報告題名：

即時人臉影像偵測系統

Real Time Face Detection System

作者：謝明娟、連珍慧

系級：通訊系 四年甲班

學號：D9156706

開課老師：林立謙 教授

課程名稱：專題研究

開課系所：通訊工程學系 多媒體通訊學程

開課學年：94 學年度 第 1 學期

摘要

本專題研究報告利用人臉邊緣和膚色特徵，找出人臉的位置。

利用 Canny edge detection 找出物體的邊緣，找到一個較正確的人臉邊界。並用 HSI color information detection 的方法找出膚色訊息，為了是判別人臉部分。再結合兩者的結果以找出整張影像中人臉位置。

在不同的環境因素條件下，對其實驗結果做比較。討論其原因所在，並提出設定環境初始值以改善膚色偵測上不足和環境因素的影響，進而提升的整個系統的偵測準確率。

關鍵字：人臉偵測、人臉追蹤、人臉辨識、膚色偵測

目　錄

圖　目　錄

表　目　錄

第一章　前言

1.1 研究動機

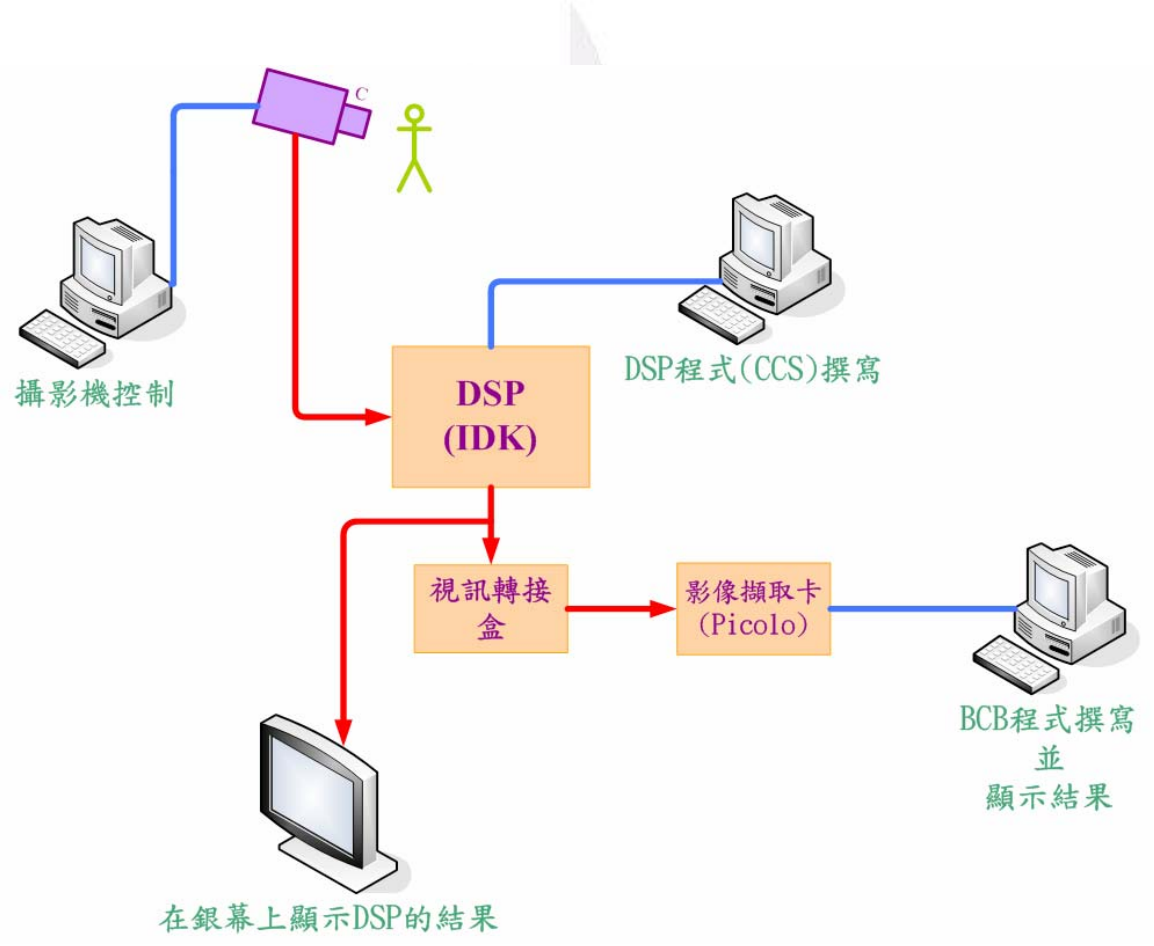
人臉辨識在監視系統中佔著重要的一環，關係著系統的精確度。越好的人臉辨識讓監視系統的效能更好，於是人臉辨識在監視系統為一個關鍵技術。

科技發展讓人類生活趨於便利。將繁重單調的工作交給電腦處理，能把更多的精神放在更有意義的地方。對於人臉辨識系統也是如此，將人臉偵測出來，去除累贅資訊留下對於系統有意義的部份，可以縮短處理速度達到高效能的最終目的。而人臉偵測的應用非常廣泛，包括人臉即時追蹤、人臉註冊、特徵比對...等。基於一項技術能廣泛應用的前提之下，啟發我們的研究興趣。

對於人臉偵測，膚色以及人臉形狀為關鍵之一。我們透過實驗室的設備與演算法結合，得到人臉偵測的實驗效果。

1.2 系統架構

本文所使用的演算法都在個人電腦上模擬，若軟體模擬無誤，便將演算法的程式寫入 DSP 晶片中。所以系統架構如圖 1.1 所示，攝影機將畫面轉為數位訊號傳到 DSP，此時有兩個途徑：一、經過 DSP 運算過後將顯示在銀幕上。二、將原始訊號經由影像擷取卡把訊號傳到電腦上，在電腦上以軟體模擬演算法的結果。



▲圖 1.1 系統架構圖

1.3 實驗設備

(1) DSP：在軟體模擬時，因為即時影像資料量很大，而 DSP 晶片可以加快演算處理，避免動態影像延遲。如圖 1.3。

(2) 擷取卡：擷取卡用途是將攝影機的影像資料輸入到電腦，供軟體模擬使用。

(3) 高畫素攝影機：取得即時影像訊號，給 DSP 與擷取卡做即時處理。如圖 1.2。

▲圖 1.2 攝影機

▲圖 1.3 DSP 子母卡

第二章　人臉辨識之原理與簡介

2.1 簡介

一張影像以人眼來看，便可以很快的分辨是誰，年紀多大、是哪一國人。但對電腦而言，影像只是一堆數值，並無任何意義。於是人要對這些影像的資料加以定義，讓它對電腦而言是有意義的。這些定義可能對人來說是很理所當然的，平常到不會去發覺的細節。這些細節包括五官的個別位置、人臉邊界 ...等。在本文中人臉辨識著重的是邊界與膚色，透過影像處理的理論，可以使用演算法來找尋邊界以及定義膚色，來達到人臉辨識的第一步驟。

2.2 人臉辨識方法

人臉辨識是指圖像中，辨識人臉的真實身分。一般而言，目前做人臉辨識的方法大致可分為下兩種：Feature-based 和 Color-based。

(1) Feature-based

利用人臉部的一些特徵來偵測人臉，例如：人臉上有眼睛、鼻子、嘴巴，而這些器官之間的幾何距離；就整張臉型來說，約略呈橢圓形，且臉和背景之間大致會有邊緣線等，有許許多多的特徵可利用。

(2) Color-based

利用人的臉部顏色來判斷某影像是否為臉；若影像是彩色的，可用 RGB、HSI、YCbCr 色彩系統來統計膚色數值，以作為辨別的依據；若影像為灰階的，膚色部分的灰階值與眼睛、嘴巴、頭髮等部位的灰階值相較之下較小。[1]

以不同角度選擇圖像可得許多分類法，演算法必須隨著不同圖像分類而調整、修正，因此圖像的選擇即為一重要課題(如表 2.1)。本文仿照表 2.1 列出表 2.2 之圖像條件探討實驗。

在 Feature-based 部分找出物件的邊緣；而 Color-based 方面對膚色作運算，以當作辨識人臉的依據。

▼表 2.1 人臉辨識的問題分類

分類依據	類　　別	
圖像來源	靜止圖像(包括如數位化的照片、數碼相機拍攝的圖片等，目前考慮的主要問題是演算法的適應性，演算法速度在其次)	動態圖像(即時監控攝影、影視資料等，為考慮即時處理，因此對演算法的速度有很高的要求)
顏色資訊	彩色(RGB、HSI、YCbCr 系統)	灰階度
鏡頭類型	頭肩部圖像(可避免與身體部位的膚色混淆)	半身/全身圖像(臉在圖像中所佔的比例減少，則可採信之臉部特徵資訊亦減少許多)
人臉姿態	正面(包括端正及平面內旋轉)	側面(包括俯仰、側影及旋轉)
人臉數目	單人(又可以稱為人臉定位，是人臉辨識問題在已知人臉數目情況下的特例)	未知(需要判定圖像中是否存在人臉、人臉的數目以及各個人臉的尺度和位置，即是完全的檢測問題)
圖像背景	簡單背景(背景的特徵被嚴格約束，在該條件下只利用人臉的輪廓、顏色、運動等少量特徵，就能夠進行準確檢測)	複雜背景(指背景的類型和特徵不受約束，某些區域可能在色彩、紋理等特徵上與人臉相似，必須利用較多的人臉特徵才能做到準確檢測)

▼表 2.2 圖像條件

分類依據	類　別　與　簡　述	
圖像來源	靜止圖像(動態圖像所擷取之單張圖片，用來計算膚色之 HSI 值)	動態圖像(即時動態攝影，演算法的速度將影響是否能夠達成即時影像處理)
顏色資訊	彩色(圖像的 RGB 與 HSI 資料為本實驗重要運算資訊)	
鏡頭類型	頭肩部圖像	
人臉姿態	正面(包括端正及平面內旋轉)、側面(包括俯仰、側影及旋轉)	
人臉數目	人臉越少，效果越好(邊緣偵測部分之故)	
圖像背景	簡單背景(無雜物，避免影響邊緣偵測效果)	

第三章　邊緣偵測

3.1 邊緣偵測的方法

邊緣偵測是為了尋找影像中物體輪廓的一個過程。當影像以灰階度表示時，物件邊緣像素與其臨近像素間的灰階值變化較大。而灰階變化度可視為灰階之梯度，當灰階梯度值越大表示灰階變化度越大，因此可用演算法突顯出物件邊緣。物件邊緣在灰階圖上，會有幾種常見的邊緣：灰階值變化較慢的斜面邊緣(Ramp edge)與峰頂邊緣(Roof edge)、灰階值變化突然的步階邊緣(step edge)與直排邊緣(line edge)，如圖 3.1。

在邊緣偵測中可以用一階導數(First-order derivative)來求出灰階梯度值，又可稱為梯度運算子(Gradient Operator)，如(3-1)式。(3-2)式表梯度的強度。實際上為了運算便利性，在求梯度強度時通常是用(3-3)式或(3-4)式。[7]

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad \text{.....(3-1)}$$

$$|G[f(x,y)]| = \sqrt{G_x^2 + G_y^2} \quad \text{.....(3-2)}$$

$$|G[f(x,y)]| \approx |G_x| + |G_y| \quad \text{.....(3-3)}$$

$$|G[f(x,y)]| \approx \max(|G_x|, |G_y|) \quad \text{.....(3-4)}$$

一般來說邊緣偵測都會分為 X 方向與 Y 方向做個別處理，且會利用遮罩(mask)方式與圖像灰階陣列作旋積。如法線方向(3-5)與水平方向(3-6)，但通常會在相反方向作平滑處理(3-7)。將兩個步驟合併起來即 Prewitt 遮罩(3-8)。但 Prewitt 遮罩過於平滑會使得部份邊緣模糊掉，為改善此問題而衍生 Sobel 遮罩(3-9)。另外還有一種叫作 Roberts cross-gradient 遮罩對斜線方向作偵測(3-10)。以上遮罩都屬於一階導數的應用，其共同缺點為對雜訊敏感。

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{.....(3-5)}$$

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \text{.....(3-6)}$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \text{和} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad \text{.....(3-7)}$$

Prewit 遮罩：$P_X=\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ 和 $P_y=\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ .....(3-8)

