以單晶片實現之嵌入式人臉偵測系統

魏先鴻　　魏守德　　鄭芳炫
中華大學資工系　　弘光科技大學資工系　　中華大學資工系
e09602017@chu.edu.tw　　shouderwei@gmail.com　　fhcheng@chu.edu.tw

***摘要*—人臉偵測在許多方面有很廣泛的應用，例如安全監控、家用機器人、互動性電玩、行車安全與智慧型家電等，嵌入式系統中實現人臉偵測演算法的需求非常強烈。雖然在 PC 上已經有快速又精確的人臉偵測演算法，但是這些演算法需要大量的記憶體資源與運算資源，需要進一步的改良才能在嵌入式系統上實現。目前已有一些用 ASIC 實現的硬體人臉偵測線路與在高單價的 DSP 上實現的人臉偵測系統，這些系統通常價格較高，不易普及到各種應用中。單晶片具有低耗電與低成本的優勢。本文提出了一個用單晶片實現的人臉偵測系統，其中的分類器採用 OpenCV 已訓練好的人臉偵測器。輸入影像為 160x120，最後結果 frame rate 為 2 frames/sec，在資源方面使用 RAM 28K bytes 與 ROM 35 K bytes。**

***關鍵詞*—人臉偵測、單晶片、嵌入式系統**

一、簡介

人臉偵測方法迄今已有十幾年，已有各式各樣的方法被提出[4]，例如 feature-based [2]、利用膚色的方法[14]、Eigenface [5]、Neural Network [6][7]、Support Vector Machine [8]與 Adaboost [9]等。Sung [6]、Rowley [7]與 Osuna[8]在差不多時間提出了高偵測率的人臉偵測方法，這三個方法都是用機器學習的方法來訓練人臉偵測系統。Viola and Jones [9]提出了一個精確的即時人臉偵測演算法，他們用很容易計算的 haar-like feature，再利用 Integral Image 來加速特徵的計算。這個方法利用 adaboost 來挑選合適的 weak classifier，來組成一個高偵測率的 strong classifier。他們也利用一個 cascade 結構來早期剔除掉不可能是人臉的區塊，以達到即時偵測。

人臉偵測有很廣泛的應用。一開始只是用做人臉辨識的先前的步驟而已[5]。當 Viola and Jones [9]推出第一個在 PC 上實現的即時人臉偵測系統之後，各式各樣的應用也愈來愈多。像是在數位相機上，可利用偵測到的人臉資訊來做更精確的白平衡、自動對焦與自動曝光等。在智慧型家電上，如電視或冷氣，自動偵測人臉存在與否來開啟或關閉家電，以達到節能的效果。在行車安全上，可用來偵測駕駛者的精神狀態，再以聲音提醒駕駛者。另外還可以應用在電子寵物，家用機器人，安全監控，互動性電玩上。

雖然在 PC 上已有很成熟的演算法，想要達成上述的應用，將人臉偵測演算法用嵌入式平台實現是有必要的。目前已有不少廠商在 DSP 平台上開發人臉偵測演算法，也有些廠商直接將人臉偵測演算法 ASIC 化。這類的產品成本較高。目前尚未有單晶片版本的人臉偵測平台出現。單晶片在全世界被普遍使用，開發工具齊全，多半具有低耗電，低成本的優點。我們提出了一個以駿億電子公司的單晶片為平台的人臉偵測系統，其中分類器採用 Intel OpenCV [10][11]裡面的人臉偵測器。

本文的架構如下，在第二節描述人臉偵測系統。第三節簡介 Intel OpenCV 的人臉偵測器。第四節說明人臉偵測方法的複雜度與所需要的資源。接下來第五節詳述我們所提出來改進後的方

法。最後則是結果與結論。

二、人臉偵測系統架構

人臉偵測的目的是在輸入影像中，找到不同位置與不同大小的所有人臉。人臉偵測系統包含了一個 moving window 在輸入影像上移動，用來框出不同位置的 subimage，一個 downscale 的單元，用來縮小輸入影像，以便找到不同大小的人臉。還有一個人臉分類器，用來判斷每個 subimage 是否為人臉，如圖 2 所示。系統的運作方式如圖 1 所示，利用 moving window 在輸入影像上掃瞄，取出每個位置的 subimage，再推入 face classifier 判斷是否為人臉。

