Bab 14

Pemampatan Citra Fraktal

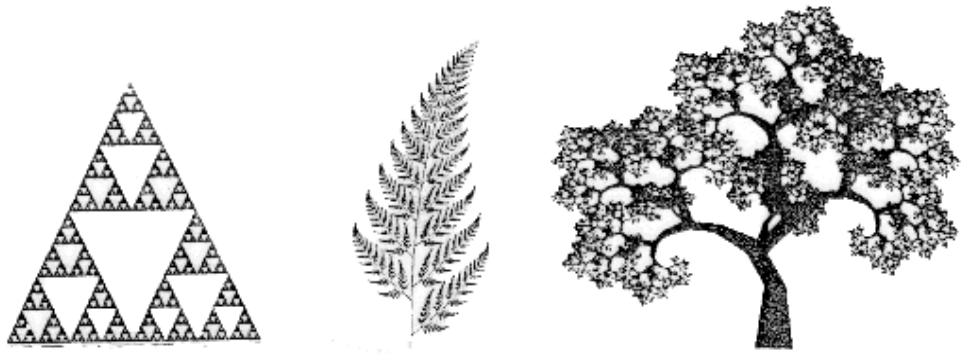
Metode pemampatan cira fraktal (*fractal image compression*) adalah metode *lossy* compression yang relatif baru. Metode ini mengeksploitasi kemiripan bagian-bagian di dalam citra dan menghitung transformasi yang memetakan bagian-bagian caitra yang memiliki kemiripan tersebut. Pembahasan di dalam Bab 14 ini dimulai dari teori mengenai fraktal, kemudian konsep pemampatan fraktal, lalu aplikasi pemampatan fraktal untuk memampatkan sembarang citra.

14.1 Definisi Fraktal

Sejarah fraktal dapat dirunut dari buku Benoit Mandelbrot yang berjudul *The Fractal Geometry of Nature*. Jika geometri Euclidean digunakan untuk merepresentasikan bentuk-bentuk yang dibuat oleh manusia (bujursangkar, lingkaran, bloa, dsb), maka geometri fraktal merupakan cara yang alami untuk merepresentasikan bentuk-bentuk objek di alam.

Fraktal dapat didefinisikan dari dua buah properti [MUN99]: 1) *self similarity*, dan 2) matra (*dimension*).

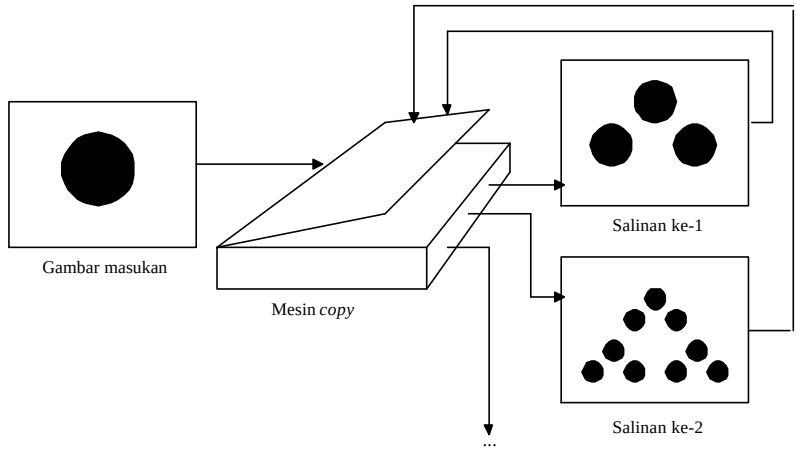
1. Fraktal adalah obyek yang memiliki kemiripan dirinya-sendiri (*self-similarity*) namun dalam skala yang berbeda. Ini artinya, bagian-bagian dari obyek akan tampak sama dengan obyek itu sendiri bila dilihat secara keseluruhan. Gambar 14.1 memperlihatkan tiga buah contoh fraktal, yaitu segitiga Sierpinski, daun pakis Barnsley, dan pohon fraktal.
2. Fraktal adalah objek yang memiliki matra bilangan riil atau pecahan (*fractional*). Kata terakhir inilah yang menurunkan kata **fraktal**.