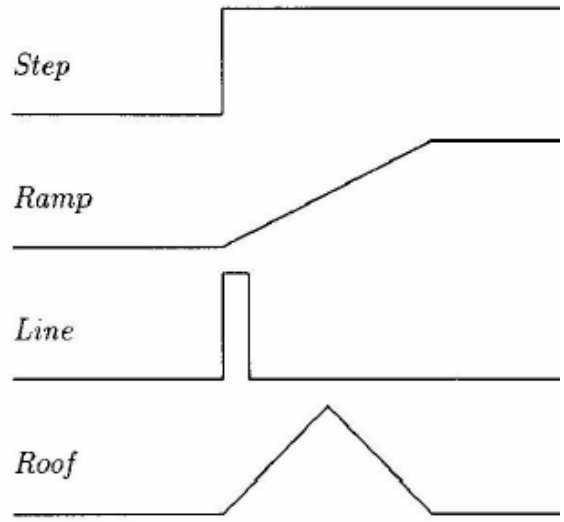
Sobel 遮罩：$S_X=\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ 和 $S_y=\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ .....(3-9)

Roberts cross-gradient 遮罩：$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ 和 $\begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ .....(3-10)

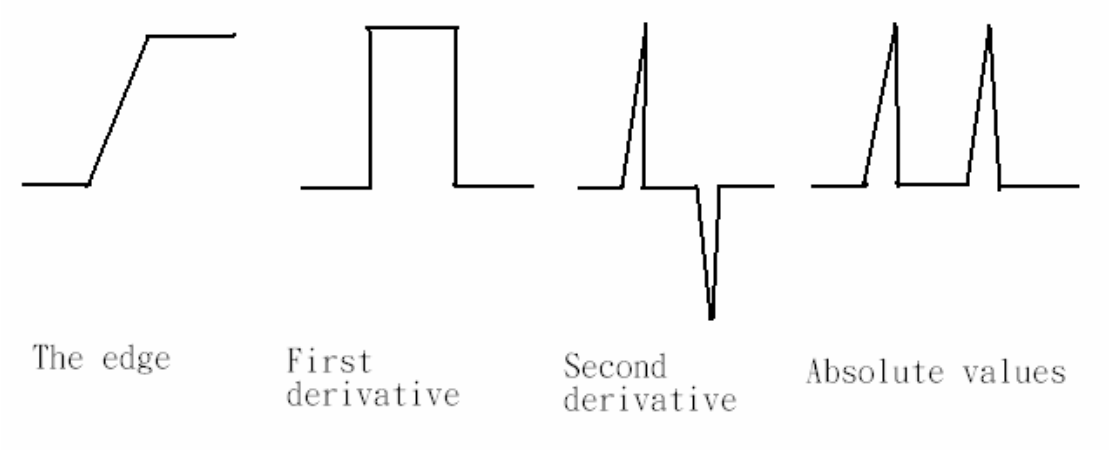
第二種方式是經過二階導數(Second-order derivative)運算灰階值後，由正數轉為負數或負數轉為正數的越零點(Zero Crossings)在圖像的物件上為凸點或凹點，連結後即可得到約略的邊緣。又可稱為拉普拉斯運算子(Laplacian opterator)[2]，常用的遮罩如(3-11)式[6]。一階導數與二階導數比起來，一階導數對於偵測的效果較好；二階導數對於將邊緣細化的部份作的較好。

但二階導數的缺點是對於雜訊的干擾很敏感。而經由視訊所得到的影像，常常會受到雜訊的干擾，使得物件的邊緣被切斷、將非邊緣的地方偵測出來或是偵測出來的邊緣變形。這些問題是以上邊緣偵測法所難以解決的，然而 Canny edge detection 卻可以解決這個問題。

Laplacian 遮罩：$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ 或 $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ 或 $\begin{bmatrix} -2 & 1 & -2 \\ 1 & -4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$ .....(3-11)



▲圖 3.1 灰階影像邊緣示意圖



▲圖 3.2 邊緣經過一階到二階的過程

3.2 Canny edge detection 演算法[3][6][7]

Step1 Smoothing

將整張影像 I(i,j)先經過高斯平滑濾波器 G(i,j)(3-12)式，然後再分別對 x 方向與 y 方向計算其 Laplacian (3-13)式~(3-14)式。意思同於對影像作線性 Laplacian of Gaussian 濾波器。實作上即利用圖 3.3 (a)遮罩對影像作旋積(convolution)可以原始圖像看起來更為平滑，避免在對圖像作運算時，少數極端的灰階值影響偵測結果，如：雜訊、歧異點。

$$S(i,j) = G(i,j,\sigma) * I(i,j) \quad \text{.....(3-12)}$$

$$P(i,j) \approx (S(i+1,j) - S(i,j) + S(i+1,j+1) - S(i,j+1))/2 \quad \text{.....(3-13)}$$

$$Q(i,j) \approx (S(i,j+1) - S(i,j) + S(i+1,j+1) - S(i+1,j))/2 \quad \text{.....(3-14)}$$

※註：本文使用 Grayscale Morphology 代替高斯濾波器，關於型態學介紹於 3.3 應用於影像處理的形態學(Morphology)。

Step2 Gradient

計算其梯度向量和大小，梯度向量表該點在平面上變化最大的方向。在實際運算上，水平(垂直)方向上的梯度計算是以經過平滑處理後的灰階圖像與 Sobel 遮罩各別作旋積後再利用(3-4)式，以及對平滑後的 Q(i ,j)與 P(i ,j)取 inverse tangent，算出梯度近似值如(3-15)式。

$$\theta(i,j) = \tan^{-1}\left(\frac{Q(i,j)}{P(i,j)}\right) \qquad \text{.....(3-15)}$$

Step3　Non-maximum suppression

將上一步所算出的 inverse tangent 值，如同圖 3.4 所劃分的方式辨別屬於哪個區域。然後該凸起點延著梯度方向比較前後兩點，若小於則該點設為 0。此步驟可以使偵測出來的邊緣不會過寬。

Step4　Hysteresis Thresholding

利用雙臨界值演算法(thredholding)，做邊界的判別與連結。首先設定兩個臨界值(較小臨界值 T_L、較大臨界值 T_H)。

－當梯度大於 T_H，表示該點為邊緣上的一點。

－當梯度介於 T_H與 T_L之間，表示可能為邊緣上的一點。必須還要看看附近的點是否為邊緣，才能判定。

－當梯度小於 T_L，表示該點並非邊緣。

這一步驟主要是為了要避免因為雜訊干擾下，所造成的幾個問題：影像的整條邊緣變得斷斷續續，將不是邊緣的部份當成邊緣偵測出，或是扭曲原來影像的邊緣。

Canny edge detection 擁有以下幾點好處，因此本文選用此演算法。[3]

(1) 偵測錯誤率低：幾乎可以完全偵測到影像的邊緣。

(2) 定位效能好：圖像中真實邊界與演算之後邊界距離差異小。

(3) 良好的反應值：邊緣上的點都有唯一的反應值，因此偵測出來的邊緣不會過寬。

$\frac{1}{8}$ X

0	1	0
1	2	1
0	1	0

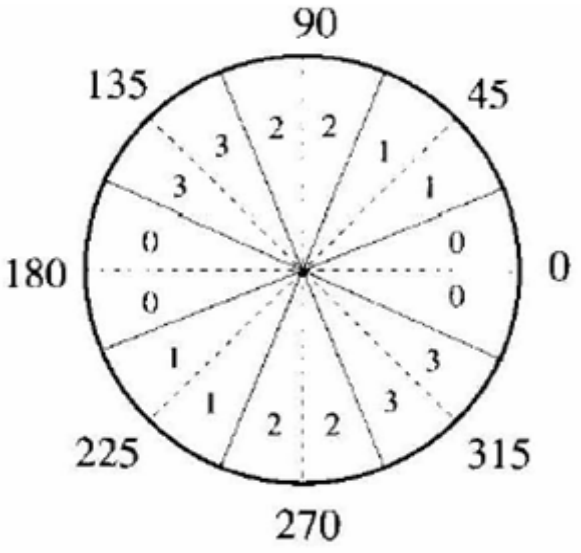
(a)

$\frac{1}{81}$ X

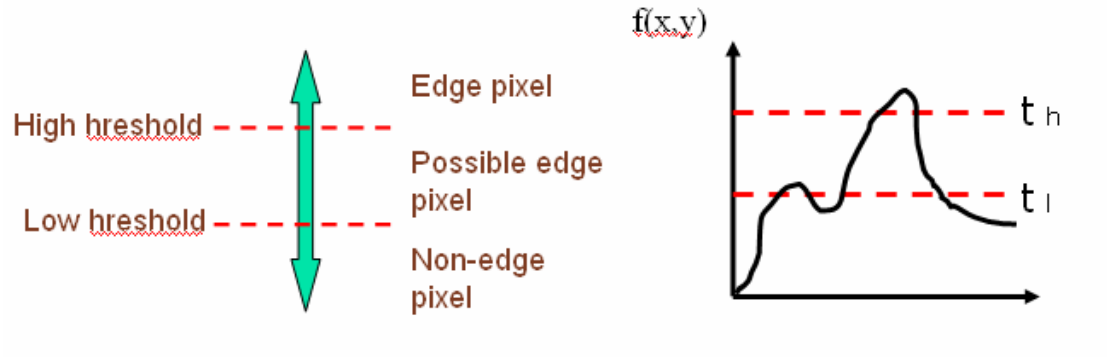
1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

(b)

▲圖 3.3　(a) 3x3 mask　(b) 5x5 mask



▲圖 3.4　方向畫分圖



▲圖 3.5　Hysteresis Thresholding 示意圖

3.3 應用於影像處理的形態(Morphology)[4][5][6]

形態學是一種對物件作邊界處理的演算法，其特點為減少細部資訊又可保留邊緣。基本上是利用一個 n*m 的像素矩陣與被處理的物件進行像素運算。其中可以分為四種，所有形態學的應用都是以四種運算混合使用。

假設 A 像素遮罩為被處理之圖像，B 像素遮罩為運算處理之遮罩，B 像素遮罩又稱為結構元素(structuring elements)。

(1)擴散(dilation)

定義為 $A \oplus B = \cup A_x; x \in B$。擴散以結構元素形狀來定義一個鄰域內選擇 A+B 的最大值，所以灰階影像上進行擴散的效果為：

－如果 B 值均為正，則輸出影像傾向比輸入影像明亮。

－A 的黑暗細節會依據 B 的值與形狀減少或消除。如圖 3.6。

(2)浸蝕(erosion)

浸蝕則是可以讓圖像 A 的外形均勻縮小，若以數學式表示即 $A \ominus B = \{w : B_W \subseteq A\}$。浸蝕以結構元素形狀來定義一個鄰域內選擇最小值為基礎，所以灰階影像上進行浸蝕的效果為：

－如果 B 值均為正，則輸出影像傾向比輸入影像暗。

－輸入影像中比 B 小的區域，其亮度細節將被減少，減少程度取決於亮細節周圍灰階值與 B 的形狀及振幅。如圖 3.7。

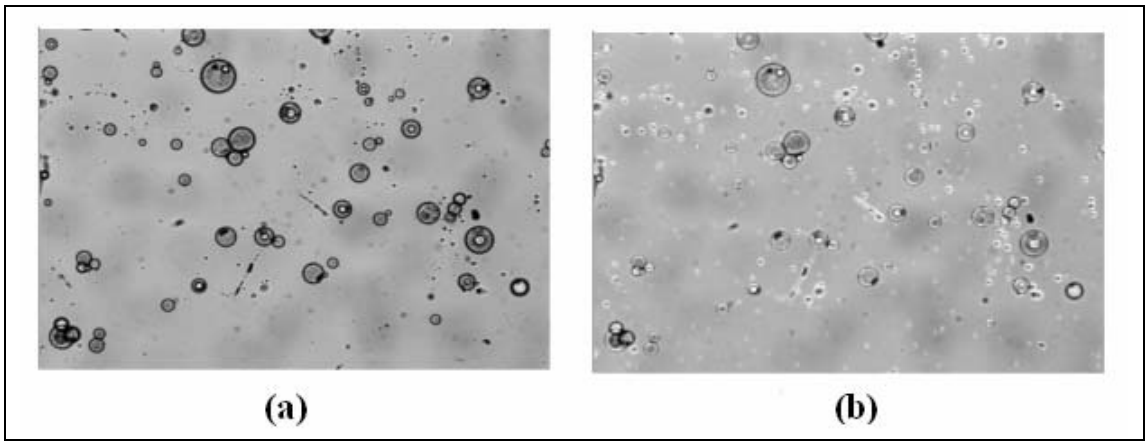
(3)斷開(opening)

