An Efficient Run-Based Algorithm for Morphological Image Processing

基於長度編碼之

快速外形影像處理演算法

指導教授：連國珍 教授

研究生：林子皓 撰

中華民國一零六年七月

摘要

為了使外形影像處理更有效率，本論文沿用前人所提出的長度編碼表，從而提出一種新的演算法來進行外形影像處理。在傳統外形影像處理的過程中，運算往往會受到影像大小影響。在本論文中，我們將影像做長度編碼的壓縮，建成新的長度編碼表，並利用此長度編碼表，使外形影像處理的速度不再受到影像大小所影響，再利用外形影像處理中的基本理論來進行改進，加入移除影像中較短長度編碼的判斷技術，加速侵蝕的計算，除了應用在外形影像處理中，我們也可以將此長度編碼表進行連通區域標記，減少了影像轉換和重新建立長度編碼表的時間，實驗結果將證明本論文所提出的演算法運算速度高於傳統的外形處理演算法，提高外形影像處理的效率。

關鍵詞：外形影像處理、長度編碼、侵蝕、影像處理

Abstract

In order to improve the efficiency of morphological image processing, this paper presents a new algorithm by the previously proposed run-length table. In the traditional process of morphological image processing, the operation tends to be image size. In this paper, we compress image by run-length encoding to build a run-length table. Using the table, morphological image processing of speed no longer by image size by effect, and then use the basic theory of morphological processing to be improved, adding the decision technology to remove the shorter run of the image. This information is then used to speed up the calculations of erosion operator. In addition to morphological image processing, we can also use it on connected components labeling. Reduced time for image conversion and reestablishment of run-length tables. Experimental results show that the provided algorithm is much better than the traditional algorithm. Improve the efficiency of morphological image processing.

Keywords: Morphological Operators, Run-based Algorithm, Run-length Code, RLE, Erosion, Morphological Image Processing.

目錄

圖目錄

第一章　緒論

1.1　研究動機與目的

影像是泛指所有帶有訊息的信號，在現今的社會中已經被廣泛的使用，手機拍出來的照片、衛星影像圖、監視器所拍攝的畫面、雷達掃描、視訊會議等這些皆是影像的範疇，我們想從中獲取甚麼訊息，依照不同的需求，將同一影像用不同的技術來進行加工，再藉此分析，或者是在降低資料量的情況下不改變影像的品質和資訊，來增進影像傳遞的效率，這些都是影像處理衍生的應用。

而外形影像處理常常被運用為影像分析的前置處理，經過處理後的影像能夠更加方便的提取影像中的物件資訊。一個好的外形影像處理演算法能夠直接影響到整體工作的效率，為了能夠得到更好的結果，前人們也分別提出了許多演算法來使其進步，長度編碼表就是其中的產物。

本篇論文將沿用前人們所提出的長度編碼表，將二值化影像轉變為長度編碼表後使用提出的新的演算法來完成外形影像處理，處理後的長度編碼表進行連通區域標記，能運用在圖形分析和辨識，希望透過本論文所提出的外形影像處理演算法，來提升整體系統的效率。

1.2 相關工作與研究方法

由於科技的日益發達、網路的普及，影像的取得越來越容易，往往人們會需要從影像中得到許多資訊(監視器、指紋等)，相較於用肉眼分析，使用電腦分析的效率更能提升許多，因此影像分析進步的課題時常被提出，若想要分析存在於影像中的資訊，需要將二值化影像的前景提取出來，而對於已經分離前景和背景的二值化影像，時常會有雜訊存在或是有前景互相重疊在一起的情形，在這種情況下做影像的分析，往往會因為誤把雜訊當前景或前景互相重疊而得到一個錯誤的結果。

為了得到影像資訊，必須有效的去除雜訊，外形影像處理是最常見的使用方式，已經有許多改進外形影像處理演算法的方法被提出[3][9][10]，其中又可分為以像素點運算和基於長度編碼，傳統的影像處理方式都是以像素點運算來進行處理，本論文中所使用的 OpenCV 中的函數亦是以像素點運算來進行，但在處理影像的過程中，基於像素的傳統方法處理速度會受到影像大小所影響，為此有人對像素改進進而提出了利用編碼來處理影像，目前已經有許多不同編碼的外形影像處理演算法被提出，以下我們將詳細介紹編碼。

編碼是將有限的影像用二進位數字來進行表示，其目的除了便於電腦資料處理外，亦希望能達成資料壓縮。資料壓縮是研究原資料的特性，找出一種以最少位元來代表此資料，一般而言資料壓縮的程序是將原資料中重複表示的資

訊減低到最小，亦可稱為原編碼(source encoding)。而壓縮編碼的方式又可大致分為兩大類，失真編碼與不失真編碼[8]。

失真編碼的方式為壓縮後的影像還原回來和原始影像不同，影像的些微亮暗或粗細並不會影響人體對影像的認知，因為允許資料的損失，藉此可以做到較高的壓縮比(原影像的資料量和壓縮影像資料量的比值)。而長度編碼則是不失真編碼，對於某些不能有偏差的影像來說，長度編碼是最好的選擇，既能達到資料壓縮，資料亦不會有所遺失。

以像素點做運算，一個 N*N 像素的影像以一個 M*M 像素的結構元素(Structuring Element，以下以 SE 簡稱)做外形影像處理，所需要使用的時間複雜度$O(N^2M^2)$，在[1]中，作者提出先將影像透過長度編碼壓縮後，再應用於外形影像處理，透過[1]提出的長度編碼結構，將二值化影像中連續相同的像素值表示為集合，L 為像素結構轉為長度編碼表結構所壓縮的資料筆數，再以一個 M*M 像素的 SE 進行影像處理，所需要使用的時間複雜度則為$O(M^2L)$；[5]論文中延續[1]所提出的長度編碼表結構，將[1]實際求出影像平移的長度編碼方法，改進成只需要用指標來記錄影像平移的資訊，如此可以提高處理的速度也減少記憶體的耗費，所使用的時間複雜度亦與[1]相近為$O(M^2L)$。

在[2]中，提出了另一種長度編碼表結構，來做外形影像處理，[6]則是延續[2]的長度編碼表，並運用外形影像中的分析，來跳過一些可以減少判定的標準，改進外形影像處理可以更有效率的進行。透過[6]中的侵蝕演算法，我們有

了改進[5]的平移方式的想法，藉由改進平移的方式和順序，對於以一個 M*M 的 SE 做侵蝕，其時間複雜度可以降至$O(2ML)$。

在[4]中是先將影像透過長度編碼壓縮後，運用於連通區域標記法，傳統的連通區域標記法需要經過兩次的掃描才可以將前景的數量標記完成，[4]方法中，只需要一次性的掃描即可將影像標記完成，[7]改進了[4]方法後加入了新的結構，將相同標籤號的長度編碼連起來，加速了合併標籤的效率。

本論文提出了新的演算法，是針對 M*M 像素的 SE 來進行實驗比較，但實際上本演算法亦能適用於任意的 SE，運用[1][5]提出的長度編碼結構，再藉由[6]的方法來做改進外形處理演算法，加速外形影像處理演算法的效率，結合外形影像處理和連通區域標記不再需要轉換影像或長度編碼的結構，減少在轉換間所耗費的時間，提升整體效能。

1.3 論文架構

本論文章節編排如下，第一章論述研究動機與目的，第二章將介紹外形影像處理及長度編碼相關的背景知識，第三章將詳細介紹[5][6]中演算法及其理論，第四章將介紹本論文所改進的演算法，第五章為實驗結果，第六章為結論。

