具高辨識度的邊緣追蹤演算法之 SOC 設計

李燿宇
嘉義大學資工系
goodmorning7337@yahoo.com.tw

章定遠
嘉義大學資工系
dychan@mail.ncyu.edu.tw

徐超明
嘉義大學資工系
rchsu@mail.ncyu.edu.tw

摘要

這篇論文提出一個可以從一個封閉區段物體圖片中取出一個具有高度形狀保存，並且把邊緣的多餘物去除的邊緣偵測的設計，這篇所提出來的方法主要可以分為兩大部分來考慮，一個是 progressive boundary linking (PBL) 方法與 synchronous redundancy pruning(SRP)方法。所提出來的設計是針對 IP 核心部分使用 AMBA 匯流排作 SOC 設計，與現今的方法，也就是數學型態學邊緣取出演算法來做比較下，以硬體 SoC 的角度來看，所提出的方法的硬體架構實現，無論是更精確的取出邊緣，計算效率，以及硬體的使用率等等，都會有較佳的效能。

關鍵詞：嵌入式系統晶片 (SoC)、即時影像處理 (Real-Time Image processing)、邊緣追蹤 (Contour Tracing)、邊緣取出 (Contour Extraction)

Abstract

A (System on a Chip) design of contour tracing with shape-preserving and redundancy removal for segmented object image is presented in this paper. The proposed design consisted of two processes named progressive boundary linking (PBL) and synchronous redundancy pruning (SRP). The proposed design is realized on a space-qualified SoC with open-source processor IP core utilization AMBA bus. SoC realization results shows that, compared with existing methods such as mathematical morphology contour extraction, the proposed design achieves better results in precise detecting the contour, computational efficiency and in spatial cost.

Keywords: SOC (system on chip)、Real-time Image processing、Contour tracing、Contour extraction.

1. 介紹

邊緣追蹤或邊緣取出為一個可以從數位圖片中精確取出物體邊緣 (object contour)的方法，並且要保留住物體本身的邊緣特色，也就是與原圖片的邊緣要差異性不大，才算是一個精確的邊緣取出方法。而一個精確的具有形狀保存特性的邊緣追蹤方法，無論是在圖片是否有無被雜訊污染或邊緣有附著多餘物的情形下都可以取出精確的邊緣。這麼做的目的是為了達到圖片高品質 (因為去除了多餘雜訊)，低資料量，並且最重要是，使得圖片的邊緣部份的資訊免於因為作了某些運算而遺失，可以有更好的物體辨識度。因為取出邊緣通常是很多的數位影像處理應用的前處理，例如類型辨識（Pattern Recognition），它依據了物體的形狀特色所做的辨識，跟目標追蹤 (target tracing)，是依據物體姿勢所做的追蹤應用，以及型態編碼等等的應用。為了為往後的應用處理使用，一個好的並且具有高度物體辨識度的邊緣取出演算法是必須的。

當數位圖片再圖片取得時(數位化過程)或圖片在網路之間作傳遞的動作，有可能會遭受到雜訊的污染而使得物體邊緣的辨識度降低。在圖片取得階段，例如像機本身的亮度或溫度等等都會造成圖片有雜訊的產生，而圖片在網路間傳遞，無論內部外部的溝通都會使得圖片本身產生多餘的雜訊。不明顯的物體邊緣可能會造成所取出的物件邊緣難以辨識。一些不必要的雜訊，將會使得物體邊緣擷取之後的型態辨識變的困難。特別是在兩相異物體邊緣很靠近的時候又有雜訊產生在它們邊緣部份產生的雜訊，這樣的狀況將會使得邊緣取得更加困難，所以雜訊對邊緣取得的困難度是有很大影響的。

而物體的邊緣部份，可能會有一些多餘物(redundancy)存在，過多的多餘物附著在物體邊緣上，也會影響到物體的辨識度，並且也會對往後的影像處理應用的效率上造成影響，越

多的多餘物，往後的處理將會更耗費效能並且錯誤性更高，所以這些多餘物也必須要被去除掉，才能夠取得一個具有高辨識率，並且也為往後的應用作一個很好的前處理步驟。在這篇論文中多餘物會附著在邊緣上並且可能被視為是雜訊來處理，其他的邊緣取得演算法以此為前提作處理的話，其實是有很大的風險的，因為無法保證所消除的是否是雜訊，換句話說，也有可能被消除的是邊緣，這將會造成取出錯誤的邊緣。在這篇論文中，多餘物可以包含了斷裂點 (crack)與分支 (branches)這兩種，斷裂點會造成邊緣破碎，而分支會使得物體邊緣的辨識度降低，使得邊緣模糊。所以本篇論文提出一個使用邊緣追蹤的方式，並且同步的去除掉多餘的雜訊與多餘物，可以得到一個具有高辨識度，高邊緣保留性的一個物體邊緣，將可以作為是往後影像處理應用的一個很好的前處理工具。