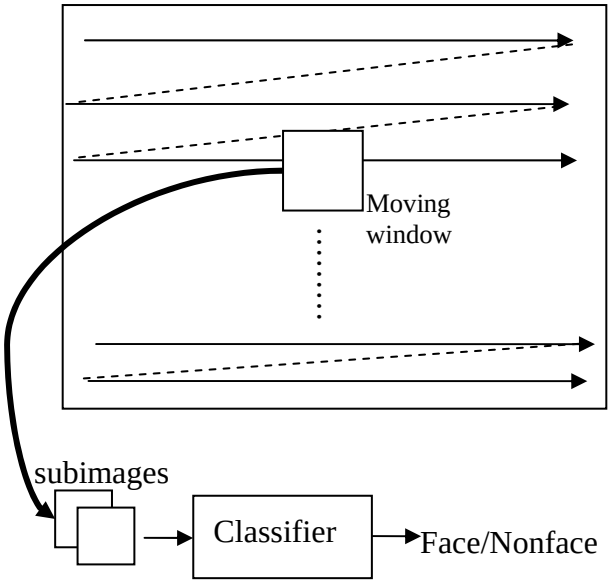

圖 1. 人類偵測系統運作方式。利用 moving window 在整張輸入影像上掃描，將框到的 subimage 用分類器來判斷是否為人臉。

當目前的層數的影像被掃瞄完成後，經過 downscale 單元把原來的影像縮小，通常都是以 1/1.2 的比例來縮小。如圖 2 所示。縮小的層數與系統想要偵測到最大的人臉有關，也與輸入影像有關。

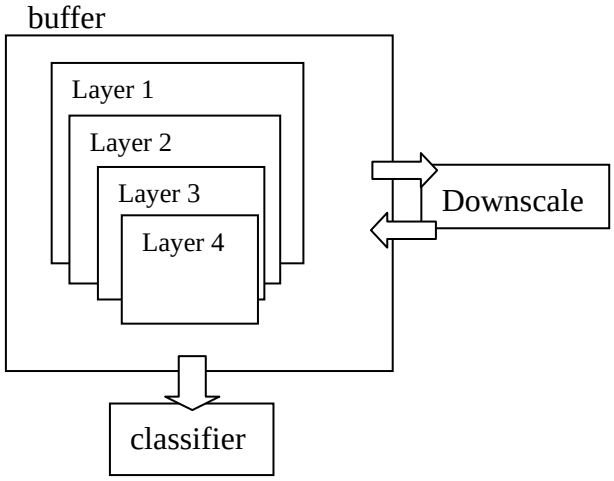

圖 2. 人臉偵測系統架構

三、分類器(Classifier)簡述

在我們的單晶片人臉偵測系統中，我們使用 Intel OpenCV 已經訓練好的人臉偵測器。此人臉偵測器是 boosting-based 的方法，利用簡單的區塊像素值總合相減後的差值當做特徵值，此種特徵值稱為 Haar-like feature，每一個特徵皆可形成一個弱分類器。利用 boosting-based 的方法可以在訓練階段從 training data 中找出較有鑑別力的弱分類器，這些弱分類器加上權重可以組合成一個強大的分類器。為了快速計算 haar-like feature，引進了 Integral Image 的概念。此方法更進一步採用了 cascade 架構，使人臉偵測器運算更快速。底下描述此方法的運作方式。

(一) Haar-like feature

人臉有很多地方有高低反差，像是額頭比眼睛亮，嘴角的陰影比臉頰還要黑，這些區塊像素值總合相減之後的差值當做人臉偵測時的特徵。此種特徵稱為 haar-like feature。這類的特徵的好處是簡單且容易計算，用 Integral Image 可以加速特徵的計算，以達到即時偵測的目的。總共有五種 haar-like feature，如圖 3 所示。白色區塊內的像素值的總合，與黑色區塊的像素值總合之間的相差值當做 feature。

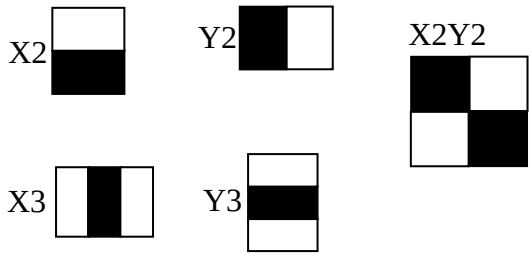

圖 3. 五種 haar-like feature

為了更容易表達與運算，特徵的計算方法如下：

$$f_I(x) = \sum_{i \in I=\{1,\cdots,N\}} \omega_i \cdot \mathrm{Re}cSum(r_i) \qquad (1)$$

上述的式子所表示的意思是任意個區塊總合 RecSum 的 weight sum 即是特徵值。N 是區塊的個數，權重 ω_i 屬於實數域，r_i 是在 subimage (24x24)裡的任意區塊。這樣的特徵範圍太大了，在實作上限縮了特徵的範圍，只使用如圖 3 的五種特徵，其中白色區塊與黑色區塊的大小是一樣的。

