BAB III

METODE KOMPRESI DAN DEKOMPRESI

Kompresi citra fraktal memodelkan citra sebagai limit dari suatu proses iterasi. Jika diberikan suatu citra $A \in \mathrm{H}(X)$, metode ini akan mencari suatu proses W sedemikian sehingga titik tetap dari W adalah A dan $A = W(A)$ serta $A = \lim_{n \to \infty} W^{\circ n}(B)$ untuk setiap $B \in \mathrm{H}(X)$. Oleh karena itu, untuk menyimpan citra A, cukup dengan menyimpan proses W [10] [6].

Bab 3 ini menjelaskan tentang metode kompresi dan dekompresi citra fraktal. Penjelasan dimulai dengan pengenalan *Multiple Reduction Copy Machine* (MRCM) yang menjadi metode dasar pembentukan citra fraktal.

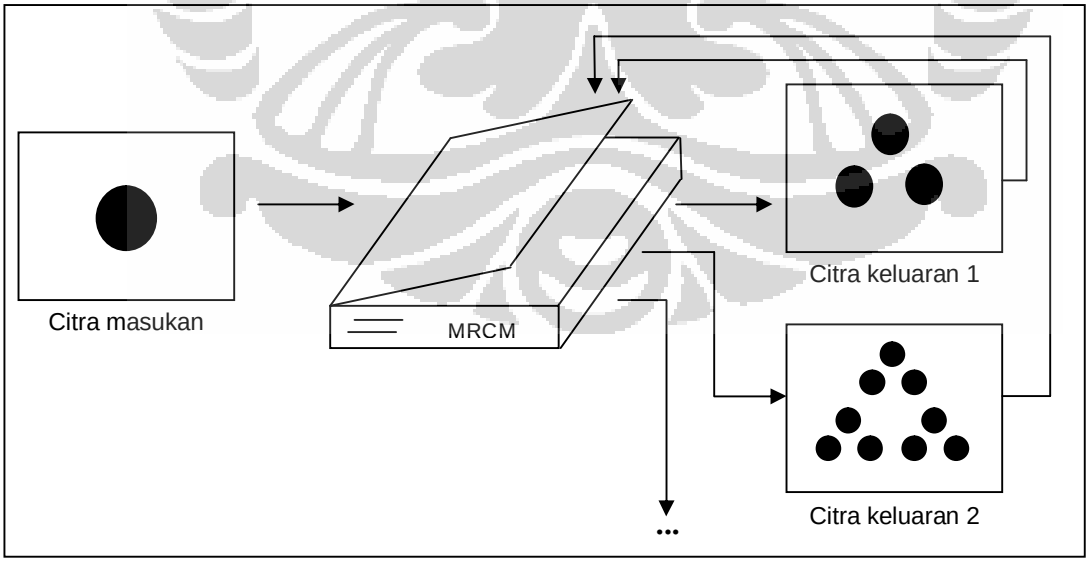
3.1 *MULTIPLE REDUCTION COPY MACHINE* (MRCM)

Multiple Reduction Copy Machine (MRCM) merupakan mesin fotokopi yang menerima citra masukan dan melakukan penyalinan terhadap citra masukan. MRCM memiliki beberapa lensa yang masing-masing lensa berfungsi melakukan penyalinan terhadap citra masukan dengan dimensi yang lebih kecil. Selain itu, setiap lensa pada MRCM juga melakukan

Kompresi Citra..., Lhuqita Fazry, FMIPA UI, 2008

pemutaran atau penggeseran serta menentukan posisi citra salinan. Citra keluaran dari MRCM merupakan gabungan dari citra salinan masing-masing lensa [12].

Proses penyalinan pada MRCM dilakukan menggunakan skema umpan balik. Citra keluaran pertama dijadikan sebagai citra masukan kedua. Setelah itu, MRCM melakukan proses penyalinan seperti proses yang pertama dan menghasilkan citra keluaran kedua. Citra keluaran kedua dijadikan sebagai citra masukan ketiga sehingga dihasilkan citra keluaran keempat. Proses ini dilanjutkan untuk beberapa iterasi. Prinsip dari skema umpan balik adalah citra keluaran pada proses sebelumnya dijadikan sebagai citra masukan untuk proses selanjutnya. Gambar 11 menunjukan skema umpan balik pada MRCM.

