25 - Pemampatan Citra

(Bagian 2)

IF4073 Interpretasi dan Pengolahan Citra

Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2021

Fractal Image Compression

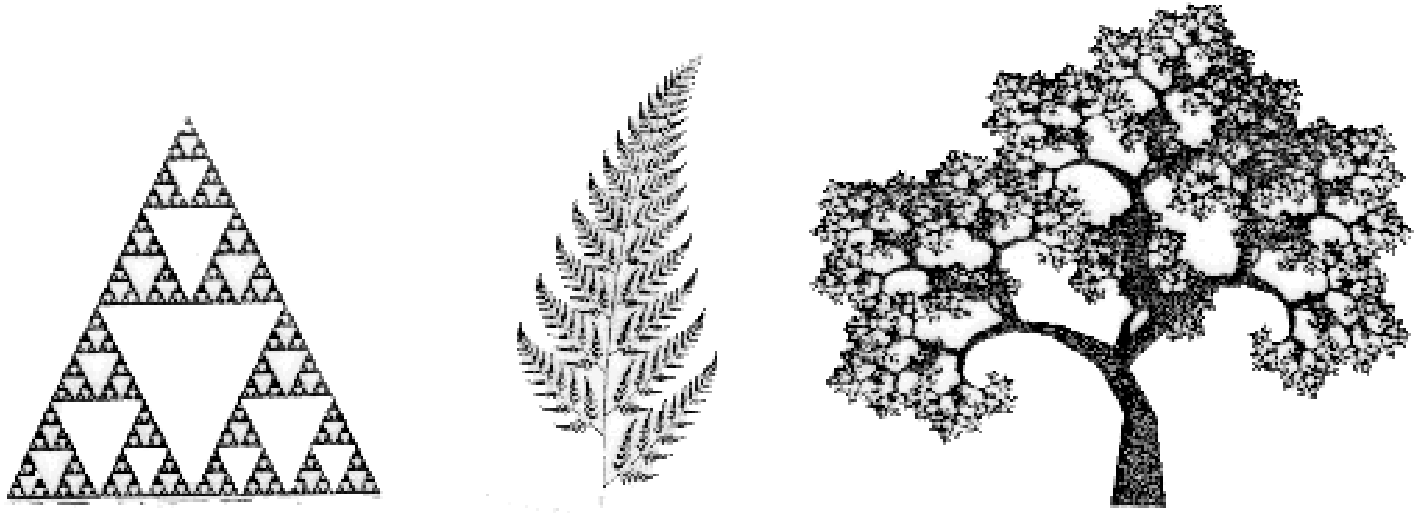
- Algoritma pemampatan citra dengan cara kerja yang unik.
- Prinsip: mencari bagian di dalam citra yang memiliki kemiripan dengan bagian lainya namun ukurannya lebih besar (*self similarity*).
- Cari matriks yang mentransformasikan bagian yang lebih besar tersebut dengan bagian yang lebih kecil.
- Simpan hanya elemen-elemen dari sekumpulan matriks transformasi tersebut (yang disebut matriks transformasi *affine*).
- Pada proses penirmampatan, matriks ransformasi *affine* di-iterasi sejumlah kali terhadap sembarang citra awal.

Fraktal

Definisi

Fraktal: objek yang memiliki kemiripan dirinya-sendiri (*self-similarity*) namun dalam skala yang berbeda.

Fraktal: objek yang memiliki matra berupa pecahan (*fractional*). Kata terakhir inilah yang menurunkan kata **fraktal**