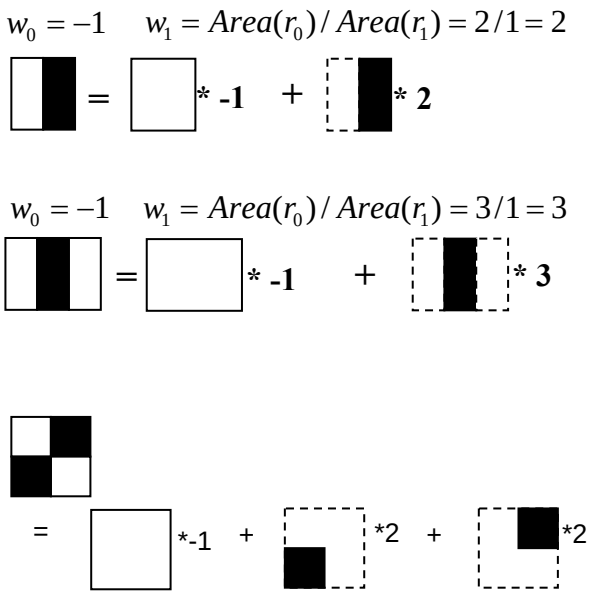

圖 4. 三種 haar-like feature 計算方式與其權重值。

二個區塊的權重不同號，而且與其大小有關。權重不同號是為了計算區塊的差值，而以區塊大小的倍數當做權重。比如說 haar_x3，大白塊比小黑塊大三倍，所以白塊的權重是-1，而黑塊的權重是+3。而 haar_x2，因為大白塊比小黑塊大二倍，所以大白塊的權重為-1，而小黑塊的權重則為+2，如圖 4 所示。

(二) Integral Image

Integral Image 用來加速計算 harr-like feature，其定義如圖 5 與式 2 所示。先對每 row 累加，再對每 column 即可得到 Integral Image。因為原始影像的像素值是正值，所以 Integral Image 的左下角的值，永遠大於右上角的值。有了輸入影像的 Integral Image 我們就可以用簡單的運算得到影像中任意大小區塊的像素值總合，如圖 6 所示，區塊 A 內像素值總合，可用 Integral Image 中的 a+d-b-c 得到。

Integral Image 內每個元素的大小與輸入影像大小有關。若輸入影像大小為 640x480，則 Integral Image 每個元素所需要的大小為 640x480x255，即 27bit。

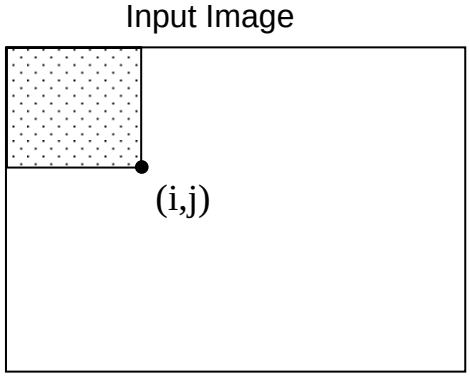

圖 5. Integral Image 的定義與圖示

$$II(x,y) = \sum_{i \le x} \sum_{j \le y} I(i,j) \qquad (2)$$

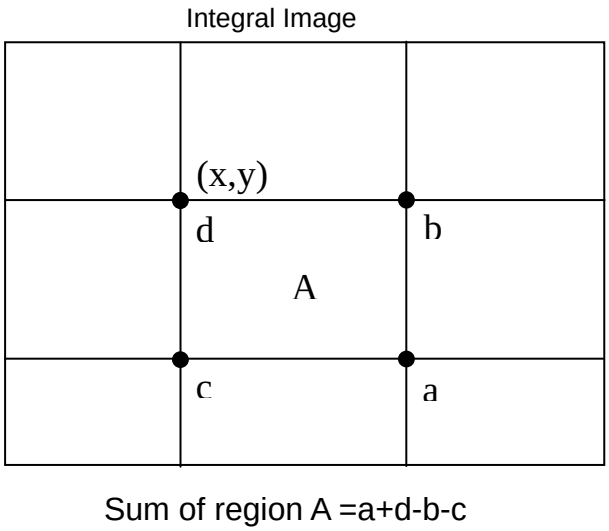

圖 6. 利用 Integral Image 可以快速的計算出區塊內像素值的總合。區塊 A 內像素值總合，可用 Integral Image 中的 a+d-b-c 得到。

(三) Weak classifier and strong classifier

Boosting 演算法的精髓就是將許多弱分類器(weak classifier)結合成一個強分類器(strong classifier)。

弱分類器的定義如下：

$$h_i(x)=\begin{cases}\alpha_L & f_i(x)<\theta \\ \alpha_R & f_i(x)\geq\theta\end{cases} \quad (3)$$