先對圖像 A 作浸蝕再作擴散。一般表示為 A。$B=(A\ominus B)\oplus B$ 亦可表示為 A。$B=\cup\{B_W : B_W\subseteq A\}$。其功用在於可以除去影像中之雜訊小點。如圖 3.8。

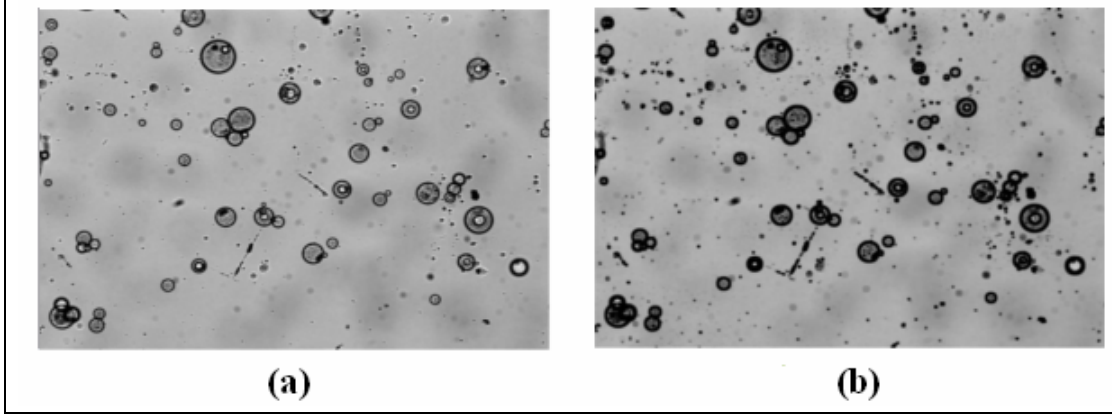
(4)閉合(closing)

與斷開相反，先對圖像 A 作擴散再作浸蝕。一般表示為 $A\cdot B=(A\oplus B)\ominus B$。其功用在於可以補影像中之小洞及將一些斷線連接起來。如圖 3.9。

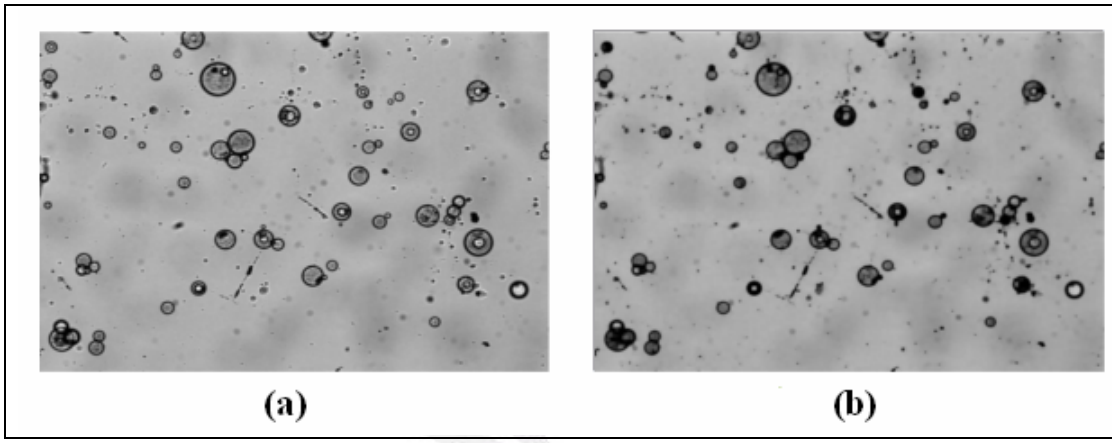
本文所使用的 Grayscale Morphology 是先用斷開再作閉合，對影像作平滑的處理。以 Morphology 代替 Laplacian of Gaussian，是因為前者定位效能與邊緣細化效果都比後者優良。



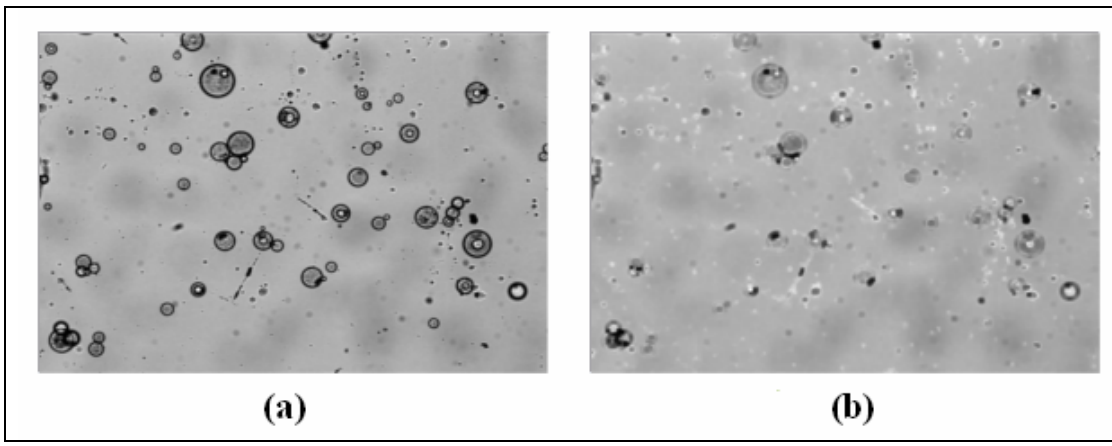
▲圖 3.6 (a)原圖 (b)擴散示意圖



▲圖 3.7 (a)原圖 (b)浸蝕示意圖



▲圖 3.8 (a)原圖 (b)斷開示意圖



▲圖 3.9 (a)原圖 (b)閉合示意圖

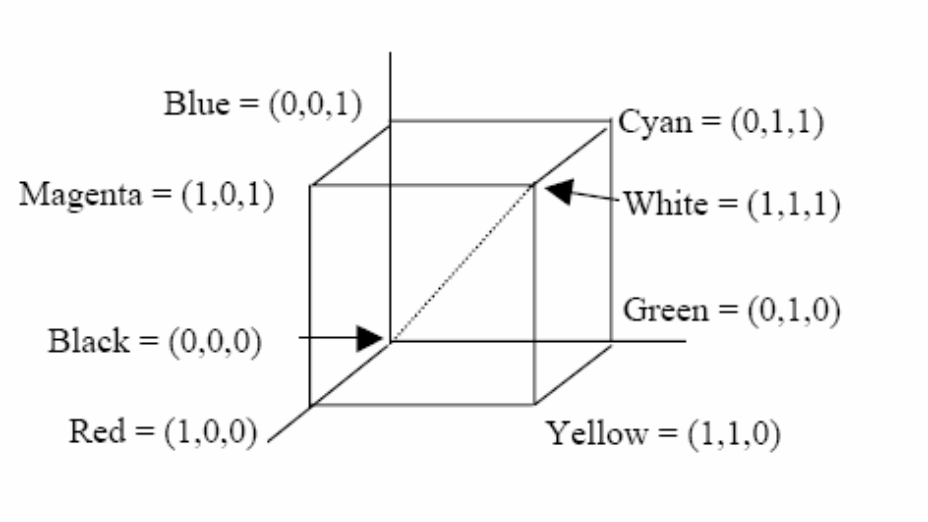
第四章　色彩系統

4.1 RGB 色彩系統

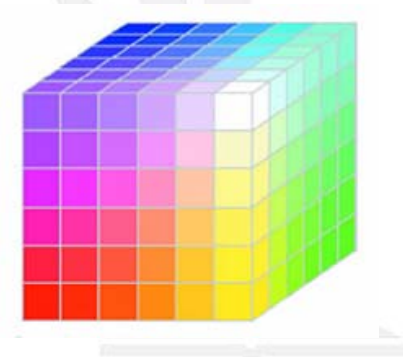
RGB 系統為電腦常採用的色彩資訊，原因可推至早期 CRT 螢幕時的設計原理，由電子槍所產生的電子流打到屏幕磷光點上反射出的是三原色光，而 RGB 正是以此為定義的色彩系統。

RGB 色彩系統是由紅色(Red)、綠色(Green)和藍色(Blue)的色光三原色所組成。被稱為色光三原色的原因，是由於它們無法再進行分解，而且也不能由其他色光混合出來。因為其混合的結果會得到越明亮的色光，稱為「加色混合」，當三原色光皆以 100%混合時，則會呈現白色光。

RGB 彩色模式是屬於一個三維的直角座標系，可以視作為立方體座標系，如圖 4.1。其 R 相當於 X 軸；G 相當於 Y 軸；B 相當於 Z 軸。因此若用座標來表示，(1,0,0)代表紅色；(0,1,0)代表綠色；(0,0,1)代表藍色。黑色在原點(0,0,0)的位置，白色則在立方體的另一端，座標(1,1,1)，如圖 4.2。當 R:G:B 等比例混合時，當 R、G、B 等量相加時，因不受 RGB 任一成份的影響可視為灰階。



▲圖 4.1 RGB 色彩系統座標係



▲圖 4.2

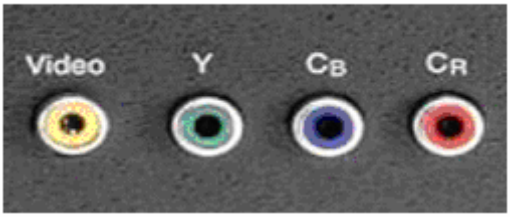
4.2 YCbCr 色彩系統

YCbCr 和 YUV、YIQ 是常用於視訊、電視影像傳輸的色彩模式，如圖 4.3。原理是依據人類對於亮度的變化比顏色的改變來的敏感，所以利用減少顏色的頻寬而增加亮度的頻寬，使頻寬達到有效利用。

YUV 主要用於 PAL 視訊標準而 YIQ 則是用於 NTSC 視訊標準上。其中的 Y 都代表亮度，即是由彩色轉換成灰階影像的灰階值或稱之為亮度值(luminace)，而其轉換公式(4-1)主要是依據人類的眼睛對綠色(0.587)、紅色(0.299)、藍色(0.114)三原色的不同敏感度而來，其中係數值越大則表人的眼睛對於該顏色較為敏感。

YCbCr 色彩系統又稱為 YPbPr 系統，簡稱 YCC 系統。它是由 YUV 系統正規化而來的一種色彩空間。定義於 CCIR Recommendation 601 規格[9]，規定 YCbCr 各為 8bit(即 0-255)的編碼，其中 Y 為明亮程度，其值為 16 到 235，代表明亮程度由暗到亮，共分為 220 個亮度等級。16 代表黑色，235 代表白色，並將其值±16 作為補償，使其值正規化至(0, 255)。而彩度 Cb 表偏藍色差向量，Cr 表偏紅色差向量，其值皆為-112 到+112，並將其值+128 作為補償，使其值(16, 240)，共有 225x225 種彩度。

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.533 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -84.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{.....(4-1)}$$



▲圖 4.3

4.3 HSI 色彩系統

人眼並不能分解色彩的組成量，所以 RGB 或是 YCbCr 系統對於人類是難以理解的，若有一系統能先將顏色分類，然後再控制色彩的細節，那正符合我們人眼的視覺，依照此需求所設計的色彩系統正是 HSI 色彩系統。

HSI 色彩系統將顏色分為色度(Hue)、飽和度(Saturation)、亮度(Intensity)三個部分來分別作表示，如圖 4.4。此系統將色度定義在 0~360 的角度範圍，而飽和度與亮度的值定義為 0~1 之間，其轉換公式為(4-2)~(4-5)式。

$$H = \begin{cases} \theta & \text{if B} \le \text{G} \\ 360 - \theta & \text{if B} > \text{G} \end{cases} \qquad \text{.....(4-2)}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}} \right\} \qquad \text{.....(4-3)}$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \qquad \text{.....(4-4)}$$

$$I = \frac{1}{3}(R+G+B) \qquad \text{.....(4-5)}$$

(1) Hue 色度

色度或稱為「色相」，指的是色彩相位，因此為一個角度。色度就像一張對照表，將顏色指定到特定角度。例如：在 H=0° 時為紅色系，H=120° 時為綠色系，H=240° 時為藍色系。由於色度其值在 0° ~360° 不斷循環，因此可視作一色相環，如圖 4.5。

