Kompresi *Fractal* dengan Metode Partisi *Adaptive Quadtree*

Torkis Nasution
Jurusan Teknik Informatika, STMIK-AMIK Riau
torkisnasution@stmik-amik-riau.ac.id

Abstrak

Kompresi fractal merupakan teknik yang tergolong baru dalam sejarah kompresi citra. Dengan memanfa-atkan kesamaan bentuk dengan bagian yang lebih kecil pada citra, metode kompresi fractal diperkenalkan. Dalam kompresi fractal, bagian kecil yang memiliki suatu kemiripan dengan bagian yang lebih besar pada citra akan dilakukan transformasi scan berulang-ulang sehingga membentuk bagian besar tersebut. Proses ini disebut juga dengan iterated function system. Penelitian ini membahas mengenai pengompresan citra hitam putih menggunakan metode partisi adaptive quadtree. Maksud dari metode ini adalah compresi fractal yang skema partisinya menggunakan skema partisi quadtree dengan threshold yang berubah-ubah (adaptive threshold). Penelitian ini merancang skema partisi quadtree menggunakan adaptive threshold untuk lebih mengoptimalkan hasil suatu kompresi citra baik dari segi kualitas, rasio, dan waktu pengompresan. Pada skema partisi standar. suatu citra dipartisi menjadi bagian-bagian berbentuk persegi dengan ukuran yang sama. Partisi jenis ini memiliki kelemahan utama yaitu terpartisinya bagian citra yang sebenarnya memiliki tingkat kompleksitas rendah. Hal ini sangat tidak efisien dikarenakan pemborosan bit yang terjadi untuk merekam koefisien dari bagian-bagian tersebut untuk pengompresan sehingga menyebabkan ukuran file menjadi besar. Skema partisi quadtree memungkinkan untuk mempartisi citra menjadi bagian-bagian yang memiliki ukuran yang berbeda sesuai dengan tingkat kompleksitas bagian tersebut berdasarkan threshold yang sudah ditetapkan sebelumnya.

Kata kunci : citra, kompresi fractal, adaptive quadtree

Abstract

*Fractal compression is categorized as a new technique in history of image compression. By using the similar shapes in smaller portion of image, this fractal compression is introduced. In fractal compression, the smaller part will have similarity with bigger part in image, tranformation scan will be carried out repetitively so that it creates the bigger part. This p*rocess is called *iterated function system.The research discusses white and black image compression by using adaptive quadtree partition method. It means that fractal compression which its partition scheme uses quadtree partition scheme with adaptive threshold. The intention is to optimize the result of image compression in its quality, ratio and time. In standard partition scheme, an image is parted into similar size of squares. The main weakness of this kind of partition is the parted of low complexity of the real image. It is inefficient because of its wasting bits in recording the coefficient the parts for compression that the file size will be bigger. Quadtree partition scheme enables image partition in different sizes as per its complexity based on the determined threshold.*

Keywords : image, fractal compression, adaptive quadtree

1. Pendahuluan

Kualitas dari sebuah citra tergantung dari karakte-ristik citra (ukuran, warna resolusi) serta cara mem-proses citra tersebut. Semakin baik kualitas citra yang diproses maka semakin besar juga ukuran file dari citra tersebut. Untuk itu dibuat berbagai jenis metode kompresi citra seperti JPEG, GIF, dan PNG. Tetapi akibat-nya terjadi beberapa penurunan kualitas pada citra yang disebabkan hilangnya informasi pada citra saat diproses. Oleh karena itu dibutuhkan suatu metode baru untuk mengecilkan ukuran file citra dengan rasio tinggi tetapi dapat meminimalisasi penurunan kualitas tersebut. Salah satu metode kompresi yang dapat mem-enuhi kriteria ini adalah kompresi *fractal.*

Kompresi *fractal* merupakan suatu metode kom-presi yang sangat potensial pada rasio tinggi. Konsep dari kompresi *fractal* adalah merubah suatu citra asli menjadi *koefisien fractal* dan menghasilkan citra kembali dengan cara melakukan proses dekompresi koefisien *fracta* tersebut. Dengan hanya menyimpan beberapa koefisien *transformasi affine,*maka secara otomatis ukuran data citra akan lebih kecil dibanding menyimpan keseluruhan citra.

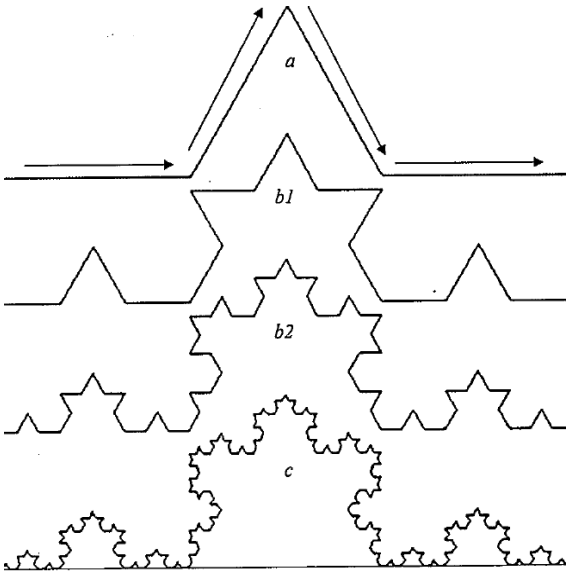
Pada skripsi ini, dikembangkan suatu metode yang mengkombinasikan kompresi *fractal* dengan metode partisi *adaptive quadtree* untuk menghasilkan suatu bentuk kompresi citra yang memiliki rasio kompresi yang tinggi dan dapat meminimalisasi penu-runan kualitas. Peluang bagi *kompresi fractal* untuk berkembang di masa depan amatlah luas, terutama di-karenakan kompresi *fractal* dapat dikombinasikan dengan banyak metode partisi dan algoritma kompresi.

Dalam kompresi *fractal* terdapat beberapa faktor yang mempengaruhi hasil citra yang dikompresi baik dalam segi waktu, rasio dan kualitas. Diantaranya adalah karakteristik citra yang dipakai, metode partisi dan tahapan-tahapan yang diimplementasikan dalam proses pengompresan ataupun dekompresi. Oleh karena itu, penulisan skripsi ini memfokuskan penelitian terhadap hal sebagai berikut :

a. Perancangan suatu metode dalam kompresi *fractal* dengan metode partisi *quadtree* yang menggunakan *adaptive threshold.*
b. Citrayang dipakai menggunakan mode *grayscale*.
c. Karakteristik citra yang dipergunakan dalam pro-ses dekompresi adalah koefisien citra global hasil proses *transformasi-affine* dengan menggu-nakan fungsi rotasi dan translasi.