其中 $f_i(x)$ 是式 1 中定義的 haar-like 特徵值，θ 是門檻值，而 α_L 與 α_R 分別是人臉與非人臉的機率值。這些值都是在訓練階段，由 positive data 與 negative data 所決定。

強分類器的定義如下：

$$StageOutput=\sum_{i=0}^{i<wcNum} h_i(x) \quad (4)$$

$$\begin{cases} StageOuptut > StageThreshold & pass \\ otherwise & reject \end{cases} \quad (5)$$

由多個弱分類器加總所組成。當其輸出值大於門檻值時，則此子影像為人臉。

(四) Cascade structure

強分類器是由多個弱分類器所組成，其運算複雜度與弱分類器的個數有關，一個精確的強分類器通常有上百到上千個弱分類器。在一張影像中，大部份都是非人臉，人臉佔極少數。以 VGA 大小為例，若 moving window 一次移動一個 pixel，downscale factor 為 1/1.2 時，共掃瞄 10 層的話，總共有 84 萬個 subimage，其中是人臉的 subimage 大概不會超過 100 個。這 84 萬皆經過一個強分類器來分辨是否為人臉，會很耗時。

為了要達到快速的人臉偵測，利用多個強分類器形成階層式結構來代替一個強分類器。如圖 7 所示，每一個 stage 是一個 strong classifier。每個 stage 的主要任務是過濾非人臉。每個 stage 的在訓練時要求其在寬鬆的人臉標準下，盡量過濾掉非人臉。在訓練時，後一層的 stage 的非人臉資料，是經過前一層已訓練好的 stage 過濾後所剩下的，愈後面的非人臉會愈來愈少，也愈來愈難分類，所以愈後面的 stage 會有愈多的弱分類器。大部份的非人臉在前幾個 stage 就會被過濾掉。最後通過所有 stage 的 subimage 即為人臉。

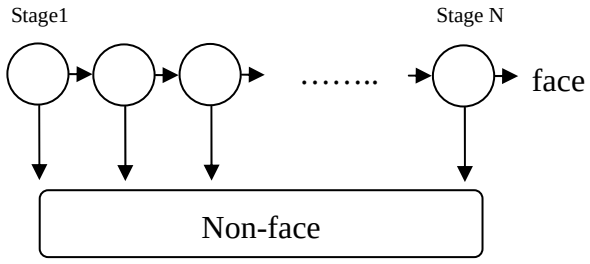

圖 7. 每一個 stage 是一個 strong classifier，其主要任務是過濾非人臉，最後通過所有 stage 的 subimage 即為人臉。大部份的非人臉在前幾個 stage 就會被過濾掉。

(五) lighting compensation

由於輸入影像可能在不同的光源與不同的曝光下所拍攝，會造成人臉的亮度不一樣，所以必須做光源補償(lighting compensation)。光源補償如下列式子所示：

$$\bar{I}(x,y) = \frac{I(x,y) - \mu}{c\sigma} \quad (6)$$

每一個像素值減掉區塊像素的平均值後，再除以整個區塊的 variance。我們可以利用 integral image 得到像素平均值，計算 variance 需要區塊內像素值平方總合，這個值可以由一個 Squared Integral Image(SII)快速計算得到，其定義如下：

$$SII(x,y) = \sum_{i \le x} \sum_{j \le y} I(i,j)^2 \quad (7)$$

四、人臉偵測方法的複雜度與資源使用度

人臉偵測系統用一個 moving window 掃過所有可能的 subimage 來偵測輸入影像中不同位置的人臉。為了要偵測不同大小的人臉，系統還必須多層次的縮小輸入影像，再重新掃瞄一次。以 VGA 大小的輸入影像為例，當 moving window 大小為 24x24，而每次只移動一個 pixel 時，則必須要掃瞄 28 萬次((640-24)x(480-24))。若設定最大偵測人臉大小為 256x256，輸入影像 down-scale factor 為 1/1.2 時，掃瞄 10 層時，總共有 84 萬個候選區塊，系統的複雜度很高。

在判斷每個 subimage 是否為人臉時，都要對記憶體做存取。而因為 subimage 太多了，造成記憶體頻寬需求量很大。為了要使整個系統達到準確又快速，縮減記憶體使用量，加速 classifier 的判斷都是必要的。

在原來的方法中，使用 Integral Image (II)來快速計算 Haar-like 特徵，使用 Squared Integral Image (SII)計算 variance，用來做光源補償。以 VGA 來計算，II 裡面最大值為 255x640x480，每個元素將近要 27bits。而 SII 最大值為 255^2x640x480，也將近需要 35bits。記憶體需求太大。

為了在嵌入式系統上實現，許多人提出不同的方法來。Wong and Chiu[12] 用硬體線路來實現人臉偵測系統，他們主要概念是利用一個 Local II 來取代整張 II，以減少記憶體使用量。每個 moving window 都產生一個 Local II，由於 Local II 數量太多，他們設計了一個特殊的硬體線路來快速的計算 moving window 的 local II。