第二章　研究背景知識

本章將會介紹外形影像處理所會使用到的背景知識，從基本的結構元素(Structuring Element)、侵蝕(Erosion)、增長(Dilation)定義的介紹，以及外形影像處理的基本理論，長度編碼(Run-length encoded)與本論文使用的長度編碼表(Run-length Table)所有相關技術，這些背景知識都將逐一介紹。

2.1　外形影像處理

外形影像處理在處理影像的外形和骨架，是將一張二值化影像依照一些設定的轉換規則轉成另一圖形，如將圖加粗、變細、外框處理或找出圖形的骨架等處理，為影像處理的基本理論，常用的外形處理如侵蝕(Erosion)、增長(Dilation)、開運算(Opening)、閉運算(Closing)等基本運算。經過外形影像處理後，可以完成許多影像處理中的工作，例如：雜訊處理(noise suppression)、邊界探測(edge detection)、物件分割(object segmentation)等。

2.1.1　外形影像處理以像素點運算表示

外形影像藉由空間上點和點之間的關係定義出欲尋找出的外形，基本運算可分為侵蝕(Erosion)和增長(Dilation)，通常以 M*M 範圍來考慮圖形的關係

[8]，這裡以 3*3 做為舉例，圖形定義如下：

$$\begin{matrix} x_3 & x_2 & x_1 \\ x_4 & x & x_0 \\ x_5 & x_6 & x_7 \end{matrix}$$

圖 2.1　M*M 影像定義圖

$$g(i,j) = x \cap P(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) \qquad (1)$$

$$g(i,j) = x \cup P(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) \qquad (2)$$

(1)(2)分別為侵蝕和增長運算的通式，(1) 若$P(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$或 x 任意一方為 0，則輸出即為 0； (2)若$P(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$或 x 任意一方為 1，則輸出為 1。

$$g(i,j) = x \cap (x_0 \cap x_1 \cap x_2 \cap x_3 \cap x_4 \cap x_5 \cap x_6 \cap x_7) \qquad (3)$$

$$g(i,j) = x \cup (x_0 \cup x_1 \cup x_2 \cup x_3 \cup x_4 \cup x_5 \cup x_6 \cup x_7) \qquad (4)$$

(3)(4)分別為 8 鄰邊侵蝕與增長，侵蝕為 x 周圍有一點為 0，x 為 0；增長為 x 周圍有一點為 1，x 為 1。若要將影像左右侵蝕與增長，以下用(5)(6)公式表示：

$$g(i,j) = x \cap (x_0 \cap x_4) \qquad (5)$$

$$g(i,j) = x \cup (x_0 \cup x_4) \qquad (6)$$

2.1.2 結構元素(Structuring Element)

在數學形態學中(mathematical morphology)，是一種用來分析影像中幾何特性的理論與工具。在理論上，具有完整的數值特性作為基礎；在應用中，透過運用結構元素(Structuring Element 以下以 SE 來做簡稱)來處理影像。

外形影像處理以集合表示時，影像與任一集合運算得出侵蝕的結果，此集合我們稱之為結構元素， SE 最主要的兩個特徵為形狀和大小；從形狀來說，可以是球狀，亦可以是任意矩形，透過不同的形狀或特殊空間的設置來達成區分影像的功能；大小可以為 3*3 的矩陣，也可以為 9*9 的矩陣，藉由調整大小和形狀設定所需要的 SE，再進行影像處理來得到所需的目標。

2.1.3 外形影像處理以集合理論表示

給予一張已經二值化的影像 A，結構元素 B，這邊將透過集合理論，來介紹侵蝕(Erosion)和增長(Dilation)[5]，其定義如下：

$$A \ominus B = \{c|c = a - b, a \in A, b \in B\} \qquad (7)$$

$$A \oplus B = \{c|c = a + b, a \in A, b \in B\} \qquad (8)$$

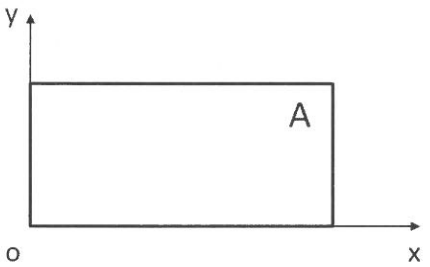
(7)和(8)分別代表侵蝕與增長，a 屬於影像 A 中的任一元素，b 則屬於結構中的元素，c 為 a 與 b 做二進制邏輯運算的結果。定義亦可用[6]來表示：

$$\varepsilon_B(A) = \{p \in \mathbb{Z}^d | B_p \subseteq A\} \quad (9)$$

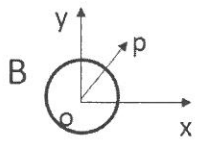
$$\delta_B(A) = \{p \in \mathbb{Z}^d | (B^t)_p \cap A \neq \emptyset\} \quad (10)$$

(9)和(10)亦為侵蝕與增長，用不同的方式來呈現定義，B_p是 B 做向量$p \in \mathbb{Z}^d$的位移，B^t為結構 B 的反射位置，(9)的意思是將 SE 做向量$p \in \mathbb{Z}^d$的位移仍包含於影像 A，此集合即為 A 的侵蝕，(10)的意思是將 SE 的位置反射至原點對側，做向量p的位移後交集於影像 A，此集合即為 A 的增長。以下將用圖例來做表示：

給予一張已經二值化的影像 A，結構元素 B：



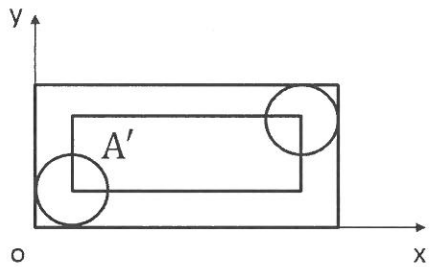
(a)影像 A



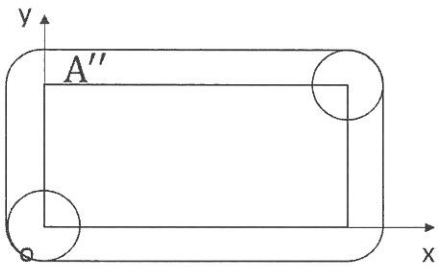
(b)結構元素 B

圖 2.2　影像 A 以結構元素 B 進行外形影像處理

以結構元素 B 對影像 A 進行侵蝕與增長，我們可得結果如下圖：



(c)A'為影像 A 受到侵蝕的結果



(d) A''為影像受到增長的結果

$$A \ominus B = \bigcap_{b \in B} A_b = B \ominus A = \bigcap_{a \in A} B_a \quad (11)$$

$$A \oplus B = \bigcup_{b \in B} A_b = B \oplus A = \bigcup_{a \in A} B_a \quad (12)$$

(11)和(12)則為(7)和(8)的延伸，$\bigcap_{b \in B} A_b$和$\bigcup_{b \in B} A_b$代表的是每一張影像間的運行方式，$\bigcap_{a \in A} B_a$和$\bigcup_{a \in A} B_a$則代表每一個像素點的運算方式。假設將 B 定義為：{(0,0),(1,0),(-1,0)}，對於(11)、(12)會得到以下結果：

$$A \ominus B = A_{(0,0)} \cap A_{(1,0)} \cap A_{(-1,0)}$$

$$A \oplus B = A_{(0,0)} \cup A_{(1,0)} \cup A_{(-1,0)}$$

A 與 B 的結果都是將原影像與向右向左位移的影像做二進制邏輯運算，藉此達成外形影像處理，與(5)(6)表示方式不同，但結果相同。上述是為(7)(8)兩式的介紹[5]，(9)(10)兩式則會用到$Z = \cup_{n=1}^{N} R_n$，Z 為一個二維的二值影像，N 是所代表的是 Run 的數量，這邊提到的 Run 是指[5]中的長度編碼表結構，長度編碼表會在下面做詳細敘述，R_n代表的是第 n 個 Run，將每一個 Run 聯集，即為二值影像 Z，在本論文中，我們將運用侵蝕來介紹此演算法。

