基於多重解析度搜尋之碎形影像編碼

[1]張軒庭, [2]許志仲
光電與資訊實驗室, 國立雲林科技大學電機工程系
{[1]htchang, [2]g9412716}@yuntech.edu.tw

摘要

本論文提出一種利用多重解析度搜尋 (Multi-resolution searching) 的演算法應用於碎形影像編碼 (Fractal coding). 傳統碎形影像編碼的定義域區塊 (Domain block) 爲值域區塊 (Range block) 四倍大, 然而定義域區塊可能在不同解析度下, 再經過次取樣之後可以保留更多細節。所以, 當搜尋到最佳匹配的定義-值域區塊時, 便往較高/較低一層之解析度的定義域區塊中搜尋是否有更匹配的區塊, 若有便在設定之範圍內往更高/低層解析度的定義域區塊搜尋。實驗證明, 本論文所提出的方法可以有效增進重建影像品質。

關鍵字: 碎形編碼, 影像壓縮, 資料壓縮, 多重解析度。

1 序論

碎形影像編碼 (Fractal coding) [13] [1]是由 Jacquin 學者在 1992所提出, 在理想上可達到超高壓縮比, 因此吸引許多學者投入研究。傳統碎形影像編碼是將影像切割成許多的值域區塊 (Range block) 與定義域區塊 (Domain block) 後, 搜尋由定義域區塊所組成的定義池 (Domain pool), 將定義域區塊透過收縮彷射轉換 (Contractive affine transform, CAT) 來找最接近的值域區塊。稱之爲最佳匹配區塊。接著, 將收縮彷射轉換的參數保留下來, 此即爲碎形壓縮編碼的輸出格式。

由於值域區塊越小, 重建的影像品質就會越好, 但相對必需付出較大的計算量與位元率 (bit rate), 因此, 四元樹切割法 (Quadtree partition) [2] [3] [5]普遍被使用在值域區塊的切割上, 並在位元率與重建品質上取得較佳的平衡。在本文中, 我們設定起始值域區塊爲 8×8大小, 當最佳匹配區塊找到後, 其平均誤差 (Mean square error, MSE) 大於門檻值, 則將此值域區塊切割成 4×4大小。在值域區塊中, 有些值域區塊相當平滑, 因此可以計算值域區塊平均值 (Mean) 與變異數 (Variance), 當值域區塊的變異數小於門檻值, 則只要保留此值域區塊的平均值即可 [8], 在本論文中只保留平均值的前六個位元。

大多數碎形影像編碼的研究著重在混合其它技術提升編碼效能 [10] [12] [6]、加快編碼速度 [9] [11]與影像之間相似性的探討 [4]爲主。傳統碎形影像編碼的定義域區塊爲值域區塊四倍大, 然而定義域區塊可能在不同解析度下, 經過次取樣之後可以保留更多細節。因此, 本論文便提出利用不同解析度的定義池大小來增進品質, 稱之爲多重解析度搜尋法 (Multi-resolution searching, MRS)。在原本解析度下所找到最佳匹配區塊, 往高/低一層解析度的定義池中進行搜尋, 若有找到更佳匹配區塊, 則再繼續往更高/低解析度的定義池搜尋, 若無, 則停止搜尋以節省計算量, 如此一來可以增加可利用的定義域區塊, 改善重建影像的品質。

本論文之結構一開始是序論, 接著爲碎形影像編碼之介紹, 第三節提出多重解析度搜尋法, 第四節爲實驗結果, 最後是本文的結論。

2 碎形影像編碼

2.1 傳統碎形影像編碼

碎形影像編碼的精神是建構在影像自我相似性 (Self-similarity) 之上, 每個影像都可以在裡面互相有相似的地方, 如圖1所示, 左上角的帽沿與右下角鏡子反射的部份有著局部的自我相似性, 肩膀的大影像區塊與其小影像區塊也具有此一特性。

碎形影像編碼的過程是將影像切割成許多值域區塊與定義域區塊, 並從中找到最佳匹配區塊。每個定

圖 1: 自我相似性

義域方塊與值域區塊進行比較前, 必須先經過 CAT, 其公式定義爲

$$\widehat{R} = i\{\alpha \cdot (S \circ D) + \Delta g\} \qquad (1)$$

其中定義域區塊 D經過 S縮小轉換後, 再經過 α對比的縮放, 最後加上 Δg亮度平移值, 最後透過 i (Isometry) 來得出彷射值域區塊 $\widehat{R}$, 最後計算與值域區塊的 MSE, 進而找到最佳匹配區塊。而每個參數的位元分配如下: (1) α 分配三位元。(2) i 分配三位元。(3) Δg 只保留前六個位元。