在論文第二部分，我們先介紹本篇論文的相關工作。本篇論文的核心演算法，雜訊與多餘物去除的邊緣追蹤演算法將會在第三部分作介紹。而所提出方法的硬體架構與實現將會在第四部份作描述。而所提出方法的 SoC 硬體實現的一些實驗結果將會再第五部分作討論。第六部分則對此論文所提出的方法作一個結論。

2.相關工作

數位影像處理程序[9]對圖片作雜訊去除通常是第一步驟，能夠讓圖片的品質提升。雜訊去除的硬體架構因所使用的演算法而不同。一個基本的非線性率波 (non-linear filtering)濾波器，稱為是中值濾波器 (median filter)，它的硬體實現的架構在[3,10]作介紹。在[3]中介紹了中值濾波器的處理是使用一個 3 x 3 大小的遮罩 (mask)，以這遮罩內的像素值作一個排列的動作，並且取出一個中位數(median)作為結果，因為使用硬體很容易實現此 3 x 3 大小的遮罩架構，這是其中一個使用硬體實現來做影像處理的理由。此方法提出一個排序方法叫垂直-水平-對角線排序法(vertical-horizontal diagonal sorting)來提升傳統使用氣泡排序法來做排列所花的時間複雜度，當然硬體的需求也因此降低。在[10]中，另一個使用硬體實現的理由就是，它的架構使用了平行處理 (parallel process)的機制在[3]的垂直-水平-對角線排序法上，藉著使用硬體的平行化 (parallelism)處理機制，可以幫助雜訊去除處理再一個時脈週期 (clock cycle)之間達到一個最高的硬體效能運用。

當對圖片中的雜訊作去除的動作之後，接下來要做的就是把物體的邊緣取出。硬體實現相關的邊緣取出演算法如[11,15]。在[15]中提出了邊的偵測與角 (corner)的偵測，所依據的演算法為 SUSAN 演算法，同樣的它的架構也是依據了平行處理的精神來改善單位時脈的硬體效能使用率。另外，數學型態學(mathematical morphology)方法[7,14,16]也是一個既簡單又有效率的邊緣取出方法。在早期的 80 年代就已經對數學型態學演算法作一個硬體上的運算分析[4]，而數學型態學邊緣取出方法，基本上是由兩個運算子 (operator)所組成的，它們分別是膨脹 (dilation)與侵蝕(erosion)，藉著交互使用這兩個數學型態運算子，可以取出邊緣。

.

使用硬體的技巧，FPGA[1]開發版的協助可以對影像處理程序提供一個有效率的方法，無論是在計算所花費的時間跟硬體使用空間上，都有一個較佳的表現。為了能夠更近一步的改進複雜的演算法的效能，一些硬體邏輯或算數運算能力，例如 ALU 運算的改善如[8]所討論。然而硬體的演算法架構設計，對本身演算法的複雜度利用 SoC 實現作一個改善，是很重要的。架構必須考慮到平行處理[2,6,12,13]將會大大的加速演算法的運算時間並且最重要的是，達到影像處理程序的即時性的要求。

具有去除多餘物的邊緣追蹤演算法，由[5]所提出，跟現今的邊緣取出演算法來做比較之下，所提出的方法的計算將會比較少量。跟數學型態學方法作比較，數學型態學方法無論是在做雜訊去除或是邊緣取出處理，都必須要對全部整個圖片使用 3 x 3 大小或更大的遮罩作處理，將會消耗更多的 ALU 運算時間。然而[5]所提出的方法相較之下不會去計算圖片的所有 3 x 3 大小的遮罩，取而代之的是沿著物體的邊緣作追蹤，並且同步的去除掉多餘物，所需的運算會較少。並且，這邊緣追蹤方法在追蹤處理結束前，也就是形成一個封閉物體邊緣之前，會使用一個迴圈的機制，將可以大幅的縮減所需的硬體使用空間。在[5]中所提出的邊緣追蹤演算法之硬體 SoC 架構將會在這篇論文中作描述，並且 SoC 使用一些測試圖片所做的分析結果將會在

下面章節作討論。

3.方法描述

由[5]所提出的具形狀保存的邊緣追蹤演算法主要是由兩個步驟組成，稱為是 progressive boundary linking (PBL)機制與 synchronous redundancy pruning (SRP)機制。當初始邊緣點不管是由圖片的垂直方向或水平方向作偵測，一但決定了初始的邊緣點之後，PBL 機制就會開始作處理。PBL 是用一個 3 x 3 大小的遮罩，目的是要計算在這遮罩內的點的優先順序 (priority)，就是為了要決定在這遮罩內哪個點要作為是下個邊緣點。在做邊緣追蹤處理的時候，當一旦決定了一新邊緣點之後，SRP 機制會同步的偵測並且去除掉多餘的點，一但去除了多餘點或沒有多餘點存在，PBL 的追蹤處理將會繼續運作，一樣的會藉由計算點的優先順序並且決定下個邊緣點並且遮罩中心會移動到下個邊緣點上以繼續做追蹤處理。一但發現沒有新的點可以藉由 PBL 決定的話，整個追蹤處理就會終止。所提出的具有邊緣形狀保存的邊緣追蹤演算法的細節如下所述。