Pemfokusan penelitian terhadap tahapan metode partisi dikarenakan tahap metode partisi merupakan salah satu tahapan kompresi *fractal* yang memiliki peran sangat penting dalam penentuan kualitas, rasio dan waktu pengompresan, sedangkan alasan pemilihan citra *grayscale* dikarenakan tingkat kerumitan dan karak-terisitik citra *grayscale* lebih sederhana dibanding de-ngan citra berwarna sedangkan penelitian me-ngenai kompresi *fractal* saat ini masih tergolong sangat baru dan sumber-sumber yang mendukung masih sangat sedikit.

2. Fractal

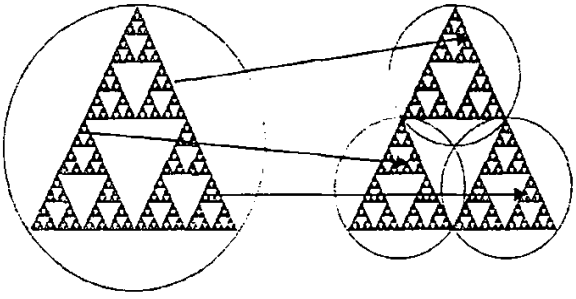


Gambar 1. Van Koch Snowflake

Fractal adalah objek geometri yang bagian-bagiannya mempunyai persamaan bentuk yang mewakili bentuk dasar objek itu sendiri dalam

segala skala. Bentuk dari *fractal* adalah *irregular* (tidak teratur) dan kompleksitas *fractal* tidak pemah berubah. *Fractal* memiliki dua ciri khas, yaitu *self-similarity* dan *infinite detail*. *Self-similarity* berarti setiap bagian *darifractal* memiliki bentuk dasar yang sama walau dilihat menggunakan skala apapun. Sedangkan *infinite detail* berarti setiap *fractal* memiliki bentuk dasar yang seakan-akan tidak habis-habis apabila diperhatikan. Berbagai bentuk *fractal* yang terkenal antara lain *Sierpinsky Triangle* dan *Von Koch Snowflake*

Pada gambar 1 dapat dilihat bahwa *fractal Von Koch Snowflake* mempunyai tiga buah objek penting yaitu *initiator, generator* dan *attractor.* Bagian ber-label *a* disebut juga sebagai *initiator*. *Initiator* dapat dianggap sebagai citra asli. Fungsi dari *initiator* adalah sebagai bahan dasar untuk membuat *fractal*. Tanpa *initiator* suatu *fractal* tidak akan dapat terbentuk. Berdasarkan definisi diatas, *initiator* dapat dianggap sebagai bagian terkecil dari sebuah *fractal*. Bagian berlabel *bl* dan *b2* biasa disebut juga sebagai *generator*. *Generator* berasal dari hasil transformasi pola yang ada pada *initiator* ke dalam bentuk dasar *initiator* itu sendiri. Pada gambar 1, terlihat jelas bagian *bl* terbentuk dari hasil proses transformasi bagian *a* ke dalam bagian *a* itu sendiri sedangkan bagian *b2* terbentuk dari hasil proses transformasi bagian *bl* ke bagian *bl* itu sendiri. Berdasarkan penjelasan diatas, *generator* dapat didefi-nisikan sebagai hasil dari trans-formasi *initiator* ke dalam dirinya sendiri. Bagian terakhir adalah bagian *c*. bagian *c* dapat disebut juga sebagai *attractor*. *Attrac-tor* merupakan basil akhir dari transformasi yang dilakukan *generator*. Oleh karena itu *attractor* sering disebut juga sebagai *fractal* itu sendiri.



Gambar 2. Sierpinsky *Triangle*

2.1. Dimensi *Fractal*

Dimensi fractal adalah suatu bilangan real yang menunjukkan derajat ketidakteraturan bidang *fractal* tersebut. Rumus dimensi *fractal* adalah :

$$D = \frac{Log\ N}{Log(1/R)}$$

Dimana *D* adalah dimensi *fractal*, *N* adalah jumlah segmen garis pada *generator,* dan *R* adalah panjang segmen garis pada *generator* dibagi jarak titik awal dan akhir *generator*

2.1.1. *Iterated Function System*

IFS adalah suatu fungsi iterasi yang terdiri dari sekumpulan *transformasi-affine* yang dipilih sedemikian rupa hingga gabungan dari *transformasi-affine* tersebut mendekati citra target. Sifat dari IFS menyerupai sifat mesin fotokopi, yaitu menyalin kembali suatu bentuk ke dalam posisi dan ukuran yang diinginkan. Prinsip terpenting dalam IFS adalah pemetaan kontraktif *(contractive mapping)* dimana setelah semua *transfor-masi-affine* ditentukan, IFS dapat diterapkan dengan mengkodekan semua koefisien transformasi. Pemetaan suatu fungsi ke dirinya sendiri dikatakan kontraktif bila selisih nilai hasil fungsi pada setiap iterasi semakin kecil. Teorema pemetaan kontraktif menyatakan bahwa setiap IFS memiliki suatu *unique fixed point* pada pemetaan fungsi $f(x) = x$ dimana nilai itu merupakan nilai dari *attractor* sebuah IFS. Rumus dari *unique fixed point* adalah sebagai berikut :

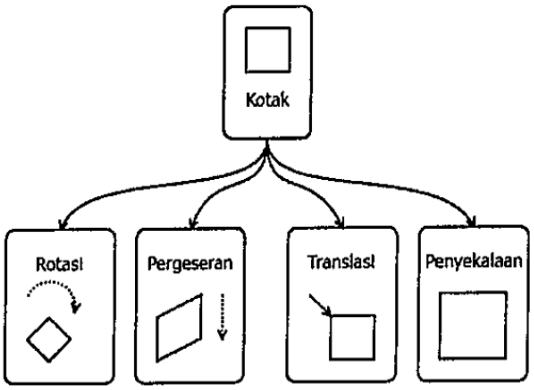
$$A = U_{n-1}^{N} wn(A)$$

Dimana :
A : Attractor
N : Jumlah iterasi
W : Transformasi Affine

2.1.2. *Transformasi-Affine*

Transformasi-affine adalah transformasi linier

yang terdiri dari empat operasi dasar yaitu rotasi, pergeseran, translasi, dan penyekalaan. Transformasi dinamakan *affine* karena terdapat afinitas (hubungan) antara visual dan struktural antara bentuk lama dan bentuk yang baru.



Gambar 3. Bentuk *transformasi-affine* dari segi empat

Transformasi titik-titik dalam bidang dapat dinyatakan dengan suatu perkalian matriks transformasi *T* dengan vektor *v* yang mewakili koordinat titik tersebut. Koordinat baru hasil transformasi *(x2.Y2)* didapatkan dengan melakukan operasi perkalian antara *T* dan *v*.