每個值域區塊都會到定義池中找到最佳的彷射區塊, 因此, 定義池大小 DP_s越大則重建品質越好。然而, 定義池過大則會造成計算量過大的問題。定義池的取樣方法有兩種, 如圖2所示, 分別是: (1) 鄰近取樣 (Neighboring), 在目前編碼的值域區塊附近以可重疊的方式取樣建立定義池, 並搜尋最佳匹配區塊之。(2) 次取樣 (Subsampling), 在影像上以不重疊的方式均勻取樣建立定義池, 並搜尋最佳匹配區塊。同樣的, 找到最佳匹配區塊時必須記錄的座標之位元分配爲 $\log_2(DP_s)$個位元。

然而, 影像中往往存在著一些相當平滑的影像區塊, 這些影像區塊可以只保留其平均值 (Mean) 即可, 因此在每個值域區塊切割時, 便計算每個值域區塊的變異數 (Variance), 其定義爲

$$\sigma^2 = \frac{1}{B} \sum_{1 \le i,j \le B} (R_{i,j} - \mu_R)^2, \qquad (2)$$

其中 σ^2爲變異數, μ爲此 $B \times B$大小的值域區塊 R之平均值。當變異數小於門檻值 V_{th}則此值域區塊只保留平均值前六個位元。

2.2 四元樹切割法

由於值域區塊的大小對峰值訊噪比 (Peak-signal to noise-ratio, PSNR)、位元率影響甚大, 若是區塊太大, 則 PSNR 下降但位元率也下降; 反之亦然。因此本論文是以四元樹切割法來決定值域區塊的大小, 如圖3所示, 在父系 (Parent level) 之值域區塊大小爲8×8, 子系 (Child level) 值域區塊大小爲4×4, 而進行子系切割與否, 是由MSE 來判斷, 其定義爲

$$\mathrm{MSE}(R, \hat{R}) = \frac{1}{B^2} \sum_{1 \le i,j \le B} (r_{i,j} - \hat{r}_{i,j})^2, \qquad (3)$$

其中 $r_{i,j}$代表編碼前的值域區塊像素值, 而 $\hat{r}_{i,j}$爲彷射區塊像素值。接著, 判斷在父系時其最佳匹配區塊之 MSE 是否有高於門檻值 E_{th}, 若有則將父系的值域區塊再分割爲四等份之子系的值域區塊。

2.3 碎形影像解碼

碎形影像解碼的方式爲先隨意輸入一張影像, 接著不斷對每個定義域區塊作 CAT 得到彷射值域區塊, 一直到所有的彷射值域區塊都已經收斂了爲止, 否則, 就繼續疊代直至收斂爲止。