某些方法為了要減少 subimage 的數量，採用一些 pre-filter 來過濾掉大量的 subimage。最常用的是利用膚色來過濾[2][14]。這類的方法的缺點是膚色模型並不穩定，很容易受到環境光源、相機白平衡、相機自動曝光等等的因素的影響。尤其在強光下，膚色幾乎變白色，很容易出錯。

上述的方法都是利用特殊的硬體線路完成人臉偵測系統，不易修改。利用 DSP 為平台者，則有高成本的缺點。因為單晶片廣泛使的被應用在不同的場合，發展工具齊全，而且俱有低耗電低成本的優點，我們提出了一個在單晶片上實現的人臉偵測系統。我們的人臉偵測演算法提出了一個精簡式的 II 來代替原本的 II。我們也利用 line buffer 來計算 moving window 內像素值平方總合，來代替 SII。而且針對 weak classifier table 中的 alpha 值做化簡，使得儲存空間縮小。我們也提出了一個 early rejection 的改良人臉分類器。

五、改良式人臉偵測系統

目前大部份即時人臉偵測演算法都在 PC 上實現。單晶片的資源並不像 PC 那麼的多，想用單晶片來實現人臉偵測方法需要針對記憶體空間的消耗做改良。人臉偵測演算法對記憶體讀取非常頻繁，由 moving window 裁切出的候選區域非常的多，記憶體讀取上需要很大的頻寬。為了達到省電與即時的目的，演算法所需要的記憶體空間需縮減到能塞入晶片內部記憶體之中。我們所提出來的方法有精簡型的 Integral Image，使用

line-based accumulative buffer 來計算 moving window 內像素值平方總合，精簡 classifier table 與 early rejection 機制，精簡 haar-like feature 儲存等。

(一) 精簡式 Integral Image

Integral Image 可以快速的計算輸入影像上任意大小區塊像素值的總合，可用來快速計算 harr-like 特徵。Integral Image 的定義如圖 6 與式 2 所示， 其大小與輸入影像大小有關，以輸入影像大小為 VGA 為例，Integral Image 每個元素需要 27 bits。事實上，在人臉偵測方法中，並不需要計算任意大小的區塊像素加總值，最大的 haar-like 特徵為 subimage 的大小。以圖 3 的 haar_X2 為例，最大的區塊的大小為 subimage 的一半。Integral Image 內部的元素只要能記錄最大區塊像素值總合的最大值即可。傳統上 II 的左下角的值一定會大於右上角的值，但是我們並不需要忠實的累加原始像素值，我們只需要 Integral Image 裡面四個點的差值就可以計算出區塊總合值。換句話說，在建立 II 時為了不讓累加值溢位所以需要比較多的記憶空間。但在計算區塊，我們只需要知道 a, b, c, d 之間的差值就好，再利用其關係 b>d, c>d，即可輕易計算出區塊像素值總合。

建立 Integral Image 時若像素加總值大於最大值時，就任其溢位，其值會變小。我們已經知道 a 大於 b, c 與 d，而且 b>d , c>d。當我們利用 II 來計算區塊像素值總合時，若 a 的值比 b, c 與 d 還要小，我們即可了解，a 真正的值是 a 加上溢位前的最大值。只要區塊像素值加總不超過 II 內部每個元素的儲存空間，即可用這個精簡的 II 正確且快速的計算出來。

舉例來說，圖 8 是一個 1 byte 的 Integral Image。其中 a 的值是 15，而 b, c 與 d 的值都比 a 大。我們即可知道 a 實際上的值是 256+15。

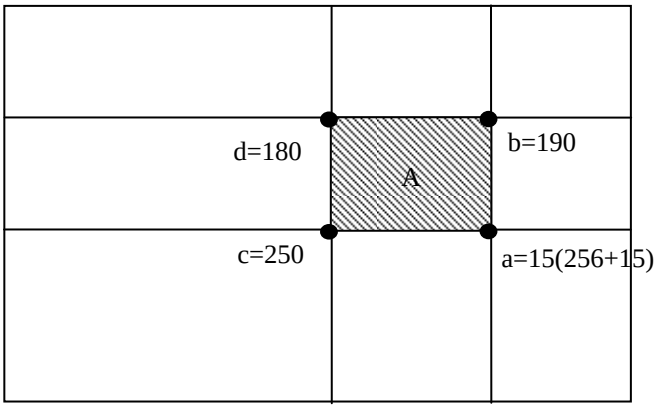

圖 8. 圖示 1bytes 的 Integral Image

在我們實作上，最大的 haar-like feature 的區塊為 24x12，我們只取 high-nibble 捨棄 low-nibble。則此最大區塊像素值的總合值最大為 24x12x16，不管輸入影像有多大，我們所使用的 II 每個元素只需要用 2 bytes 來記錄即可。