I. PBL 機制

PBL 邊緣追蹤處理中，使用的是選擇最大優先權點策略，主要是沿著物件的邊緣來選擇最有可能的邊緣點，並且把這個點加入到一個邊緣陣列 (contour array)來建構出一個連續的邊緣點的鏈結。由上所述，我們必須要有一個鏈結陣列來儲存邊緣點並且連結由 PBL 機制所選出來的連續點。我們使用一個 3 x 3 大小的遮罩來計算每個點的優先順序 (priority)並且選擇最大的優先順序的點當成是下一個的邊緣點，並加入到邊緣陣列中。以一個 3 x 3 大小的遮罩，中心點不計算它的優先順序，其餘八個點都要去計算它們的優先順序，在這裡我們會用到平行處理的觀念，同時(concurrently)計算這八個點的優先順序節省所花費的時脈週期。當一個新的點加入到邊緣陣列時，我們就要考慮目前為止我們所連接的邊緣點，如果判斷在這之中有多餘的點存在的話，我們就會去執行 SRB 機制，否則的話我們一樣繼續作 PBL 追蹤的處理，一樣用遮罩去找到下一個邊緣點並且把它加入到邊緣陣列中，當在追蹤下一個邊緣點時，如果找不到下一個新的邊緣點的話，那整個追蹤處理方法就會終止，PBL 的處理程序如下：

步驟一、 先選擇一個值得信賴的物體邊緣點作為第一個邊緣點 並且把它加入到邊緣陣列的第一個位置。

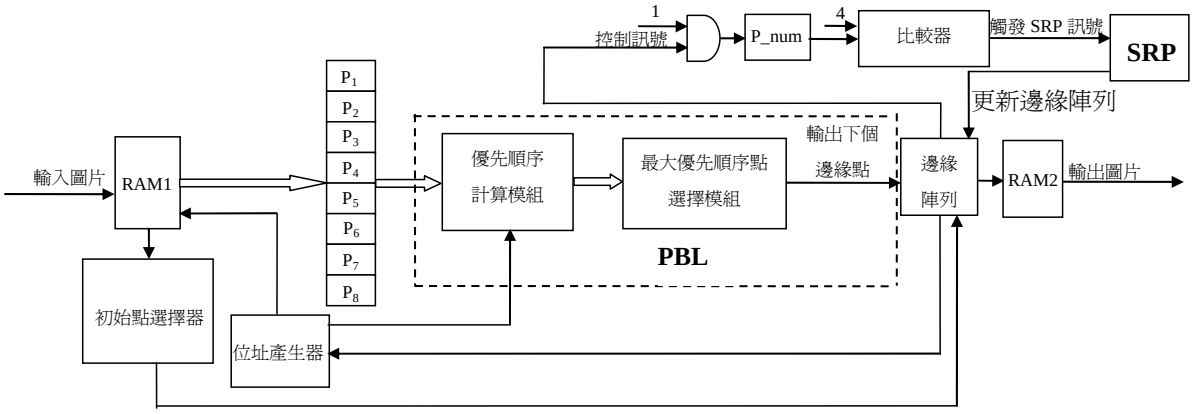
步驟二、 把遮罩中心設在目前的邊緣點位置上。

步驟三、 應用選擇最大優先權點策略(這方法會在後面作說明)去決定下一個邊緣點，並且把新決定的點加到邊緣陣列中。

步驟四、 如果 PBL 無法在步驟三找到新的點加入邊緣陣列中，那整個追蹤處理方法就終止，否則就會呼叫 SRP 機制，沿著已經在邊緣陣列的點找到多餘的點並對它們作修剪去除的動作，然後就會再回到步驟二繼續的對目前的邊緣點做追蹤的處理。

II.SRP 機制

這個機制是用在移除邊緣存在多餘點上，有可能是裂縫或是分支 (cracks or branches)，當我們決定了下一個邊緣點的時候，此機制就會開始從先前已連接過的邊緣點來判斷是否有斷裂點或分支產生，往後，我們考慮新加入的點前面三個邊緣點，如果判斷在這四個點之間有斷裂點或分支的話，將會對斷裂點作填補，有分支的話就去除掉。SRP 機制又可再分為兩個階段作處理，在第一階段會去判斷是否先前的連接點之間是否有存在斷裂點，如果有的話我們就會使用邊緣的像素值對這個斷裂點作填補的動作，並且會對邊緣陣列作更新的動作來表示我們加入了這個點。如果判斷沒有斷裂點存在或是把斷裂點填補上了之後，SRP 機制會進行到第二階段，在此階段會判斷是否這些點之間存在著一個多餘的分支，如果這些點之間存在著分支的話，我們把這些分支去除掉並且也一樣對邊緣陣列作更新的動作，即是刪除掉這些被視為是分支的邊緣點。到此完成了 SRP 機制的處理，目前所連接的邊緣點多餘的點已經被去除，並且再回到 PBL 繼續追蹤下個點，當有新的點被決定，SRP 機制會在對這些點作判斷並且修剪掉不必要的邊緣點，如此一來，沒有附著多餘點的邊緣即可藉由此方法取出。