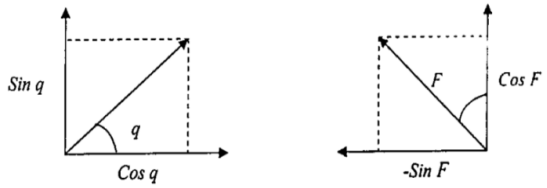
$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix};\ v = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} x \begin{bmatrix} x \\ y \end{bmatrix}$$

Beberapa bentuk transformasi dasar yaitu :

a. Rotasi, bentuk matrik transformasinya adalah sebagai berikut

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos q & -\sin F \\ \sin q & \cos F \end{bmatrix} x \begin{bmatrix} x \\ y \end{bmatrix}$$



Gam bar 4. Sin q dan Cos F

Dimana *q* adalah besar sudut rotasi terhadap sumbu *x* dengan arah berlawanan jarum jam, sedang *F* adalah besar sudut rotasi terhadap sumbu *y* dengan arah berlawananjarumjam.

b. Pergeseran *(shearing)*, bentuk dari transfom1asi ini terlihat seperti "menarik" sebagian sisi objek geometri ke sebuah arah yang parallel dengan koordinat sisi satunya. Bentuk matriks shearing adalah sebagai berikut.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} x \begin{bmatrix} x \\ y \end{bmatrix}$$

c. Translasi *(translation)*, titik-titik sebuah objek dapat dipindahkan dalam arah horisontal sebesar *tx* satuan dan dalam arab *vertikal* sebesar *ty* satuan dengan operasi penjumlahan sebagai berikut.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x + tx \\ y + tx \end{bmatrix}$$

d. Penyekalaan *(scaling)*, penyekalaan diatur oleh clcmcn diagonal matrik, yaitu nilai *sx* dan *sy*. Bentuk matrik transformasinya adalah sebagai berikut :

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix}$$

Dari empat operasi transformasi diatas. yang dipakai dalam IFS hanya tiga operasi, yaitu penyekalaan, rotasi dan translasi. Bentuk dari *transformasi- affine* dalam IFS dinyatakan dengan :

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

$$\begin{aligned} x_2 &= ax + by + tx \\ y_2 &= cx + dy + ty \end{aligned}$$

Dimana *a, b, c, d* mewakili operasi penyekalaan dan rotasi, sedangkan *tx* dan *ty* mewakili operasi translasi.

2.2 Teorema *Collage*

Proses pada IFS adalah proses untuk meng-generate suatu *fractal* dengan cara melakukan

fungsi iterasi secara kontraktif dengan koefisien *affine* ke dalam dirinya sendiri. Proses IFS dapat dianggap sebagai proses pembentukan *attractor* dari *initiator*. Lalu muncul sebuah pertanyaan baru, "Apakah mungkin proses itu dibalik, bisakah *attractor* dikembalikan menjadi sebuah *ini-tiator?"*. Masalah ini dikenal dengan *inverse problem*, bagaimana dari sebuah *fractal* kita dapat melacak *ini-tiator* yang mirip dengan *fractal* tersebut. Dan muncullah teorema *collage* sebagai solusinya. Teorema *collage* menyatakan bahwa jika *W(J)* mendekati maka *unique fixed point W(l) = W(W(W..Jf'(l)...))* juga mendekati **I.** Gambar W(l) tersusun dari semua gabu-ngan dari *W(J(l))*.

2.3. Kompresi Citra *Fractal*

Kompresi citra *fractal* pertama kali dikenalkan oleh Michael F. Bamsley pada awal tahun 1980. Lalu dikembangkan oleh Arnaud Jacquin yang akhirnya menciptakan kompresi citra *fractal* pertama, yaitu *Partitioned Iterated Function Syslem* (PIFS). Karena menggunakan karakteristik *fractal* dalam metode pengompresannya maka teknik kompresi ini sering disebut pula sebagai kompresi citra *fractal (fractal image compression)* atau dapat disingkat menjadi *kompresifractal.*

Konsep dari kompresi *fractal* adalah merubah suatu citra asli menjadi koefisien *fractal* menggunakan teorema *collage* dan menghasilkan citra dekompresi dengan cara melakukan proses IFS terhadap ko-efisien *fractal* tersebut (Bamsley, 1993). Dengan hanya menyimpan.beberapa *koesienfractal* hasil dari *trans-formasi-affine,* maka secara otomatis ukuran data citra akan lebih kecil dibanding menyimpan keseluruhan citra.

Bamsley mengemukakan bahwa sebuah *fractal* merupakan salinan dari setiap *initiator*-nya sendiri. Pada kenyataannya suatu citra tidak benar-benar me-miliki sebuah bagian kecil *(initiator)* yang sama persis dengan dirinya seperti pada *fractal.* Tetapi kemung-kinan bahwa adanya kemiripan bagian kecil dari suatu citra dengan bagian yang lainnya selalu ada. Ini berarti bahwa kumpulan *initiator* yang memiliki sifat *self-similarity* tidak dapat membentuk keseluruhan citra asli, namun dapat membentuk bagian tertentu dari citra asli.

Gambar 5 Self-similarity dari sebuah citra

Pada gambar 5 dapat ditemukan bahwa bagian topi dari citra tersebut sama dengan bayangan topi yang terpantul di kaca. Ini menunjukkan bahwa se-benarnya kemiripan *(self-similarity)* dalam sebuah citra sebenarnya ada tetapi tidak disadari. Teknik mem-partisi dan menemukan bagian-bagian citra terbaik yang saling terkoneksi tidaklah mudah. Sampai saat ini, banyak teknik partisi citra yang telah dikembang-kan. Diantaranya adalah metode partisi *quadtree* dan *horizontal-vertical.*

3. Tahapan Kompresi Citra *Fractal*

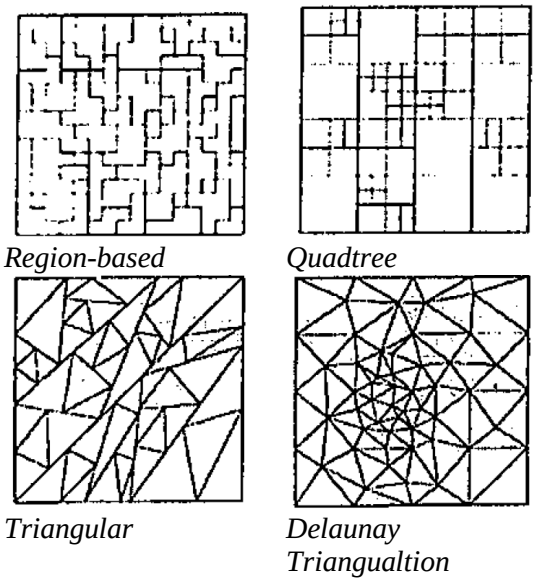
Gambaran umum tahapan proses pada kompresi citra *fractal* adalah sebagai berikut:

a. Skema Partisi

Skema partisi adalah salah satu tahapan yang sangat penting. Pemilihan metode skema partisi sa-ngat mempengaruhi kualitas, rasio dan waktu pengompresan citra. Pada tahap ini, suatu citra dipartisi ke dalam kumpulan range dan domain. Bentuk partisi dibagi menjadi dua jenis, yaitu:

1. *Right angled partition scheme* dimana bentuk partisinya sesuai dengan bentuk asli piksel dan membentuk sudut yang berpotongan secara tegak lurus. Contoh metode partisi ini adalah fixed *block, region-based* dan *quadtree.*

2. *Not-right angled partition scheme* dimana bentuk partisinya tidak sesuai dengan bentuk asli piksel dan membentuk sudut yang berpotongan diagonal. Contoh metode partisi-nya adalah *triangular* dan *delaunay triangulation.*



Gambar 6. Bentuk partisi

b. Seleksi Domain
Penentuan panjang dan lebar dari domain mempengaruhi kualitas dan waktu kompresi. Semakin kecil ukuran domain menyebabkan kua-litas citra terkomprcsi makin baik namun waktu kom-presi akan semakin lama dikarenakan banyaknya domain yang diproses, begitu juga sebaliknya.

c. Transformasi Blok
Transformasi blok adalah proses penyekalaan ukuran domain sehingga sama dengan ukuran range. Jenis proses transformasi blok sangat dipengaruhi oleh skema partisi yang dilakukan. Untuk skema partisi *right-angled,* proses 'pengecilan' domain dilakukan dengan cara mengalikan domain dengan faktor skala pengali. Untuk mendapatkan nilai piksel maka dila-kukan teknik *averaging* pada piksel tetangga. Se-dangkan pada skema partisi *not-right angled* proses transformasi blok lebih rumit karena diperlukan konversi bentuk bagian dari bentuk piksel ke dalam bentuk partisi poligon yang diinginkan.

d. *Encoding*, pada tahap ini dilakukan proses penyim-panan dan pengalokasian koefisien *transformasi-affine,* lokasi domain dan lokasi range menjadi bentuk bit.

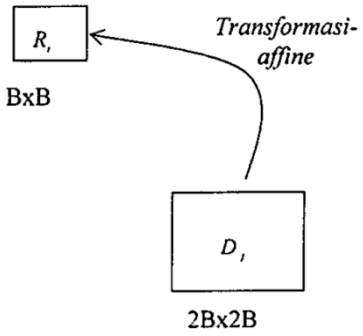
e. *Decodin*g, pada tahap decoding, proses IFS dila-kukan terhadap koefisien *fractal* yang disimpan sehingga membentuk suatu citra hasil kompresi.

4. Range, Domain, dan *Domain Pool*

Pada awalnya sebuah citra dipartisi menjadi 2 bagian, yaitu range dan domain. Range adalah hasil partisi dari suatu citra dimana setiap bagiannya tidak boleh saling menimpa *(non-overlapped block). Dai-amfractal,* range disebut juga *initiator* yang diperlukan untuk membentuk sebuah *attractor.* Range pada umum-nya berbentuk bujur sangkar, namun sesungguhnya tidak ada keharusan mengenai bentuk dan ukuran range. Pemilihan bentuk serta ukuran range yang tepat dapat menghasilkan rasio kompresi yang cukup tinggi, kualitas citra yang masih dapat dipertahankan dan waktu kompresi yang cukup rendah.

Bila ukuran range terlalu besar, maka citra basil proses kompresi akan mengalami penurunan kualitas yang besar, namun rl!sio kompresi yang dicapai akan tinggi. Demikian sebaliknya jika ukuran range terlalu kecil, maka rasio kompresi akan rendah namun kualitas citra semakin baik. Tidak seperti range, domain adalah basil dari partisi citra yang boleh saling menimpa antara satu dengan yang lainnya *(orerlapped block;.* Domain merupakan bagian dari citra yang mirip dengan range. Domain juga sering disebut dengan *codebook.* Ukuran dari domain harus lebih besar atau sama dengan range. Aturannya adalah O<s<l dimana s adalah faktor skala pengali. Faktor skala pengali *(s)* dari domain ke range pada umumnya dibagi menjadi empat ukuran, yaitu $s \in S = \{1/4, 1/2, 3/4, 1\}$. Domain dikatakan 2 kali lebih besar dari range jika $s = 1/2$, domain dikatakan 4 kali lebih besar dari range jika $s = 1/4$, dan seterusnya. Proses

kompresi dilakukan dengan menemukan pemetaan *transformasi-ajjine* terbaik dari domain ke range.



Gambar 7. Transformasi-affine memetakan domain ke range

Karena domain boleh saling bertumpuk, maka jumlah domain pada citra dipengaruhi oleh *step horizontal()* dan v*ertical()* yang ditentukan rum us per-hitungan range dan domain pada sebuah citra adalah :

a. Range, suatu citra berukuran *MxM* piksel dengan ukuran range *BxB* piksel akan memiliki range se-banyak :

$$\left(\frac{M}{B}\right) \times \left(\frac{M}{B}\right)$$

b. Domain, jika kita tentukan *step* domain sebe-sar *6* piksel, maka total domain yang ada pada citra tersebut dapat diperoleh melalui rumus berikut :

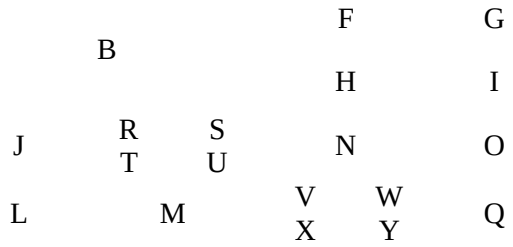
$$\left(\frac{M-2B}{\partial}+1\right) \times \left(\frac{M-2B}{\partial}+1\right)$$

Berikut adalah contoh perhitungan range dan domain. Suatu citra *grayscale* berukuran 256x256 piksel dengan ukuran range adalah 8x8 piksel. *Step horizontal (oh)* dan *step vertical (ov)* sebesar 4 piksel. Range baik vertikal maupun horisontal berjumlah 32 buah. Jumlah range total adalah 32 x 32 = 1024 range. Jumlah domain blok adalah 61 x 61 = 3721 domain blok. Dikarenakan ada 8 *transformasi-ajjine* (4 arah rotasi dan 4 arah translasi) yang dilakukan untuk mencari domain terbaik dari suatu range, maka pada setiap range dilakukan sejumlah 8 x 3721 = 29768 proses. De-ngan jumlah range sebanyak *I024* range, maka secara keseluruhan ada 29768 x I 024 = 30482432 proses. Semakin banyak proses yang dilakukan berarti sema-kin lama waktu kompresi yang dibutuhkan. Pencarian domain terbaik dengan cara membabi buta sangatlah tidak efisicn dan memakan waktu. Untuk itu dibutuhkan strategi untuk mengefisienkan pcncarian domain terbaik. yaitu dengan cara mengurangi jumlah domain yang dicari. Proses ini disebut dengan *domain pool.*

Domain pool atau *virtual codebook* adalah beberapa kumpulan domain yang dekat dengan suatu range. Landasan pemikiran *domain pool* menyatakan bahwa pada umumnya domain terbaik dari suatu range terletak tidak jauh dari range tersebut. Pada *domain pool,* pemilihan domain terbaik tidak dilakukan pada keseluruhan domain yang ada pada citra, melainkan hanya pada *domain pool* yang yang bersangkutan.