2.2 SE 的平移理論

在執行外形影像處理，原點$o = (0, \ldots, 0) \in \mathbb{Z}^d$包含在 B 且$(o \in B)$，使用公式(3)來證明再進行侵蝕前可以藉由 SE 位移向量 q 使$o \in B_p$[6]。

證明理論：

$$\varepsilon_{\mathrm{B}}(X) = \left[\varepsilon_{\mathrm{B}_q}(X)\right]_q$$

使影像$\mathrm{X} \subseteq \mathbb{Z}^d$,結構元素 $\mathrm{B} \subseteq \mathbb{Z}^d$以及向量$q \in \mathbb{Z}^d$，我們可以得出

$$\left[\varepsilon_{\mathrm{B}_q}(X)\right]_q = \{p | B_{p+q} \subseteq X\}_q$$

$$= \{p + q | B_{p+q} \subseteq X\}$$

$$= \{\tilde{p} | B_{\tilde{p}} \subseteq X\}$$

$$= \varepsilon_{\mathrm{B}}(X) \qquad \textit{by subdtitution:} \tilde{p} = p + q$$

藉由這個理論來處理外形影像處理中的平移問題，使我們可以更有效的運用長度編碼表完成外形影像處理。

2.3 移除短的 Run

影像$X = \cup_{n=1}^{N} R_n^X$和 SE $B = \cup_{m=1}^{M} R_m^B$將這兩項長度編碼資訊代入侵蝕的例子會得到：$\varepsilon_{\mathrm{B}}(X) = \cap_{m=1}^{M} \cup_{n=1}^{N} \varepsilon_{R_m^B}(R_n^X)$，運用長度編碼來做侵蝕非常有效，但在合併和比較的運算往往花了更多效能，利用移除短的 Run 來減少合併的運算，使外形影像處理有更高的效率。

$$L_{min} := min_{m \in \{1,\ldots,M\}}\{|R_m^B|\}$$

$$X_{L_{min}} := \bigcup_{\substack{1 \le n \le N \\ |R_n^X| \ge L_{min}}} R_n^X$$

當影像中所有的 Run 都小於 SE 中的最小 Run，$|R_m^B| < L_{min}$，我們可以知道侵蝕後會沒有影像，$\varepsilon_{R_m^B}(R_n^X) = \emptyset$。

使$E_m^n \coloneqq \varepsilon_{R_m^B}(R_n^X)$

$$\varepsilon_{\mathrm{B}}(X) = \bigcap_{m=1}^{M} \bigcup_{n=1}^{N} E_m^n$$

$$= \cap_{m=1}^{M}([\bigcup_{\substack{1\le n\le N \\ |R_n^X|\ge L_{min}}} E_m^n] \cup [\bigcup_{\substack{1\le n\le N \\ |R_n^X|< L_{min}}} E_m^n])$$

$$= \cap_{m=1}^{M} \bigcup_{\substack{1\le n\le N \\ |R_n^X|\ge L_{min}}} E_m^n \qquad \text{因為當} |R_m^B| < L_{min} \text{時} \varepsilon_{R_m^B}(R_n^X) = \emptyset.$$

$$= \varepsilon_{\mathrm{B}}\left(X_{L_{min}}\right)$$

2.4 長度編碼(Run-length encoding)

長度編碼(Run-length encoding)，又稱為遊程編碼或行程長度編碼，是一種無損數據壓縮(即為不失真壓縮)的技術。目前已被運用的相當廣泛，主要是用在壓縮來源資料中重複的內容達到節省空間及減少傳遞資料的時間，儲存長度編碼內容的資料會因為用途而有所不同。舉例來說，一組資料串“AAABBCCCDD”，經過長度編碼可以將資料壓縮為 3A2B3C2D，資料量由 10 個單位降至 8 個，優點在於將重複性的資料量壓縮成小單位，但其缺點是若資料的重覆率不高，也可能導致結果資料量比原始資料大。

2.5 長度編碼表(Run-length Table)

影像經過二值化處理後，會產生只有由黑色和白色(背景與前景)所組成的影像，長度編碼表將後續研究分析需要使用到的資料做儲存，由於僅儲存所需要的資訊，減少了全部影像的完整資訊，故可以降低資料量，來提升影像傳遞的效率。這裡將介紹[5][6]提出之長度編碼結構做說明：

[5] 提出的長度編碼結構只儲存前景(1)和背景(0)，連續前景及背景交錯編碼並壓縮，使二維影像轉為一維，我們將其中每個前景(1)和背景(0)稱之為一個 Run，並使第一個 Run 固定為背景(0)做開頭，若第一個像素為背景(0)則會繼續累積，反之則插入一個長度為 0 的 Run，以方便做影像處理，如下圖所示。給予一 7*7 像素影像中，有 3*3、3*1、1*3、1*1 的前景，透過將影像以[5]長度編碼來表示：

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

(a)原影像

8	3	1	1	2	3	1	1	2	3	1	1	9	3	1	1	8

(b)長度編碼表

圖 2.3 [5]長度編碼表之結構

[6]提出的長度編碼結構由$R = \langle lx, rx, y\rangle = \{(lx, y), (lx + 1, y), (lx + 2, y), \ldots, (rx, y)\}$組成，$lx$、$rx$和$y$分別為前景像素點的最左邊的起始位置、最右邊的結束位置及 Run 的列數，影像A $= \cup_{n=1}^{N} R_n$，如下圖所示。

	0	1	2	3	4	5	6
0	■	■	■	■	■	■	■
1	■				■		■
2	■				■		■
3	■				■		■
4	■	■	■	■	■	■	■
5	■				■		■
6	■	■	■	■	■	■	■

(a)原影像

R1(1,3,1)
R2(5,5,1)
R3(1,3,2)
R4(5,5,2)
R5(1,3,3)
R6(5,5,3)
R7(1,3,5)
R8(5,5,5)

(b)長度編碼表

圖 2.4　[6]長度編碼表之結構

第三章　長度編碼用於外形影像處理的相關論文

本章所參考的長度編碼用於外形影像處理的論文主要是[5]及[6]，在本章中，我們將介紹這兩篇的論文。

論文[5]「一種長度編碼之外形影像處理演算法」，於 2015 由黃上銘、連國珍所發表，是使用[1]"Morphology Image Processing on Run-length Encoded and Its Performance Analysis"提出的將影像轉為長度編碼表結構，接著再進行外形影像處理。[6] "Fast algorithms for morphological operations using run-length encoded binary images"，由 G. Ehrensperger, A. Ostermann, F. Schwitzer 於 2015 年所發表，使用[2] "Fast algorithms for binary dilation and erosion using run-length encoding"由 Wook-Joong Kim, Seong-Dae Kim, Kyuheon Kim 等人所提出的長度編碼結構來進行外形影像的處理，兩篇都是使用 C++的程式語言來進行外形影像處理。

3.1　[5]外形影像處理演算法

[5]的外形影像處理法，是將影像進行長度編碼壓縮，編碼為前景背景交錯，再藉由壓縮後的長度編碼表，透過平移的長度編碼表，以 M*M 的結構元素對影像進行侵蝕，平移後的長度編碼表，做二進制邏輯運算，即完成外形影像處理；由於[5]並沒有將平移影像的長度編碼真實求出，是使用指標來紀錄影像平移的資訊，再與原始影像的長度編碼做二進制邏輯運算，藉以提高處理速度亦減少記憶體的耗費。以下我們將一一介紹[5]演算法的步驟：