Gambar 14.1 *Segitiga Sierpinski, daun pakis Barsnsley, dan pohon fractal*

14.2 *Iterated Function System* (IFS)

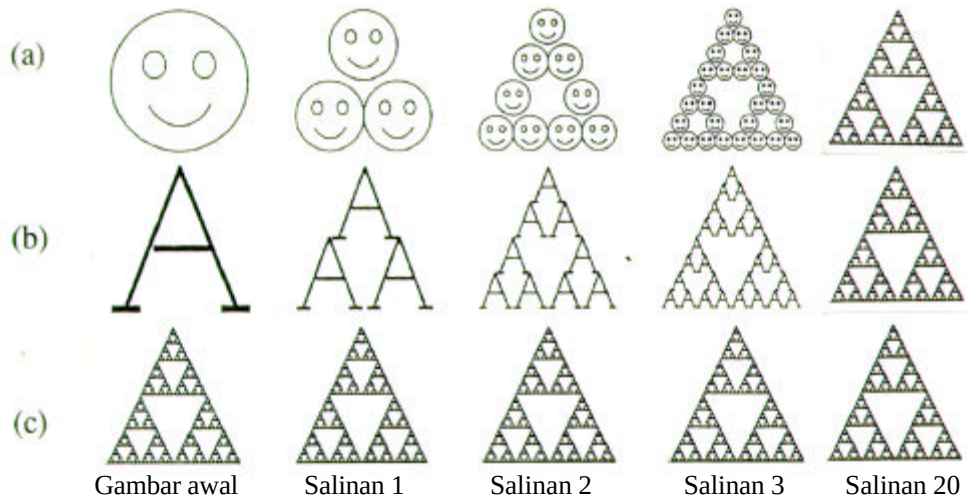
Michael Barnsley (1988) merepresentasikan fraktal ke dalam model matematika yang dinamakan IFS (*Iterated Function System*), melalui buku, *Fractals Everywhere.* IFS dimetaforakan sebagai sebuah mesin *foto copy* yang disebut *Multiple Reduction Copy Machine* (*MRCM*). *MRCM* memiliki banyak lensa dan setiap lensa melakukan pengecilan gambar dalam jumlah yang banyak. Gambar dihasilkan dari mesin *copy* dioperasikan kembali sebagai masukan untuk membuat salinan berikutnya (Gambar 14.2).



Gambar 14.2 *Multiple Reduction Copy Machine (MRCM)*

Hal yang menarik dari *MRCM* adalah, apapun gambar awal yang digunakan, MRCM selalu konvergen ke gambar akhir yang sama. Gambar 14.3 memperlihatkan hasil salinan setelah beberapa kali lelaran dari *MRCM* yang

disusun oleh tiga buah lensa, setiap lensa memiliki faktor pengecilan setiap lensa adalah ½. Gambar akhir yang dihasilkan selalu segitiga Sierpinski.



Gambar 14.3 *Apapun gambar awalnya, MRCM selalu menghasilkan segitiga Sierpienski .*

Secara matematis, sistem lensa pada *MRCM* dapat dinyatakan dengan sekumpulan transformasi *affine* w_1, w_2, …, w_n. Setiap transformasi w_i melakukan pencondongan, pemutaran, pengecilan, dan penggeseran terhadap salinan (*copy*) citra masukan.

Setiap transformasi *affine* dinyatakan sebagai matriks dengan enam buah elemen:

$$w = \begin{bmatrix} a & b & e \\ c & d & f \end{bmatrix} \tag{14.1}$$

Sembarang titik (x,y) pada gambar masukan ditansformasikan oleh w menjadi

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = w \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = Ax + t \tag{14.2}$$

Setiap transformasi *affine* w_i menghasilkan salinan citra yang lebih kecil; yaitu, untuk sembarang citra awal A yang diberikan, dihasilkan salinan *affine*, $w_1(A)$, $w_2(A)$, …, $w_n(A)$. Gabungan dari seluruh salinan tersebut adalah $W(A)$, yang merupakan keluaran dari mesin,

$$W(A) = w_1(A) + w_2(A) + \ldots + w_n(A) \tag{14.3}$$

W, yang dinamakan operator Hutchinson, adalah gabungan (*collage*) dari sejumlah transformasi individual w_i, yaitu

$$W = w_1 \cup w_2 \cup ... \cup w_n = \bigcup_{i=1}^{n} w_i \qquad (14.4)$$

Setiap transformasi *affine* w_i bersifat kontraktif, yaitu w_i memetakan dua buah titik menjadi lebih dekat. Ini berlaku untuk semua titik di bidang citra. Akibatnya, *MRCM* selalu menghasilkan salinan gambar yang ukurannya lebih kecil daripada ukuran gambar semula.