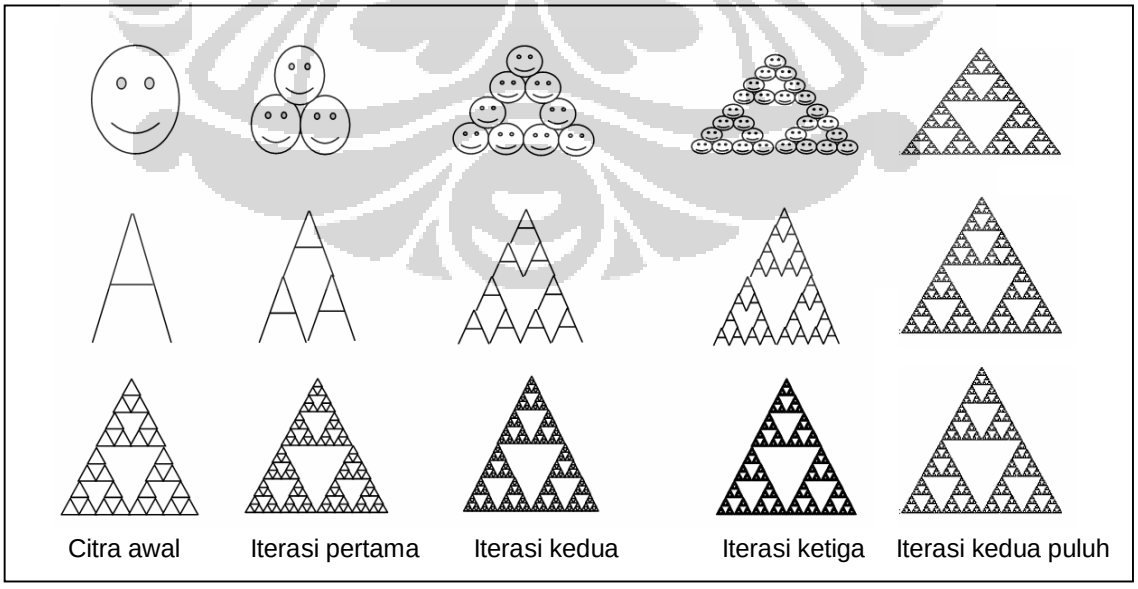


Gambar 11. Skema umpan balik pada MRCM [12]

Hal yang menarik dari MRCM adalah apapun citra masukan, citra keluaran yang dihasilkan akan konvergen ke citra akhir yang sama. Citra akhir ini disebut *attractor* dari MRCM [12] [6].

Setiap susunan lensa dalam MRCM akan menghasilkan *attractor* yang berbeda. Dengan kata lain, *attractor* yang dihasilkan dari MRCM tergantung kepada susunan lensa MRCM dan tidak tergantung pada citra masukan. Untuk suatu susunan lensa MRCM, apapun citra masukannya, *attractor* yang diperoleh adalah sama.

Berikut adalah contoh *attractor* dari suatu MRCM untuk beberapa citra masukan yang berbeda. MRCM yang dipakai memiliki tiga lensa, yang masing-masing lensa melakukan pengecilan terhadap sailnan citra masukan dengan faktor skala $\frac{1}{2}$ [12].



Gambar 12. *Attractor* dari MRCM untuk beberapa citra masukan [12]

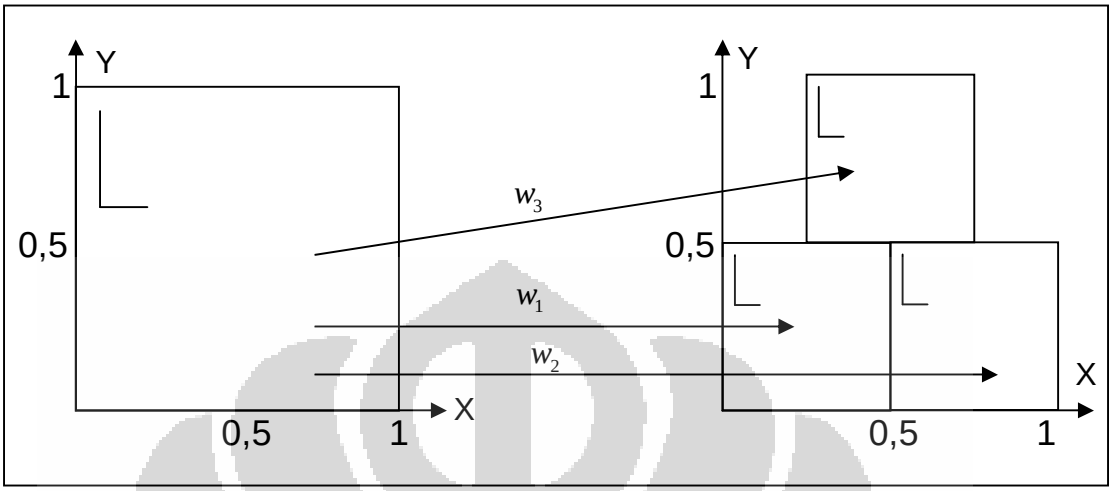
3.2 IFS MERUPAKAN MODEL MATEMATIS UNTUK MRCM

Secara matematis, sistem lensa pada MRCM dapat dimodelkan sebagai *Iterated Function System* (IFS) yang merupakan kumpulan transformasi *affine* kontraktif $w_1,\ w_2,\ w_3,\ ...,\ w_n$. Setiap transformasi *affine* w_i mewakili sebuah lensa pada MRCM. Untuk setiap citra masukan A, akan dihasilkan salinan $w_1(A),\ w_2(A),\ w_3(A),\ ...,\ w_n(A)$. Citra keluaran dari MRCM adalah $W(A)$ yang merupakan gabungan dari $w_i(A)$, yaitu $W(A)=\bigcup_{i=1}^{n} w_i(A)$.