Step 1：影像轉為長度編碼：先判斷影像第一個像素是否為 0(背景)，若否，則在長度編碼的第一個值插入一個長度為 0 的 Run；接著將每個連續的 0(背景)、1(前景)的個數存入 Run 中，每一個 Run 都代表著連續的背景或前景；判斷最後一個像素是否為 0，若否，亦在最後一個值插入一個長度為 0 的 Run。

Step 2：利用指標來記錄影像平移的資訊，採用$shift = (x + y \times width)$的公式，因為長度編碼已將影像由二維轉為一維影像，shift 為一維影像需平移的個數，x 和 y 分別表示影像水平或垂直方向的位移，width 表示影像的寬度；因為此論文是對於四鄰邊做侵蝕，故考慮的方向只有上、右、下、左，其中向右和向下的平移會使長度編碼起始的 Run 增加，最後一個 Run 減少，[5]使用的指標中必須記錄兩個索引值，分別記錄開頭和結尾以及開頭目前的值和結尾目前的值，向左和向上的平移則與向右和向下平移相反，若開頭的 Run 加上 Shift 求出的值小於 0，則會向下一個 Run 去計算，最後得到開頭的 Run 的值為相減後的值；若結尾的 Run 加上 Shift 求出的值小於 0，會向上一個 Run 去計算，最後得到的結尾 Run 的值為相減後的值。

Step 3：最後是將兩個長度編碼做二進制邏輯運算：長度編碼裡面的值為 0 或 1 的個數，較小的值會先被放入目的陣列[i]，判定其值的索引值與目的陣列的索引值，相同則加入目的陣列[i]，不同則加入目的陣列[i+1]，再由大的值減去小的值，求出的值即為剩下要被運算的個數，若兩個長度編碼的個數相同，

則只需取其中一個放入目的陣列，再以索引值與目的陣列的索引值做判定，當一個 Run 的值已經為 0 則會進入到下一個 Run 來繼續運算，直到整段長度編碼都運算結束，即可完成外形影像處理。

3.2 [6]Fast algorithm for morphological operations using run-length encoded binary images

[6]的長度編碼表是先將影像做長度編碼壓縮，影像$Z = \cup_{n=1}^{N} R_n$，$R = \langle lx, rx, y\rangle = \{(lx, y), (lx+1, y), (lx+2, y), \ldots, (rx, y)\}$，$lx$為前景像素點的最左邊的起始位置，$rx$為最右邊的結束位置，$y$ 為 Run 的列數，並利用數學形態學的集合理論，來跳過一些分析，這裡將簡介其論文及演算法。

在進行侵蝕之前，[6]先對影像和 SE 做一些前處理，透過証明 2.2，利用侵蝕過影像的平移來減少侵蝕的次數，再藉由証明 2.3，將影像中 Run 的長度比 SE 中 Run 的長度較小的 Run，使之移除，以減少之後的判定。

藉由 Jump-Miss 理論找出幾種舉例並提出 Jump-Hit 理論來執行侵蝕，利用侵蝕與膨脹的二元性，來完成外形影像處理。

第四章　本文提出外形影像處理演算法

本章將會介紹本文所提出的外形影像處理演算法，我們將針對 M*M 的結構元素，來進行外形影像處理演算法，並介紹本篇章所使用的長度編碼資料結構，接著介紹本篇演算法，最後我們將分析以上所提的不同演算法之時間複雜度。

4.1　本論文的資料結構說明

由上述篇章可以知道，長度編碼表有許多種，本論文使用的是[5]的長度編碼表，為何會選用[5]而不選用[6]的長度編碼表，以下將一一分析：

優點一，由於[5]所使用的長度編碼為 0、1 交錯(背景、前景)，每一個 Run 只儲存前景和背景的長度，Run 的個數多，但資料量少；[6]只儲存前景，每個前景中包含前景像素點的最左邊的起始位置，前景最右邊的結束位置，和所在的列值，雖然 Run 的個數較少，但資料空間卻較[5]來的多。

優點二，因為是外形影像處理，進行邏輯運算或是影像平移，長度編碼都必須要移動。對於[5]的長度編碼來說，移動只需要將前後的數值做調整，判定快速且有效率；對於[6]的長度編碼來說，移動需要對每一個 Run 中的頭座標和尾座標分別進行比對，並改變結構，此方式較為耗時，為了使外形影像處理能達到最高效率，本論文選擇的是[5]的長度編碼結構。

本論文的方法仍亦有缺點，對於侵蝕來說，在影像周圍要是 0(背景)才會正確，為 1 的話，影像長度會因為侵蝕(將 1 轉為 0)而有所誤差，但此缺點的處理並不會太過複雜，只需要在建立長度編碼表時將周圍像素判定為 0 即可。

8	3	1	1	2	3	1	1	2	3	1	1	9	3	1	1	8

圖 4.1　[5]長度編碼表

R1(1,3,1)
R2(5,5,1)
R3(1,3,2)
R4(5,5,2)
R5(1,3,3)
R6(5,5,3)
R7(1,3,5)
R8(5,5,5)

圖 4.2　[6]長度編碼表

4.2　本論文的演算法說明

在進行完長度編碼後，我們將開始進行外形影像處理，利用[6]中提到的 $\varepsilon_{\mathrm{B}}(X)=\cap_{m=1}^{M}\cup_{n=1}^{N}\varepsilon_{R_m^B}(R_n^X)$侵蝕定理，為了使效率更加明顯，我們將針對 M*M 的結構元素來進行侵蝕，結構元素 M*M=M*(1*M)，R_1^B, R_2^B, R_3^B分別為以 1*M 進行侵蝕的結果，我們只需要將R_1^B求出，並根據[6]平移定理，即可得出R_2^B和R_3^B，將R_1^B, R_2^B, R_3^B進行邏輯運算，即可完成侵蝕。

[5]所進行的外形影像處理演算法只有針對四個方位進行處理，若對於M*M 的結構元素則需要多進行左上、右上、左下、右下等方位的處理，在此藉由[6]所提侵蝕的基本運算，改進[5]快速演算法，以下將詳細介紹。

改進[5]快速演算法的以 M*M 的結構元素執行外形影像處理，在第三章中，有詳細提到[5]的架構，為一四鄰邊演算法，若改 SE 為 3*3 的侵蝕，會增加左上、右上、左下、右下等方位侵蝕，八方位的指標位移後，再做二進制邏輯運算；藉由[6]所提的侵蝕運算，給了我們改進的想法，先將影像對於結構元素 M*M 中 1*M 的侵蝕求出，此時只需要紀錄指標為影像平移的資訊，先對所獲得的資訊做二進制邏輯運算，又 SE 為一 3*3 像素影像，故所得的邏輯運算影像可繼續運用至 SE 原點的上、下列，對於原點的上、下列亦是運用紀錄指標的平移資訊，接著再對平移資訊進行二進制運算，即完成侵蝕。

我們使用的是[5]的長度編碼結構，為 0、1 交錯的長度編碼，在上述的例子中$\cup_{n=1}^{5} R_n^X$的 Run 即為影像的長度編碼，為了使效率更加明顯，我們針對 M*M 的結構元素來進行侵蝕，將 1*M 的長度編碼表侵蝕求出後，根據平移定理，將長度編碼表平移進行邏輯運算。與[6]不同的地方是，其每一個 Run 皆需要分別去跟 SE 做運算，若使用[5]的長度編碼，在進行侵蝕時，侵蝕次數將等同於 SE 的列數，這樣將大幅降低外形影像處理的時間複雜度，以下將詳細介紹本論文外形影像處理的演算法步驟：