Sifat kontraktif saja tidak begitu penting. Sifat ini menjadi penting bila MRCM dijalankan dengan skema kalang umpan-balik terhadap gambar awal. Yaitu, diberikan citra awal A_0, diperoleh

$$A_1 = W(A_0) = \bigcup_{i=1}^{n} w_i(A_0)$$
$$A_2 = W(A_1) = W(W(A_0)) = W^2(A_0)$$
$$A_3 = W(A_2) = W(W(A_1)) = W(W(W(A_0))) = W^3(A_0)$$
$$\vdots$$
$$A_n = W^n(A_0)$$

Jika *W* seluruhnya kontraktif, maka untuk $n \to \infty$ lelarannya konvergen ke sebuah citra yang unik, A_∞. Citra A_∞ disebut titik-tetap (*fixed-point*) atau *invariant* dari proses lelaran, dan *attractor* dari *W*. Titik-tetap adalah citra A_∞ sedemikian sehingga

$$A_\infty = W(A_\infty) \qquad (14.5)$$

Jadi, jika A∞ dipilih sebagai citra awal, maka tidak ada perubahan pada hasil transformasinya. Sedangkan *attractor* dari *W* adalah citra A_∞ sedemikian sehingga

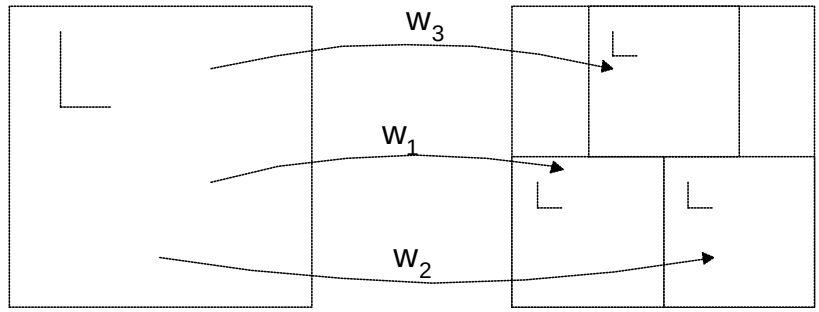
$$A_\infty = \lim_{n \to \infty} W^n(A_0) \text{ untuk sembarang citra awal } A_0 . \qquad (14.6)$$

Persamaan yang terakhir ini menyatakan bahwa tidak peduli apa pun citra awal yang digunakan, limit lelarannya selalu menghasilkan citra akhir yang sama. Dengan kata lain, citra *attractor* adalah unik.

Tansformasi *affine* yang menghasilkan citra titik-tetap segitiga Sierpinski adalah

$$w_1 = \begin{bmatrix} 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0.5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \quad w_3 = \begin{bmatrix} 0.5 & 0.0 & 0.25 \\ 0.0 & 0.5 & 0.5 \end{bmatrix}$$

Skema pembentukan segitiga Sierspinski dengan ketiga transformasi w_1, w_2, dan w_3 ditunjukkan pada Gambar 14.4.



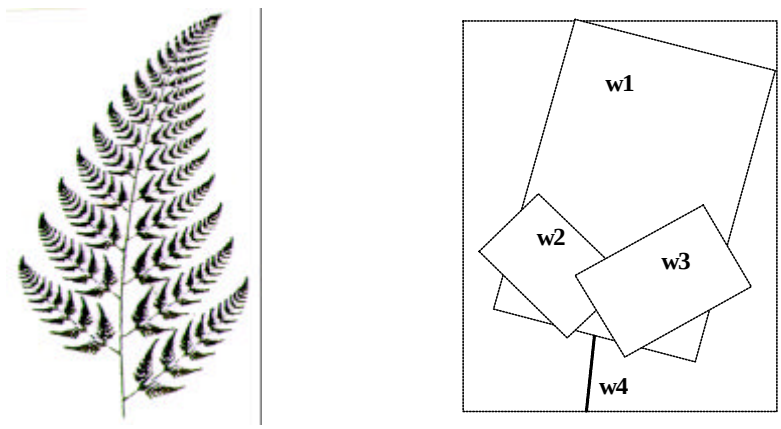
Gambar 14.4. *Cetak biru MRCM untuk membentuk segitiga Sierpinski.*