Berikut adalah IFS yang mengkodekan segitiga *Sierpinski*:

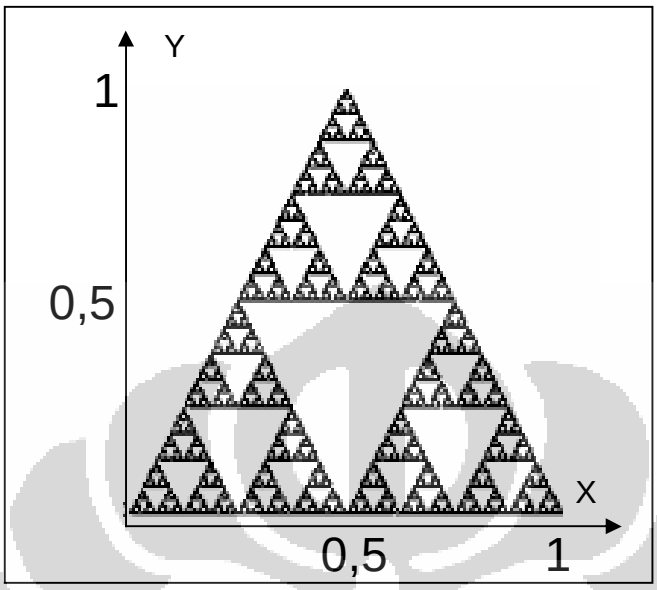
$$
\begin{aligned}
w_1\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 0,5 & 0 \\ 0 & 0,5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \\
w_2\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 0,5 & 0 \\ 0 & 0,5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+\begin{pmatrix} 0,5 \\ 0 \end{pmatrix} \\
w_3\begin{pmatrix} x \\ y \end{pmatrix} &= \begin{pmatrix} 0,5 & 0 \\ 0 & 0,5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}+\begin{pmatrix} 0,25 \\ 0,5 \end{pmatrix}
\end{aligned}
\tag{3.1}
$$

IFS yang mengkodekan segitiga *Sierpinski* terdiri dari tiga transformasi *affine* $w_i,\ \ i=1,\ 2,\ 3$. Masing-masing transformasi *affine* w_i melakukan proses pengecilan dengan faktor skala $\frac{1}{2}$. Perhatikan Gambar 13.



Gambar 13. Skema IFS yang mengkodekan segitiga *Sierpinski*

Transformasi w_1 melakukan pengecilan dengan faktor skala $\frac{1}{2}$ dan tidak melakukan pergeseran. Transformasi w_2 melakukan pengecilan dengan faktor skala $\frac{1}{2}$ dan melakukan pergeseran ke arah sumbu-x^+ sebesar 0,5 satuan. Transformasi w_3 melakukan pengecilan dengan faktor skala $\frac{1}{2}$ dan melakukan pergesaran kearah sumbu-x^+ sebesar 0,25 satuan dan ke arah sumbu-y^+ sebesar 0,5 satuan. Gambar 14 menunjukan *attractor* yang diperoleh dengan mengaplikasikan transformasi $w_i,\ i = 1,\ 2,\ 3$ pada sembarang citra awal dengan skema umpan balik.



Gambar 14. Segitiga *Sierpinski* sebagai *attractor* dari $w_i,\ i = 1,\ 2,\ 3$ [12]

3.3 KOMPRESI CITRA MENGGUNAKAN IFS

Segitiga *Sierpinski* dapat dikodekan menjadi kumpulan transformasi *affine* kontraktif (IFS). Kemampuan IFS untuk mengkodekan segitiga *Sierpinski* melahirkan gagasan untuk mengkodekan sembarang citra menggunakan IFS. Pengkodean seperti ini merupakan suatu metode kompresi citra, karena memori yang dibutuhkan untuk menyimpan IFS jauh lebih sedikit dibandingkan menyimpan setiap *pixel* penyusun citra *attractor* dari IFS.