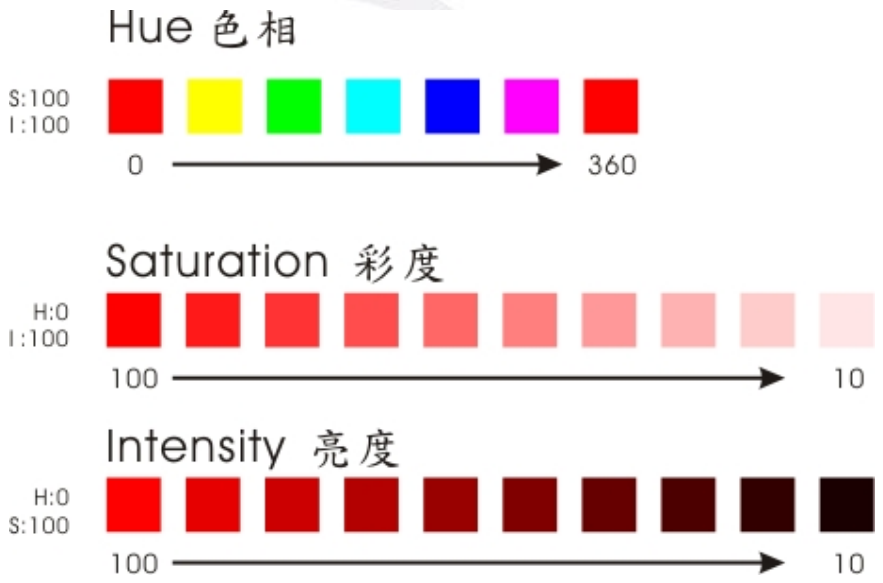
(2) Saturation 飽和度

飽和度是指色彩混濁的程度。飽和度越高，表顏色越鮮艷，飽和度越低，表顏色越混濁。在同一色相中，飽和度最高者稱為「純色」，其色彩中完全不含黑白成份。例如紅色係中，其飽和度最高的就是純紅色。

(3) Intensity 亮度

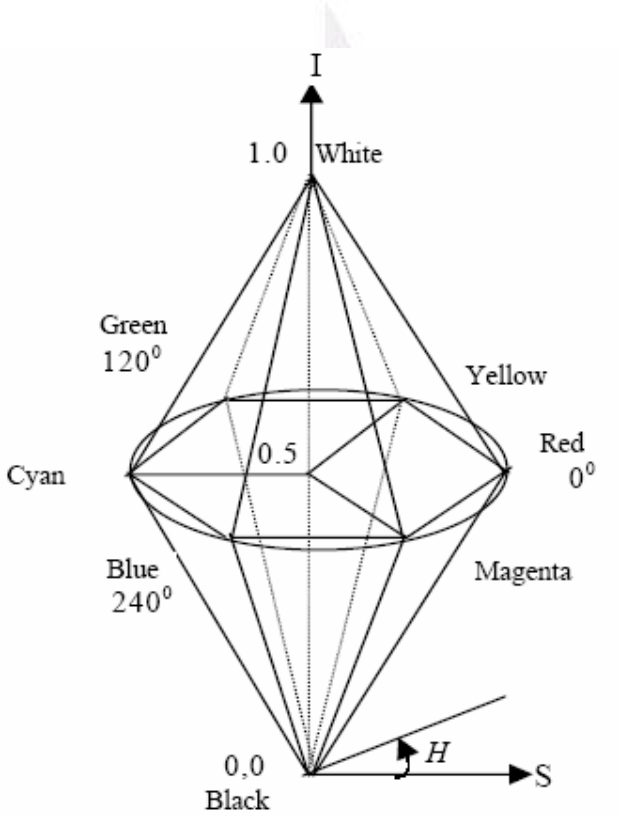
亮度又可稱「明度」，指的是物體反射光的程度，也可以說是光的能量。因此一張彩色影像的亮度資訊，相當於該影像的灰階影像。

HSI 彩色模式可視為一個圓柱座標系，H 對應到Φ(切線向量)，S 對應到 R(半徑)，I 對應到 Z(高度)，如圖 4.6 所示。



▲圖 4.4 色彩三要素

▲圖 4.5 色相環



▲圖 4.6 HSI 色彩系統座標係

第五章　演算法

5.1 演算架構

5.1.1 程式構思

本文使用的人臉偵測方法，利用「邊緣偵測」和「膚色偵測」找出人臉的皮膚部分。再依兩者所得到的結果作判斷，找出人臉位置。

5.1.2 步驟

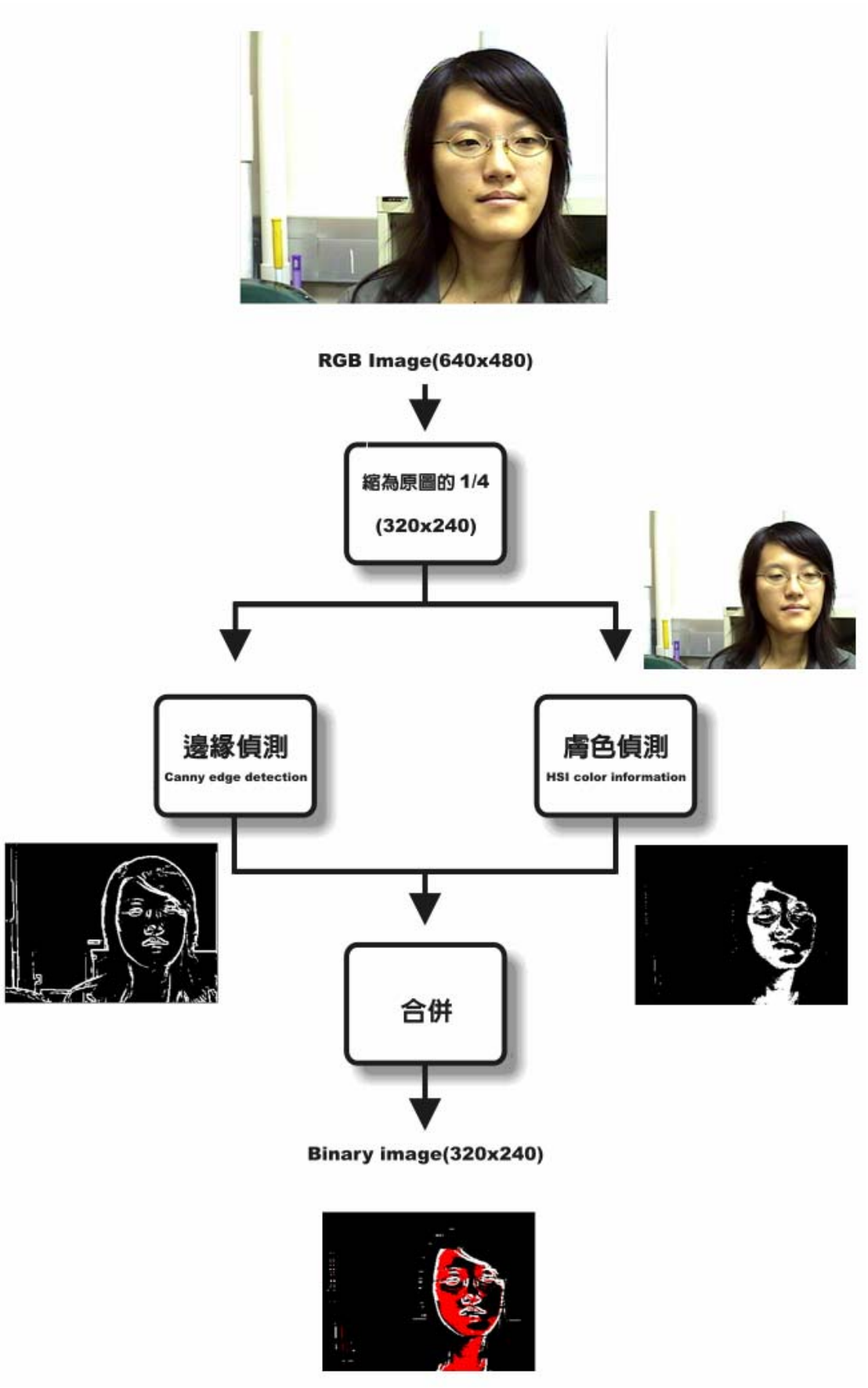
本文使用的人臉偵測方法的程式其流程便如圖 5.1 所示，步驟如下:

(1) 將處理前的即時影像縮為原影像的 1/4：因為電腦無法處理 640x480 及時影像這麼龐大的資料量，因此先作縮圖的動作。

(2) 邊緣偵測（canny edge detection）：主要在作抓取的人臉的形狀，可以準確的框出人臉位置。

(3) 膚色偵測（HSI color information）：利用人臉膚色色彩資料，判別可能為人臉訊息。利用 HSI 色彩系統，判別是否為膚色。

(4) 合併：將邊緣偵測和膚色偵測的結果，判斷人臉位置。


RGB Image(640x480)
縮為原圖的 1/4
(320x240)
邊緣偵測
Canny edge detection
膚色偵測
HSI color information
合併
Binary image(320x240)

▲圖 5.1 程式架構圖

5.2 邊緣偵測

5.2.1 程式構思

因為膚色色彩非均勻分佈於臉上，只利用膚色偵測，臉部邊緣陰影處可能無法被偵測到，造成所偵測到的人臉是不完整的。因此本文選用 Canny edge detection 偵測邊緣，其目的在找出人臉位置。

5.2.2 步驟

本文使用的邊緣偵測方法為 Canny edge detection，其流程便如圖 5.2 所示，步驟如下:

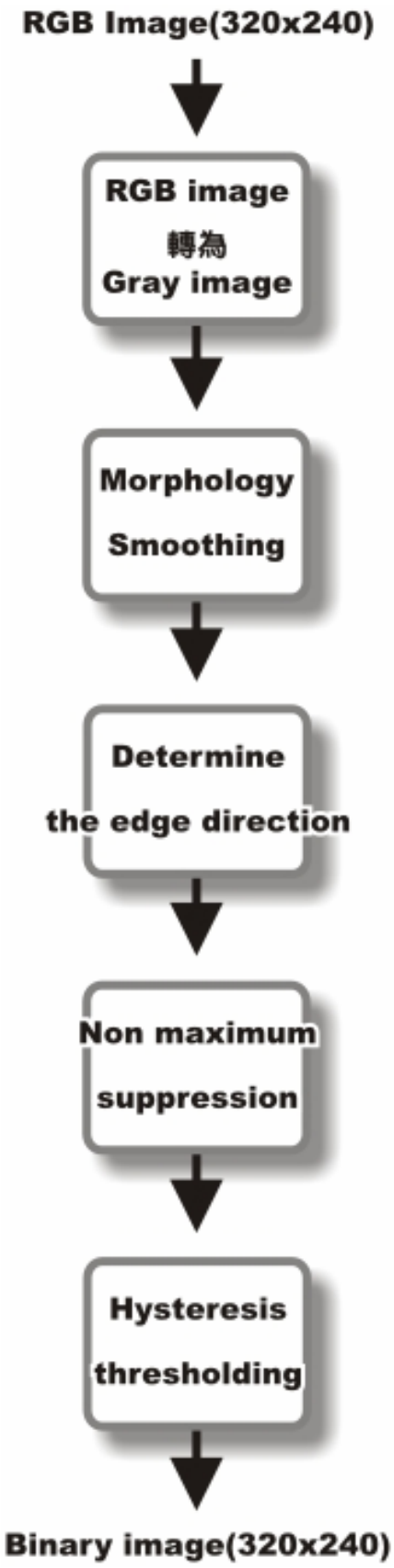
(1) 先將 RGB 彩色影像轉為灰階影像。

(2) Morphology smoothing 消除雜訊。

(3) 求出其每個 pixel 的梯度向量和大小。

(4) 找出每個 pixel 梯度方向最大值。

(5) 設臨界值 TH、TL 找邊緣。


RGB Image(320x240)
RGB image
轉為
Gray image
Morphology
Smoothing
Determine
the edge direction
Non maximum
suppression
Hysteresis
thresholding
Binary image(320x240)