(二) 使用 line-based accumulative buffer 計算區塊像素平方值總合

如第二節所述，人臉偵測演算法利用每個 pixel 除上 subimage 的 variance 值來克服在不同光影變化下拍攝到的人臉的問題。我們可以用 subimage 裡像素值的平方和來快速計算 variance。傳統的實現方法是建立一張 Squared Integral Image (SII)，再用 moving window 在 SII 上 4 個角落的點，如圖 6 般計算出區塊內像素值平方的總合。此 SII 比 II 還要耗記憶體，當 frame 大小為 VGA(640x480)時，SII 上的每個元素需要 35 bits (256^2x640x480)，整個 SII 需要 1.3M bytes 左右。在 face detection 方法中，每個 subimage 只需要一個 variance 值，而且 subimage 的大小是固定的。我們不需要計算任意大小區塊的像素值變異數，所以我們使用 line-based accumulative buffer 來代替 SII。

我們的做法是對 moving window 內的 subimage 的每條 column 上的 pixel 平方值總合存到 line buffer 之後，再沿著 line buffer 加總，即

可得到 subimage 內部所有 pixel 平方值的總合。而由於 moving window 是順序移動的，所以要計算下個 moving window 所框到的 subimage 的 variance 值時，我們只要在 line buffer 內，減掉前一個值，加上最後一個值即可。

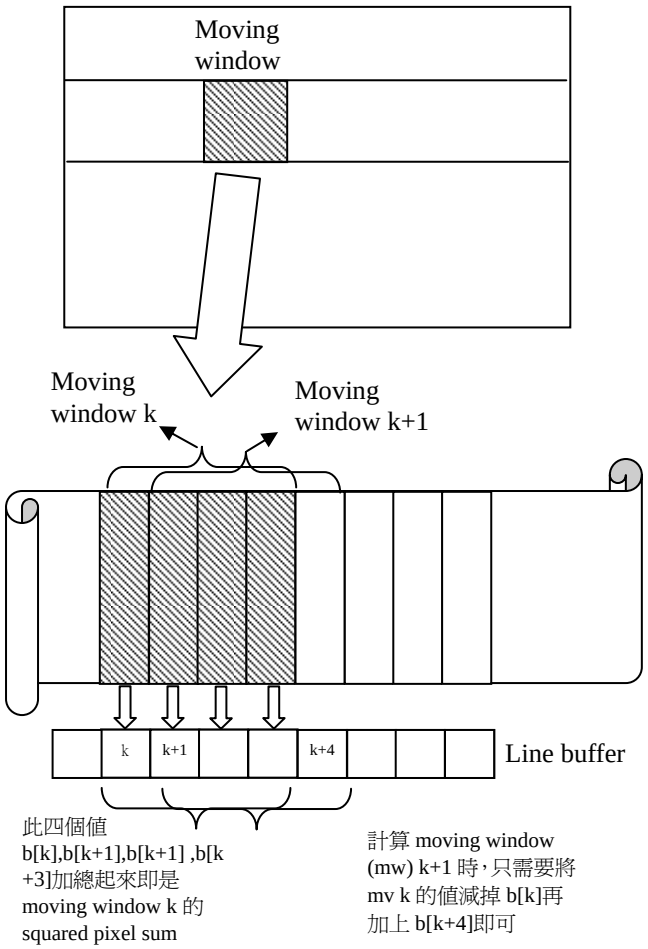

圖 9. Line-based square sum calculation，以 4x4 的 moving window 為例。

如圖 9 所示，我們以大小為 4x4 的 moving window (mw) 為例，其中 mw_k 與 mw_{k+1} 只差一排 column。mw_k 的四排 column 內的像素值平方加總後，分別儲入 line buffer 的 b[k], b[k+1] , b[k+2] , b[k+3]之中。此四個值是 mw_k 的 squared pixel sum。計算 mw_{k+1} 時，只需要將 mw_k 的值減掉 b[k]再加上 b[k+4]即可。

line buffer 的長度與輸入影像的寬度一樣，而 line buffer 每個元素的大小，則與 subimage 的大小有關。在我們的實作上，輸入影像為 160x120， subimage 大小為 24x24，所以 line buffer 中最大值為 160x24x24x256，每個元素需要 4 bytes 的儲存空間。也就是說，line buffer 的大小為 160x4 bytes。

(三) 化簡 classifier

原本的 classifier 有 102,128 bytes，有化簡的必要。其中有兩個 alpha 值，每個 alpha 值用定點數儲存需要 4 bytes 的空間。為了要降低 classifier 的儲存空間，我們將其中一個 alpha 省略掉。詳述如下：