Sebagai contoh: penyimpanan segitiga *Sierpinski* yang berdimensi 128×128 *pixel* dalam media penyimpanan (*storage*) membutuhkan memori

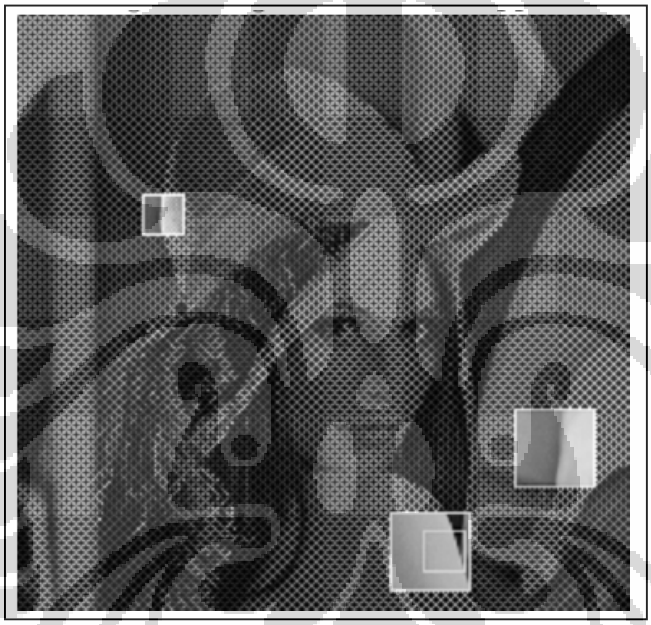
sebanyak $128 \times 128 \times 1 = 16.384$ bit (segitiga *Sierpinski* diatas adalah contoh citra binari atau citra 1 bit, karena hanya memiliki 2^1 skala keabuan yaitu 0 dan 1. Nilai 0 menunjukan intensitas hitam dan angka nilai 1 menunjukan intensitas putih). Akan tetapi jika yang disimpan adalah koefisien transformasi *affine* yang mengkodekan segitiga *Sierpinski*, memori yang dibutuhkan hanyalah 3 transformasi $\times 6$ (6 koefisien per transformasi) $\times 4$ byte = 72 byte (1 koefisien = 4 byte). Dengan mengkodekan segitiga *Sierpinski* menjadi IFS berarti diperoleh perbandingan ukuran (perbandingan antara ukuran citra semula dengan ukuran citra hasil pengkodean) yaitu 16.384:72 atau 227:1.

3.4 KOMPRESI CITRA MENGGUNAKAN *PARTITIONED ITERATED FUNCTION SYSTEM* (PIFS)

IFS dapat digunakan untuk mengkodekan segitiga *Sierpinski* yang merupakan citra fraktal, yakni bersifat *self-similarity*. Kesulitan utama dalam mengkodekan sembarang citra menggunakan IFS adalah menemukan bagian citra yang sama dengan keseluruhan citra. Pada sembarang citra, seringkali sangat sulit mendapatkan bagian citra yang sama dengan keseluruhan citra, bahkan mungkin tidak ada. Pada kondisi yang demikian, sangatlah sulit untuk mencari IFS yang mengkodekan citra tersebut. Dengan

kata lain, IFS tidak dapat digunakan untuk mengkodekan sembarang citra [12] [3].

Akan tetapi, pada sebagian besar citra, terdapat kemiripan antara satu bagian citra dengan bagian citra yang lain. Kemiripan ini disebut kemiripan lokal. Gambar 15 memperlihatkan kemiripan lokal pada citra.



Gambar 15. Kemiripan lokal pada citra [5]

Pada Gambar 15, kemiripan lokal terdapat pada daerah yang ditandai dengan kotak. Kotak yang lebih kecil mirip dengan kotak yang lebih besar. Untuk selanjutnya kotak-kotak ini disebut blok citra atau partisi citra. Kemiripan lokal antara satu blok dengan blok lainnya berarti bahwa kedua blok tersebut tidak tepat sama *pixel* per *pixel*, melainkan dimungkinkan terdapat kesalahan (*error*).

Kemiripan-kemiripan lokal ini memungkinkan untuk mencari transformasi *affine* w_i yang memetakan suatu blok D_i ke blok R_i sehingga $d\left(w_i\left(D_i\right),\ R_i\right)$ cukup kecil, dengan d adalah metrik rms. Blok D_i disebut blok ranah dan blok R_i disebut blok jelajah. Ukuran blok ranah dibuat lebih besar dari ukuran blok jelajah agar transformasi *affine* w_i bersifat kontraktif. Kumpulan transformasi w_i disebut *Partitioned Iterated Function System* (PIFS).

Perbedaan antara PIFS dengan IFS terletak pada daerah asal (*domain*) transformasi *affine* pada PIFS dan IFS. Daerah asal transformasi *affine* pada IFS adalah keseluruhan citra, sedangkan daerah asal transformasi *affine* pada PIFS dibatasi pada blok citra tertentu. Berbeda dengan IFS, PIFS dapat digunakan untuk mengkodekan sembarang citra [4] [12] [3] [6].