Jika jumlah transformasi *affine* meningkat menjadi empat dengan setiap w_i adalah sebagai berikut:

$$w_1 = \begin{bmatrix} 0.85 & 0.04 & 0.0 \\ -0.04 & 0.85 & 1.6 \end{bmatrix} \qquad w_2 = \begin{bmatrix} 0.20 & -0.26 & 0.0 \\ 0.23 & 0.22 & 1.6 \end{bmatrix}$$

$$w_3 = \begin{bmatrix} -0.15 & 0.28 & 0.0 \\ 0.26 & 0.52 & 0.44 \end{bmatrix} \qquad w_4 = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.16 & 0.0 \end{bmatrix}$$

maka *MRCM* konvergen ke citra fraktal yang terkenal, yang dinamakan tanaman pakis Barnsley (*Barnsley's fern*) – Gambar 14.5. Di sini, w_1 mengendalikan keseluruhan bentuk, w_2 membangkitkan daun kiri, w_3 membangkitkan daun kanan, dan w_4 menghasilkan batang.



Gambar 14.5 *Pakis Barnsley dan empat buah transformasi affine-nya*

14.3 Pengkodean Citra dengan *IFS*

Menyimpan citra sebagai kumpulan *pixel* membutuhkan memori yang besar, namun bila yang disimpan adalah transformasi *affine*-nya, maka memori yang dibutuhkan jauh lebih sedikit. Cara ini melahirkan gagasan pengkodean citra dengan nisbah pemampatan yang tinggi.

Pakis Barnsley misalnya, dibangkitkan dengan empat buah transformasi *affine*, masing-masingnya terdiri atas enam buah bilangan riil (4 *byte*), sehingga dibutuhkan $4 \times 6 \times 4$ *byte* = 96 *byte* untuk menyimpan keempat transformasi itu. Bandingkan bila citra pakis Barnsley disimpan dengan representasi *pixel* hitam putih (1 *pixel* = 1 *byte*) berukuran 550 × 480 membutuhkan memori sebesar 264.000 *byte*. Maka, nisbah pemampatan citra pakis adalah 264.000 : 96 = 2750 : 1, suatu nisbah yang sangat tinggi.

Kesulitan utama pemampatan dengan *IFS* adalah menemukan bagian citra yang mirip dengan keseluruhan citra. Intervensi manusia diperlukan untuk memandu menemukan bagian citra yang mirip dengan citra secara keseluruhan. Selain itu, pemampatan citra dengan IFS yang telah dikemukakan di atas hanya dapat diterapkan untuk citra yang memiliki *self-similarity* saja. Citra alami, disamping mempunyai warna, hampir tidak pernah *self-similar* secara keseluruhan. Karena itu, pemampatan sembarang citra (baik citra berwarna maupun citra *greyscale*) dengan *IFS* tidak dapat dilakukan.

Terobosan penting dibuat oleh Arnaud D. Jacquin, mahasiswa Barnsley. Melalui tulisan monumentalnya pada tahun 1992, Jacquin meyajikan skema otomatis pengkodean citra yang dikenal dengan nama *Partitoned Iterated Function System* (*PIFS*). *PIFS* dapat digunakan untuk memampatkan sembarang citra, baik citra skala-abu maupun citra berwarna, dan tidak terbatas untuk citra fraktal saja.

14.4 *Partitioned Iterated Function System* (PIFS)

Citra alami (*natural image*) umumnya hampir tidak pernah *self-similar* secara keseluruhan. Karena itu, citra alami pada umumnya tidak mempunyai transformasi *affine* terhadap dirinya sendiri. Tetapi, untunglah citra alami seringkali memiliki *self-similarity* lokal, yaitu memiliki bagian citra yang mirip dengan bagian lainnya, misalnya citra berskala-abu (*greyscale*) Lena pada Gambar 14.6 (bagian yang mirip ditandai di dalam kotak putih. Sebagai contoh, bagian topi Lena mirip dengan bagian topi di dalam bayangan cermin, bagian bahu mirip dengan bagian bahu yang lebih besar, dan sebagainya).