▲圖 5.2 邊緣偵測

5.3 色彩資訊偵測

5.3.1 程式構思

HSI 色彩系統是經過比較後，發現最符合人眼視覺的色彩系統。因此本文選用 HSI 色彩系統的色彩訊息來判斷皮膚部分。其目的在找出一個最佳中心點(H0,S0,I0)和半徑 R0。以(H0,S0,I0)為球心，R0 為半徑的球體，當其值在球體內，判別為皮膚；球體之外，視為非重要訊息。

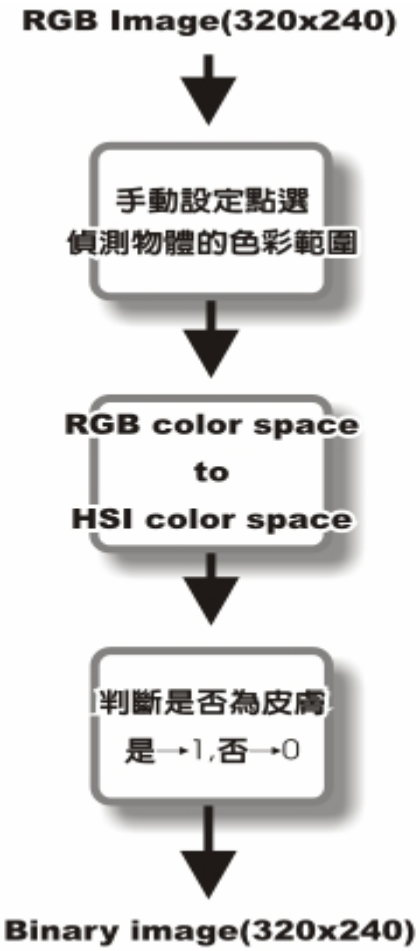
5.3.2 步驟

本文使用的膚色偵測的色彩模式 HSI color information，其流程如圖 5.3 所示，步驟如下:

(1) 手動設定

(2) RGB 色彩系統轉為 HSI 色彩系統

(3) 判斷是否為膚色



▲圖 5.3 膚色偵測

5.4 設定不同環境的膚色取樣範圍

5.4.1 程式構思

起初利用單張影像去找其 HSI 值，再給定 HSI 範圍，發現其偵測效果不彰，只對該張圖像有效。又鑒於在不同的環境、光線強弱和攝影器材的不同，都會影響膚色取樣範圍。因此開始有「手動設定偵測物體的色彩資訊」的想法，在新的環境下，初次使用時，需做手動

設定點選要測量的物體，由程式計算其中心點(H0,S0,I0)和半徑 R0，做為之後演算的依據。

5.4.2 步驟

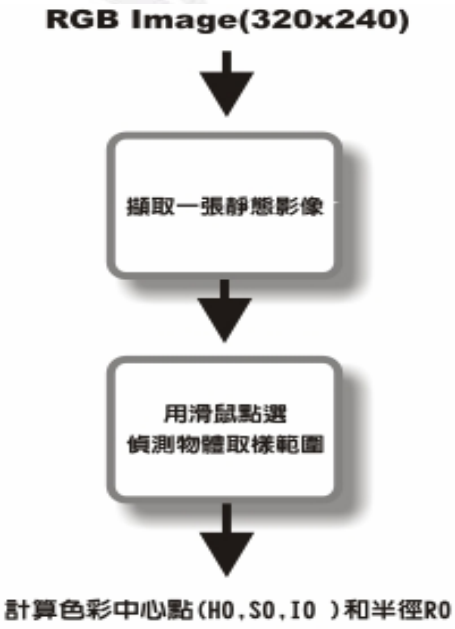
手動點選偵測物體，計算其中心點和半徑。其流程如圖 5.4 所示，步驟如下:

(1) 擷取一張靜態影像。

(2) 用滑鼠點選所要偵測的色彩範圍。

(3) 滑鼠點選的影像，RGB 彩色訊息轉為 HSI 色彩訊息。

(4) 計算其中心點(H0,S0,I0)和半徑 R0。



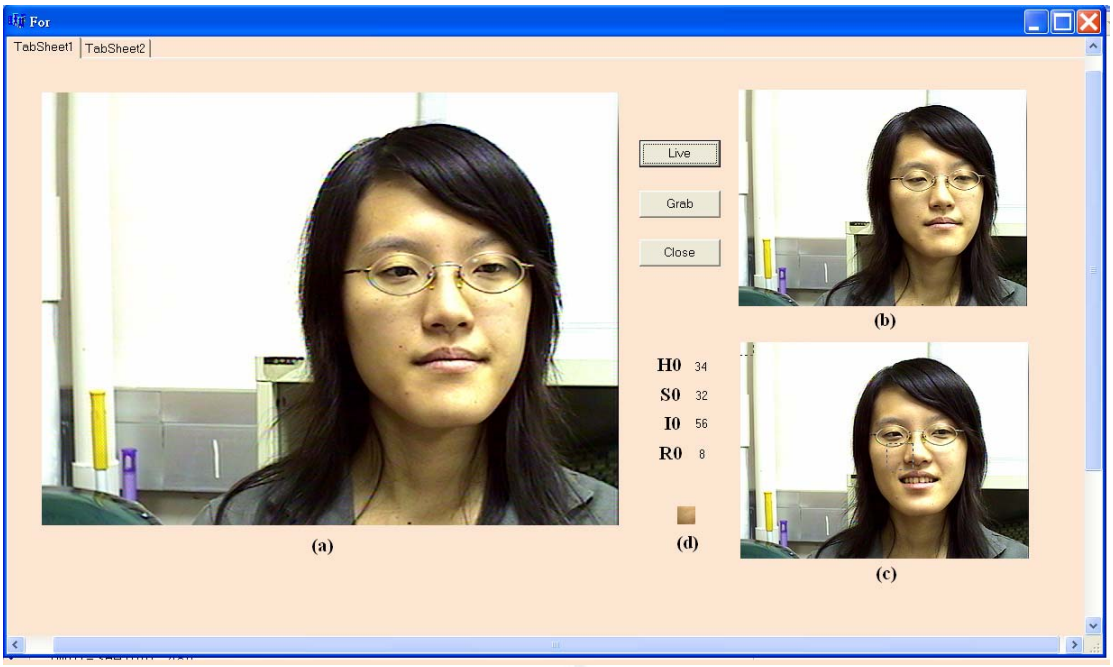
▲圖 5.4 手動設定點選偵測物體的色彩範圍

第六章　結果討論與未來研究方向

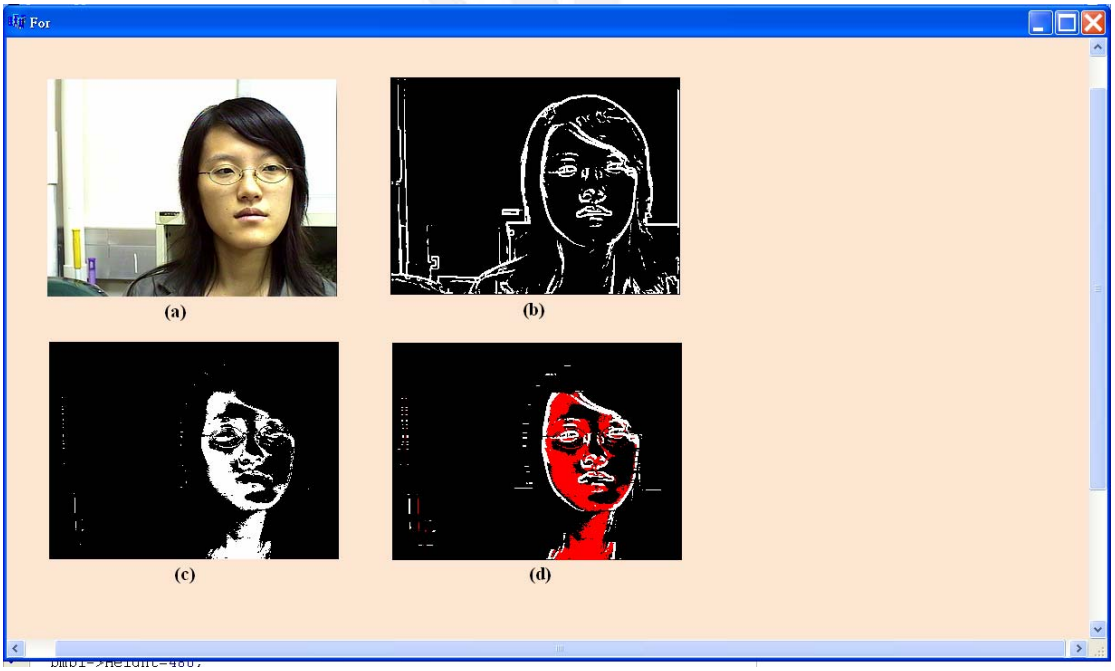
6.1 實驗結果

我們將分成視窗介面分為兩頁。第一個頁面，設定在不同環境下，偵測顏色的平均值：中心點和半徑，如圖 6.1。圖 6.1(a) 640x480 的即時影像；圖 6.1(b) 縮為原影像 1/4 的即時影像；圖 6.1(c) 擷取影像，用滑鼠移動視窗點選偵測物的色彩資訊；圖 6.1(d) 點選視窗的影像。

第二頁為偵測的結果，並將 Canny edge detection 和膚色偵測所得到的結果，將是邊緣也是膚色的部份用白色表示；非邊緣但是膚色的部份用紅色表示；其餘視為非臉部訊息，以黑色表示。如圖 6.2。圖 6.2(a) canny edge detection 結果；圖 6.2(b) 膚色偵測的結果；圖 6.2(c) 合併的結果。



▲圖 6.1 實驗結果(一)



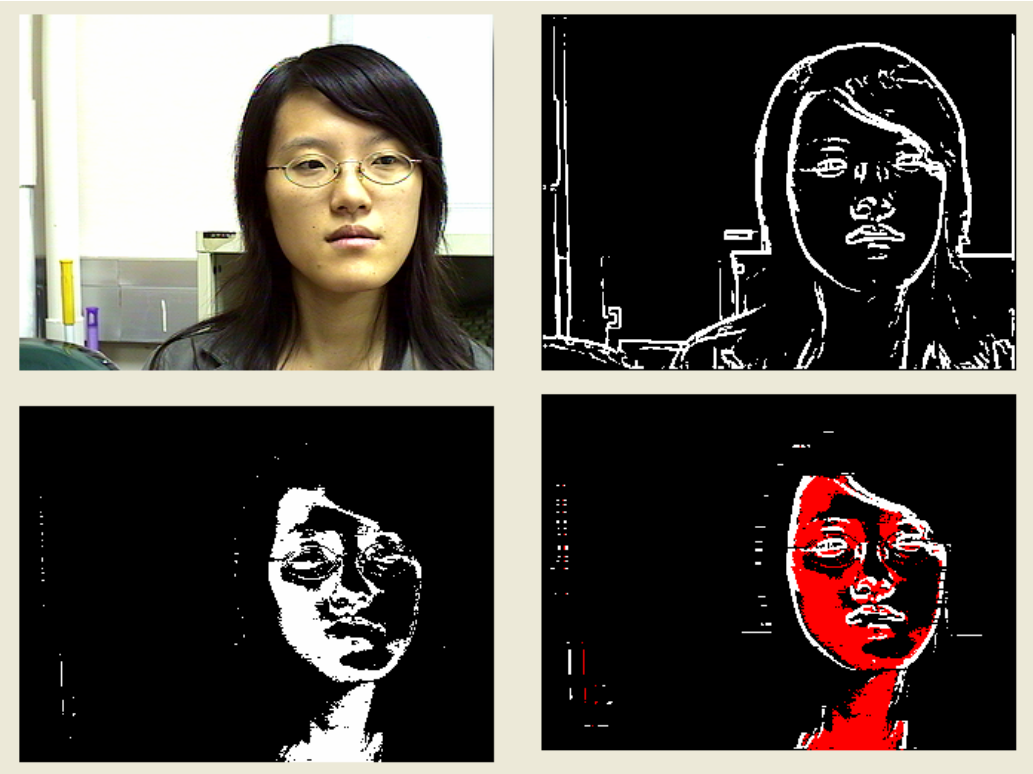
▲圖 6.2 實驗結果(二)