3 多重解析度搜尋法

傳統碎形影像編碼的定義域區塊大小都爲值域區塊的四倍大, 可以直接進行次取樣得到與值域區塊相同

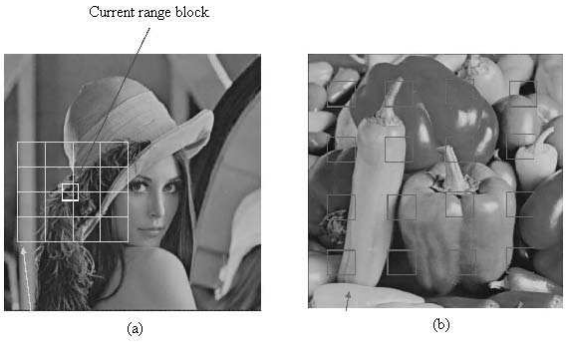

圖 2: 鄰近取樣與次取樣

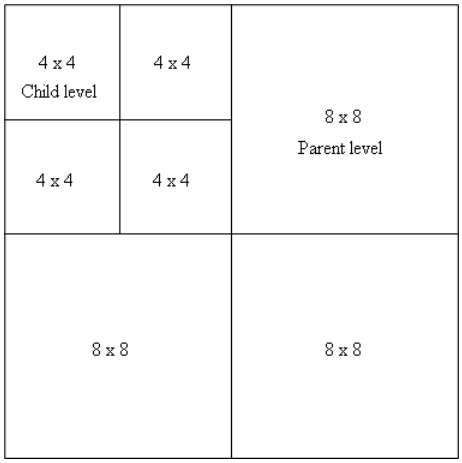

圖 3: 值域區塊的四元樹切割

大小的彷射區塊。然而, 某些情況之下, 四倍大小進行次取樣之後資訊損失嚴重, 如圖4 (a) 所示, 原本波形在經過次取樣之後, 其波形只有其兩個取樣點的平均值。然而, 圖4 (b) 中採用較大/較小間隔的取樣點的方式, 可以得到較多訊息, 以取樣點一, 每個兩點取樣的方式來看, 由三個值取平均, 避免了 (a) 的情形。而取樣點二的方式, 每隔一點半取樣, 也可得到較佳之結果。在影像上, 當影像區塊較小時, 取樣會有重複的點產生, 在隨後的章節, 我們將詳述重複取樣點的處理方式。

在碎形影像編碼中, 值域區塊平滑的部份將只用其平均值編碼, 換句話說, 平滑的值域區塊不會到定義池中搜尋最佳匹配區塊, 因此, 定義池中有平滑的定義域區塊, 將造成定義池效能的低落。舉例來說, 設定 $V_{th} = 5$的情況, 當值域區塊的變異數小於5則不去定義池中搜尋, 因此有許多變異數小於5的定義域區塊, 將不易找到最佳匹配的值域區塊. 若將取樣點往前延伸或往後收縮 (意即使用較大或較小的定義域區塊取樣), 如圖4 (b) 所示, 則可從較高/低的解析度得到更多的訊息, 增加定義池中可以利用的定義域區塊.

然而, 若使用多層解析度的定義池時 (例如: 10層不同解析度之定義池), 則所耗費在搜索最佳匹配區塊的計算將會過大, 因此下一小節我們將探討並解決這個問題。

3.1 條件式多重解析度搜尋

爲了減少花費在多層解析度之定義池中的搜尋時間,

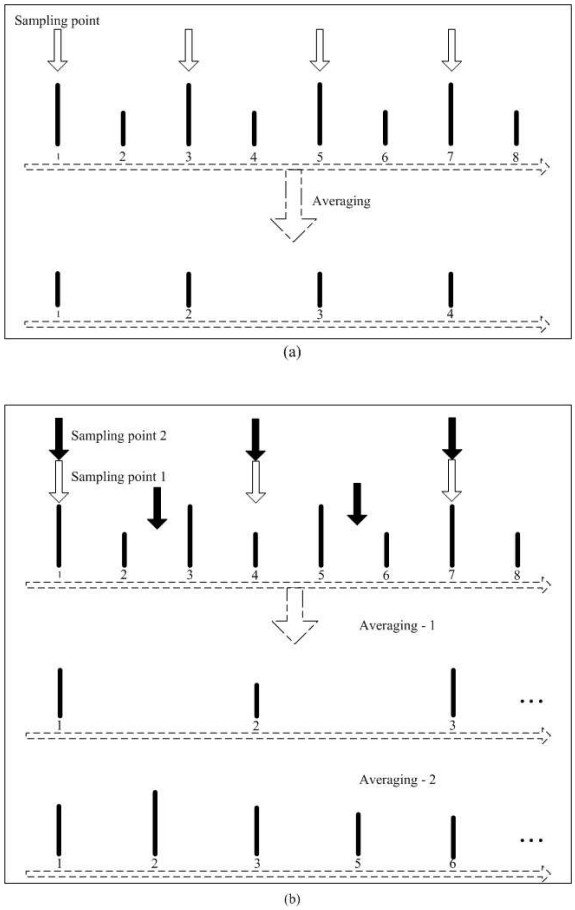

圖 4: 次取樣所造成之損失示意圖

因此本論文提出一種條件式多重解析度搜尋, 亦即符合條件才進行多重解析度搜尋。假設在 N_p解析度中已找到最佳匹配區塊, 則便往上一層解析度 $N_p + 1$來搜尋, 若有更佳之匹配區塊, 則再往上層 $N_p + 2$之解析度的定義池中搜尋, 依此類推。但是當往上一層 $N_p + 1$並無法找到更佳匹配區塊, 則往低一層解析度 $N_p - 1$的定義池搜尋。然而, 爲了節省計算時間, 尚可以 (1) 取較高解析度的定義池, 如此一來只要搜尋高一層解析度的定義池來判斷即可, 不需要搜尋上下兩層。(2) 限制其它解析度之定義池大小, 以增加判斷、尋找的速度。