Kemiripan lokal yang banyak terdapat pada citra alami bersifat *self-transformability*, yaitu bagian citra yang lebih kecil dapat diperoleh dengan

mentransformasikan bagian citra yang lebih besar namun mirip dengan bagian citra yang lebih kecil itu [FIS94]. Setiap transformasi itu disebut *IFS* lokal. Gabungan dari seluruh hasil transformasi itu adalah citra fraktal yang menyerupai (atau menghampiri) citra semula.

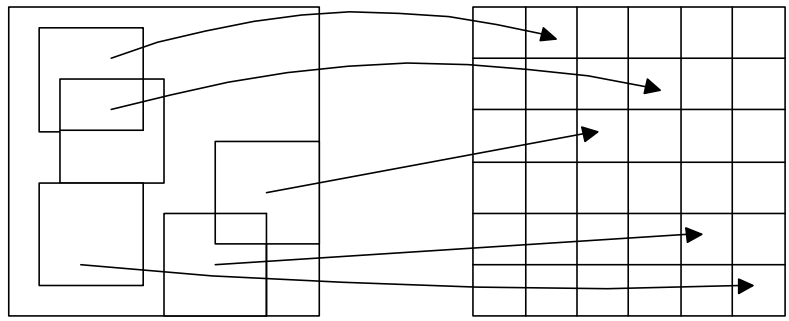
Gambar 14.6 *Kemiripan lokal pada citra Lena*

Langkah pertama yang dilakukan dalam proses pemampatan adalah membagi citra atas sejumlah blok yang berukuran sama dan tidak saling beririsan, yang disebut blok jelajah (*range*). Untuk menyederhanakan masalah, blok jelajah diambil berbentuk bujursangkar. Untuk setiap blok jelajah, dicari bagian citra yang berukuran lebih besar dari blok jelajah –yang disebut blok ranah (*domain*)- dan paling mirip (cocok) dengan blok jelajah tersebut (Gambar 14.7), kemudian turunkan transformasi *affine* (*IFS* lokal) w_i yang memetakan blok ranah ke blok jelajah. Hasil dari semua pemasangan ini adalah *Partitioned Iterated Function System* (*PIFS*).

Kemiripan antara dua buah (blok) citra diukur dengan metrik jarak. Metrik jarak yang banyak digunakan dalam praktek adalah metrik *rms (root mean square*):

$$d_{rms} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(z'_{ij} - z_{ij})^2} \qquad (14.7)$$

dengan *z* dan *z*' adalah nilai intensitas *pixel* dari dua buah citra, dan *n* adalah jumlah *pixel* di dalam citra.



Gambar 14.7 *Pemetaan dari blok ranah ke blok jelajah*

Seperti yang sudah dijelaskan, blok ranah dan blok jelajah keduanya berbentuk bujursangkar, tetapi ukuran blok ranah diambil dua kali blok jelajah. Untuk blok jelajah berukuran 8×8 *pixel* dan blok ranah berukuran 16×16 *pixel*, citra 256×256 misalnya, dapat dibagi menjadi 1024 buah blok jelajah yang tidak saling beririsan dan $(256 - 16 + 1)^2 = 58.081$ buah blok ranah berbeda (yang beririsan). Himpunan blok ranah yang digunakan dalam proses pencarian kemiripan dimasukkan ke dalam *pul ranah* (*domain pool*). Pul ranah yang besar menghasilkan kualitas pemampatan yang lebih baik, tetapi membutuhkan waktu pencocokan yang lebih lama.

Sebelum proses pencarian kemiripan dimulai, setiap blok ranah diskalakan sehingga ukurannya sama dengan ukuran blok jelajah. Penskalaan ini dimaksudkan agar jarak antara blok jelajah dan blok ranah mudah dihitung dengan persamaan 14.7. Penskalaan dilakukan dengan menjadikan 2×2 buah *pixel* menjadi satu buah *pixel*. Nilai satu buah *pixel* tersebut adalah rata-rata nilai keempat *pixel*.