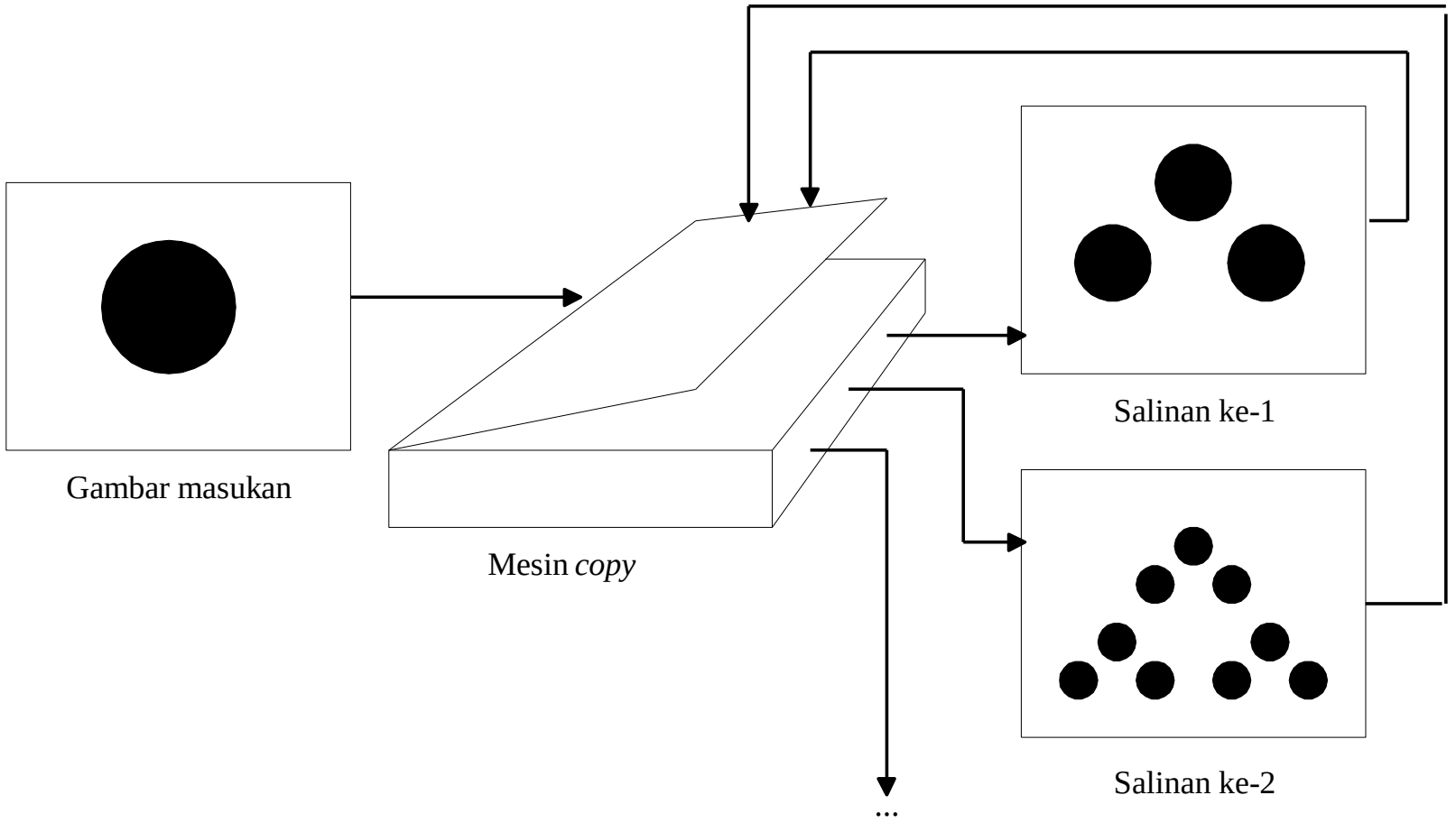


Segitiga Sierpinski, daun pakis Barsnsley, dan pohon fractal

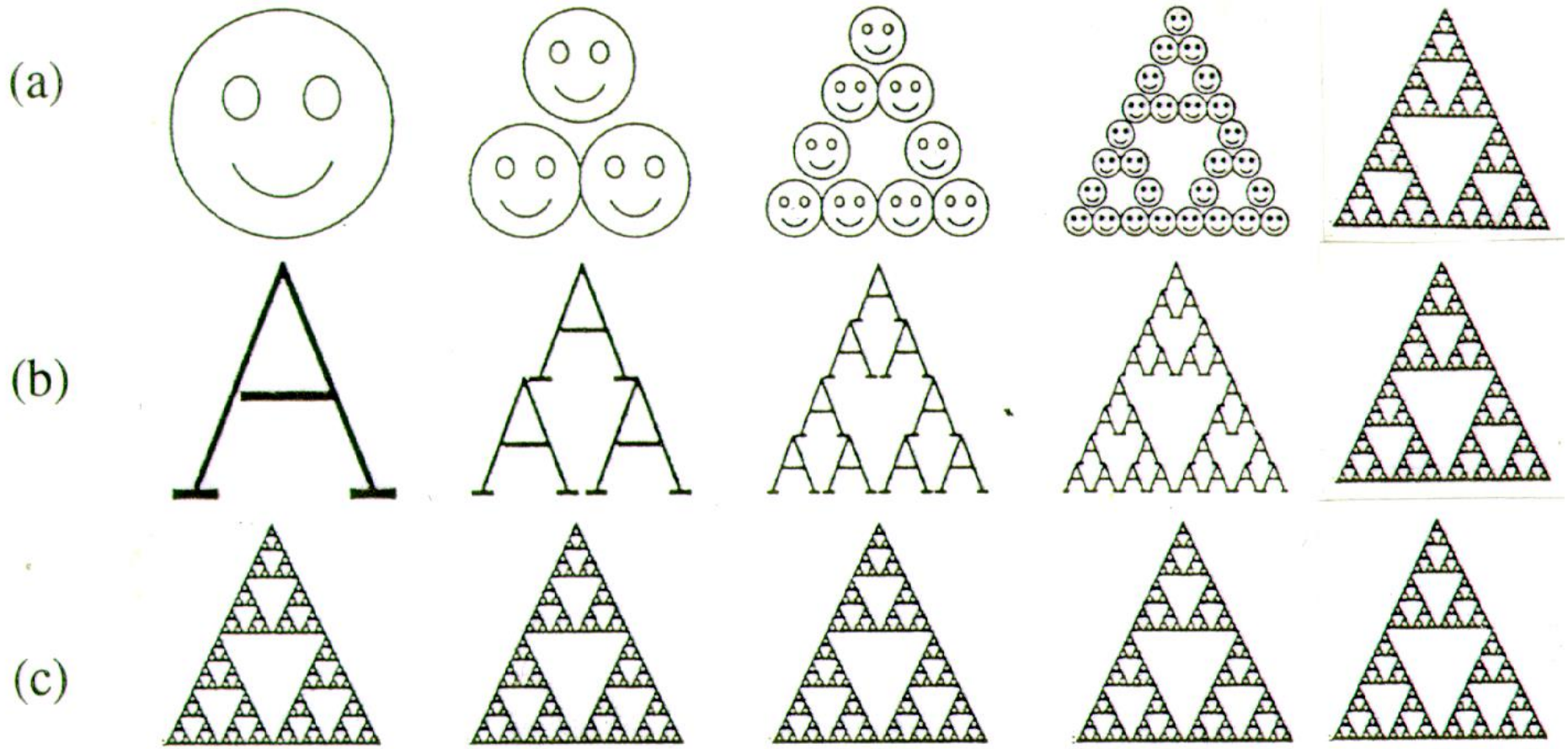
Fraktal di alam

Iterated Function System (IFS)

Penemu: Michael Barnsley (1988)



Multiple Reduction Copy Machine (MRCM)



Apapun gambar awalnya, MRCM selalu menghasilkan segitiga Sierpienski .

- MRCM dapat dinyatakan dalam bentuk transformasi *affine*:

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

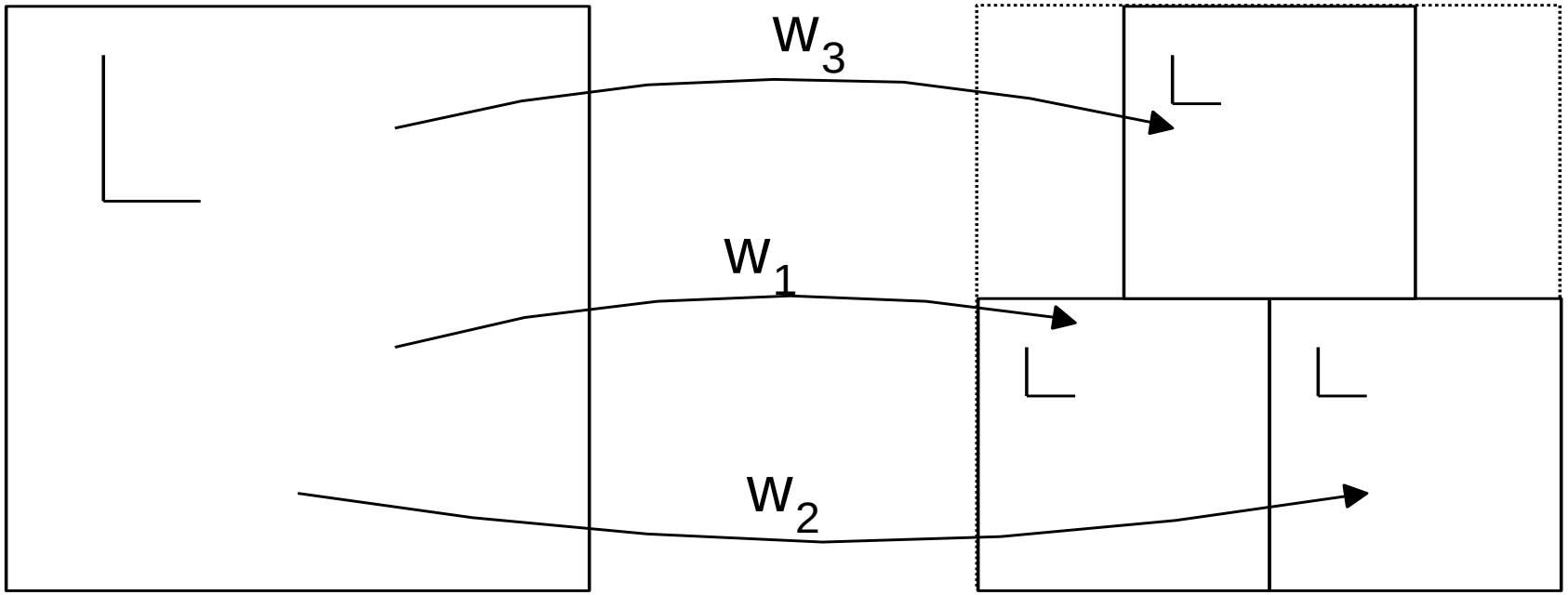
- Untuk sembarang citra awal A, dihasilkan salinan *affine*, $w_1(A)$, $w_2(A)$, …, $w_n(A)$.

- Gabungan dari seluruh salinan tersebut adalah $W(A)$, yang merupakan keluaran dari mesin,

$$W(A) = w_1(A) + w_2(A) + \ldots + w_n(A)$$

- Tansformasi *affine* yang menghasilkan citra segitiga Sierpinski:

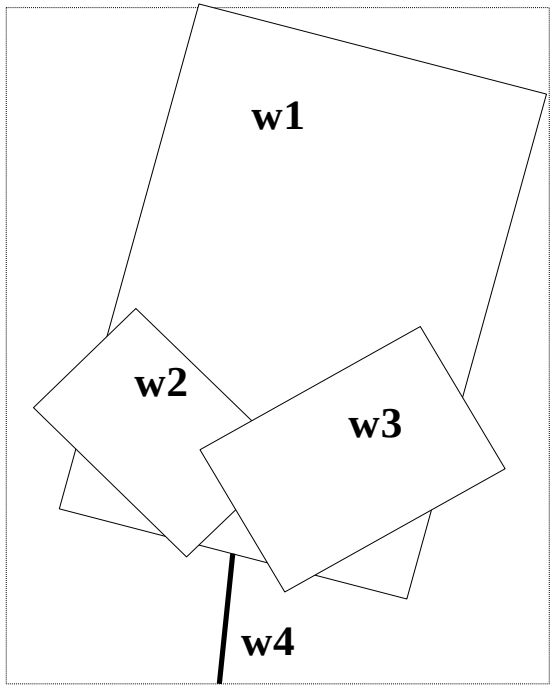
$$w_1 = \begin{bmatrix} 0.5 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0.5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 \end{bmatrix} \quad w_3 = \begin{bmatrix} 0.5 & 0.0 & 0.25 \\ 0.0 & 0.5 & 0.5 \end{bmatrix}$$



- Tansformasi *affine* yang menghasilkan citra daun pakis:

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

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



- 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 x 6 x 4 byte = 96 byte untuk menyimpan keempat transformasi itu.

- Bandingkan bila citra pakis Barnsley disimpanxdengan representasi pixel hitam putih (1 pixel = 1 byte) berukuran 550 x 480 membutuhkan memori sebesar 264.000 byte. Maka, nisbah pemampatan citra pakis adalah 264.000 : 96 = 2750 : 1, suatu nisbah yang sangat tinggi.