接著, 若是在父系與子系的值域區塊都進行多重解析度的搜尋, 易造成計算量提升但編碼效能提升不多的情況. 因此, 只有在父系值域區塊才進行多重解析度搜尋, 這可以讓分配到子系值域區塊編碼的機率降低, 換言之便是降低整體的位元率。

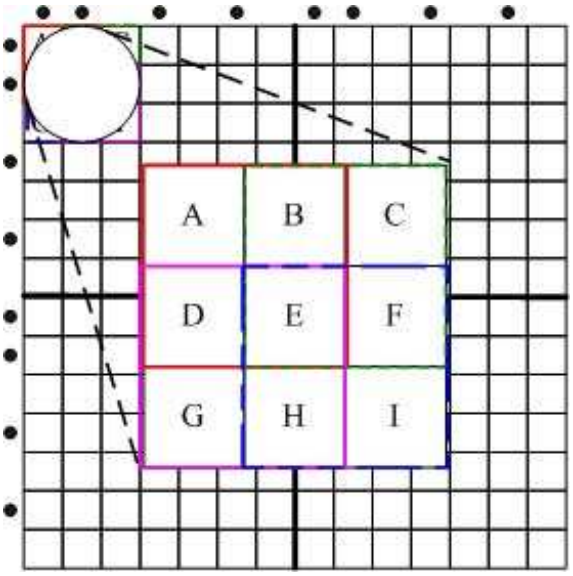

圖 5: 重複取樣點之示意圖

3.2 多重解析度之取樣法

傳統碎形影像編碼的次取樣公式爲

$$D_s(x,y) = (D(x,y) + D(x+1,y) + D(x,y+1) + D(x+1,y+1))/4. \quad (4)$$

其中 $D_s(x,y)$爲次取樣後的定義域區塊。式10只適用在定義域區塊解析度爲值域區塊的四倍大時, 因此, 若非四倍大的比例則必須先計算縮放比例

$$C_s = \frac{S_D}{S_R}, \quad (5)$$

其中 S_D與 S_R 分別爲定義域區塊與值域區塊的大小。接著, 計算定義域區塊 x 與 y 軸的下一個取樣點 P_x與 P_y, 藉由

$$P_x = \text{Round}(C_s \times X_{count}), \quad P_y = \text{Round}(C_s \times Y_{count}), \quad (6)$$

可得出. 其中 X_{count}爲目前次取樣的計數器, $0 \le X_{count} \le S_R$, Round 爲無條件捨去. 因此, 從 14×14次取樣到 8×8的 x 軸取樣點即爲 $S = \{0,\ 1,\ 3,\ 5,\ 7,\ 8,\ 10,\ 12\}$, 式10中的 $x+1$與 $y+1$原本定義爲下一個像素, 此例下一個取樣點便以 S中下一個取樣點位置的像素取代, 如式7所示.

$$D_s(x,y) = (D(x,y) + D(S_x(x+1),y) + D(x,S_y(y+1)) + D(S_x(x+1),S_y(y+1)))/4. \quad (7)$$

其中 S_x與 S_y分別代表了 x 與 y 軸上的下一個取樣點.

接著, 當解析度較大時, 某些取樣的間隔距離會超過一個像素點, 因此必須先判斷是否符合以下條件: 目前取樣點與下個取樣點大於一個像素單位。若否則使用式10來計算, 若是, 則採用

$$D_s(x,y) = \frac{1}{AB}\sum_{a=0}^{A-1}\sum_{b=1}^{B-1} D(x+a,y+b), \quad (8)$$