6.2 不同條件下實驗結果比較

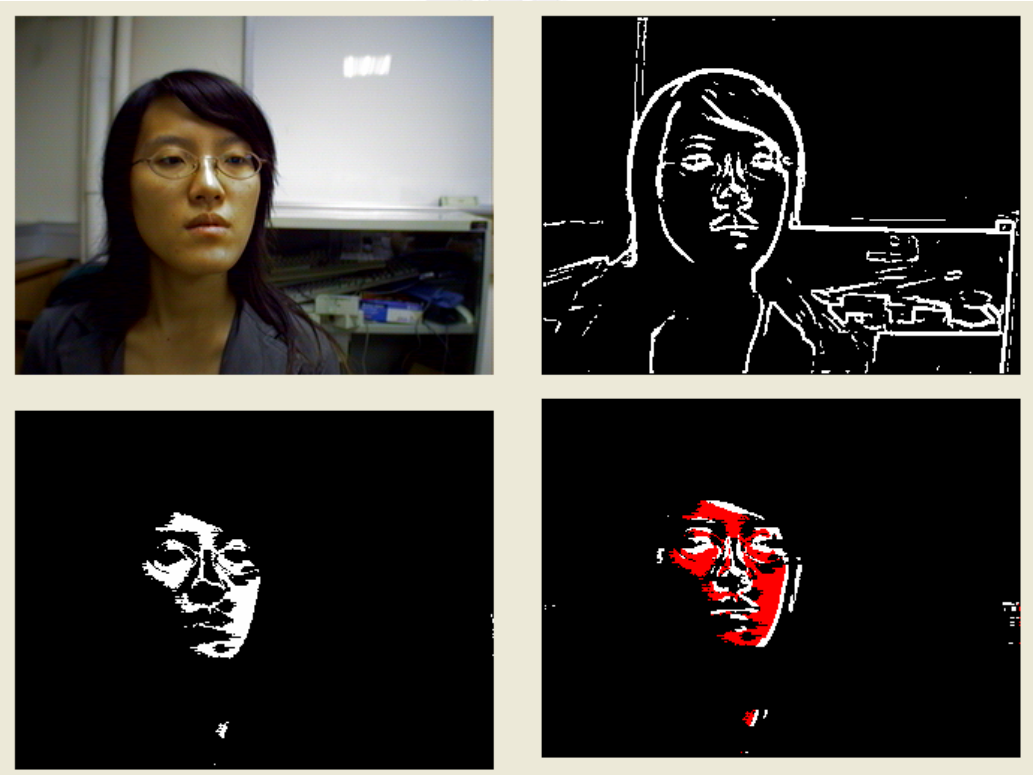
本文利用在不同條件下，比較偵測的效果的不同。表 6.1 是就各種可能發生情況的實驗結果並加以探討。

▼表 6.1 不同條件下偵測結果

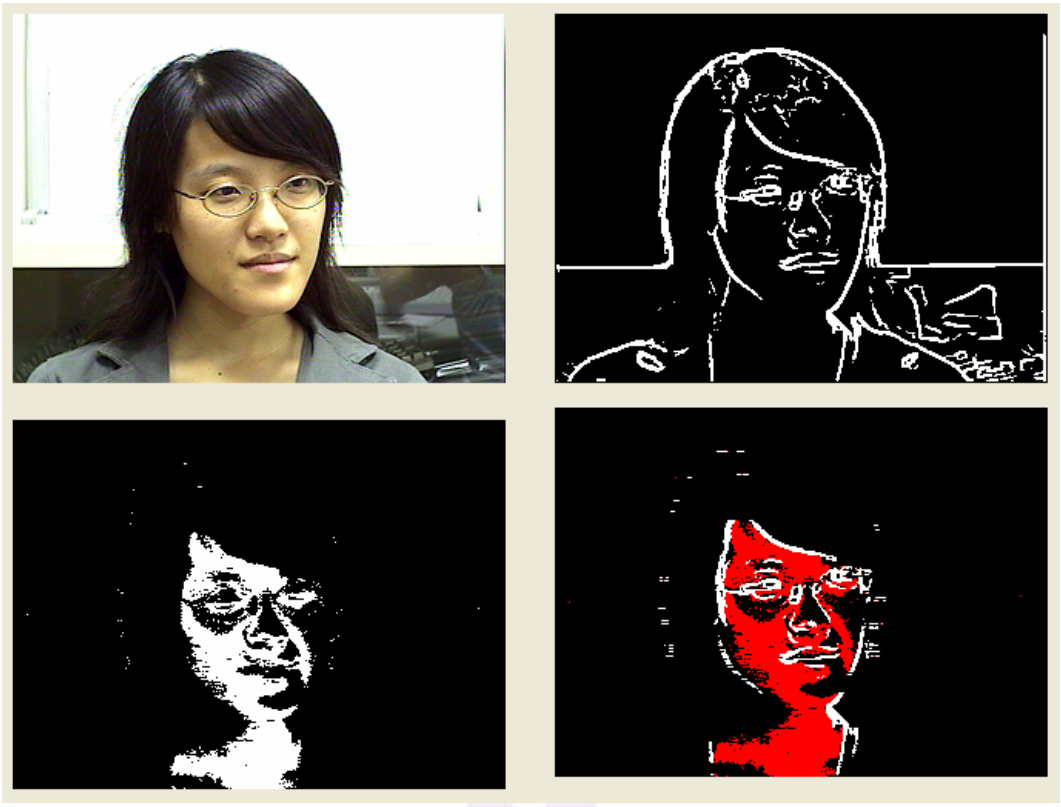
分類依據	結果
圖像來源	不同的攝影設備經過擷取卡，讀取進來的色彩資訊不同。中心點(H0,S0,I0)和 R0 的範圍都有所差異，因此偵測效果也會有所不同。如圖 6.3-6.4。
圖像背景複雜程度	簡單背景較複雜背景的偵測效果好。其原因是簡單的背景影響偵測的不確定因素較少，如背景顏色和膚色相近時，背景可能也會被偵測到。
光　　線	不同方向的光源，會造成偵測效果的不同。
鏡頭類型	頭肩部圖像的效果較半身圖像來得好。如圖 6.5-6.6。
人臉姿態	在初始值未更新中心點和半徑的情況下，有些方向可能會偵測不到。如圖 6.7-6.10。
人臉數目	只利用膚色色彩訊息作為偵測的依據，因此人數並不會造成偵測上的問題。如圖 6.11。



▲圖 6.3 攝影設備(一)



▲圖 6.4 攝影設備(二)

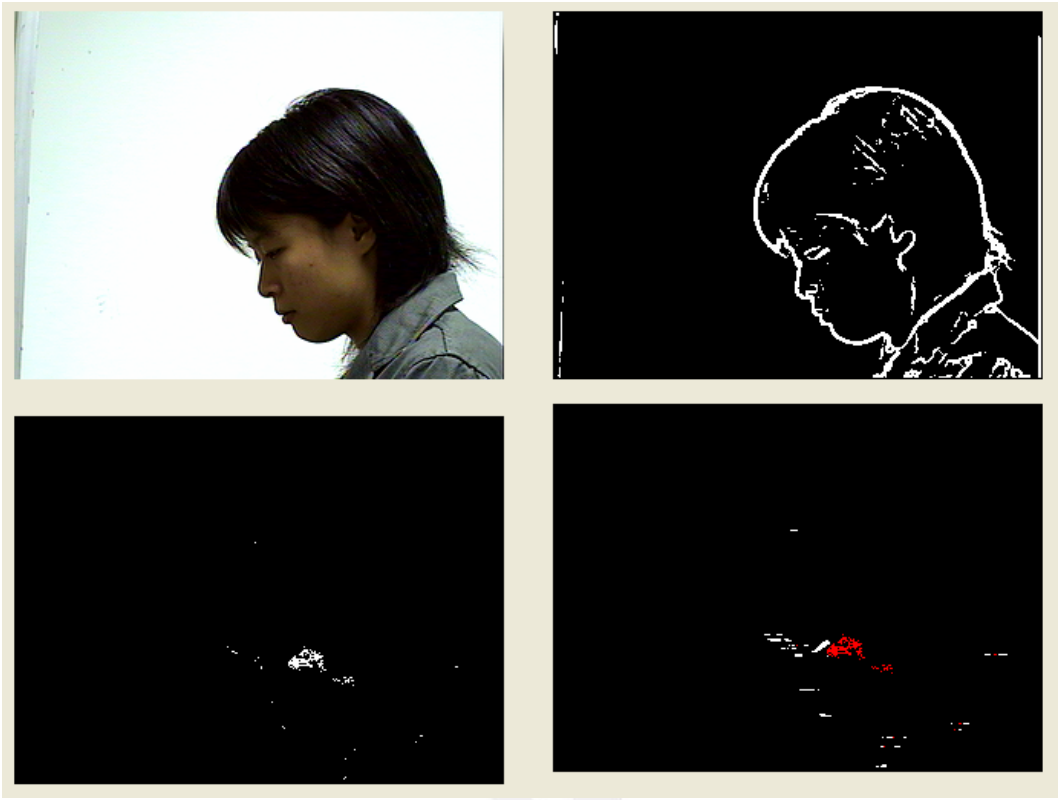


▲圖 6.5 頭肩部位

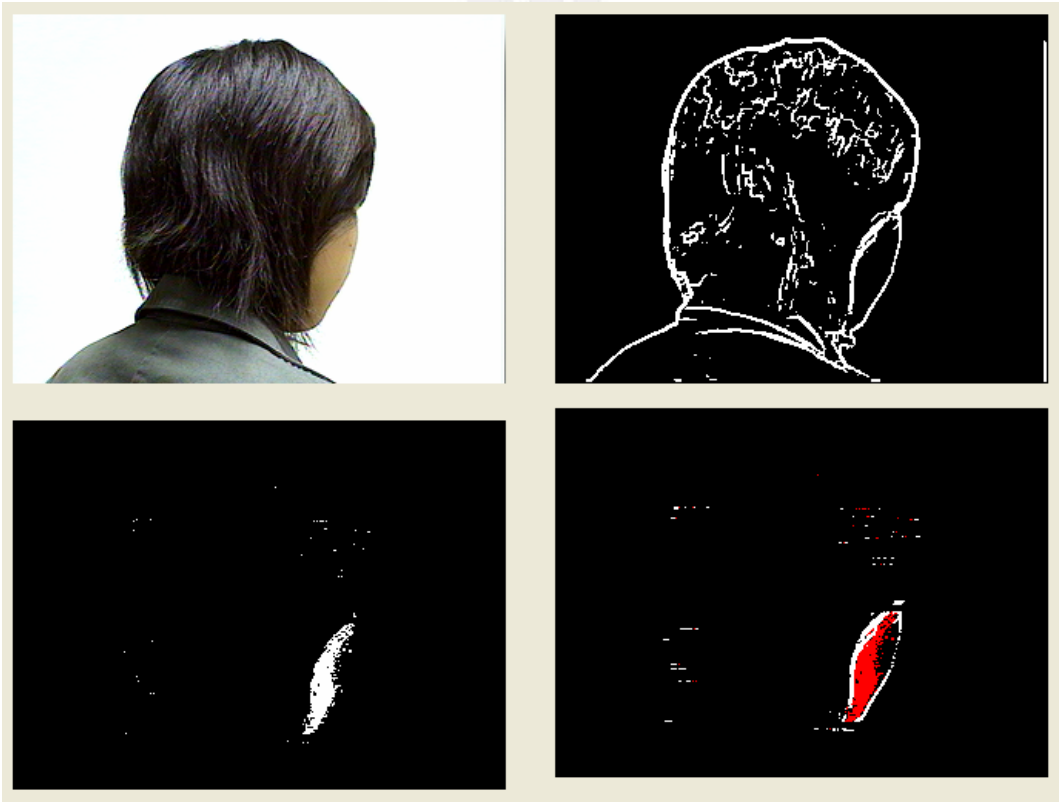
▲圖 6.6 半身

▲圖 6.7 仰角

▲圖 6.8 側面



▲圖 6.9 側面



▲圖 6.10 背面

▲圖 6.11 人臉數數目為複數

▲圖 6.12 人臉重疊

6.3 討論

本文所利用的人臉偵測方法還有許多的問題和改善空間。影響偵測結果除了外在因素攝影設備外，最大影響因素就是光線。當光線不均勻時，偵測效果較差，其原因是臉部陰影和光照處的色彩資訊有所不同。因此，不論正面或側面偵測上效果有所差異時，其主要原因是因為光的分佈均勻與否，而頭肩部圖像和半身/全身圖像也可以就此說明。