5. Skema Partisi *Quadtree*

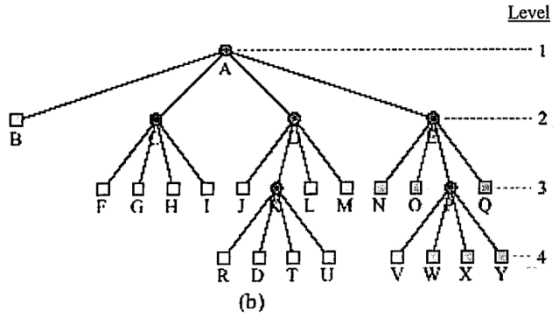
Inti metode partisi *quadtree* adalah mempartisi citra ke dalam 4 sub node berbentuk bujur sangkar yang ber-ukuran sama.



Gam bar 8. (a) Bentuk representasi citra menggunakan metode quadtree

Perhatikan gambar 8 (a). Pada awalnya citra *A* dibagi menjadi 4 sub node yaitu *B,* C, *D, E* kemudian node *C* dipartisi menjadi 4 sub node *F,G,H,l,* node *D* dibagi menjadi 4 sub node *J. K, L, M,* dan seterusnya. Hubungan

dalam metode *quadtree* juga dapat direpre-sentasikan melalui sebuah *tree.* Setiap node dapat dibagi menjadi empat sub node. Sebuah *quadtree* membagi range ke dalam beberapa level sesuai ukuran-nya.



Gambar 9. (b) Korespondensi dalam suatu quadtree

Perhatikan gam bar 9 (b). Node *A* sebagai *root* menunjukkan representasi partisi dalam bentuk *quadtree.* Setiap node yang terbagi menjadi empat sub node baru memiliki level baru yang menggambarkan tingkat kedalaman partisinya. Begitu pula dengan sub node dibawahnya. Representasi dari suatu citra meng-gunakan *quadtree* ditentukan oleh dua parameter yaitu level minimum dan level maksimum. Ukuran *quadtree* menentukan bahwa suatu citra minimal harus dipartisi sampai level minimumnya dan level partisi tidak boleh melebihi level maksimurnnya. Level minimum adalah level dimana pengkodean *quadtree* dimulai sedangkan level maksimum adalah batas maksimal pengkodean *quadtree.*

Pengkodean *quadtree* hanya dilakukan di antara level minimum dan level maksimum. Node yang mem-iliki level dibawah level minimum dan diatas level maksimum tidak akan dikodekan. Node yang terpecah memiliki nilai I, sedangkan node yang tidak terpecah memiliki nilai 0.

Perhatikan gambar 9(b), urutan pembacaan struktur *quadtree* menggunakan *breadth first search* dengan *level minimum* = 2 dan *level maksimum* = 4 adalah sebagai berikut :

A-B-C-D-E-F-G-H-l-J-K-L-M-N-0-P-Q-R-S-T-U-V-**W-X-Y**

Kode *quadtree* yang dihasilkan adalah :

I-0-l-l-l-0-0-0-0-0-I-0-0-0-0-l-0-0-0-0-0-0-0-0

Tetapi karena setelah mencapai level maksimum sudah pasti node tidak dapat dibagi lagi menjadi sub node baru, maka kita dapat mengabaikan kode di level maksimum. Jadi kode *quadtree* yang sudah dikurangi dengan level maksimumnya adalah :

l-0-1-l-l-0-0-0-0-0-l-0-0-0-0-l-0

6. *Adaptive Treshold*

Pada awalnya teknik *quadtree* menggunakan satu nilai *threshold* error tetap *(fixed threshold).* Namun teknik ini memiliki kelemahan karena satu nilai *threshold* tetap tidaklah cocok untuk diaplikasi-kan terhadap ukuran range yang berbeda-beda. Untuk itu dipergunakan *adaptive threshold* dimana setiap level *quadtree* memiliki nilai *threshold* error yang berbeda-beda sesuai dengan levelnya.

Teknik *adaptive threshold* sebelumnya telah di teliti dan dibuktikan bahwa j ika kita memakai *adaptive threshold* pada level *quadtree,* maka kualitas peng-kodean citra akan lebih baik daripada menggunakan *jixed threshold* pada bit rate yang sama (Shusterman dan Feder, 1994). Berikut adalah tahapan *adaptive threshold* dan penggunaannya dalam partisi *quadtree*

a. Untuk setiap blok range, cari koefisien fractal ter baik yang memiliki minimal error diantara *domain pool-nya.*
b. Jika error minimal lebih besar dari *initial threshold* dan level node masih dibawah maksimum level, partisi blok range menjadi empat bagian sama besar.
c. *Initial threshold (eJ)* di level pertama menjadi menjadi sub *threshold* bagi level selanjutnya. Rumus dari *adaptive threshold* adalah sebagai berikut :

$$e_i = k.e_{i-1}$$

dimana:

e_i *threshold* dari level i.

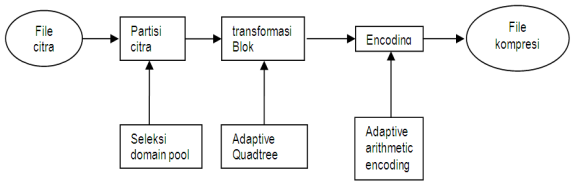
k nilai *threshold* yang pas bagi kondisi level. Berdasarkan penelitian Shusterman dan Feder, nilai *k* yang paling tepat pada *adaptive threshold* adalah *2*.

d. Ulangi tahap a dan b sampai level node memenuhi batas maksimum level.