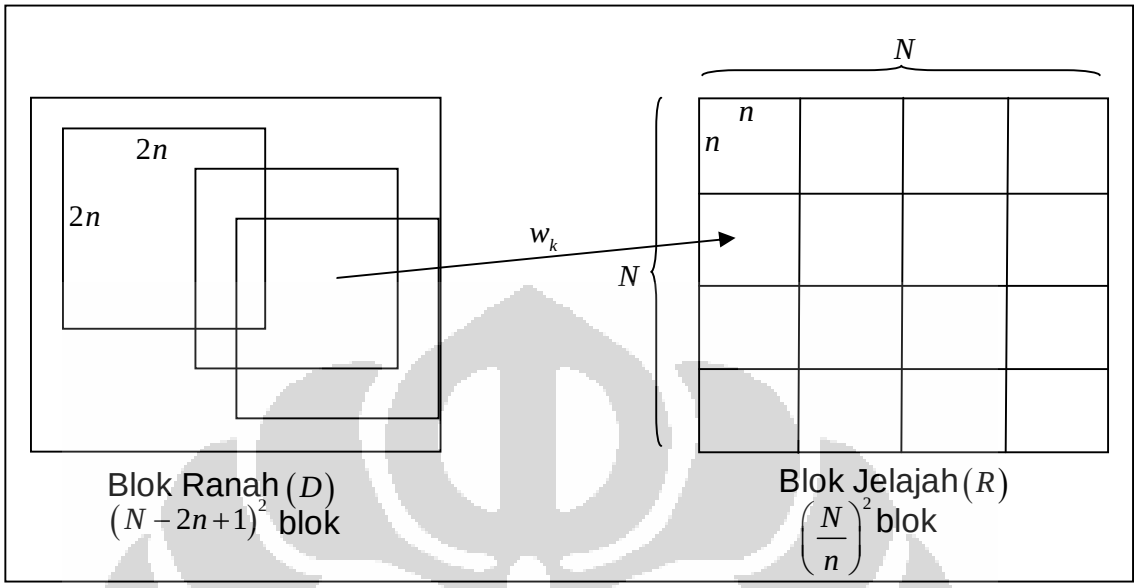
Alur proses kompresi citra menggunakan PIFS adalah sebagai berikut:

1. Partisi citra menjadi sejumlah blok yang berukuran sama dan tidak beririsan (*nonoverlaping*) disebut dengan blok jelajah (*range block*) dan dinotasikan dengan R.
2. Partisi citra menjadi sejumlah blok yang berukuran sama dan boleh beririsan (*overlaping*) disebut blok ranah (*domain block*) dinotasikan dengan D. Dalam aplikasi ukuran blok ranah dibuat dua kali ukuran blok jelajah.

3. Untuk setiap blok jelajah (R_i), dicari blok ranah (D_i) yang paling mirip dengan blok jelajah (R_i). Kemudian diturunkan transformasi *affine* w_i yang memetakan blok ranah (D_i) ke blok jelajah (R_i) sehingga $d\left(w_i\left(D_i\right),\ R_i\right)$ cukup kecil. Dalam aplikasi, transformasi *affine* yang digunakan adalah satu dari transformasi *affine* pada Tabel 2.

Sebagai ilustrasi proses kompresi, diberikan citra berdimensi $N \times N$. Partisi citra menjadi blok jelajah yang tidak saling beririsan dan berukuran $n \times n$, kemudian partisi citra menjadi blok ranah yang boleh beririsan dan berukuran $2n \times 2n$. Blok jelajah yang terbentuk adalah $\left(N-2n+1\right)^2$ blok dan blok ranah yang terbentuk adalah $\left(\frac{N}{n}\right)^2$ blok.

Setiap blok jelajah $\left\{R_i\right\}_{i=1}^{\left(\frac{N}{n}\right)^2}$ dipasangkan dengan setiap blok ranah $\left\{D_j\right\}_{j=1}^{\left(N-2n+1\right)^2}$ dan setiap transformasi *affine* $\left\{w_k\right\}_{k=0}^{7}$ pada Tabel 2 dan untuk setiap blok jelajah dipilih blok ranah dan transformasi *affine* w_k yang membuat $d\left(w_k\left(D_j\right),\ R_i\right)$ minimal. Gambar 16 menunjukan skema pemetaan blok ranah ke blok jelajah.