Step 1：影像轉為長度編碼：用我們定義的結構 RunLength 並建立一個 Struct e 使用 malloc 配置記憶體空間來儲存 0 和 1(背景和前景)的長度，及建立一個 sum 值計算 0 和 1 的個數(Run 的個數)，藉由 sum 值來找 e 中 0、1 的長度，接著使用 threshold 將影像二值化區分，先判斷影像的一個像素是否為 0(背景)，如果不是則在長度編碼(sum)第一位插入一個長度為 0 的 0(背景)；接著記錄連續 0(背景)或 1(前景)的個數，如果不連續則將新的值加入至下一個 sum 中；判斷最後一個像素是否為 0(背景)，如果不是則在最後一個 Run 後插入一個長度為 0 的 0(背景)，藉由維持長度編碼的開頭和結尾的索引值為 0 以方便之後的運算。

給予一 7*7 像素影像中，有 3*3、3*1、1*3、1*1 的前景，我們將此影像轉為長度編碼表如下：

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

8	3	1	1	2	3	1	1	2	3	1	1	9	3	1	1	8

圖 4.3　影像 X 轉為長度編碼 L

Step 2：移除比 SE 中最短 Run 的長度編碼：依照輸入的 SE，和 Step 1 所建立的長度編碼表比較大小，若 SE 中最短的 Run 大於長度編碼表中 1(前景)位置的 Run，將這個位置(sum)前景中的 Run 與 sum 前一位後一位背景的 Run 做加總，將合併的 sum 清除，以避免 0、1 的排序錯誤，以下舉例皆以 3*3 的結構元素來做表示。

8	3	1	1	2	3	1	1	2	3	1	1	9	3	1	1	8

8	3	4	3	4	3	11	3	10

圖 4.4　將比 SE 中最短 Run 移除後的長度編碼L'

Step 3：藉由 1*M 的 SE 將影像做侵蝕：對一個影像做侵蝕，前景會減少，背景會增加，對於 SE 為 3*1 的白色影像侵蝕結果來說，影像左右兩端各會被侵蝕 1 格，若 SE 為 5*1 的白色影像，影像兩端則會各被侵蝕 2 格，使用長度編碼可以便於計算侵蝕，將長度編碼中 0(背景)的長度增加 1，而 1(前景)的則長度減少 2，在這邊我們將 M*M 的 SE 以 3*3 為例，區分為R_0、R_1、R_2且原點設置於R_1，$SE = \cup_{m=1}^{3} R_m$，因為每一段以 1*M 進行侵蝕所得到的 R 只會有平移

上的差異，以下做的是以R_1做侵蝕所得到的長度編碼，用受到侵蝕的影像和長度編碼表來做表示。

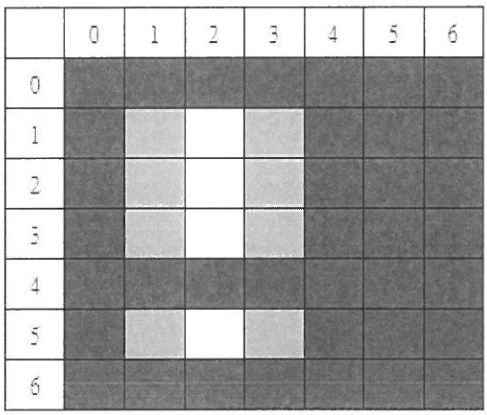


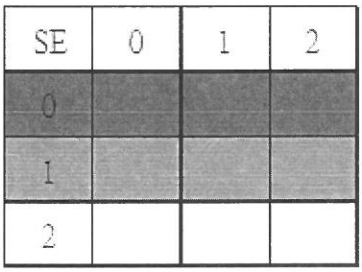
圖 4.5 (a)影像 X 經過R_1侵蝕後所得影像，灰階為受到侵蝕的區域

8	3	4	3	4	3	11	3	10
+1	-2	+2	-2	+2	-2	+2	-2	+1
9	1	6	1	6	1	13	1	11

(b)長度編碼L'經過R_1侵蝕後的長度編碼L''

Step 4：以 SE 中 M*(1*M)的 M 對影像做侵蝕：對於 SE 為 M*M 的像素影像來說，影像在不同的 SE 列數中，所得到的影像會是以 1*M 侵蝕後位移的原影像，而在長度編碼中，上下位移僅需要改其開頭與結尾，因為使 SE 中心在原點的關係，在此將其分成兩種判定：第一種是 SE 的列數在原點上方，在原點上方的 SE，得出的影像會向下位移，長度編碼則是第一個 Run 做增加，最後一個 Run 則會減少，增加或減少的值皆為影像位移的長度，若最後一個 Run 的值小於向下位移的長度，則會從最後一個 Run 的值向前一個 Run 累加，直到 Run 的值大於向下位移的長度，Run 才會減去其長度，整段長度編碼即是原點

上方 SE 所做侵蝕的長度編碼；第二種則是 SE 的列數在原點下方，與第一種相反，所得出的影像會向上位移，長度編碼則是第一個 Run 減少，最後一個 Run 增加，第一個 Run 的值若是小於向上位移的長度，會向後累加直到 Run 的值大於其位移長度，減去後的整段編碼則為原點下方 SE 侵蝕後的長度編碼。



(a)結構元素 SE：3*3 像素影像

9	1	6	1	6	1	13	1	11

(b)L'經過 SE (R_1)侵蝕後的結果L''

16	1	6	1	6	1	13	1	4

(c) L'經過 SE (R_0)侵蝕後的結果L'''

2	1	6	1	6	1	13	1	18

(d) L'經過 SE(R_2)侵蝕後的結果L''''

圖 4.6　長度編碼以 SE 不同列進行侵蝕

Step 5：長度編碼的二進制邏輯運算：在進行完侵蝕之後，要將每一列 SE 所得到的長度編碼兩兩進行二進制運算，需要判定的列數等於 SE 的列數(Step 4)，因為長度編碼是 0、1 交錯編碼，故邏輯運算的方式與[5]相似，長度編碼中所存放的是 0 和 1 的個數，兩兩做比對，較小的值會被放入給予的 dest 值中，dest 值為我們的輸出值，並藉由兩個長度編碼中的索引值與 dest 中的索引值是否相同，相同則可將值放入 dest 值中，若不同，則會加入至 dest 的下個編碼中，其放入後較大的值將減去較小的值，得到的數值為剩餘未被計算的個數；

若兩邊的個數相同，只需取其中一邊放入 dest 值中，接著便可進入下個長度編碼中，以此類推，直到完成比對即可得到侵蝕的結果。

4.3 時間複雜度分析

在傳統影像分析中，若藉由一 M*M 像素的結構元素來對影像 N*N 像素進行外形影像處理，因為會使每一個結構元素的像素與每一個影像像素都分別進行分析，所使用的時間複雜度為$O(N^2M^2)$。

在[1]論文中，作者提出了將影像壓縮為長度編碼來進行外形影像處理，將長度編碼以 L 來表示所壓縮的資料個數，利用將長度編碼表平移，再將其進行二進制邏輯運算；[5]所提出的快速外形影像處理演算法即是利用此長度編碼來進行處理，透過將資料平移的資訊轉為指標來進行記錄，除去每一次將影像平移的長度編碼計算出來的時間，所使用的時間複雜度為$O(M^2L)$。

在本篇論文中，我們只針對 M*M 的結構元素進行外形影像處理，透過[6]的集合理論，將[5]演算法進行改良，以 M*M 的結構元素進行侵蝕，若為原[5]演算法，時間複雜度將會是$O(M^2L)$；改進後的演算法，因為只分別進行列與行的次數的侵蝕，時間複雜度會是$O(2ML)$；本篇的演算法提出了對於 SE 對長度編碼做侵蝕的快速運算，降低長度編碼對 SE 行數侵蝕的次數，時間複雜度可以壓縮至$O((1+M)L)$。以時間複雜度上討論，改進過的[5]演算法與本文演算