Transformasi *affine* w_i untuk citra berskala-abu disusun oleh bagian spasial yang memetakan posisi *pixel* di blok ranah D_i ke posisi *pixel* di blok jelajah R_i, dan bagian intensitas yang mengubah nilai intensitas *pixel*. Titik (x,y) dengan intensitas z yang termasuk di dalam blok ranah dipetakan oleh w_i menjadi:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \qquad (14.8)$$

Dengan pemetaan w_i di atas, intensitas tiap *pixel* juga diskalakan dan digeser, yaitu

$$z' = s_i z + o_i \qquad (14.9)$$

Parameter s_i menyatakan faktor kontras *pixel* (seperti tombol kontras di TV). Bila s_i bernilai 0, maka *pixel* menjadi gelap, bila s_i sama dengan 1 kontrasnya tidak berubah; antara 0 dan 1 *pixel* berkurang kontrasnya, di atas 1 kontrasnya bertambah. Parameter o_i menyatakan ofset kecerahan (*brightness*) *pixel* (seperti tombol kecerahan di TV. Nilai o_i positif mencerahkan gambar dan nilai o_i negatif menjadikannya gelap. Kedua parameter tersebut dapat memetakan secara akurat blok jelajah berskala abu ke blok jelajah berskala abu.

Untuk menjamin efek kontraktif dalam arah spasial, maka blok ranah harus berukuran lebih besar daripada blok jelajah. Untuk alasan praktis, ukuran blok ranah diambil dua kali ukuran blok jelajah (perbandingannya 2:1). Jadi, jika ukuran blok jelajah adalah $B \times B$ *pixel*, maka ukuran blok ranah adalah $2B \times 2B$ *pixel.* Perbandingan ini membuat transformasi *affine* menjadi lebih sederhana, yaitu

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = w_i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{14.10}$$

Parameter e_i dan f_i mudah dihitung karena keduanya menyatakan pergeseran sudut kiri blok ranah ke sudut kiri blok jelajah yang bersesuaian. Sedangkan s_i dan o_i dihitung dengan menggunakan rumus regresi berikut [FIS94]: Diberikan dua buah blok citra yang mengandung n buah *pixel* dengan intensitas $d_1, d_2, \ldots, d_n$ (dari blok ranah D_i) dan $r_1, r_2, \ldots, r_n$ (dari R_i). Nilai s dan o diperoleh dengan meminimumkan total kuadrat selisih antara intensitas *pixel* blok D_i yang diskalakan dengan s lalu digeser sejauh o dan intensitas *pixel* blok R_i:

$$E = \sum_{i=1}^{n} (d'_i - r_i)^2 = \sum_{i=1}^{n} (s \cdot d_i + o - r_i)^2 \tag{14.11}$$

Nilai minimum E terjadi bila turunan parsialnya terhadap s dan o adalah nol, yang terjadi bila turunan pertama E sama dengan 0, atau

$$E' = 0$$

yang dipenuhi oleh

$$s = \frac{\left[n \sum_{i=1}^{n} d_i r_i - \sum_{i=1}^{n} d_i \sum_{i=1}^{n} r_i \right]}{\left[n \sum_{i=1}^{n} d_i^2 - \left(\sum_{i=1}^{n} d_i \right)^2 \right]} \tag{14.12}$$

dan

$$o = \frac{1}{n} \left[\sum_{i=1}^{n} r_i - s \sum_{i=1}^{n} d_i \right] \tag{14.13}$$

Jika $n\sum_{i=1}^{n} d_i^2 - (\sum_{i=1}^{n} d_i)^2 = 0$ maka $s = 0$ dan $o = \frac{1}{n}\sum_{i=1}^{n} r_i$. Dengan nilai s dan o yang telah diperoleh, maka kuadrat galat antara blok jelajah dan blok ranah adalah

$$E = \frac{1}{n}\left[\sum_{i=1}^{n} r_i^2 + s\left(s\sum_{i=1}^{n} d_i^2 - 2\sum_{i=1}^{n} d_i r_i + 2o\sum_{i=1}^{n} d_i\right) + o\left(no - 2\sum_{i=1}^{n} r_i\right)\right] \qquad (14.14)$$