Partitioned Iterated Function System (PIFS)

Penemu: Arnaud D. Jacquin (1992), mahasiswa bimbingan Michael Barnsley

Dasar pemikiran:

- 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.

- Setiap transformasi itu dari bagian citra ke bagian citra lain yang mirip dapat direpresentasikan dengan transformasi *IFS* lokal atau *Partitioned Iterated Function System* (PIFS)

Kemiripan lokal pada citra Lena

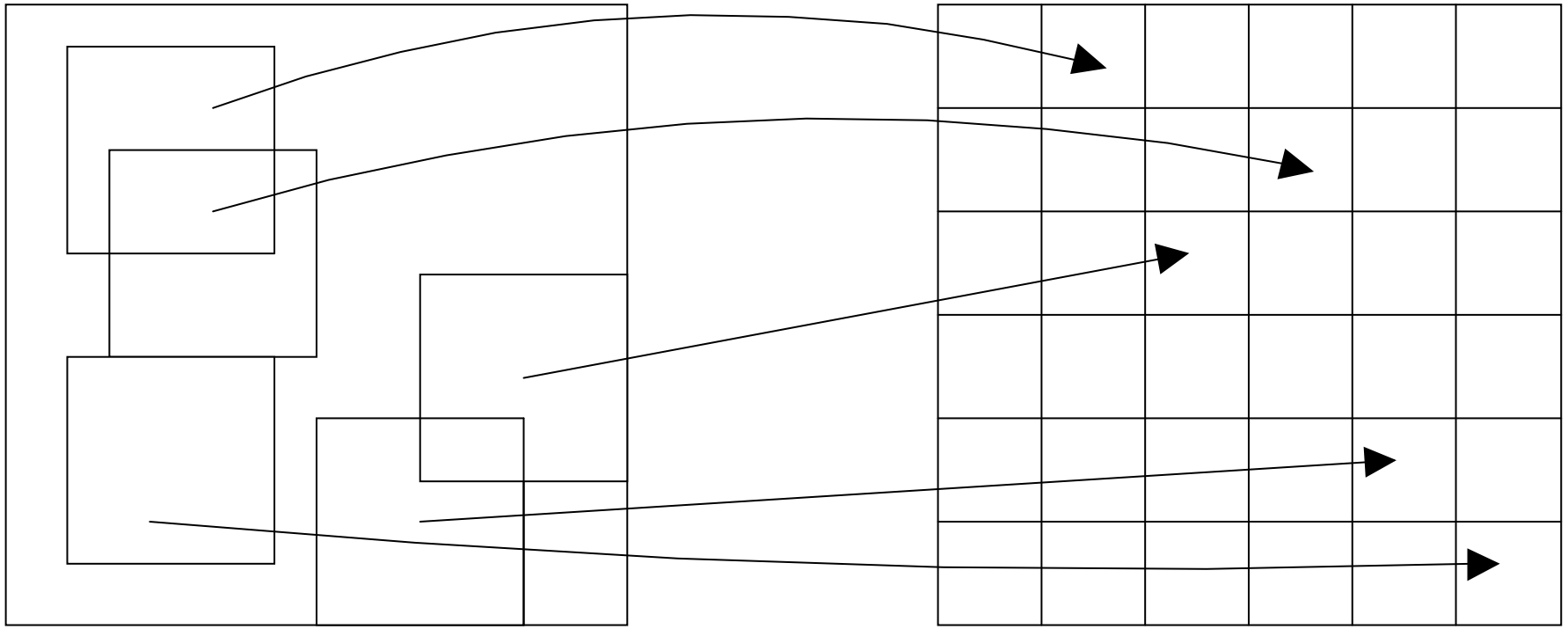
Algoritma:

1. Bagi citra atas sejumlah blok yang berukuran sama dan tidak saling beririsan, yang disebut blok jelajah (*range*).

2. Untuk setiap blok jelajah, cari bagian citra yang berukuran lebih besar dari blok jelajah –yang disebut blok ranah (*domain*)- dan paling mirip (cocok) dengan blok jelajah tersebut.