我們將 stage 內所有的 weak classifier 的 α_L 相加，成為 default stage output。每個 weak classifier 只儲存 $\hat{\alpha}_R = \alpha_R - \alpha_L$。此時 stage 的運算方式可用下列式子表示。

$$StageOutput += \begin{cases} 0 & f_i(x) < \theta \\ \hat{\alpha}_R & f_i(x) \geq \theta \end{cases} \qquad (8)$$

我們的概念是，先假設所有的 weak classifier 每次都是 $f_i(x) < \theta$，預先把 α_L 加到 stage output 內。當 $f_i(x) < \theta$ 的時，不做任何事。而當 $f_i(x) \geq \theta$ 時，我們猜錯了，必須要在 stage output 值中，加上 α_R，再減去 α_L，此即為 $\hat{\alpha}_R$ 的值。如此，我們在每個 stage 中多儲存一個 default output，但我們卻可以省下每個 weak classifier 的 α_L。

我們還加入了 early rejection 的觀念，在不需要計算全部的 weak classifier 的 haar-like feature 值的情況下，預先就判定某個 subimage 是否被剔除。我們的做法是，先將 $\hat{\alpha}_R$ 的值轉成負的，再對 $\hat{\alpha}_R$ 值的大小做排序。每次計算 weak classifier 時，當 $f_i(x) \geq \theta$，加上 $\hat{\alpha}_R$，會使得 stage output 值愈來愈小。當 stage output 最後小於 stage threshold 時，此 subimage 則會被剔除。有了這樣的機制後，不需要將整個 weak classifier 都做完，即可提前將此 subimage 剔除。當 $\hat{\alpha}_R$ 並不是負的，我們將 haar-like feature 的計算顛倒過來，以 haar_x2 來說，原本來大白塊的權重是

負的，改成小黑塊是負的。這樣一來就能保證所有的$\hat{\alpha}_R$的值是負的，我們用扣分的概念，達到早期剔除的效果。還有對 haar-like feature 儲存方式做調整以減少儲存空間。最後總共需要 34,956 bytes，因為 classifier 是固定的，我們將此 table 儲放到 ROM 裡面。

六、實驗結果

由於人臉偵測時需要大量讀取記憶體，所以我們希望能與所有的工作空間都塞入 IC 之中，以避免外部記憶體頻寬不足的問題。我們採用駿億電子公司的單晶片編號 KBDM810，其工作時脈為 64MHz，效能差不多是 64MIPS。其內部有 ROM 128K bytes、SRAM 48K bytes 與 PROGRAM RAM 80K bytes。此單晶片有支援 1 個 clock 做乘加的指令，在計算圖 4 中有乘加動作的 haar-like feature 時非常合適。我們利用第四節所述的方法將 OpenCV 上的 face classifier 化簡，整個輸入影像、精簡式的 II，計算 subimage 變異數時所用的 line buffer 與 classifier table 都可置入晶片中。

我們的系統中，輸入影像為 160x120，moving window 為 24x24。Downscale factor 為 1/1.2，將影像縮小至小於 moving window 為止，共有 8 層。Moving window 每次跳兩個 pixels，總共有 28,602 個 subimages。最後我們的結果是 2 frames/sec。

在資源使用方面，我們儲存整張的輸入影像，共需要 19,200 bytes。建立 Integral Image (II) 時只取 high nibble，加上 haar-like feature 最大為 24x12，所以 II 的每個元素只需要 2 bytes。我們並沒有用一整張的 II，而是用 160x25 的帶狀 II，如圖 10 所示，當 moving window 掃描到下一條 row 時，捨棄上一條 row，加入新的 row 即可，總共需要 160x25x2=8000 bytes。實作圖 10 時，每次捨棄一條 row 並加入新的 row 時，必需將整個帶狀 Integral Image buffer 中每條 row 都向上 shift 一列，搬動 24 條 buffer 相當的耗時。此單晶片提供硬體架構 circular buffer 可以有效減少資料的搬移。