法皆為$O(ML)$，但實際實驗中，兩者的差別在於斜率不同，以實驗結果來說，仍是本論文演算法的效率較高。

第五章　實驗

本論文共進行三項實驗，實驗一，以本文外形影像處理演算法與傳統外形影像處理演算法進行比較，以 3*3 的結構元素，分別進行對影像一次、三次、五次和七次的侵蝕，將對結果比較；實驗二，以 M*M 的結構元素做外形影像處理中的影像侵蝕，用實驗結果來證明本演算法的時間複雜度$O((1+M)L)$是以線性進行成長；實驗三，用本文所提的外形影像處理演算法與[5]的外形影像處理演算法進行比較，利用一些常見的影像來進行實驗，比較兩者演算法的效率，並以圖表做為呈現。

5.1　實驗環境

本實驗的實驗環境如下：

作業系統：Windows 7

程式平台：Visual Studio 2015 Community

程式語言：C++、*OpenCV(Open Source Computer Vision)*函式庫

處理器：Intel ® Core CPU i5-2410M @ 2.30GHz

記憶體：8.00 GB

5.2　實驗一

實驗一，將本文演算法與傳統外形影像處理演算法比較效率，以 3*3 的結構元素對不同影像分別進行一次、三次、五次、七次侵蝕，本文演算法的效率

在進行多次侵蝕後，所耗費時間將會小於傳統演算法，以下為實驗影像與實驗結果。

5.2.1 實驗影像

本實驗使用的影像如下圖 5.1 所示，實驗影像的資訊如下表格 5.1：

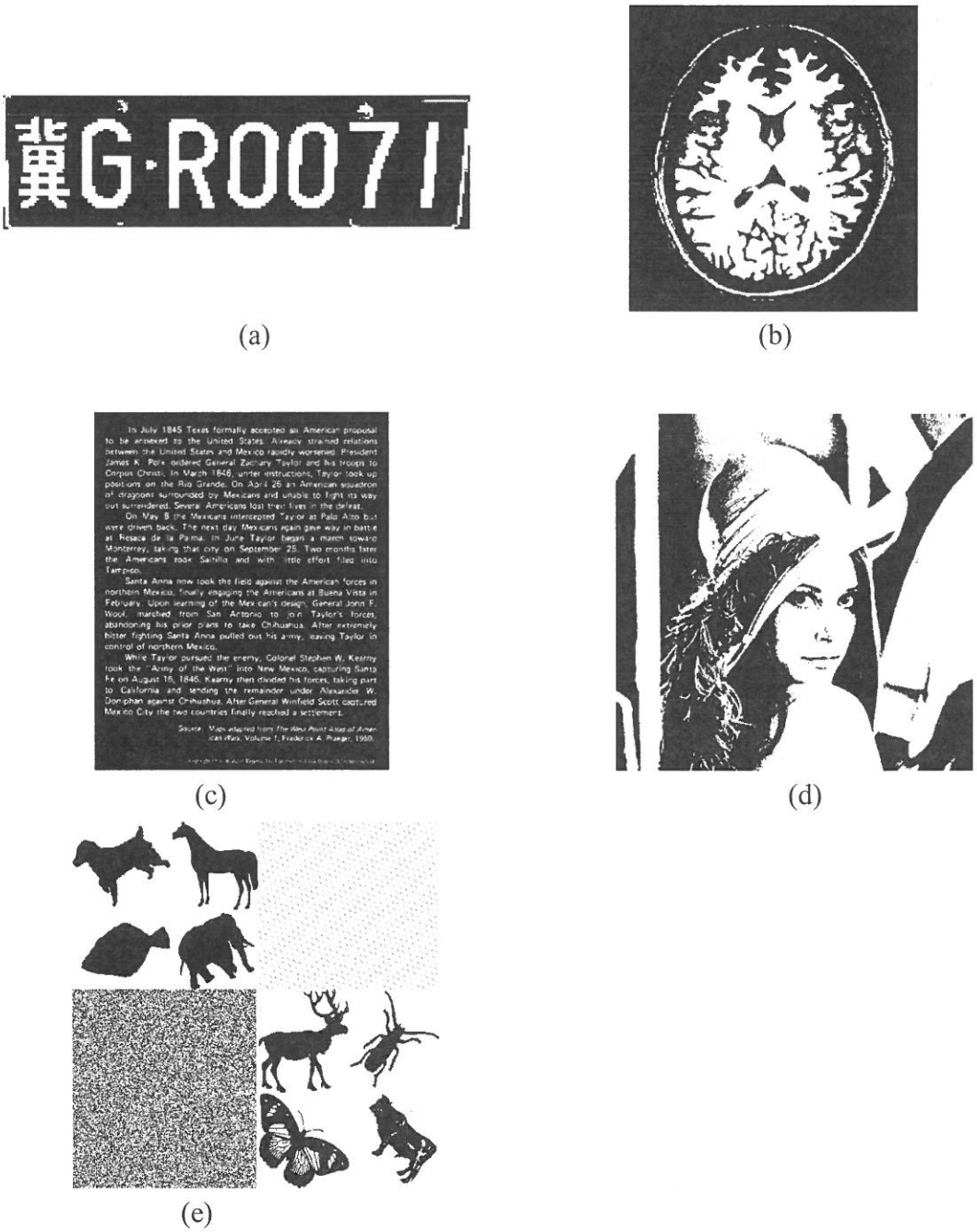


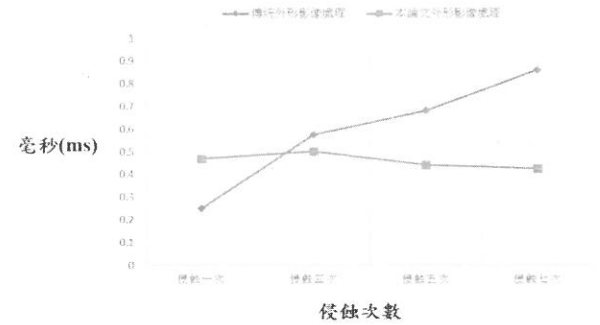
圖 5.1 實驗影像(a)車牌(b)大腦(c)文字(d)人物(e)綜合圖

表 5.1 實驗影像資訊：

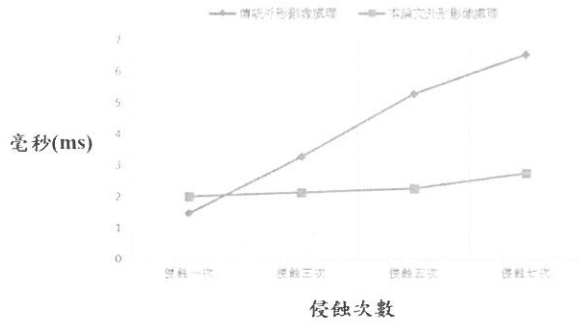
影像	(a)	(b)	(c)	(d)	(e)
大小(列*行)	422*122	514*544	622*754	514*514	2502*2502
長度編碼數	2276	7562	42254	12768	474306
前景長度編碼數	1138	3781	21127	6384	237153

5.2.2 實驗結果

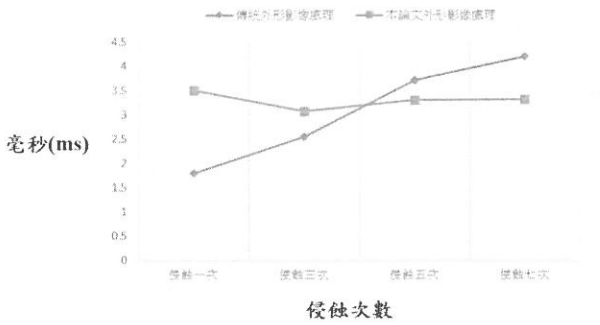
使用同一張影像進行多次實驗時，每次實驗都可能多少會出現一些誤差，為此，我們將執行程式 200 次，加總後取其平均，得出執行的時間，這邊只執行 a、b、c、d 四種影像進行實驗，以下為實驗結果圖：