來取得縮小的定義域區塊。其中 $0 \le a \le A$且 A爲下個取樣點與目前取樣點的像素間隔距離。

然而, 進行較低解析度的定義域區塊次取樣時, 有些取樣點會重複, 因此必須要給其較低的權重, 其它取樣點則給較高的權重。以一個例子, 如圖5且定義域區塊大小由14×14次取樣到8×8的情況, 其 x 與 y 軸之取樣點間格如黑點所示, 爲 $x = \{0,\ 1,\ 3,\ 5,\ 7,\ 8, ...\}$, 每隔兩次取樣就會出現重複取樣點, 因此, 進行次取樣之前必須先判斷兩個條件: (1) 當目前取樣點與下個取樣點只間隔一個像素單位時. (2) 當目前取樣點與上個取樣點只間隔一個像素單位時. 若符合其中一點, 就必須計算各像素點的權重, 否則使用原始的次取樣公式。在圖5中各像素點的重複次數分別是: A、G、C、I 點爲一次, B、D、H、F 點則爲兩次, E 點則四次。因此, 若符合狀況一的條件, 使用

$$D_s(x,y) = (D(x,y) + 0.5D(x+1,y) + 0.5D(x,y+1) + 0.25D(x+1,y+1))/2.25, \quad (9)$$

來取得次取樣後的值。其中要注意下一個取樣點是 $x+1$而不需要帶入 $P_x(x+1)$, 因爲狀況一的條件已經清楚說明, 是距離一個單位像素時才需要套用此公式. 若是狀況二, 則使用

$$D_s(x,y) = (0.5D(x,y) + D(x+1,y) + 0.25D(x,y+1) + 0.5D(x+1,y+1))/2.25, \quad (10)$$

來取得次取樣後的值。相同的, 在 y 軸上的處理方式與 x 軸相似, 在此省略。

3.3 多重解析度搜尋法之步驟

本文所提出的多重解析度之搜尋法整理如下, 參照

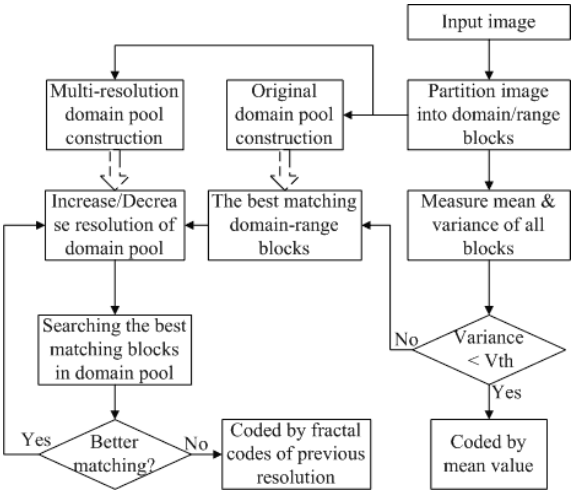

圖 6: 多重解析度搜尋法之流程圖

圖6:

1．輸入欲編碼影像, 將影像分割出值域區塊。

2．建立正常解析度的定義池, 由使用者決定建立幾個不同解析度之定義池以及其大小。

3．利用變異數臨界值 V_{th}來判別值域區塊是否只保留其平均值即可, 否則跳至下步驟。

4．利用 CAT 將定義域區塊轉換成彷射值域區塊, 並從原始定義池中搜尋最佳匹配區塊, 記錄此轉換參數與位置。

5．往高/低一層解析度之定義池找是否有更佳匹配區塊, 若無則紀錄上步驟之結果, 若有則往下步驟。

6．記錄目前定義池解析度、匹配區塊位置與轉換參數, 並回到上步驟, 直至到最高/低解析度之定義池止。

7．若最佳匹配區塊之間的 MSE 大於門檻值 E_{th}, 則將值域區塊切成四等份, 並跳至步驟 3。

(a)

(b)

圖 7: 測試影像 (a) Lena (b) Goldhill

表 1: 提出方法各參數之位元分配

Coded type	bit allocation
Fractal code	i=3 bits α=3 bits Δg=6 bits
Coordinate	$\log_2(DP_s)$ bits [1]
Mean value	6 bits
Header	bit allocation
Quadtree partition flag	1 bit
Coded by fractal code	1 bit
N_{th}-resolution domain pool	N_{th} bits

[1] DP_s denote the domain pool size.