6.4 改善空間

本文中所介紹的人臉偵測，因為人臉演算法有限並只利用到人臉 color-based 的特性，但因未利用臉部特徵，因此還有許多問題和待改進的地方，以下是就問題和解決方法：

(1) 未利用臉部特徵

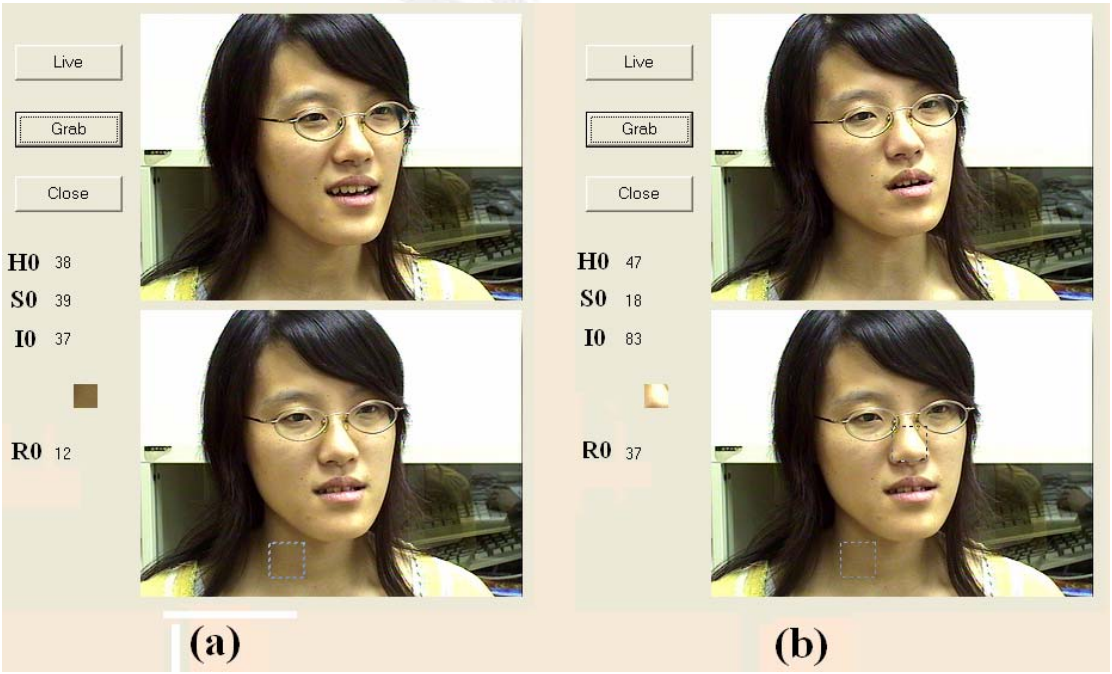
－臉型(橢圓形)：可避免脖子和四肢膚色部分被偵測到問題，但側面偵測可能會有問題。

－五官的幾何圖形(三角形)：可解決正面、側面，臉部旋轉各方向的問題，但運算上較為複雜。

(2) 手動設定測量物取樣範圍

－因點選範圍不同造成偵測差異：因為手動點選可能因人為因素點而得到不同的中心點(H0,S0,I0)和半徑 R0。如圖 6.13。若點選膚色較均勻處其 R0 太小，可能會造成偵測到的膚色區塊不完整性；點選膚色差異度較大處其 R0 太大，背景近膚色訊息部分又會被偵測出來。進而影響偵測的結果。

－未完全自動化：需要手動設定範圍，在應用上較不實際。解決方法利用適應性回傳參數值(T_H、T_L、H0、S0、I0、R0)，系統自動計算中心點和半逕，算出環境整體光線強弱、差異度，自動計算出 T_H、T_L和 H0、S0、I0、R0 的值回傳給系統使用。



▲圖 6.13 (a)點選位置一 (b)點選位置二

6.5 未來發展

越來越發達的時代中，通訊佔人類的重要性越來越大，不只是網際網路普遍化，多媒體的應用也與人們的生活幾乎已經結合在一起了。犯罪手法也因為科技的進步而越來越防不勝防，因此監視系統不能只靠錄影以及人工辨識來達到防止犯罪。經過這次實驗讓我們發覺到，不同的演算法、偵測條件的多寡會大大影響到偵測效果。

因此，我們將朝著利用更多的人臉特徵，例如：人臉約略呈橢圓形、眼睛與嘴唇略呈等腰三角形等。或是讓系統自行偵測環境，取得所需的參數，減少人工部分讓系統更為自動化。更進一步，希望能夠達到人臉追蹤，將追蹤到的人臉資訊能夠加以放大、解析或作到更多的處理。最後，基於這些種種開發研究也是學生未來想達到的目標與期望。

附　　錄

程式 Canny edge detection

```
//*************************************************************
// Dilation (擴散)
//*************************************************************
int findmaximum(unsigned char *in)
{
   int i,j,s,t;
   for (i=-1;i<=1;i++) {
      for (j=-1;j<=1;j++) {
         if (i==-1 && j==-1)
            s = (*(in+i*Im_Width+j));
         else {
            t = (*(in+i*Im_Width+j));
            if (t>s)
               s = t;
         }}}
   return (s);
}
//*************************************************************
// Erosion (浸蝕)
//*************************************************************
int findminimum(unsigned char *in)
{
   int i,j,s,t;
   for (i=-1;i<=1;i++) {
      for (j=-1;j<=1;j++) {
         if (i==-1 && j==-1)
            s = (*(in+i*Im_Width+j));
         else {
            t = (*(in+i*Im_Width+j));
            if(t<s)      s = t;
         }}}
   return (s);
}
```

```
//*************************************************************
// Morphology Smoothing
//*************************************************************
void Morphology(unsigned char *org,unsigned char *postSmoothing)
{
    int i,j;
    unsigned char a[Im_Height*Im_Width];
    unsigned char b[Im_Height*Im_Width];
    unsigned char c[Im_Height*Im_Width];
    //-------------------------------------------------------------------------
    // Opening
    //-------------------------------------------------------------------------
    // ----- Erosion ----------
    for (i=0;i<Im_Height;i++)
    {
       for (j=0;j<Im_Width;j++)
       {
          if (i==0||i==(Im_Height-1)||j==0||j==(Im_Width-1))
             *(a+i*Im_Width+j) = *(org+i*Im_Width+j);
          else
             *(a+i*Im_Width+j) = findminimum(org+i*Im_Width+j);
       }
    }
    //----- Dilation ----------
    for (i=0;i<Im_Height;i++)
    {
       for (j=0;j<Im_Width;j++)
       {
          if (i==0||i==(Im_Height-1)||j==0||j==(Im_Width-1))
             *(b+i*Im_Width+j) = *(a+i*Im_Width+j);
          else
             *(b+i*Im_Width+j) = findmaximum(a+i*Im_Width+j);
       }
    }

    //-------------------------------------------------------------------------
    // Closing
    //-------------------------------------------------------------------------
```

```
    //----- Dilation -----
    for (i=0;i<Im_Height;i++)
    {
       for (j=0;j<Im_Width;j++)
       {
          if (i==0||i==(Im_Height-1)||j==0||j==(Im_Width-1))
             *(c+i*Im_Width+j) = *(b+i*Im_Width+j);
          else
             *(c+i*Im_Width+j) = findmaximum(b+i*Im_Width+j);
       }
    }
    // ----- Erosion ----------
    for (i=0;i<Im_Height;i++)
    {
       for (j=0;j<Im_Width;j++)
       {
          if (i==0||i==(Im_Height-1)||j==0||j==(Im_Width-1))
             *(postSmoothing+i*Im_Width+j) = *(c+i*Im_Width+j);
          else
             *(postSmoothing+i*Im_Width+j) = findminimum(c+i*Im_Width+j);
       }
    }
}

//************************************************************
// Canny
//************************************************************
void Canny(unsigned char *Gausian, unsigned char *postCanny)
{
    int GX[3][3], GY[3][3];
    int sumX, sumY, SUM;
    int r,c, i, j;
    double          ORIENT;
    int             edgeDirection;
    int             HighThreshold, lowThreshold;
    int             leftPixel, rightPixel;
    int             P1, P2, P3, P4, P5, P6, P7, P8;
```

```
//-------------------------------------------------------------------------
// 計算其梯度大小和方向
//-------------------------------------------------------------------------
//----- 3x3 GX Sobel mask ----------
GX[0][0] = -1;   GX[0][1] = 0;   GX[0][2] = 1;
GX[1][0] = -2;   GX[1][1] = 0;   GX[1][2] = 2;
GX[2][0] = -1;   GX[2][1] = 0;   GX[2][2] = 1;
//----- 3x3 GY Sobel mask ----------
GY[0][0] = 1 ;   GY[0][1] = 2 ;   GY[0][2] = 1 ;
GY[1][0] = 0 ;   GY[1][1] = 0 ;   GY[1][2] = 0 ;
GY[2][0] = -1;   GY[2][1] = -2;   GY[2][2] =-1;
//-----  對 x 和 y 方向作梯度運算 ----------
for(r=0; r<=(Im_Height-1); r++) {
   for(c=0; c<=(Im_Width-1); c++) {
      sumX = 0;
      sumY = 0;
      if(r==0 || r==Im_Height-1)
         SUM = 0;
      else if(c==0 || c==Im_Width-1)
         SUM = 0;
      else {
         for(i=-1; i<=1; i++) {
            for(j=-1; j<=1; j++) {
            sumX = sumX+(int)(Gausian[(r+i)*Im_Width+c+j] * GX[i+1][j+1]);
            }
         }
         for(i=-1; i<=1; i++) {
            for(j=-1; j<=1; j++) {
            sumY = sumY+(int)(Gausian[(r+i)*Im_Width+c+j]* GY[i+1][j+1]);
            }
         }
         //-----  計算梯度大小 ----------
         SUM = abs(sumX) + abs(sumY);
         if(SUM>255)       SUM=255;
         if(SUM<0)         SUM=0;
    }
 }
```

```
//-----  計算梯度向量方向(角度:0~360 度) ----------
if(sumX == 0) {
   if(sumY ==0)
      ORIENT = 0.0;
   else if (sumY<0) {
      sumY = -sumY;
      ORIENT = 90.0; }
   else
      ORIENT = 90.0; }
else if(sumX<0 && sumY>0) {
      sumX = -sumX;
      ORIENT = 180 - ((atan((float)(sumY)/(float)(sumX))) * (180/M_PI)); }
else if(sumX>0 && sumY<0) {
   sumY = -sumY;
   ORIENT = 180 - ((atan((float)(sumY)/(float)(sumX))) * (180/M_PI)); }
else
   ORIENT = (atan((float)(sumY)/(float)(sumX))) * (180/M_PI);

//-------------------------------------------------------------------------
// Step Three
//-------------------------------------------------------------------------
// 將梯度向量分為八個區域
if      (ORIENT <   22.5)   edgeDirection =    0;
else if (ORIENT <   67.5)   edgeDirection =   45;
else if (ORIENT < 112.5 )   edgeDirection =   90;
else if (ORIENT < 157.5 )   edgeDirection =  135;
else                        edgeDirection =    0;

 if(edgeDirection == 0) {
    leftPixel = (int)(Gausian[r*Im_Width + c - 1]);
    rightPixel = (int)(Gausian[r*Im_Width + c + 1]); }
 else if(edgeDirection == 45){
    leftPixel = (int)(Gausian[(r+1)*Im_Width + c - 1]);
    rightPixel = (int)(Gausian[(r-1)*Im_Width + c + 1]); }
 else if(edgeDirection == 90) {
    leftPixel = (int)(Gausian[(r-1)*Im_Width + c ]);
    rightPixel = (int)(Gausian[(r+1)*Im_Width + c ]); }
```

```
        else {
           leftPixel = (int)(Gausian[(r-1)*Im_Width+c-1]);
           rightPixel = (int)(Gausian[(r+1)*Im_Width+c+1]); }

        if(SUM < leftPixel || SUM < rightPixel)
           SUM = 0;
        //------------------------------------------------------------------------
        // Step Four
        //------------------------------------------------------------------------
        else {
           HighThreshold = 70;
           lowThreshold = 0;
           //----- 梯度大小 > TH，判斷為邊界 -----
           if(SUM >= HighThreshold)
              SUM = 255;
           //----- 梯度大小 < TL，判斷為非邊界 -----
           else if(SUM <= lowThreshold)
              SUM = 0;
           //----- 梯度介於 TH 和 TL 之間，判別鄰近點是否為邊界 ----------
           else {
              P1 = (int)(Gausian[(r-1 )*Im_Width+c-1 ]);
              P2 = (int)(Gausian[(r-1 )*Im_Width+c    ]);
              P3 = (int)(Gausian[(r-1 )*Im_Width+c+1]);
              P4 = (int)(Gausian[(r    )*Im_Width+c-1 ]);
              P5 = (int)(Gausian[(r    )*Im_Width+c+1]);
              P6 = (int)(Gausian[(r+1)*Im_Width+c-1]);
              P7 = (int)(Gausian[(r+1)*Im_Width+c    ]);
              P8 = (int)(Gausian[(r+1)*Im_Width+c+1]);

              if (P1 > HighThreshold || P2 > HighThreshold || P3 > HighThreshold ||
P4 > HighThreshold || P5 > HighThreshold || P6 > HighThreshold ||
P7 > HighThreshold || P8 > HighThreshold)
                 SUM = 255;
              else
                 SUM = 0; }}
              postCanny[r*Im_Width+c] = (unsigned char)(SUM); }}
}
```