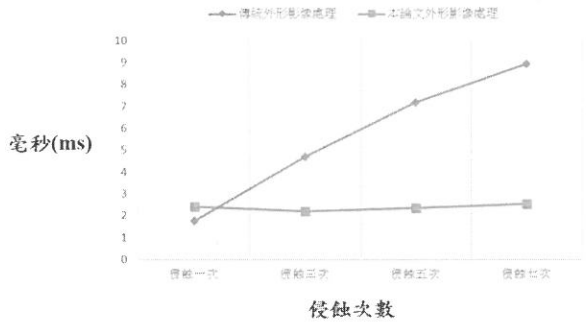
(a)車牌



(b)大腦



(c)文字



(d)人物

圖 5.2 實驗一實驗結果(a)車牌(b)大腦(c)文字(d)人物比較圖

由實驗結果得知，影像在進行第一次侵蝕時，傳統式影像會在讀影像時同

時進行侵蝕，但本論文演算法會先進行長度編碼建表，才進行外形影像處理，因為多掃描了一次的關係，時間可能會較傳統式外形影像處理演算法慢；影像在侵蝕三次時，傳統演算法仍對會對每個影像進行掃描並執行侵蝕處理，而本文演算法則只需針對建好的長度編碼表去進行處理，時間上已經大幅低於傳統演算法；之後侵蝕次數的比較，顯示出若外形影像處理執行的次數越多，長度編碼的方法效果就會越好。比較特別的是圖(c)，因為是文字的關係，影像可能在侵蝕一次時前景就被侵蝕完畢，因此本文演算法對其並沒有產生太大的效果。

5.3 實驗二

實驗二利用 5.2.1 的實驗影像，以 M*M 的結構元素對不同影像分別進行侵蝕，我們將以 3*3、5*5，7*7 像素的 SE 侵蝕影像並分別進行計時，以結果來證明本演算法的時間複雜度為線性曲線。

實驗結果

為了減少誤差，我們與實驗一一樣將程式執行 200 次，加總後取其平均，得出執行的時間，以下為本文演算法以 M*M 的結構元素對影像進行侵蝕時，執行時間比較圖：

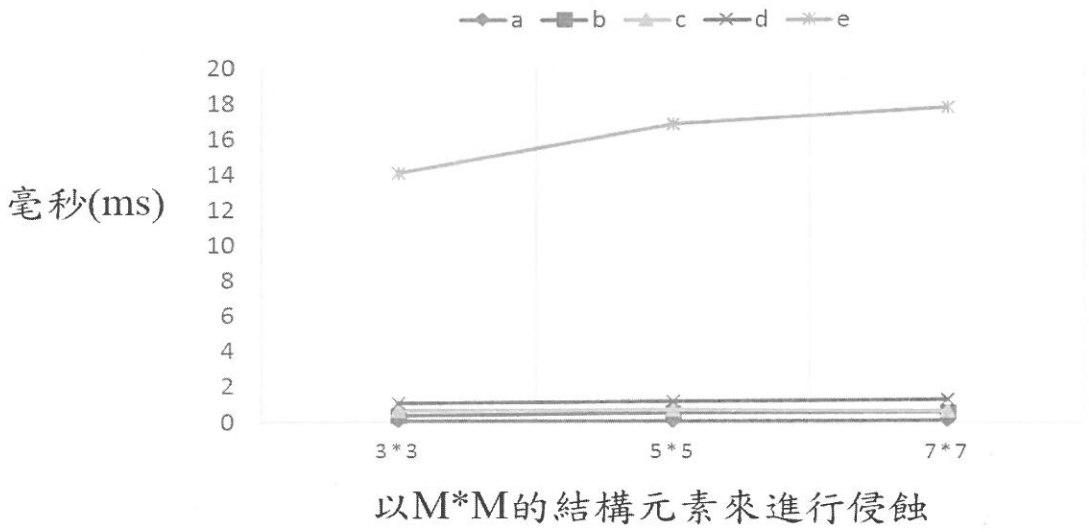


圖 5.3 實驗二實驗結果比較圖

由圖 5.3 的結果中得知，本文演算法不論是使用何種 M*M 的結構元素，先求出 1*M 的侵蝕後的長度編碼，藉由平移來得出其他的長度編碼，再進行邏輯運算，都是以 SE 的列數成線性成長；由於影像 e 的長度編碼數相較於其他影像大的多，所使用的建表時間也相對較多，為了使其他實驗結果更加明顯，我們將影像 a、b、c、d 用以下圖表再進行一次比較。

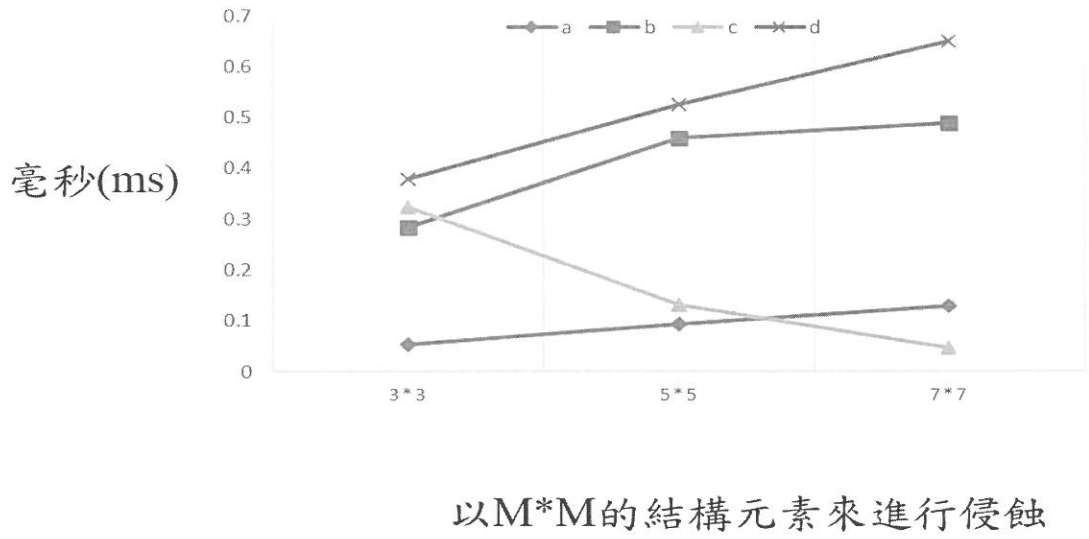


圖 5.4 實驗二實驗結果比較圖(四種影像)

以圖 5.4 的結果來說，影像 a 和影像 d 為 worst case，但仍是以線性曲線做為成長，若影像中有可以移除的 run，其侵蝕速度仍可進行提升，如影像 b 和影像 c，以 SE 為 7*7 對影像侵蝕可能會比以 SE 為 3*3 進行侵蝕的還要快。

5.4 實驗三

利用 5.2.1 的實驗影像，進行本文的外形影像處理演算法與[5]的外形影像處理演算法的比較，分析兩種演算法的執行效率。

實驗結果

為了減少誤差，故兩種程式皆執行 200 次，再取其平均，得出執行的時間，因為本文的長度編碼建表與[5]的長度編碼表相同，故我們只計時演算法的部分並比較，以下為實驗結果：