圖一 邊緣追蹤演算法之硬體架構方塊圖

4.硬體架構設計

根據 PBL 機制與 SRP 機制的基本描述，將提出來一個適合邊緣追蹤處理演算法的 FPGA 硬體架構，如圖一所示。

圖一基本上的硬體構成有幾個部分。記憶體元件 (memory components)、一個位址產生模組 (address generating module)，跟幾個重要的主計算模組 (main calculating modules) 構成。記憶體元件是硬體用來對輸入圖片作儲存跟對輸出圖片作儲存以便傳回個人電腦用。而位址產生模組則是產生可能會被選為下個邊緣點的所有候選邊緣點。主計算模組是由初始點選擇器 (initial point selector)、PBL 模組，跟 SRP 模組所組成的。初始點選擇器是去選擇第一個加入到邊緣陣列的邊緣點，從這一個點開始作 PBL 追蹤處理。而 PBL 跟 SRP 模組則是在先前所描述的處理模式。一開始我們先對這整個圖一的執行路徑作一個描述，之後我們在對每個元件作更細部的解釋。

I. 執行步驟

如圖一所示，一開始我們要處理的圖片，將會被儲存到 RAM1 記憶體中，第一次的追蹤處理步驟，首先我們必須要先作一個初始點的決定，使用初始點選擇器先選擇一個確切的邊緣點並且把這個點加入到邊緣陣列中。接下來處理的步驟為，根據目前點的所在圖片中的位置，依據位址產生模組，去從 RAM1 中抓取目前點的周圍八個點的二元值並存到 P_1 到 P_8 這八個暫存器中，接著就會開始作 PBL 的處理選出這八個點的其中一點作為是下個邊緣點並且依序的把新決定的邊緣點連接在前一個點的後面，到此結束第一次的追蹤處理。下個處理將會開始在由前一次所決定的邊緣點上，再一次的使用位址產生模組產生下個邊緣點的候選點並且執行 PBL 模組決定下個邊緣點並且在儲存新的點到邊緣陣列中。這是一個迴圈的機制，當沒有新的點被 PBL 決定的話，那整個*追蹤*處理就結束了。而當 PBL 決定了一個新的點，SRP 模組就會開始由先前所連接過的點判斷是否有斷裂點或分支存在其中，有做修剪刪除多餘點的動作的話，還要更新邊緣陣列的內容，最後會再回到這個迴圈機制繼續的追蹤新的邊緣點。

II. 架構元件描述

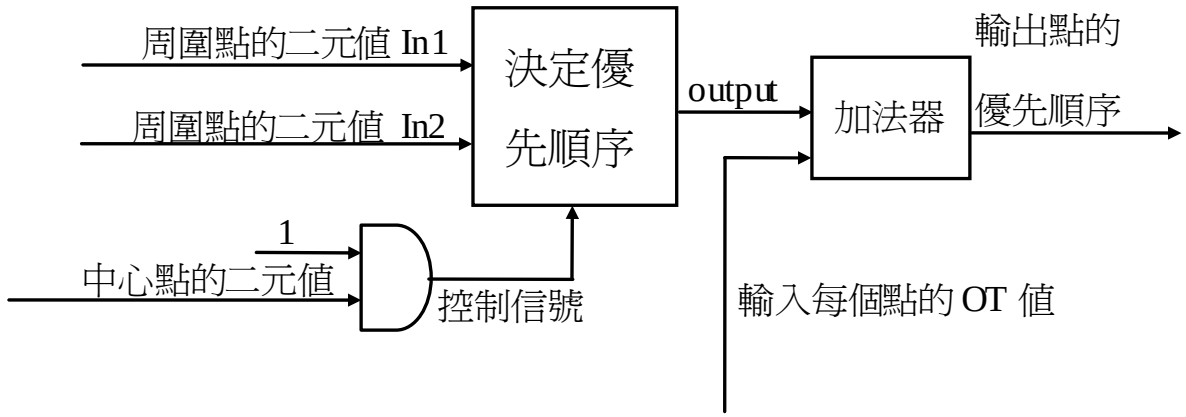
A. 記憶體元件

a. 兩個 RAMs

RAM1 儲存要被處理的影像(每個點的二元值是用一個 bit 的儲存空間)，RAM2 是儲存結果影像並且把影像由 FPGA 傳送到 PC 端。

b. 八個暫存器(P_1 到 P_8)

每個暫存器儲存單一點的二元值(每個暫存器有一 bit 的記憶空間)，這八個暫存器的目的在於存放目前邊緣點的周圍八個點的二元值，來組成一個遮罩就能夠在接下來的 PBL 模組處理。



圖二 優先順序計算模組-細部方塊圖

c. 一個暫存器(P_num)