程式 color information detection

```
//*************************************************************
// color information 偵測
//*************************************************************
void HSI_processing (unsigned char *in , unsigned char *postHSI)
{
    int i,j;
    int P,sum,ans,m;
int R,G,B,H,S,I;
    float x,y,z,sita;

    for (i=0;i<Im_Height;i++)
    {
       for (j=0;j<Im_Width;j++)
       {
          P = i*Im_Width+j;
          B = *(in+3*P    );
          G = *(in+3*P+1);
          R = *(in+3*P+2);

          sum = R+G+B;
          //-------------------------------------------------------------------------
          // 計算 H 值
          //-------------------------------------------------------------------------
          x = 0.5*((R-G)+(R-B));
          y = sqrt((R-G)*(R-G)+(R-B)*(G-B));
          if (y==0)
             z = 0;
          else
             z = x/y;

          sita = (acos(z)*360)/(2*M_PI);

          if (G>B)
             H = (int) (sita);
          else
             H = (int) (360-sita);
```

```
        //--------------------------------------------------------------------
        // 計算S 值
        //--------------------------------------------------------------------
        if (sum==0)
           S = 100;
        else {
           m = findmin(R,G,B);
           S = (100-(3*m*100)/sum); }
        //--------------------------------------------------------------------
        // 計算I 值
        //--------------------------------------------------------------------
        I = (sum*100)/(3*255);
        //--------------------------------------------------------------------
        ans = in_or_out (H,S,I);
        if (ans==0)    postHSI [P] =    0;
        else           postHSI [P] = 255;
      }
    }
}

//*************************************************************
// 中心點(H0,S0,I0)，半徑為 R0 所形成的球體
// 包含在球內的判別為皮膚；其他判別為非皮膚
// 判別是否為膚色
//*************************************************************
int in_or_out (int H , int S , int I)
{
   int x,y;
   float ans;

   x = get_RH (H0 , H);
   y = x*100/180;
   ans = sqrt(y*y+(S-S0)*(S-S0)+(I-I0)*(I-I0));

   if (ans > R0)    return 0;
   else             return 1;
}
```

```
//*****************************************************************
// 算 H 和 H0 的夾角( 0~180 度)
//*****************************************************************
int get_RH (int a, int b)
{
   int x=0, y, c;
   if(a<180) {
      c = a+180;
      if(b >= c) {
         y = 360-b+a;
         if(y > x)
            x = y; }
      else {
         if(b > a) {
            y = b-a;
            if(y > x)
               x = y; }
         else {
            y = a-b;
            if(y > x)
               x = y; }}}
   else {
      c = a-180;
      if(b >= a) {
         y = b-a;
         if(y > x)
            x = y; }
      else {
         if(b >= c) {
            y = a-b;
            if(y > x)
               x = y; }
         else {
            y = 360-a+b;
            if(y > x)
               x = y;
          }}}
}
```

程式 計算中心點(H0,S0,I0)和球半徑 R0

```
//*************************************************************
// 計算中心點(H0,S0,I0)和球半徑 R0
//*************************************************************
void __fastcall TForm1::Form1_Im2MouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
  int P,i,j,sum,m,H1,H2;
  int R, G, B, R_H, R_S, R_I, R0;
  float x,y,z,sita;
  unsigned char H[C_Height*C_Width];
  unsigned char S[C_Height*C_Width];
  unsigned char I[C_Height*C_Width];
  //---------------------------------------------------------------------------
  // RGB color mode 轉換成 HSI color mode
  //---------------------------------------------------------------------------
  for (j=0;j<C_Height;j++)
  {
    for (i=0;i<C_Width;i++)
    {
      P = (j*C_Width)+i;
      TCColor c = Form1_Im3->Canvas->Pixels[i][j];
      B = (int)c.Blue;
      G = (int)c.Green;
      R = (int)c.Red;
      sum = R+G+B;
      //----- 計算 H 值 ----------
      x = 0.5*((R-G)+(R-B));
      y = sqrt((R-G)*(R-G)+(R-B)*(G-B));
      if(y==0)     z = 0;
      else         z = x/y;
      sita = (acos(z)*360)/(2*M_PI);
      if (G>B)
        H[P] = (int)(sita);
      else
        H[P] = (int)(360-sita);
```

```
        //-----  計算 S 值----------
        if(sum==0)    S[P] = 0;
        else
        {
           m = findmin(R,G,B);
           S[P] = (100-(3*m*100)/sum);
        }
        //-----  計算 I 值  ----------
        I[P] = (sum*100)/(3*255);
     }
  }
  //----------------------------------------
  bubble_sort (S);
  bubble_sort (I);
  //-------------------------------------------------------------------------
  //  計算 H0 和 R_H 之值
  //-------------------------------------------------------------------------
  //-----計算 H0 ----------
  x = 0; y = 0; sum = 0;
  for (i=0;i<C_Size;i++)
  {
     H1 = H[i];
     H2 = new_sita(H1);
     sum += H2;
     if(H1>180)    x++;
     else          y++;
  }
  H0 = (int)(sum/C_Size);
  if (x>y)    H0 = 360-H0;
  else        H0 = H0;
  //-----計算 R_H ----------
  x = 0;
  for(i=0;i<C_Size;i++)
  {
     H1 = H[i];
     y = get_RH(H0,H1);
  }
  R_H = x*100/180;
```

```
    //---------------------------------------------------------------------------
    // 計算 S0 和 R_S 的值
    //---------------------------------------------------------------------------
    //----- 計算 S0 ----------
    sum = 0;
    for(i=0;i<C_Size;i++) {
       sum += S[i];
    }
    S0 = (int)(sum/C_Size);
    //----- 計算 R_S ----------
    x = S0-S[10];
    y = S[C_Size-11]-S0;
    if (x>y)    R_S = x;
    else        R_S = y;
    //---------------------------------------------------------------------------
    // 計算 I0 和 R_I 的值
    //---------------------------------------------------------------------------
    //-----  計算 I0 ----------
    sum = 0;
    for(i=0;i<C_Size;i++) {
       sum += I[i];
    }
    I0 = (int)(sum/C_Size);
    //-----  計算 R_I ----------
    x = I0-I[10];
    y = I[C_Size-11]-I0;
    if (x>y)    R_I = x;
    else        R_I = y;
    //---------------------------------------------------------------------------
    // 計算 R0 的值
    //---------------------------------------------------------------------------
    R0 = (int)(sqrt((R_H)*(R_H)+(R_S)*(R_S)+(R_I)*(R_I))) ;
    //---------------------------------------
    Label1->Caption = H0;
    Label2->Caption = S0;
    Label3->Caption = I0;
    Label4->Caption = R0;
}
```

參 考 文 獻

◎文章資料

[1] Four Approaches to Face Detection 陳必衷 pp1.

[2] 利用三維形態分析診斷肺臟腫瘤之系統 葉嘉芬 pp26.

[3] 以邊為基礎之衛星影像中的車輛偵測 王彥棋(國立中央大學資訊工程研究所碩士論文)

[4] 視訊監控系統之物件抽取研究 李俊儀 (國立雲林科技大學電機工程系碩士論文)

[5] 文字辨識論文研討 第二章 pp13. (國立嘉義大學資訊工程所洪燕竹)

[6] Introduction to Digital Image Processing with MATLAB pp234~277.

[7] http://www.me.ncu.edu.tw/teacher/Subject/B91_2/A/ME564/Ch_5.doc

[8] http://www.cee.hw.ac.uk/hipr/html/sobel.html

[9] CCIR, Encoding parameters of digital television for studios, CCIR Recommendation 601- 2, Int. Radio Consult.Committee, Geneva, Swizerland,1990.

[10] Crowley, J. L. and Coutaz, J., "Vision for Man Machine Interaction," Robotics and Autonomous Systems, Vol. 19, pp. 347-358 (1997).

[11] Cahi, D. and Ngan, K. N., "Face Segmentation Using Skin-Color Map in Videophone Applications," IEEE Transaction on Circuit and Systems for Video Technology, Vol. 9, pp. 551-564 (1999).

[12] Kjeldsen, R. and Kender., J., "Finding Skin in Color Images," Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, pp. 312-317 (1996).

[13] http://www.cse.iitk.ac.in/users/pg/

[14] Cai, J., Goshtasby, A. and Yu, C., "Detecting Human Faces in Color Images," Proceedings of International Workshop on Multi-Media Database Management Systems,pp. 124-131 (1998).

[15] Yang, J., Lu, W. and Waibel A., "Skin Color Modeling and Adaptation," (CMUCS-97-146,) CS Department, CMU, PA,U.S.A. (1997).

[16] Yang, M. H. and Ahuja., N., "Detecting Human Faces in Color Images", Proceedings of IEEE International Conference on Image Processing, Vol.1, pp. 127-130 (1998).

[17] Yang, M. H., Kriegman, J. and Ahuja, N.,"Detecting Faces in Images: A Survey,"IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 24, pp. 34-58 (2002).

[18] http://www.csie.fju.edu.tw/~jykuo/semi/Face_Detection_ Report.ppt

[19] http://content.edu.tw/junior/computer/tp_ct/ch06/ch6-1/ graph27.htm

[20] http://andrew.csie.ncyu.edu.tw/DOC2/文字辨識.pdf

[21] A Robust Skin Color Based Face Detection Algorithm Sanjay Kr. Singh1, D. S. Chauhan2, Mayank Vatsa3, Richa Singh3*

[22] 數位影像處理，原著(Digital Imaging Processing, by R. C. Gonzalez and E. Woods)，繆紹綱編譯，2003 年，高立圖書公司。

[23] 利用環場及 PTZ 攝影機建構室內環境監控系統作臉部辨識 陳志銘 (國立中央大學資訊工程研究所碩士論文)

◎表格參考

▼表一：人臉檢測研究綜述梁路宏 艾海舟 徐光佑 張鈸 pp2. (清華大學電腦系，智慧技術與系統國家重點實驗室，北京)

◎圖片來源

▲圖 3.1：http://www.me.ncu.edu.tw/teacher/Subject/B91_2/A/ME564 /Ch_5.doc

▲圖 3.2~3.4：Introduction to Digital Image Processing with Matlab pp237、243.

▲圖 3.5：http://nmlab.cs.nchu.edu.tw/index.php?blogid=1&archive= 2004-05 (國立中興大學資訊科學系-網路多媒體實驗室 嚴志軒)

▲圖 3.6 ~3.9：http://www.me.lhu.edu.tw/~tin/PPT/4-Grayscale/ 43-GrayMorphology.files/frame.htm

▲圖 4.1、4.5、4.6：A Robust Skin Color Based Face Detection Algorithm Sanjay Kr. Singh1, D. S. Chauhan2, Mayank Vatsa3, Richa Singh3*

▲圖 4.2：http://neural.ee.ncku.edu.tw/Links/MTable/Course/Image/ Chapter06-Art.pdf