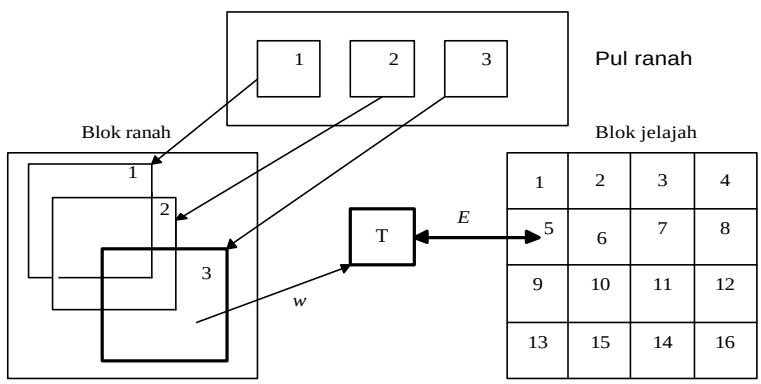
Metriks *rms*, d_{rms}, tidak lain adalah

$$d_{rms} = \sqrt{E}\ /n \qquad (14.15)$$

Selanjutnya, transformasi *affine* w_i diuji terhadap blok ranah D_i untuk menghasilkan blok uji $T_i = w_i(D_i)$ (Gambar 14.8). Jarak antara T dan R_i dihitung dengan persamaan 14.15. Transformasi *affine* yang terbaik ialah transformasi w yang meminimumkan jarak antara R_i dan T.

Runtunan pencarian dilanjutkan untuk blok jelajah berikutnya sampai seluruh blok jelajah sudah dipasangkan dengan blok ranah. Hasil dari proses pemampatan adalah sejumlah *IFS* lokal yang disebut *PIFS*. Seluruh parameter *PIFS* di-pak dan disimpan di dalam berkas eksternal. Parameter *PIFS* yang perlu disimpan hanya e_i, f_i, s_i, o_i, dan jenis operasi simetri untuk setiap blok jelajah. Dalam praktek, parameter e_i dan f_i diganti dengan posisi blok ranah yang dipetakan ke blok jelajah. Parameter a_i, b_i, c_i, dan d_i tidak perlu disimpan karena nilainya sudah tetap, yaitu ½ untuk a_i dan d_i, dan 0 untuk b_i dan c_i.

Algoritma pencocokan blok yang dijelaskan di atas adalah algoritma *brute force*, karena untuk setiap blok jelajah pencocokan dilakukan dengan seluruh blok ranah di dalam pul untuk memperoleh pencocokan terbaik.



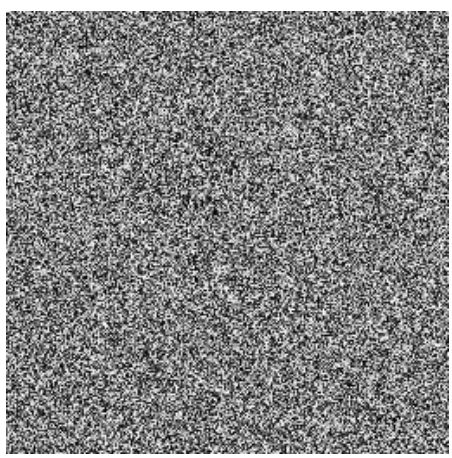
Gambar 14.8. *Blok jelajah 5 dibandingkan dengan blok ranah 3 di dalam pul ranah. Transformasi w ditentukan, lalu blok ranah 3 ditransformasikan dengan w menghasilkan T. Jarak antara T dengan blok jelajah 5 diukur.*

14.5 Rekonstruksi Citra

Rekonstruksi (*decoding*) citra dilakukan dengan melelarkan *PIFS* dari citra awal sembarang. Karena setiap *IFS* lokal kontraktif, baik kontraktif dalam matra intensitas maupun kontraktif dalam matra spasial maka lelarannya akan konvergen ke citra titik-tetap *PIFS*. Kontraktif intensitas penting untuk menjamin konvergensi ke citra semula, sedangkan kontaktif spasial berguna untuk membuat rincian pada citra untuk setiap skala. Jika *PIFS* yang ditemukan selama proses pemampatan bagus, yaitu gabungan dari transformasi seluruh blok ranah dekat dengan citra semula (diukur dengan persamaan 14.7), maka titik-tetap *PIFS* juga dekat dengan citra semula tersebut.