3. Turunkan transformasi *affine* (*IFS* lokal) w_i yang memetakan blok ranah ke blok jelajah.

4. Hasil dari semua pemasangan ini adalah *Partitioned Iterated Function System* (*PIFS*).



Blok ranah

Blok jelajah

Pemetaan dari blok ranah ke blok jelajah

- Kemiripan antara dua buah (blok) citra diukur dengan metrik jarak. Metrik jarak yang digunakan misalnya *rms (root mean square*):

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

z dan *z*' adalah nilai *pixel* dari dua buah blok, dan *n* = jumlah *pixel* di dalam citra

- Ukuran blok ranah diambil dua kali blok jelajah.
- Contoh: Untuk blok jelajah 8×8 *pixel* dan blok ranah berukuran 16×16 *pixel*, citra 256×256 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.

- Transformasi affine di dalam PIFS memiliki komponen z selain komponen koordinat (x,y):

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

  - s_i menyatakan faktor kontras *pixel* (nilanya antara 0 dan 1)
  - o_i menyatakan ofset kecerahan (*brightness*) *pixel*
  - $z' = s_i z + o_i$

- Dengan asumsi ukuran blok ranah = dua kali ukuran blok jelaja (2:1), maka transformasi *affine* menjadi lebih sederhana, yaitu:

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

- 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:

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

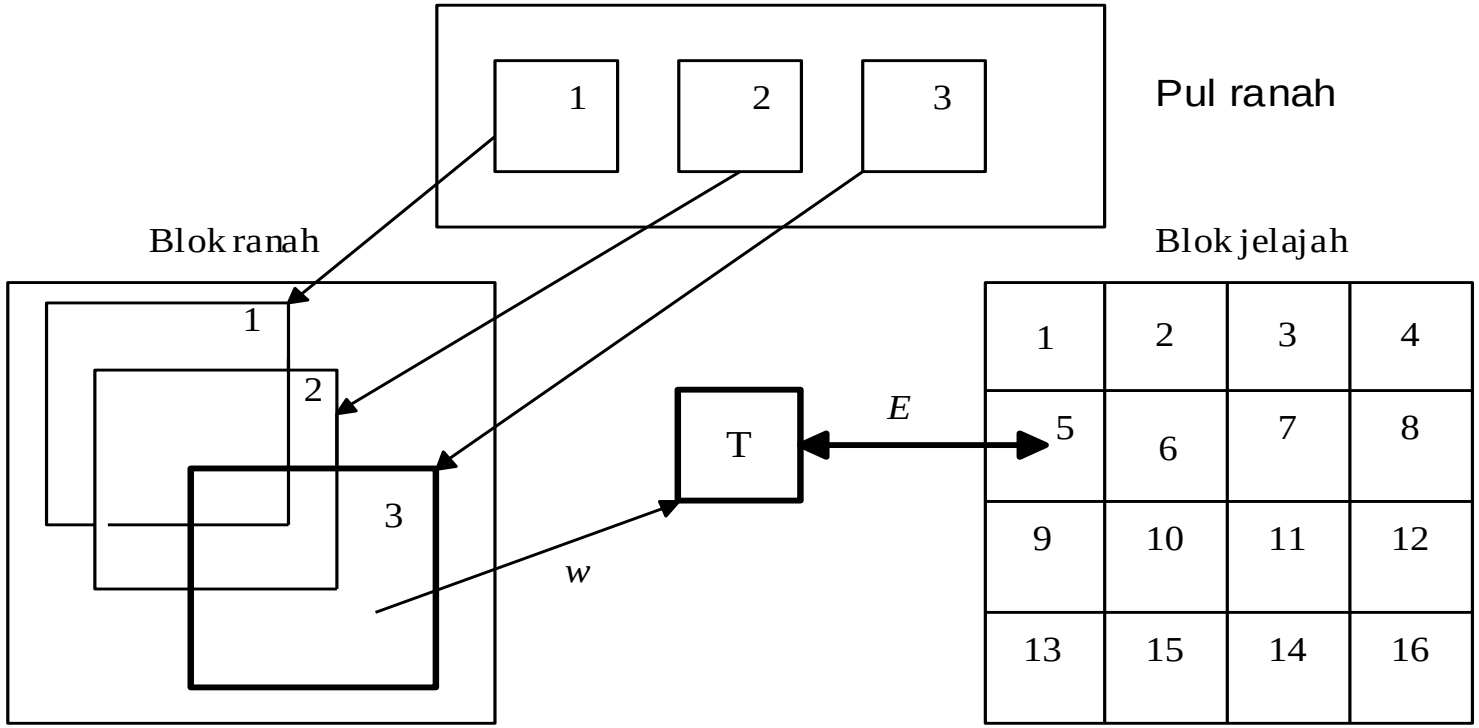
- Dengan nilai *s* dan *o* yang telah diperoleh, maka kuadrat galat antara blok jelajah dan blok ranah adalah