這個暫存器儲存在邊緣陣列內邊緣點的數目，一開始初始為 0，當有一個新的邊緣點被加入到邊緣陣列，這個暫存器的值就會加 1。

d. 邊緣陣列

這個陣列儲存由 PBL 決定的邊緣點，當一個新的點被決定，這個陣列會在目前邊緣點的後面加入這個新邊緣點的資訊。點的資訊包含了點的座標跟這個點在邊緣陣列中出現的次數(occurrence times 縮寫為 OT 值)，每個點的 OT 值一開始都為 0，如果一個新的點加入到這個陣列中，這個點的 OT 值就會加 4，而當一個新的點加入到這陣列時，這個陣列也會傳送一個控制信號 令 P_num 加 1。

B. 位址產生模組(address generator)

這模組會產生一連續的記憶體位址，並依據此位址去 RAM1 讀取目前邊緣點周圍八點的二元值並且儲存到 P_1 到 P_8 暫存器內建立出一個遮罩，為了後面的 PBL 處理。另外，這個模組也傳送每個點的 OT 值給位在 PBL 模組內部的優先順序計算模組(priority calculation module)計算優先權用。

C. 主計算模組

a. 初始點選擇器

這個選擇器決定邊緣陣列內的第一個邊緣點，決定的方法是，首先選擇圖片內的一固定的 y 座標，然後再由 x 座標由左自右的順序作搜尋來找到一個合適的邊緣點，這個點一但決定後就會把它的資訊儲存到邊緣陣列中。如此一來，我們就有了第一個邊緣點，接著再利用 PBL 模組繼續的對物體邊緣做追蹤的處理。

b. PBL 模組

這個模組是由兩個子模組 (sub-modules)所組成，優先順序計算模組 (priority calculation module)跟最大優先順序點選擇模組(selection of maximum priority point module)這兩模組所組成。PBL 會先由優先順序計算模組計算每個點的優先順序，之後再依據最大優先順序點選擇模組比較它們優先順序，選出擁有最高優先順序的點作為是下一個邊緣點。

子模組 I. 優先順序計算模組

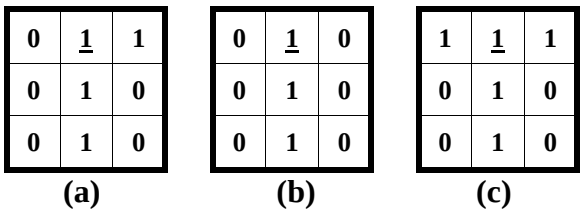
這個模組會去計算每個候選點的優先順序(P_1 到 P_8)，如圖一所示這個模組將會有八個一樣模組並行的計算這八個候選點的優先順序。圖二為優先順序計算模組的一個細節架構圖，而計算優先順序的判斷如下。首先，如果中心點的二元值為 1 的話，那這個點就可能是下個邊緣點的候選點。再來我們考慮此點的邊緣點(同時是中心點的 3 x 3 遮罩範圍內的點也是此點的四鄰點)，共有兩點，是否僅有一點的二元值等於 0，如果滿足的話，那 output 值會被設為 0。如果這兩點的二元值都為 0 的話，那 output 值會被設為 1，如果以上的條件都不滿足，那此點將會被設為"不考慮(don't care)"，表示此點將不會被選為是下個邊緣點的候選點。

表一所示為優先順序決定 (priority decision)方塊圖的真值表，“x”表示不考慮這個點作為是下個邊緣點的候選點。如圖二所示，有三個輸入，其中兩個是相鄰點的二元值，另一個輸入則是控制信號(control signal)，如果它是 0 的話，表示這個點絕對不是邊緣點，不會在考慮到另外兩個輸入的部分，其 output 一定是”x”。

表一
優先順序決定方塊圖-真值表

In1	In2	control	output
0	0	0	x
0	0	1	1
0	1	0	x
0	1	1	0
1	0	0	x
1	0	1	0
1	1	0	x
1	1	1	x

為了要更明確的對優先順序計算模組，如何運作的作一個說明，如圖三給定一個 3 x 3 大小的遮罩，劃有底線的點跟它兩個鄰近點的關係，其 output 是如何決定的，在圖三作一個明確的描述。圖三(a)中，劃有底線的點跟它兩個鄰近點的關係為，僅有一個鄰近點的二元值為 0，所以 output 為 0。在圖三(b)中鄰近點的二元值皆為 0，所以 output 為 1。而(c)皆不滿足上述的兩種情況，所以 output 為”x”。