Gambar 16. Pemetaan blok ranah ke blok jelajah

Tabel 2. Transformasi *Affine* [3]

No	Matriks	Deskripsi
0	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	Identitas
1	$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$	Pencerminan terhadap sumbu-
2	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Pencerminan terhadap sumbu $-x$
3	$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$	Pemutaran 180^0
4	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	Pencerminan terhadap garis $y = x$
5	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	Pemutaran 90^0
6	$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$	Pemutaran 270^0
7	$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$	Pencerminan terhadap garis $y = -x$

Penyesuaian nilai intensitas *pixel* perlu dilakukan antara blok ranah dan blok jelajah. Setiap *pixel* $(x, y) \in D_i$ dengan nilai intensitas *pixel* z, dipetakan menggunakan persamaan:

$$z' = s_i z + o_i \tag{3.2}$$

Dengan pemetaan diatas, maka intensitas *pixel* juga diskalakan dan digeser sehingga transformasi *affine* yang digunakan berbentuk:

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix} \tag{3.3}$$

Parameter s_i menyatakan faktor kontras *pixel* (seperti tombol kontras pada televisi). Bila s_i bernilai 0, maka *pixel* akan menjadi gelap, bila s_i bernilai 1, kontras dari *pixel* tidak berubah, bila nilai s_i antara 0 dan 1 maka *pixel* berkurang kontrasnya, sedangkan nilai s_i diatas 1 menyebabkan kontras bertambah [12].

Parameter o_i menyatakan kecerahan (*brightness*) suatu *pixel* (seperti tombol kecerahan pada televisi). Nilai o_i positif mencerahkan citra dan nilai o_i negatif menjadikan citra gelap [12].

Diberikan dua blok citra yang mengandung n *pixel* dengan nilai intensitas *pixel* $d_1, d_2, ..., d_n$ (dari blok ranah D_i setelah dipetakan) dan $r_1, r_2, ..., r_n$ (dari blok jelajah R_i), nilai s dan nilai o diperoleh dengan

meminimumkan total kuadrat selisih antara intensitas *pixel* blok D_i yang diskalakan dengan s lalu digeser sejauh o dan intensitas *pixel* R_i:

$$E = \sum_{i=1}^{n} \left(d'_i - r_i \right)^2 = \sum_{i=1}^{n} \left(s \cdot d_i + o - r_i \right)^2 \tag{3.4}$$

Nilai minimum terjadi bila turunan parsial terhadap s dan o adalah 0, yang terjadi bila $E' = 0$, yang dipenuhi oleh:

$$s = \frac{\left[n \sum_{i=1}^{n} d_i r_i - \sum_{i=1}^{n} d_i \sum_{i=1}^{n} r_i \right]}{\left[n \sum_{i=1}^{n} d_i^2 - \left(\sum_{i=1}^{n} d_i \right)^2 \right]} \tag{3.5}$$

dan

$$o = \frac{1}{n} \left[\sum_{i=1}^{n} r_i - s \sum_{i=1}^{n} d_i \right] \tag{3.6}$$

Jika $n \sum_{i=1}^{n} d_i^2 - \left(\sum_{i=1}^{n} d_i \right)^2 = 0$ maka $s = 0$ dan $o = \frac{1}{n} \sum_{i=1}^{n} r_i$.

Berikut adalah algoritma kompresi citra fraktal:

Algoritma Kompresi Citra Fraktal

```
{input: citra I(NxN pixel), n(ukuran blok jelajah);
output: PIFS}
1  Partisi citra menjadi blok jelajah(R) dengan ukuran n
2  Partisi citra menjadi blok ranah(D) dengan ukuran 2n
3  for i = 1 to (N/n)^2 do
4      w = -1, dom = -1
5      counter = infinity
6      for j = 1 to (N-2n+1)^2 do
7          for k = 0 to 7 do
8              A = Wk(Dj)
9              s = d(A, Ri)
10             if s < counter then
11                 counter = s
12                 w = Wk, dom = j
13             end if
14         end for
15     end for
16     simpan transformasi w dan dom
17 end for
18 PIFS = gabungan w dan dom
```

3.5 ALGORITMA GENETIKA PADA KOMPRESI CITRA FRAKTAL

Pencarian transformasi-transformasi (PIFS) dari blok jelajah ke blok ranah merupakan permasalahan dengan kompleksitas cukup tinggi. Sebagai ilustrasi: citra yang akan dikompresi berdimensi 256×256; ukuran blok jelajah diambil sebesar 8×8 sehingga blok jelajah yang terbentuk sebanyak 1024 blok, sedangkan blok ranah berukuran 16×16 (dua kali ukuran blok jelajah) sehingga blok ranah yang terbentuk sebanyak $(256 - 16 + 1)^2 = 58.081$ blok.

Dengan kondisi tersebut maka terdapat sebanyak $(1.024 \times 58.081 \times 8)$ atau $475.799.552$ pemasangan. Angka 8 menunjukan delapan kemungkinan transformasi *affine* pada Tabel 2. Jumlah pemasangan ini sangatlah besar sehingga memperlama waktu kompresi.

Algoritma genetika dapat digunakan untuk menyelesaikan permasalahan pencarian PIFS pada kompresi citra fraktal [7][15]. Penggunaan algoritma genetika dalam kompresi citra fraktal diharapkan mampu memperkecil jumlah pemasangan blok ranah dan blok jelajah sehingga dapat mempersingkat waktu kompresi.

