0
                hue = 90
            Else
                satuation = CInt(maxValue * (tmpMax - tmpMin) / tmpMax)
                If _R = tmpMax Then hue = CInt(60 * (tmpB - tmpG))
                If _G = tmpMax Then hue = CInt(60 * (2 + tmpR - tmpB))
                If _B = tmpMax Then hue = CInt(60 * (4 + tmpG - tmpR))
                If hue >= 360 Then hue = hue - 360
                If hue < 0 Then hue = hue + 360
            End If
            hue = CInt(hue / 2)
            tbH.Value = hue : lblH.Text = tbH.Value
            tbS.Value = satuation : lblS.Text = tbS.Value
            tbV.Value = value : lblV.Text = tbV.Value
        End If
End Sub

'------------------------------------------------------------------------------

Private Sub btnSwitch_Click(sender As Object, e As EventArgs) Handles btnSwitch.Click
        If btnSwitch.Text = "Start" Then
            If _Capture Is Nothing Then
                Try
                    _Capture = New Capture()
                Catch excpt As NullReferenceException
                    txtInfo.Text = excpt.Message
                End Try
            End If
            If _Capture IsNot Nothing Then
                captureInProgress = True
                btnSwitch.Text = "Stop"
            End If
        Else 'Stop
            If _Capture IsNot Nothing Then
                captureInProgress = False
                btnSwitch.Text = "Start"
            End If
        End If
End Sub

Private Sub tbH_Scroll(sender As Object, e As EventArgs) Handles tbH.Scroll
        lblH.Text = tbH.Value
End Sub

Private Sub tbS_Scroll(sender As Object, e As EventArgs) Handles tbS.Scroll
        lblS.Text = tbS.Value
End Sub

Private Sub tbV_Scroll(sender As Object, e As EventArgs) Handles tbV.Scroll
        lblV.Text = tbV.Value
End Sub

Private Sub tbTolerant_Scroll(sender As Object, e As EventArgs) Handles tbTolerant.Scroll
        lblTolerant.Text = tbTolerant.Value
End Sub

Private Sub tbSize_Scroll(sender As Object, e As EventArgs) Handles tbMinSize.Scroll, tbMaxSize.Scroll
        lblMinSize.Text = tbMinSize.Value
```

```
            lblMaxSize.Text = tbMaxSize.Value
        End Sub

End Class
```

【評語】040803

本作品可應用於記錄運動物理之時間及位置。待擴充的功能為速度的限制及物體的辨認，還有視角、距離的問題。完成之後未來性非常看好。