圖三 (a) 劃有底線點的 output = 0, (b) 劃有底線點的 output = 1, (c) 不考慮此底線點作為是下個邊緣點之候選點。

output 的結果再與每個點的 OT 值使用加法器加起來就會得到最後的結果，每個點的優先順序了，如下式所示。

$$Priority = OT + output \qquad (1)$$

一旦所有點的優先順序都被計算好了，再來就會從這幾個點中選出一個有最大優先順序的點，藉由執行下個子模組，最大優先順序點選擇模組來決定這個最大優先順序點。

子模組 II. 最大優先順序點選擇模組

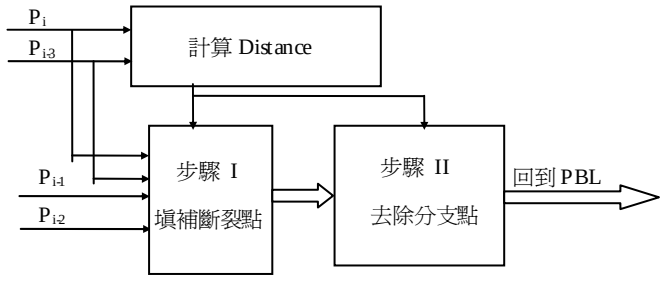
這個模組作的事情很簡單，目的是選擇一個擁有最大優先順序的點作為輸出。我們用 2-input 的比較器來比較兩個點的優先順序，藉著依序的比較兩個優先順序並保留較大的優先順序點，最後的輸出將會是擁有最大優先權的點，我們再把它送到邊緣陣列並存起來，整個 PBL 模組的處理就結束。

c. SRP 模組

因為 SRP 模組需要至少有四個邊緣點才可以對邊緣是否存在著多餘點作判斷，因此在架構上，在暫存器 (P_num)的值大於 4 的情況之下，SRP 才會觸發，並由圖一的比較器，分別輸入 P_num 變數跟 4 這兩個數字作比較，如 P_num 大於 4 的話產生的觸發訊號即為 1，表示 SRP 觸發，如為 0 表示未滿足條件，SRP 模組將會依據觸發訊號決定是否執行 SRP 運算，一旦 SRP 運算終了，將會更新邊緣陣列的結果，如有必要更改，SRP 會發出訊號 1 表示要更改，訊號 0 表示不更改。圖四為 SRP 的細部架構圖，圖中的 P_i 表示新加入的邊緣點，而 P_{i-1}、P_{i-2}，跟 P_{i-3} 依序是連接在 P_i 之前的點，把它們由邊緣陣列取出來做為 SRP 判斷的輸入。SRP 一開始要先計算 P_i 跟 P_{i-3} 這兩個點的距離(distance)，計算公式如下所示。

$$Distance(P_i, P_{i-3}) = max(|x_i - x_{i-3}|, |y_i - y_{i-3}|) \qquad (2)$$

如果距離為 2，那 SRP 會先執行步驟 I 的動作，判斷在 P_i 與 P_{i-3} 之間存在著一個斷裂點要把它填補起來。如果距離為 0 或 1 的話，就



圖四 SRP 細部架構圖

會執行 SRP 的步驟 II，表示在這兩點間存在著分支必須要被去除。SRP 會依序的由步驟 I 到步驟 II 作斷裂點填補跟分支去除的動作，SRP 一旦有任何的邊緣點的修剪動作，邊緣陣列也要跟著更新，斷裂點的加入與分支點的移除。接著 SRP 結束並且把處理權還給 PBL 繼續做下個邊緣點的追蹤。

圖五描述了如何用 SRP 作斷裂點的填補與分支點去除的一個範例，圖五(a)為原圖，圖五(b)為經過了 SRP 步驟 I 處理過後的結果圖，而(c)是經過了 SRP 步驟 II 處理過後的最後結果。

P_{i-3}	0	0
0	P_{i-2}	0
P_i	P_{i-1}	0

(a)

1	0	0
P_{i-3}	P_{i-2}	0
Pi	P_{i-1}	0

(b)

1	0	0
P_{i-3}	0	0
P_i	0	0

(c)