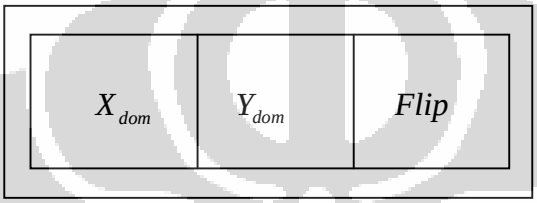
Algoritma genetika diterapkan untuk setiap blok jelajah. Jadi, untuk setiap blok jelajah akan dicari blok ranah dan transformasi *affine* pada Tabel 2 yang paling cocok. Pencarian ini dilakukan secara genetika. Oleh karena itu representasi kromosom yang digunakan adalah parameter dari PIFS, yaitu

1. Lokasi dari blok ranah yang sesuai
2. Jenis transformasi *affine* yang digunakan

Komponen algoritma genetika pada kompresi citra fraktal terdiri atas:

a. Kromosom

Berikut adalah representrasi kromosom yang digunakan [7]:



Gambar 17. Representasi kromosom

Keterangan Gambar 17:

$X_{dom} \in [1,\ L-d+1]$, L menunjukan panjang citra

$Y_{dom} \in [1,\ W-d+1]$, W menunjukan lebar citra

$Flip \in [0,\ 7]$, 8 kemungkinan transformasi affine pada Tabel 2

d adalah ukuran dari blok ranah.

X_{dom} pada kromosom di atas menyatakan absis untuk suatu blok ranah yang mungkin, sedangkan Y_{dom} menyatakan ordinat untuk blok ranah yang sama, sedangkan $Flip$ menyatakan indeks yang menyatakan salah satu transformasi *affine* pada Tabel 2.

b. Nilai *Fitness*

Nilai *fitness* (T) yang digunakan adalah [7]

$$T = \frac{100}{rms+1} \tag{3.7}$$

Nilai *fitness* merupakan nilai yang diberikan kepada setiap kromosom untuk mengukur seberapa baik kromosom tersebut dijadikan sebagai solusi. Dalam kompresi citra fraktal, hal ini berarti seberapa cocok blok ranah-yang direpresentasikan oleh kromosom tersebut- dengan blok jelajah yang diberikan. Semakin besar nilai *fitness,* maka kecocokan antara blok ranah dan blok jelajah semakin besar. Nilai *fitness* dibuat berbanding terbalik dengan *rms*, agar nilai *fitness* semakin besar ketika *rms* semakin kecil.

c. Seleksi Orangtua

Proses seleksi orangtua dilakukan menggunakan metode *roulette-wheel (roda roulette)* yang proporsional terhadap nilai *fitness*nya. Metode *roulette wheel* ini telah dijelaskan pada bab II.

d. Pindah Silang

Pindah silang antara orangtua 1 (P1) dan orangtua 2 (P2) akan menghasilkan dua kromosom anak. Skema pindah silang yang digunakan adalah sebagai berikut [7]:

Untuk anak 1:

$$X_{dom} = a * X^{P1}{}_{dom} + (1-a) * X_{dom}{}^{P2}$$
$$Y_{dom} = a * Y^{P1}{}_{dom} + (1-a) * Y_{dom}{}^{P2} \quad (3.8)$$

Untuk anak 2 :

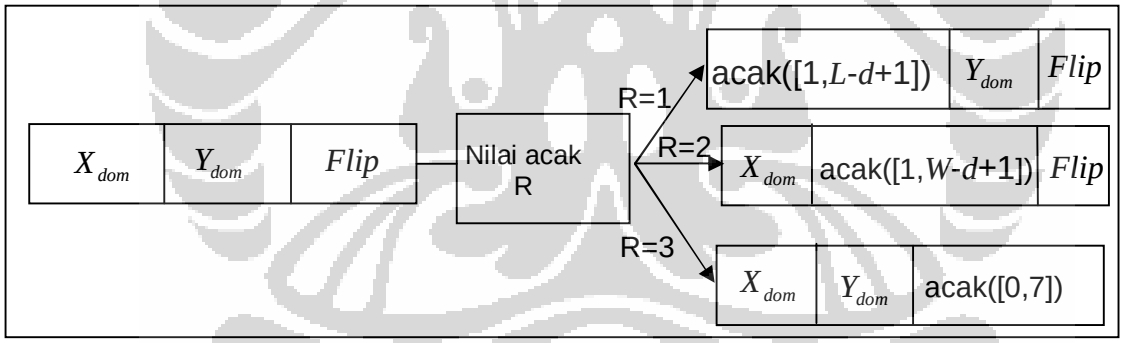
$$X_{dom} = (1-a) * X^{P1}{}_{dom} + a * X_{dom}{}^{P2}$$
$$Y_{dom} = (1-a) * Y^{P1}{}_{dom} + a * Y_{dom}{}^{P2} \quad (3.9)$$

a adalah bilangan random pada interval $[0,1]$

Proses pindah silang dilakukan terhadap beberapa kromosom dalam satu populasi. Absis (X_{dom}) dan ordinat (Y_{dom}) untuk anak 1 diperoleh dari kombinasi absis kromosom orangtuanya seperti pada persamaan (3.8), sedangkan $Flip$ yang digunakan adalah $Flip$ orangtua 1. Absis (X_{dom}) dan ordinat (Y_{dom}) untuk anak 2 diperoleh dari kombinasi absis kromosom orangtuanya seperti pada persamaan (3.9), sedangkan $Flip$ yang digunakan adalah $Flip$ orangtua 2.