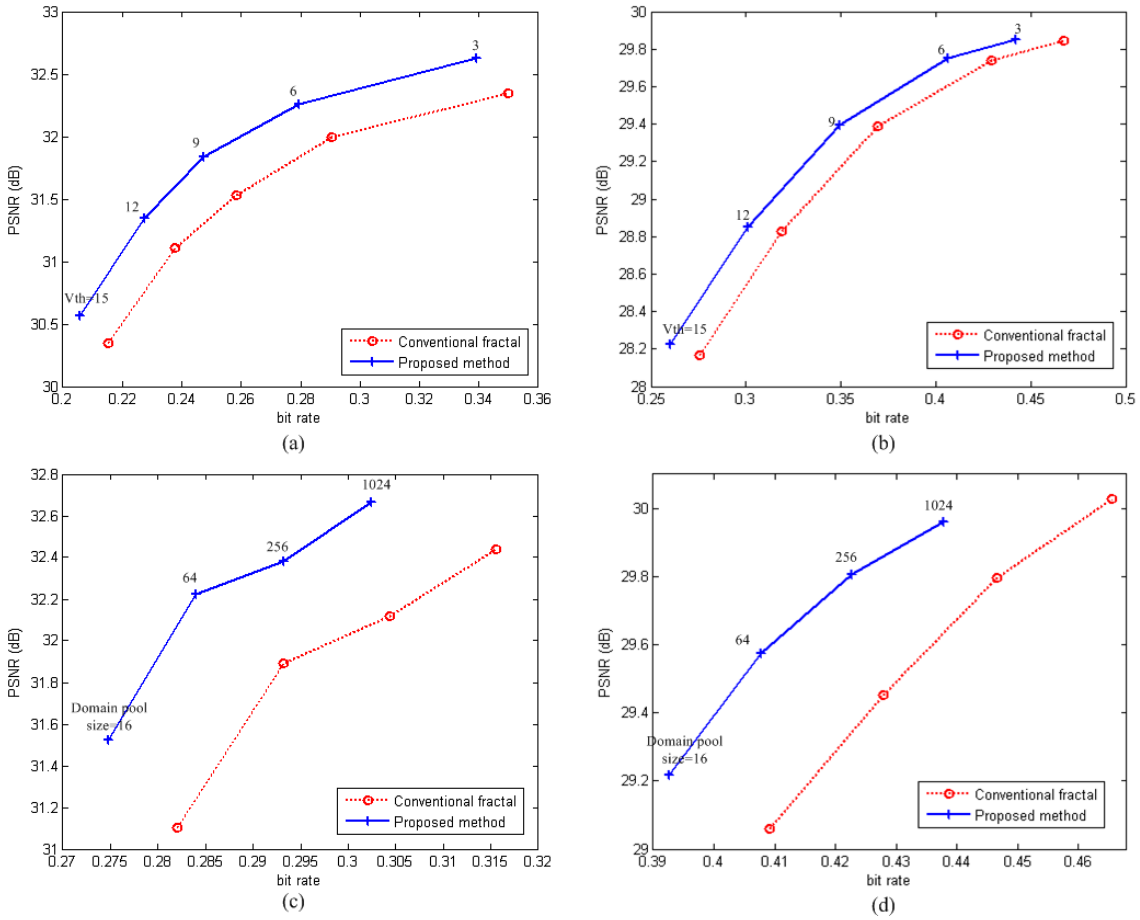

圖 8: 重建影像之 PSNR 比較圖: (a) Lena, $V_{th} = 15 \sim 3$. (b) Goldhill, $V_{th} = 15 \sim 3$. (c) Lena, 定義池大小$= 16 \sim 1024$. (d) Goldhill, 定義池大小$= 16 \sim 1024$.

4 實驗結果

圖7測試灰階影像爲 Lena 與 Goldhill, 影像大小 512×512, 父系值域區塊大小是8×8, 子系值域區塊大小爲4×4, 父系與子系定義池大小爲256, V_{th}與 E_{th}分別是$15 \sim 3$與10, 定義池的解析度分別爲 $14 \sim 25$共十層。原始解析度 16×16之定義池的取樣方式爲次取樣, 而在其它解析度的定義池採用鄰近取樣。當最佳編碼區塊位於第 N_{th}層解析度定義池時, 分配給 N_{th}個1 bit 的指示位元。所有參數的位元分配整理如表1所示, 並且沒有使用熵編碼 (Entropy coding) [7]。