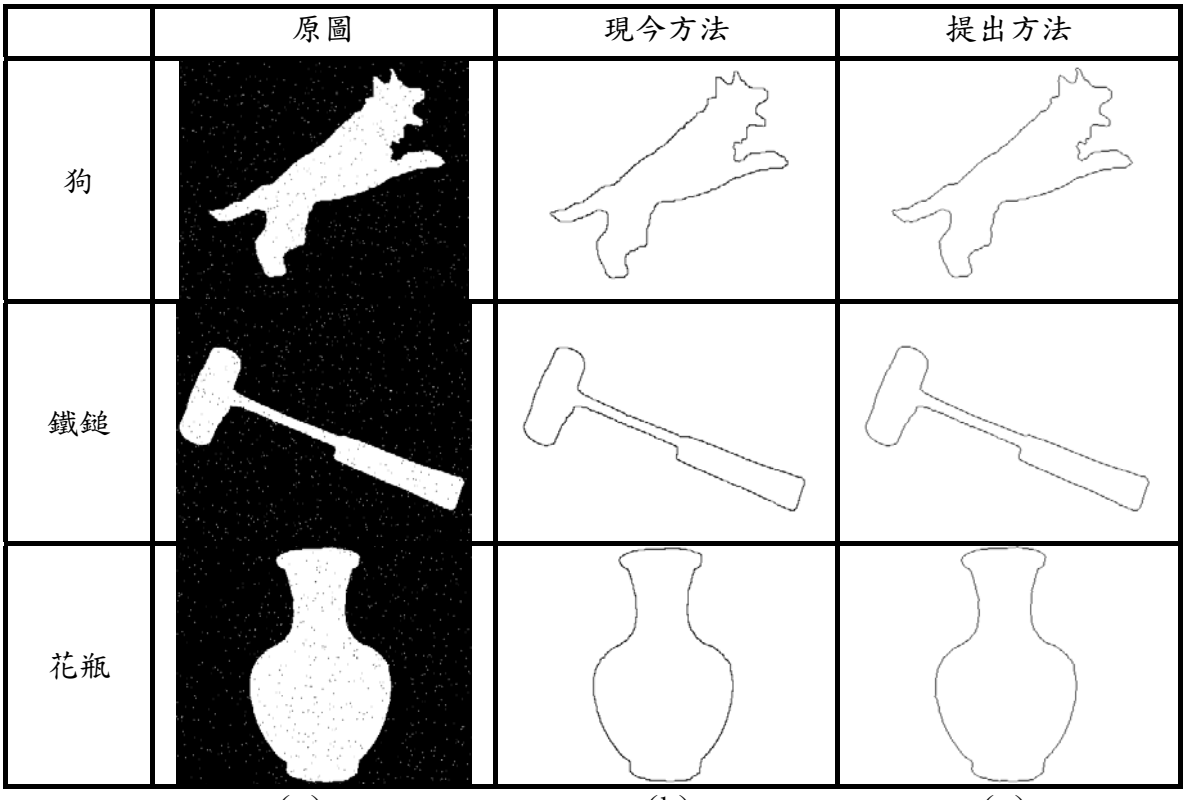
圖五 SRP 模組 (a) 步驟 I, Distance(P_i,P_{i-3})=2，填補斷裂點 (b) 填補斷裂點後的的結果，步驟 II, Distance(P_i,P_{i-3})=1，去除分支點 (c) 去除分支點後的結果。

5.實驗結果

此論文所依據的邊緣追蹤演算法將以 SOC 設計架構在 FPGA 上實現，並且跟現有的邊緣取出方法附加雜訊去除演算法(往後稱此方法為現今方法)作效能比較。現今雜訊去除與邊緣擷取的處理方法上，數學型態學是一個很簡單並且有效率的對二元圖片做雜訊去除與邊緣擷取的處理方法，現今的數學型態學方法對於雜訊去除會使用先侵蝕、再膨脹、在膨脹，最後侵蝕就可得到去除雜訊的影像。而邊緣取出有幾個步驟來做處理，主要是雜訊去除過的影像與除雜訊影像再作一次侵蝕，這兩張影像作 XOR 處理就可得到。

$$\text{邊緣取出圖片} = XOR(\text{去雜訊圖片}, \text{侵蝕(去雜訊圖片)}) \quad (2)$$

此論文中所提出的硬體架構跟現今方法將會被實作到 GR-XC3S-1500 發展板上，並以 Verilig 硬體描述語言(VHDL)來實現，對於提出的 SOC 架構方法與現今數學型態學方法，將會有三個效能的衡量與比較，它們分別是精確度(accuracy)比較，效率(efficiency)比較跟硬體空間使用率(spatial utilization)比較。



圖六 原圖(a)與經過 SOC 處理過的圖片，現今方法作處理(b)與提出方法作處理(c)過的結果圖

)

I. SOC 設計的精確度比較

BMAD (baseline-based mean absolute difference [5]是可以用來對取出的邊緣作精確度的衡量的一個標準，如下式(3)所定義。

$$BMAD = \frac{1}{N_p}\sum_{i=1}^{N_p} |\hat{d}_i - \bar{d}_i| \qquad (3)$$

其中，N_p 是基線(baseline)的點的個數，而每個點的值都是一個 BMAD 衡量的距離值，$\hat{d}_i$ 跟 $\bar{d}_i$ 是基線到第 j 個邊緣點的距離，分別是基線到所追蹤的邊緣跟基線到精確的邊緣的距離。而 BMAD 值是所有的 N_p 個距離差的平均，如上式所示，就是去對所追蹤過的邊緣跟精確無誤差的邊緣作差異衡量。當 $|\hat{d}_i - \bar{d}_i|$ 越小的話，BMAD 值也跟著越小，表示所追蹤的邊緣跟精確無誤差的邊緣也會有用小的距離差值，也就代表所追蹤的邊緣的精確度也會比較高。

在圖六中，原圖片先對它加入百分之一的雜訊再對它們作去雜訊與邊緣取出的處理。從圖六中可知，此論文所取出的結果圖片比起現今方法有較佳的邊緣保存性，如狗的喉嚨部分。現今方法與提出方法 BMAD 的比較如表二所示，根據表二的數據顯示，所提出的方法比起現今方法有較小的 BMAD 值，表示所提出的方法在所追蹤的邊緣跟精確邊緣之間有較小的距離差，也就表示所取出的邊緣較精確，跟現今的方法比較。我們可以藉由這些實驗結果得知，所提出的方法邊緣取出的精確度是較高的。