Metode partisi ini kemudian lebih dikenal dengan metode *adaptive quadtree* dimana mengguna-kan partisi *quadtree* dengan menggunakan *adaptive threshold.*

7. Analisis Kompresi Citra

Kerangka umum dari kompresi citra mengguna-kan metode *adaptive quadtree* dapat dilihat pada gambar di bawah ini :



Gambar 10. Tahapan kompresi citra metoda quadtree

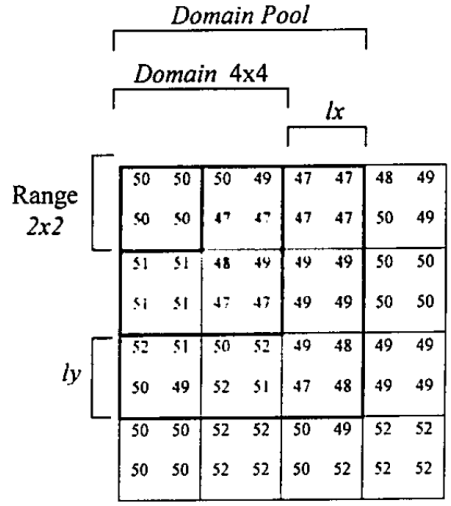
Pada gambar diatas, terdapat tiga bagian utama proses kompresi citra, yaitu :

a. Partisi citra, awalnya citra dipartisi ke dalam range-range menggunakan *adaptive quadtree.* Lalu tentu-kan parameter-parameter yang dipergunakan, yaitu range, domain, dan *domain pool.*
b. Transformasi Blok, untuk seluruh domain akan dilakukan transformasi blok menggunakan metode partisi *adaptive quadtree.* Mula-mula ukuran domain diproses sehingga sama dengan ukuran range. Se-telah itu dilakukan proses pencarian domain terbaik dari range yang ada menggunakan *transformasi-affine.*
c. *Encodin,* pada proses *encoding,* seluruh koefisien yang ada dikonver5i ke dalam *byte* yang lebih kecil menggunakan algoritma *adaptive-arithmetic coding* agar lebih menghemat kapasitas penyimpa-nan.

8. Analisis Partisi Citra

Suatu citra *T* berukuran *NxN* akan dipartisi menggunakan metode *adaptive quadtree.* Pertama-tama tentukan terlebih dahulu level minimum dan maksi-mum dari suatu citra. Setelah parameter tersebut ditentukan, partisi citra sampai batas level minimum-nya tercapai. Hasil partisi tersebut akan menjadi range dari citra. Selanjutnya akan dicari domain dan *domain pool* dari setiap range.

Bentuk dari domain dan *domain pool* tersebut adalah persegi. Pemilihan bentuk persegi ini didasarkan pada bentuk dasar dari piksel itu sendiri agar mem-permudah proses partisi. Untuk mencari domain dan *domain pool* dari sebuah range, terlebih dahulu harus ditentukan empat parameter pendukung-nya, yaitu *lx, ly, step vertical,* dan *step horizontal.* Parameter *lx* dan *ly* berguna untuk menentukan panjang dan lebar *domain pool.* Sedangkan *step vertical* dan *step hori-zontal domain pool.* Secara keseluruhan, keempat parameter ini berguna dalam menentukan jumlah domain yang ada pada *domain pool.* Contoh per-hitungan domain dan *domain pool* adalah sebagai berikut: jika terdapat suatu range berukuran B x B dan diketabui nilai *s* sebesar *Y,* maka ukuran domain adalab 2B x 2B dan didapatkan ukuran *domain pool* sebesar *(2B+lx)* x *(2B+ly).*



Gambar 11. Bentuk citra sebelum partisi dengan range 2x2 yang memiliki domain 4x4

9. Analisis Transformasi Blok

Setelah domain dan *domain pool* didapat maka akan dilakukan pencarian range-domain terbaik. Pertama-tama tentukan terlebih dabulu parameter error yang akan dipergunakan saat pemilihan domain terbaik. Ada dua jenis nilai error yang dipakai saat penghitungan, yaitu nilai error antara domain-range dan nilai *initial error* dari *threshold.*

Pemilihan pasangan range-domain dimulai de-ngan mencari domain terbaik dari *domain pool.* Proses akan dilakukan menggunakan *step horizontal* ke kanan sebanyak I piksel (sampai batas akhir *lx)* dan *step vertical* ke bawah sebanyak I piksel (sampai batas akhir *ly).* Setelah itu, proses pencarian domain terbaik dilakukan dengan cara menghitung luminansi dan menghitung error · antara range dengan setiap domain yang ada. Pasangan range-domain terbaik adalah pasa-ngan yang memiliki nilai error terkecil dari *transformasi-affine* dengan 4 arah rotasi dan 4 arah translasi diantara seluruh domain blok yang ada

0	1	2	3	12	8	4	0
4	5	6	7	13	9	5	1
8	9	10	11	14	10	6	2
12	13	14	15	15	11	7	3

15	14	13	12	3	13	14	15
11	10	9	8	2	9	10	11
7	6	5	4	1	5	6	7
3	2	1	0	0	1	2	3

(a) Rotasi 4 arah

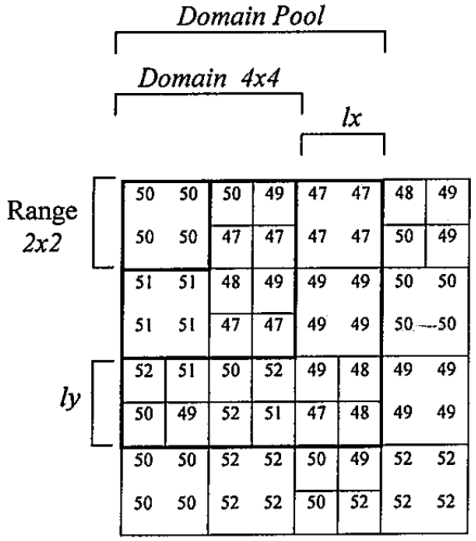
0	4	8	12	3	2	1	0
1	5	9	13	7	6	5	4
2	6	10	14	11	10	9	8
3	7	11	15	15	14	13	12

15	11	7	3	12	13	14	15
14	10	6	2	8	9	10	11
13	9	5	1	4	5	6	7
12	8	4	0	0	1	2	3

(b) Transalasi 4 arah

Gambar 12. (a) Bentuk rotasi 4 arah, (b) bentuk translasi 4 arah

Setelah dilakukan proses pencarian domain terbaik maka akan didapatkan pasangan range-domain terbaik. Sesuai dengan syarat partisi *quadtree* bahwa jika nilai *error* terkecil antara range-domain lebih besar dari *initial threshold* dan level node masih di-bawah level maksimumblok range lalu dipartisi men-jadi empat bagian sama besar.



Gambar 13. Bentuk citra sesudah partisi dengan range 2x2 yang memiliki domain 4 x 4

Setelah metode partisi *quadtree* dilakukan di seluruh bagian range maka akan didapatkan kumpulan range *(B)* dengan ukuran yang berbeda-beda. Nilai *quadtree code* lalu disimpan untuk dipergunakan saat proses dekompresi (membentuk range).