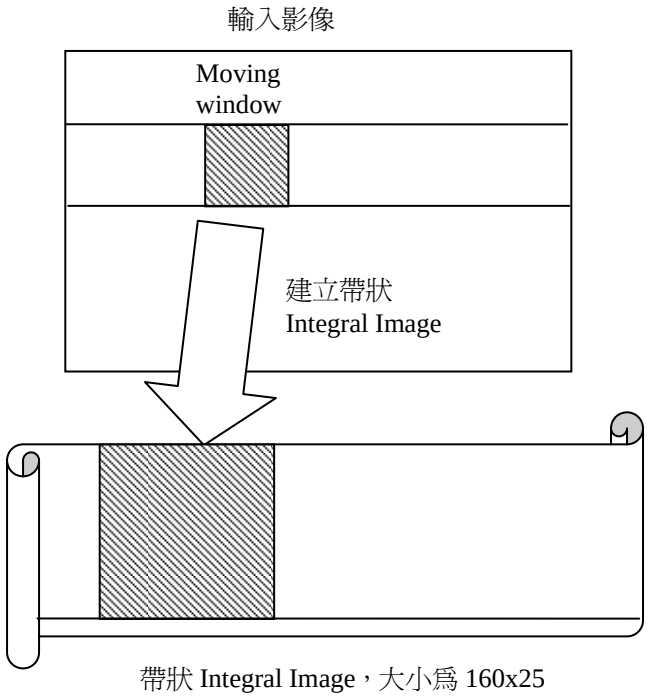

圖 10. 帶狀 II，每個元素 2 bytes，共需要 8000 bytes(160x25x2)。

表 1. 人臉偵測系統所需要的記憶體總表

	所需空間(bytes)	總計(bytes)
輸入影像	19,200	27,840 (RAM)
帶狀 II	8000	
line buffer	640	
wc table	34,956	34,956 (ROM)

原本需要用 Squared Integral Image (SII)來快速計算 variance，用於補償光源不平均，現在只需要一條 line (即 160x4 bytes)。原本 classifier 內的 classifier table 需要 102,128 bytes，用我們的方法每個 weak classifier 省掉了一個 alpha 值，並且對 haar-like feature 儲存方式做調整，最後總共需要 34,956 bytes。因為 classifier table 是固定的，我們將此 table 儲放到 ROM 裡面。表 1 列出我

們的人臉偵測平台所需要的記憶體資源，RAM 28K bytes 而 ROM 需要 35 K bytes。

七、結論

我們提出了一個以單晶片為平台的人臉偵測系統。單晶片具有低耗能低成本的優勢，非常普及於各種嵌入式系統中。我們所提出來的與影像大小無關的精簡式 Integral Image，適合應用在各種使用 haar-like feature 的物體追蹤、辨識或偵測演算法。在未來我們也計劃應用此平台發展行人偵測、物件追蹤、人臉辨識等系統。期望能將複雜的演算法應用於真實世界中。

誌謝

感謝駿憶電子公司提供協助。

八、參考文獻

[1] 陳育瑞, 陳培殷, “即時多人臉測晶片實作”, 成功大學資工所碩士論文。

[2] 張榮勝,張文鐘,”使用膚色比例前處理之即時性人臉偵測系統”,交通大學電信所碩士論文。

[3] G. Yang and T. S. Huang, “Human Face Detection in Complex Background,” Pattern Recognition, vol. 27, no. 1, 53-63, 1994.

[4] M.H. Yang, D.J. Kriegman, and N. Ahuja, “Detecting Faces in Images: A Survey”, IEEE Trans. PAMI, Vol. 24, pp.34-58, 2002.

[5] A. Pentland, B. Moghaddam, and T. Starner, “View-Based and Modular Eigenspaces of Face Recognition”, In Proc. IEEE Conf. on CVPR, pp. 84–91, 1994.

[6] H.A. Rowley, S. Baluja, and T. Kanade, “Neural Network-Based Face Detection”, IEEE Trans. PAMI, Vol. 20, pp. 22–38, 1998.

[7] K.K. Sung and T. Poggio, “Example-Based Learning for View-Based Human Face Detection”, IEEE Trans. PAMI, pp. 39-51, 1998.

[8] E. Osuna, R. Freund, and F. Girosi, “Training Support Vector Machines: An Application to Face Detection“, Proc. IEEE CVPR, pp. 17-19, 1997.

[9] P. Viola and M. J. Jones, “Robust real-time face detection“, in International Journal of Computer Vision, Vol. 57, no. 2, pp. 137-154, 2004

[10] R. Lienhart and J. Maydt, “An extended set of Haar-like features for rapid object detection”, IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[11] Intel’s open source computer vision library, http://www.intel.com/technology/computing/opencv/index.htm.

[12] Wei-Shu Wong and Ching-Te Chiu,“Design and Implementation of a Boosted Cascade Based Face Detection System”, Master thesis in Department of Computer Science, National Tsing Hua University。

[13] K. Khattab, J. Mitteran, J. Dubois, J. Matas, "Embedded System Study for Real Time Boosting Based Face Detection" IEEE Industrial Electronics, IECON 2006, pp. 3461-3465.

[14] Rein-Lien Hsu, Abdel-Mottaleb. M., Jain,A.K., “Face Detection in Color Images” IEEE Trans. on PAMI, Vol.24, Issue:5, pp.696-706, May 2002