e. Mutasi

Sebuah kromosom dimutasi menggunakan skema sebagai berikut [7]:



Gambar 18. Skema mutasi

L, W, d berturut-turut menyatakan panjang dan lebar citra serta ukuran dari blok ranah.

Proses mutasi dilakukan pada beberapa kromosom pada populasi yang dipilih dengan menggunakan metode *roulette wheel*. Mutasi yang dilakukan tergantung pada nilai R yang dibangkitkan secara acak. Nilai R yang mungkin adalah 1, 2, atau 3. Jika nilai R yang diperoleh adalah 1, maka

akan dilakukan penggantian pada absis (X_{dom}) dengan suatu nilai yang dibangkitkan secara acak dan bernilai dari 1 sampai $L-d+1$. Jika nilai R yang diperoleh adalah 2, maka akan dilakukan penggantian pada ordinat (Y_{dom}) dengan suatu nilai yang dibangkitkan secara acak dan bernilai dari 1 sampai $W-d+1$. Sedangkan, jika nilai R yang diperoleh adalah 3, maka akan dilakukan penggantian pada $Flip$ dengan suatu nilai yang dibangkitkan secara acak dan bernilai dari 0 sampai 7 yang menunjukan indeks transformasi *affine* pada Tabel 2.

Proses pindah silang dan mutasi tidak terjadi setiap saat, akan tetapi tergantung pada probabilitas pindah silang dan probabilitas mutasi.

Berikut adalah *pseucode* kompresi citra fraktal menggunakan algoritma genetika [7]:

Algoritma Genetika pada Kompresi Citra Fraktal

```
{Input : I, maksimum generasi, batas fitness
Output : PIFS}
1  Partisi citra menjadi blok jelajah(R)dan blok ranah(D)
2  for Ri pada R do
       {algoritma genetika}
3      inisialisasi populasi
4      evaluasi fitness setiap kromosom
5      while (blok ranah yang optimal belum didapat) and
       (generasi terakhir belum tercapai) do
6          generate populasi baru (menggunakan seleksi,
           pindah silang, mutasi)
7          evaluasi fitness setiap kromosom
8      end while
9      simpan transformasi wi dan blok ranah
10 end for
11 Tulis setiap transformasi wi dan blok ranah dalam file
```

Algoritma di atas menerima citra masukan I yang akan dikompresi, maksimum generasi serta batas *fitness*. Algoritma dimulai dengan mempartisi citra menjadi dua partisi citra yang berbeda yaitu blok ranah (D) dan blok jelajah (R) dengan ukuran blok ranah dua kali ukuran blok jelajah. Untuk

setiap blok jelajah (R_i) akan dicari blok ranah (D_i) dan transformasi *affine* yang cocok secara genetika. Langkah selanjutnya adalah membangkitkan inisial kromosom secara acak yang kemudian dihitung nilai *fitness* untuk setiap kromosom. Jika terdapat sebuah kromosom yang memiliki nilai *fitness* melebihi batas *fitness* yang diberikan maka blok ranah yang direpresentasikan oleh kromosom tersebut merupakan solusi bagi blok jelajah (R_i). Jika tidak ditemukan kromosom dengan nilai fitness yang melebihi batas *fitness* yang diberikan maka dilakukan proses seleksi, pindah silang dan mutasi untuk memperoleh populasi kromosom yang baru. Iterasi berlanjut sampai dicapai generasi terkahir atau diperoleh kromosom dengan nilai *fitness* yang melebihi batas *fitness* yang diberikan.

3.6 METODE DEKOMPRESI

Proses dekompresi dilakukan dengan menerapkan transformasi-transformasi (PIFS) hasil kompresi terhadap sembarang citra awal. Proses ini dilakukan menggunakan skema umpan balik seperti pada MRCM. Sebagaimana MRCM, citra keluaran dari PIFS adalah unik dan konvergen ke citra asli.

Berikut adalah *pseucode* dari proses dekompresi:

Algoritma Dekompresi

```
{input: jumlah iterasi(N), citra awal(I), PIFS(W)
output: citra asli(A)}
1  A = I
2  for i = 1 to N do
3      A = W(A)
4  end for
```