圖8爲 Lena 與 Goldhill 的重建影像品質比較圖, 其中圖8 (a) (b) 爲使用不同 V_{th}來控制位元率, 而圖8 (c) (d) 爲使用不同子系的定義池大小來控制位元率, 其父系的定義池大小固定爲256, 可以很明顯的看出, 本文提出的方法都較傳統碎形影像編碼來的好. 從圖8 (c) (d) 可以看出, 當定義池越來越大時, 改善效果較爲不明顯, 這是因爲定義池已經夠大了, 使用多重解析度搜尋只能增加較少的重建影像品質, 但依然得付出相對的位元率來指示定義域區塊的解析度. 然而, 整體而言, 多重解析度搜尋法依然可以得到較佳的編碼效能. 表2顯示出在圖8 (a) 中父系對應的最佳匹配區塊數量, 可以明顯看出, 多重解析度搜尋法有效的找到更多父系最佳匹配區塊, 節省了輸出位元且又不至於影響重建影像品質太多。

表 2: Lena最佳匹配區塊數量之比較

	MRS	Conventional fractal
$V_{th} = 12$	860	784
$V_{th} = 9$	1107	1031
$V_{th} = 6$	1578	1502
$V_{th} = 3$	2538	2462

5 結論

本論文提出多重解析度搜尋法可以有效提升碎形影像編碼之重建影像品質。結合多重解析度搜尋法, 避免了傳統碎形影像編碼的定義池在取樣上可能取到非常平滑的影像區塊, 造成定義池效能不佳的情況。並藉由判斷再高/低一層是否有較佳之匹配區塊來減少編碼時間, 避免花費大量時間在多重解析度的定義池中搜尋, 以提昇整體編碼效能。

未來將試著把多重解析度的觀念套用到免疊代碎形影像編碼, 或其它混合型碎形影像編碼上, 以增進其編碼效能。

6 致謝

本論文爲國科會研究計畫之補助, 計畫編號爲NSC-95-2221-E-224-070-MY2, 謹此致謝。

References

[1] A. E. Jacquin, "Fractal image coding: a review," *Proceedings of IEEE*, vol. 81, pp. 1451–1465, October, 1993.

[2] Hsuan T. Chang and C. J. Kuo, "Iteration-free fractal image coding based on efficient Domain pool design," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 329–339, March, 2000.

[3] Hsuan T. Chang, Chung C. Lin, "Mutual image compression based on fractal mating coding," *in Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2004)*, Taipei, June, 2004.

[4] Hsuan T. Chang and Chung C. Lin, "Mutual image compression based on fractal mating coding," *in Proceedings of IEEE International Conference on Multimedia and Expo (ICME 2004)*, pp. 1087–1090, Taipei, June, 2004.

[5] Hsuan T. Chang, Chih C. Hsu, "Mutual image compression based on iteration-free fractal mating coding," *19th Computer Vision, Graphics, and Image Processing Conference*, Taoyuan, August, 2006.

[6] K. Belloulata, "Fast fractal coding of subbands using a non-iterative block clustering," *Journal of Visual Communication Image Representation*, vol. 16, pp. 55–67, February, 2005.

[7] K. Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, Second Edition, San Francisco, pp. 530–560, 2000.

[8] M. Barnsley, "A better way to compress images," *BYTE archive*, vol. 13, pp. 215–223, January 1988.

[9] S. Furao, O. Hasegawa, "A fast no search fractal image coding method," *Signal Processing: Image Communication*, vol. 19, pp. 393–404, February, 2004.

[10] W. A. Stapleton, K. Sripaipan, D. J. Jackson, K. G. Ricks, "Hybrid fractal/jpeg image compression: a case study," *Conference on Imaging Science, Systems and Technology 2005*, pp. 161–167, Nevada, USA, June, 2005.

[11] X. Wu, D. J. Jackson and H. C. Chen, "Novel fractal image-encoding algorithm based on a full-binary-tree searchless iterated function system," *Optical Engineering*, vol. 44, pp. (107002)1–10, October, 2005.

[12] Y. Iano, F. S. d. Silva, and A. L. M. Cruz, "A fast and efficient hybrid fractal-wavelet image coder," *IEEE Transactions On Image Processing*, vol. 15, no. 1, pp. 98–105, January, 2006.

[13] Y. Fisher,*Farctal Image Compression:Theory and Applications*, Springer Verlag, New York, January, 1995.