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

- Lalu hitung *rms* sebagai berikut:

$$d_{rms} = \sqrt{E}\ /n$$

- Transformasi *affine* w_i diuji terhadap blok ranah D_i menghasilkan blok uji $T_i = w_i(D_i)$.
- Jarak antara T dan R_i dihitung dengan persamaan d_{rms}.
- Transformasi *affine* yang terbaik ialah w yang meminimumkan d_{rms}.



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.

- 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.

- 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.

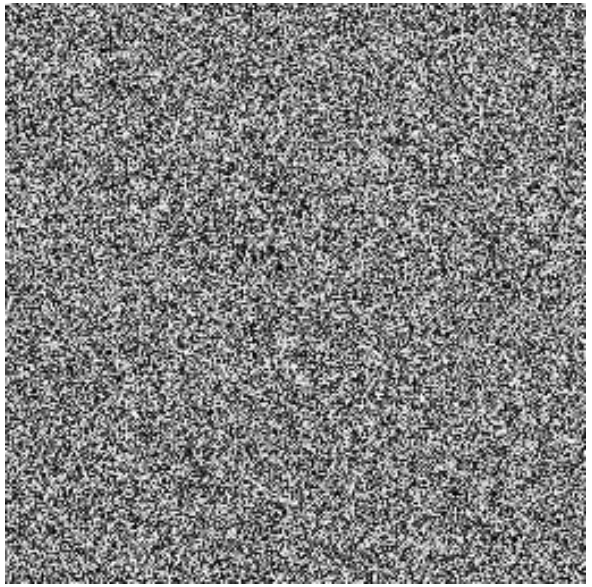
Rekonstruksi Citra (penirmpatan)

- Rekonstruksi (dekompresi) citra dilakukan dengan melelarkan *PIFS* dari citra awal sembarang.

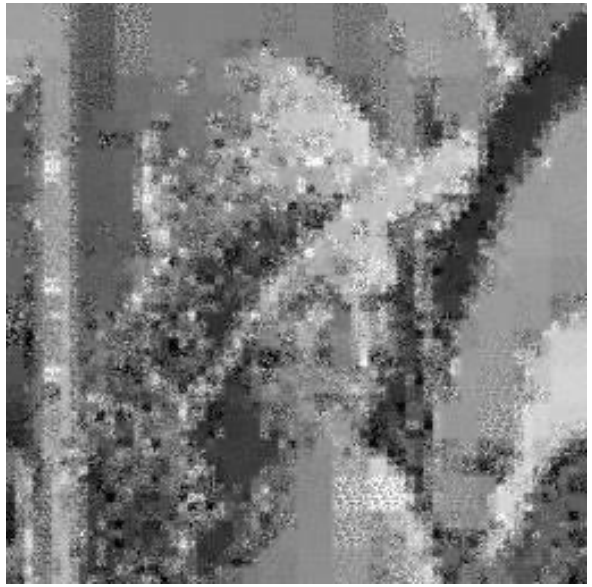
- Karena setiap *PIFS* lokal kontraktif, baik kontraktif dalam matra intensitas maupun kontraktif dalam matra spasial maka lelarannya akan konvergen ke citra titik-tetap *PIFS*.

- Kontraktif intensitas penting untuk menjamin konvergensi ke citra semula, sedangkan kontaktif spasial berguna untuk membuat rincian pada citra untuk setiap skala.

- Konvergensi ke citra titik-tetap berlangsung cepat. Konvergensi umumnya dapat diperoleh dalam 8 sampai 10 kali lelaran



Citra awal



Citra lelaran ke-1

Citra lelaran ke-2

Citra lelaran ke-6

- Beberapa hasil pemampatan fraktal:

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 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 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