Pemetaan range-domain terbaik digabungkan dengan metode partisi *quadtree* akan menghasi!kan koefisien yang terdiri dari simetri *affine,* luminansi *(contrast* dan *brightness),* posisi domain, dan *quadtree code.*

9.1. Analisis *Encoding*

Data koefisien dari proses *quadtree* dan *transformasi-ajjine* masih cukup besar. Oleh karena itu untuk lebih menghemat kapasitas penyimpanan serta mempertinggi rasio kompresi maka sebelum disimpan, data-data terse-but harus *diencode* lagi menggunakan algoritma ter-tentu. Algoritma yang digunakan

untuk *mengencode* data *quadtree code* adalah *adaptive arithmetic coding.*Tiga konsep utama dalam *adaptive arithmetic coding* adalah interval peluang, *symbol occurrence pool* dan *window size. Symbol occurrence pool* adalah suatu wadah untuk menampung simbol-simbol yang sudah muncul dalam suatu string saat proses *encode* atau simbol-simbol hasil dari proses *decode*. *Window size* adalah banyaknY.a jumlah simbol yang dapat ditampung dalam *symbol occurrence pool. Window size* ditetapkan berdasarkan jumlah simbol pada string yang akan di *encode.* Kegunaan *window size* sangat terlihatjelas pada tahap *decoding.* Berikut ini adalah contoh proses *encvJmg* mcnggunakan penghi-tungan *adaptive arithmetic coding.*

Suatu string *bccb* dimana setiap symbolnya meru-pakan anggota dari himpunan *{a,b,c}* akan di *encode* menggunakan teknik *adaptive arithmetic coding,* maka langkah langkah yang dilakukan adalah sebagai berikut

a. Inisialisasi *symbol occurrence pool* sebagai wa-dah kosong ***O*** = {} dan inisialisasi *window size* sebesar 4 (sesuai jumlah simbol yang akan di *encode)* setelah itu peluang kemunculan setlap simbol anggota himpunan *{a,b,c}* akan dihitung menggunakan rumus berikut :

$$p(a) = \frac{N(a) + 1}{N(a) + N(b) + N(c) + 3}$$

$$p(b) = \frac{N(b) + 1}{N(a) + N(b) + N(c) + 3}$$

$$p(c) = \frac{N(c) + 1}{N(a) + N(b) + N(c) + 3}$$

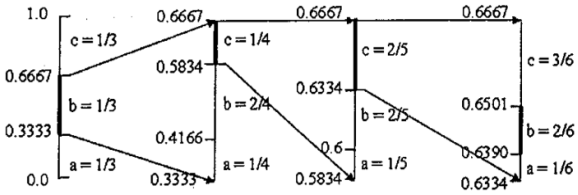
Dimana *p* melambangkan peluang kemunculan, *N* melambangkan jumlah symbol tertentu yang ada dalam *symbol occurrence pool.* Penambahan angka pada pembilang dan angka 3 pada penyebut berdasarkan rumus dasar peluang, besamya pen-ambahan pada penyebut didasarkan pada jum-lah simbol pada himpunan.

Ketika *meng-encode* simbol *b* (simbol pertama dalam string *bccb),* karena *symbol occurrence pool* (0) masih kosong maka *N(a) = N(b) = N(c)* = 0, sehingga didapatkan *p(a) = p(b)= p(c)= 113.* Interval peluang awal (0,1) akan terbagi menjadi tiga bagian sesuai dengan besar peluang kemu-nculan masing-masing simbol. Interval peluang baru dibentuk berdasarkan peluang kemunculan simbol yang di *encode* terhadap interval peluang sebelumnya. Pada tahap ini interval peluang baru yang terbentuk berdasarkan peluang kemunculan simbol *b* adalah *(low*= 0.3333. *high*= 0.6667). Saat ini, *symbol occurrence pool* (O) akan berisi simbol *b* dan dinotasikan sebagai berikut : *O={b}.*

b. Ketika *meng-encode* simbol *c* (simbol kedua dalam string *bccb), p(a), p(b)* dan *p{c)* akan dihitung ulang. Karena saat ini pada *symbol occurrence pool* berisi satu simbol *b* maka *N(b)* =I sedangkan nilai *N(a) = N(c)* = 0 setelah itu didapatkan *p(a) = p(c)* = 114 dan *p(b}* = 2/4. Selanjutnya penghitu-ngan internal peluang baru dilakukan berdasarkan peluang simbol *c.* Kemudian simbol *c* dimasuk-kan ke dalam *symbol occurrence pool.* Saat ini *symbol occurrence pool (0)* berisi 2 simbol yaitu *b* dan *c:* ***O*** = *{b,c}* sedangkan inter-val peluang baru yang terbentuk adalah *low* = 0.5834, *high* = 0.6667.

c. Simbol *c* (simbol ketiga dalam string *bccb)* di *encode. p(a), p(b) dan p(c)* akan dihitung ulang. Karena saat ini pada *symbol occurrence pool* berisi dua simbol, yaitu *b* dan *c* maka *N(a)* =0 sedangkan nilai *N(b) = N(c)* = I setelah itu akan didapatkan *p(a)* = 1/5, *p(b) = p(c)* = 2/5. Selanjutnya peng-hitungan interval peluang baru dilakukan berdasarkan peluang simbol *c* kemudian simbol *c* dimasukkan ke dalam *symbol occurrence pool.* Saat ini *symbol occurrence pool* berisi tiga simbolyaitu satu simbol *b* dan dua simbol *c :* ***O***= *{b,c,c)* sedangkan interval peluang baru yang terbentuk adalah *low* =0.6334, *high* = 0.6667.

d. Simbol *b* (simbol keempat dalam string *bccb)* di *encode. p(a), p(b) danp(c)* akan dihitung ulang. Karena saat ini pada *symbol occurrence pool* berisi tiga simbol, yaitu satu simbol *b* dan dua symbol *c maka N(a)* =0, nilai *N(b)* = 1, *N(c)* = 2 setelah itu akan didapatkan *p(a)* = *116, p(b)* = 2/6, *p(c)* = 3/6. Selanjutnya penghitungan interval peluang baru dilakukan berdasarkan peluang simbol *b* kemudian simbol *b* dimasukkan ke dalam *symbol occurrence pool.* Saat ini *symbol occurrence pool* berisi empat simbol yaitu dua simbol *b* dan dua simbol *c : O = {b,c,c,b}* sedangkan interval peluang baru yang terbentuk adalah *low* = 0.6390, *high* = 0.650 I. Karena pada tahap ini jumlah symbol pada *symbol occurrence pool* sama dengan *window size,* maka proses *encode* dihentikan kemudian pemilihan sembarang angka diantara 0.6390 *(low* terakhir) dan 0.6501 *(high* terakhir) dilakukan. Misal angka decimal 0.64 dipilih, maka *encoder* akan mengirimkan simbol 0.64.