表二
現今方法與所提出方法對三張測試圖片的ＢＭＡＤ值比較

	現今方法	提出方法
狗	0.555823	0.021231
鐵鎚	0.541666	0.003558
花瓶	0.561203	0.010914

II. SOC 設計的效率比較

以 SOC 所處理的時間分析[13]而言，時脈週期是一個可以對計算複雜度的效率提供一個好的衡量工具，以本篇論文方法跟現今方法比較的時脈週期時間比較下表所示。由表三中顯示此論文方法在硬體實現上面比較起現今方法有較短的時間表現。

表三
現今方法提出方法的時間衡量比較

	現今方法	提出方法
所需時間 (時脈週期)	5158633	2764289

III. SOC 設計的空間使用率比較

邏輯電路使用率(logic utilization)跟儲存空間使用率[13](storage utilization)用來對兩個不同 SOC 設計作空間的比較。而門栓(latch)與正反器(flip-flop)是兩個基本的儲存空間的元件。四輸入查看表(4-input look-up tables (LUT))則是為了作邏輯計算用的元件。表四與表五，以現今的方法跟所提出之方法，分別對儲存空間使用率跟邏輯使用率做比較。在表四中，相較於現今的方法，所提出之方法需較少門栓，而需要之正反器數目則與現今的方法相當。在表五中，提出方法比起現今方法有較少的 LUT 數目，即是 FPGA 的邏輯使用率較高。

表四
現今方法與提出方法的儲存空間使用率比較

	現今方法		提出方法	
	正反器	門栓	正反器	門栓
暫存器數目	4,483	30	4446	4

表五
現今方法與提出方法的邏輯使用率比較

	現今方法	提出方法
LUTs	14358	14290

參考文獻

[1] Kamai S. Ali, “Digital circuit design using FPGAS,” ***Computers & Industrial Engineering***, Vol. 31, No. 1-2, pp. 127-129, 1996.

[2] J. Batlle, J. Martí, P. Ridao and J. Amat, “A New FPGA/DSP-Based Parallel Architecture for Real-Time Image Processing,” ***Real-Time Imaging***, Vol. 8, No. 5, pp. 345-356, 2002.

[3] Gavin L. Bates and Saeid Nooshabadi, "FPGA implementation of a median filter," *TENCON '97. **IEEE Region 10 Annual Conference. Proceedings of IEEE,*** Vol. 2, pp. 437 – 440, 1997.

[4] Damien Baumann and Jacques Tinembart, "Mathematical Morphology Image Analysis on FPGA," ***AISTA 2004, Luxembourg Poster session***, 2004.

[5] Din-Yuen Chan and Roy Chaoming Hsu, "Robust Shape-Preserving Contour Tracing with Synchronous Redundancy Pruning," unpublished.

[6] Oswaldo Cadenas and Graham Megson, "A clocking technique for FPGA pipelined designs," ***Journal of Systems Architecture***, Vol. 50, No. 11, pp. 687-696, 2004.

[7] Pinliang Dong, "Implementation of mathematical morphological operations for spatial data processing," ***Computers & Geosciences***, Vol. 23, No. 1, pp. 103-107, 1997.

[8] Veselko Gustin, "An FPGA extension to ALU functions," ***Microprocessors and Microsystems***, Vol. 22, No. 9, pp. 501-508, 1999.

[9] Rafael C. Gonzalez and Richard E. Woods, ***Digital Image Processing***, 2nd ed, Pearson Education Taiwan and Cau Li Publishing, 2003.

[10] A. Hazra, J. Bhattacharyya and S. Banerjee, "Real time noise cleaning of ultrasound images," ***Computer-Based Medical Systems, 2004. CBMS 2004. Proceedings. 17th IEEE Symposium***, Vol. 24-25, pp. 379 - 384, 2004.

[11] Stephan Hussmann and Thian H. Ho, "A high-speed subpixel edge detector implementation inside a FPGA," ***The Journal of Visual Communication and Image Representation on Real-Time Imaging***, Vol. 9, No. 5, pp. 361-368, 2003.

[12] D.K. Iakovidis, D.E. Maroulis and D.G. Bariamis, "FPGA architecture for fast parallel computation of co-occurrence matrices," ***Microprocessors and Microsystems***, Vol. 31, No. 2, pp. 160-165, 2007.

[13] David A. Patterson and John L. Hennessy, ***Computer Organization and Design:The Hardware/Software Interface***, 3nd ed, Morgan Kaufmann, 2005.

[14] Frank Y. Shih and Shouxian Cheng, "Adaptive mathematical morphology for edge linking," ***Information Sciences***, Vol. 167, No. 1-4, pp. 9-21, 2004.

[15] Cesar Torres-Huitzil and Miguel Arias-Estrada, "An FPGA architecture for high speed edge and corner detection," ***Computer Architectures for Machine Perception, 2000. Proceedings. Fifth IEEE International Workshop on 11-13***, pp. 112 - 116, 2000.

[16] Jun Yang and Xiaobo Li, "Boundary detection using mathematical morphology," ***Pattern Recognition Letters***, Vol. 16, No. 12, pp. 1277-1286, 1995.