Selama proses pemulihan, setiap *IFS* lokal mentransformasikan sekumpulan blok ranah menjadi sekumpulan blok jelajah. Karena blok jelajah tidak saling beririsan dan mencakup keseluruhan *pixel* citra, maka gabungan seluruh blok jelajah menghasilkan citra titik-tetap yang menyerupai citra semula. Gambar 14.8 memperlihatkan hasil rekonstruksi citra Lena setelah 1, 2, dan 6 lelaran [MUN99]. Citra awal yang digunakan adalah citra adalah citra dengan nilai *pixel* yang dibangkitkan secara acak.

Konvergensi ke citra titik-tetap berlangsung cepat. Konvergensi umumnya dapat diperoleh dalam 8 sampai 10 kali lelaran [MUN99]. Karena transformasi *affine* kontraktif dalam arah spasial, maka semakin banyak rincian citra yang dibuat pada setiap lelaran



(a) Citra awal

(b) Lelaran ke-1

(c) Lelaran ke-2

(d) Lelaran ke-6

Gambar 14.8 *Rekonstruksi citra Lena [MUN99].*

Contoh-contoh hasil pemampatan citra fraktal [MUN99]:

Citra asli (256 × 256 *pixel*)
Lena.bmp (66 KB)

Citra hasil pemampatan fraktal
Lena.fra (7 KB)

Citra asli (256 × 256 *pixel*)
Collie.bmp (66 KB)

Citra hasil pemampatan fraktal
Collie.fra (9 KB)

Citra asli (512 × 512 *pixel*)
Kapal.bmp (258 KB)

Citra hasil pemampatan fraktal
Kapal.fra (9 KB)

Citra asli (316 × 404 *pixel*)
Potret.bmp (126 KB)

Citra hasil pemampatan fraktal
Potret.fra (17 KB)

Tabel 14.1 Perbandingan ukuran berkas citra sebelum dan sesudah dimampatkan

No.	Citra BMP (*byte*)	Ukuran (*byte*)	Citra FRA (*byte*)	Ukuran	Nisbah (%)
1	*Kapal.bmp*	263.222	***KAPAL512.FRA***	8.956	96,6
2	*Lena.bmp*	66.614	LENA256.FRA	8.137	87,6
3	*Collie.bmp*	66.614	COLLI256.FRA	9.150	86,3
4	*Potret.bmp*	128.782	POTRET.FRA	17.437	86,5

Tabel 14.2 Perbandingan ukuran citra berformat BMP, JPG, GIF, dan fraktal (FRA)

Nama Citra	Format BMP (*byte*)	Format JPG (*byte*)	Format GIF (*byte*)	Format FRA (*byte*)
Kapal.bmp	263.222	24.367	242.452	8.956
Lena.bmp	66.614	7.126	70.292	8.137
Collie.bmp	66.614	7.021	69.965	9.150
Potret.bmp	128.782	16.377	136.377	17.437

14.6 Pemampatan Citra Berwarna

Penelitian yang dilakukan tentang pemampatan citra fraktal kebanyakan ditujukan pada citra hitam-putih (*greyscale*). Karena citra berwarna terdiri atas 3 komponen (*RGB*), maka pemampatan fraktal dilakukan secara terpisah untuk masing-masing komponen. Tetapi, cara ini tidak dianjurkan karena membuat waktu pemampatan tiga kali lebih lama daripada cira skala-abunya, begitu pula ukuran berkas hasil pemampatannya tiga kali lebih besar. Teknik yang mangkus adalah dengan mentransformasi model *RGB* ke model *YIQ*, *XYZ*, atau *IHS*. Pemampatan fraktal cukup dilakukan terhadap komponen luminansinya (*Y* pada model *YIQ*, *Y* pada model *XYZ*, atau *I* pada model *IHS*) [FIS94]. Dua komponen sisanya dimampatkan dengan metode berbeda (misalnya dengan metode Huffman). Pada proses rekonstruksi citra, model *YIQ*, *XYZ*, atau *IHS* ditransformasikan kembali ke model *RGB*.