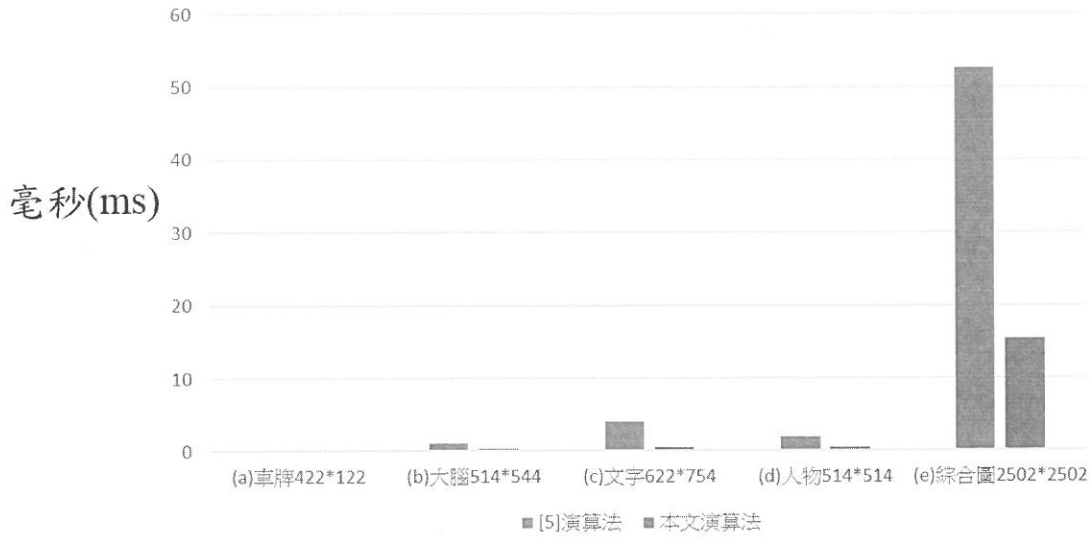


圖 5.5 實驗三實驗結果比較圖

因為影像 e 的結果過大會使其他影像結果不明顯，故我們對前面四種影像另外進行比對。

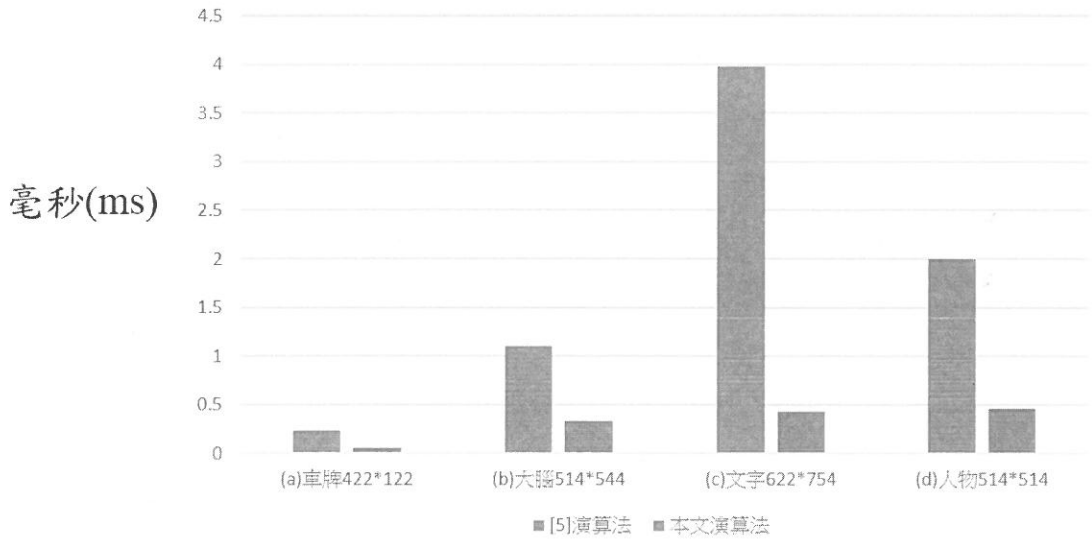


圖 5.6　實驗三實驗結果比較圖(四種影像)

如上面實驗結果所表示，本演算法與[5]演算法以 3*3 的結構元素做侵蝕，因為都是使用長度編碼表，故不計算建表時間，在[5]演算法中需要進行 M*M 次的邏輯運算，因為是以兩兩進行運算的關係，以 3*3 結構需要進行 8 次的運算；本演算法先進行一次 1*M 的侵蝕，並藉由平移，執行 M 次的邏輯運算，因為是兩兩進行運算的關係，以 3*3 結構需要進行 2 次的邏輯運算。由上述分析可以知道兩種演算法以 3*3 的結構元素執行演算法會差距 4 倍的邏輯運算，我們亦由實驗結果得出此結論，其中影像(c)的實驗結果差距甚大，是由於本演算法會先將小於結構元素最小的 Run 移除，減少需要判定的長度編碼個數，才進行侵蝕處理，故會造成此現象發生。

第六章　結論

現今科技日新月異，影像處理的應用已無所不在，一個影像處理的步驟通常先將影像轉為二值化影像接著進行外形影像處理，但若是運用長度編碼表來將影像進行壓縮，除了降低資料量外，亦能透過長度編碼來進行外形影像處理，本論文藉由外形影像處理中的定義來改進[5]利用指標記錄長度編碼表的平移資訊來完成外形影像處理，利用[1]所提出的長度編碼表容易運算的特性，通過運算來降低長度編碼表的平移和邏輯運算的次數，能更有效率的完成外形影像處理。

此長度編碼亦能運用至連通區域標記演算法，為本實驗室另一位同仁所進行的研究，透過改善外形影像處理集連通區域標記兩者之間的連貫性，減少影像轉換的時間，使整體影像處理的效率更高，成效更加顯著。

參考文獻

[1] Brian K. Lien. "Morphological Image Processing on Run-length Encoded Binary Image and Its Performance Analysis," *8th IPPR Conference on Computer Vision, Graphics and Image Processing,* 1995.

[2] W. J. Kim, S. D. Kim, and Kyuheon Kim, "Fast algorithms for binary dilation and erosion using run-length encoding," ETRI 27, pp. 814-817, 2005.

[3] J. J. Ding and P. X. Lee, "Fast Morphology Algorithm with Parallel Processing Structures," *IEEE, Language and Image Processing (ICALIP)*, pp. 324-329, 2014.

[4] G. Yunfeng, W. Feiyang, and H. Xizotian. "Connected Components Labeling Algorithm Based On Run-length Table Searching," *IEEE, Computer Science & Education*, pp. 22-24, August, 2014.

[5] 黃上銘及連國珍，「一種長度編碼之外形影像處理演算法」，*第十四屆離島資訊技術與應用研討會, ITAOI 2015*。

[6] G. Ehrensperger, A. Ostermann, and F. Schwitzer, "Fast algorithms for morphological operations using run-length encoded binary images," April, 2015.

[7] 吳柏彥及連國珍，「單向長度編碼表之連通區域標記演算法」，*輔仁大學資訊工程研究所*，新北市，2016。

[8] 連國珍，「數位影像處理 MATLAB」，四版，第 8 章。台北，儒林，2007。

[9] Xiaoping Lin and Zhihong Xu, "A Fast algorithms for Erosion and Dilation in Mathematical Morphology," *IEEE, World Congress on Software Engineering (WCSE), DOI 10.1109, 2009.367.*

[10] L. Piper and J.Y. Tang, "Erosion and Dilation of Binary Images by Arbitrary Structuring Elements Using Interval Coding," *Pattern Recognition Letter,* vol. 9, no. 3, pp. 201-209, April, 1989.