Gambar 14. Proses Penghitungan Adaptive Arithmetic Coding

Tahapan proses *encoding* menggunakan *adaptive arithmetic coding* adalah sebagai berikut :

Langkah 1 Inisialisasi *symbol occurrence pool* **O = { },** inisialisasi *window size* sesuai deng-an jumlah simbol yang akan di *encode* dan inisialisasi interval peluang dengan *low* = 0 dan *high* = I.

Langkah 2 Hitung peluang masing-masing simbol yang ada pada himpunan simbol yang ada.

Langkah 3 Hitung nilai posisi setiap simbol pada interval peluang berdasarkan peluangnya masing-masing

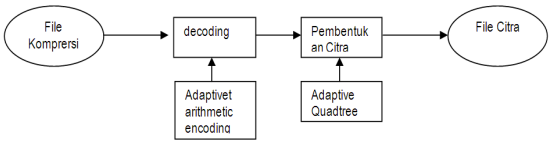
Langkah 4 *Encode* simbol awal pada string dengan cara memindahkan symbol tersebut ke dalam *symbol occurrence pool.*

Langkah 5 Buat interval peluang yang baru dengan nilai *low* dan *high* berdasarkannilai posisi *low* dan *high* pada simbol yang di *encode.*

Langkah 6 Cek apakah jumlah symbol yang ada pada *symbol occurrence pool* telah men-capai *window size.* Jika sudah, pilih dan simpan nilai posisi antara *low* dan *high* pada interval peluang terbaru kemudian akhiri prosedur *encoding.* Jika belum, ulangi langkah ke 2.

9.2 Analisis Dekompresi Citra

Kerangka umum dari dekompresi citra menggu-nakan metode *quadtree* Dapat dilihat pada gambar di bawah ini



Gambar 15. Tahapan dekompresi citra metode quadtree

Pada gambar diatas, terdapat dua bagian utama proses dekompresi citra. Kedua bagian tersebut adalah:

a. *Decodin,* data hasil kompresi yang telah di *encode* harus dikembalikan lagi ke dalam bentuk aslinya agar dapat digunakan untuk pembentukan citra. Proses ini dilakukan dengan menggunakan algoritma *adaptive arithmetic coding.*

b. Pembentukan citra, dengan menggunakan berba-gai parameter yang ada maka *quadtree code* akan disusun kembali pada citra baru sehingga menghasilkan range dengan ukuran yang berbeda. Setelah itu citra dibentuk dengan melakukan proses IFS ke dalam citra baru sesuai denganjumlah iterasi yang ditentukan.

9.3. Analisis *Decoding*

Data yang didapat dari file kompresi harus *didecode* agar kembali ke bentuk semula. Algoritma yang dipakai untuk memenuhi hal ini adalah *adaptive arithmetic coding*. Proses *decoding* dari *adaptive arith-metic coding* merupakan bentuk terbalik dari proses *encoding*. Berikut ini adalah contoh proses *decoding* menggunakan penghitungan *adaptive arithmetic coding*. Suat sirnbol 0.64 akan di *decode* menggunakan teknik *adaptive arithmetic coding.* rnaka langkah-langkah yang dilakukan adalah sebagai berikut:

a. lnisialisasi *symbol occurrence pool* sebagai wadah kosong $O = \{\}$ dan inisialisasi *windows size* sebesar 4 (sesuai dengan *windows size* sewaktu *encoding)* setelah itu peluang kemunculan setiap simbol anggota himpunan *{a,b,c}* akan dihitung menggunakan rumus:

$$\mathrm{p(a)} = \frac{N(a) + 1}{N(a) + N(b) + N(c) + 3}$$

$$\mathrm{p(b)} = \frac{N(b) + 1}{N(a) + N(b) + N(c) + 3}$$

$$\mathrm{p(c)} = \frac{N(c) + 1}{N(a) + N(b) + N(c) + 3}$$

b. dimana *p* melambangkan peluang kemunculan, *N* melambangkan jumlah simbol tertentu yang ada dalam *symbol occurrence pool*. Catatan: penamba-han angka l pada pembilang dan angka 3 pada penyebut berdasarkan rumus dasar peluang, besamya penambahan pada penyebut didasarkan pada jumlah simbol pada himpunan. *Decode* simbol 0.64 yang pertama dilakukan. Karena *symbol occurrence pool (0)* masih kosong maka *N{a) = N(b) = N(c)* = 0, sehingga didapatkanp{a) = *p{b)= p(c)*= 113. Interval peluang awal (0,1) akan terbagi menjadi tiga bagian sesuai dengan besar peluang kemunculan masing-masing simbol. Pada tahap ini 0.64 berada diantara nilai posisi *low* = 0.3333, *high* = 0.6667 yang mewakili simbol *b.* maka simbol *b* dimasukkan ke dalam *symbol occurrence pool (0)* dan dinotasikan sebagai berikut : *O={b}* sedangkan interval peluang baru yang terbentuk adalah *low* = 0.3333, *high* = 0.6667. Karena jumlah simbol pada *symbol occurrence pool* belum mencapai *window size*, maka proses *decode* simbol 0.64 akan terns diulangi

c. Pada proses *decode* simbol 0.64 yang ke dua, *p(a), p(b)* dan *p{c)* akan dihitung ulang. Karena saat ini pada *symbol occurrence pool* berisi satu simbol *b* maka *N(b)* =! sedangkan nilai *N(a)* = *N(c)* = 0 setelah itu didapatkan *p(a)* = *p(c)* = 114 dan *p{b)* = 2/4. Pada tahap ini 0.64 berada diantara nilai posisi *low* = 0.5834, *high* = 0.6667 yang mewakili simbol *c*. maka simbol *c* dimasukkan ke dalam *symbol occurrence pool (0)* dan dinotasikan sebagai berikut : $O = \{b,c\}$ sedang-kan interval peluang baru yang terbentuk adalah *low* = 0.5834, *high* = 0.6667.

d. Pada proses *decode* simbol 0.64 yang ke tiga, *p(a), p{b)* dan *p{c)* akan dihitung ulang. Karena saat ini pada *symbol occurrence pool* berisi dua simbol, yaitu *b* dan *c* maka *N(a)* =0 sedangkan nilai *N(b)* = *N(c)* = I setelah itu akan didapatkan *p(a)* = 115, *p(b)* = *p(c)* = 2/5. Pada tahap ini 0.64 ber-ada diantara nilai posisi *low* = 0.6334, *high* = 0.6667 yang mewakili simbol *c.* maka simbol *c* dimasukkan ke dalam *symbol occurrence pool (0)* dan dinotasikan sebagai berikut : *O = {b.c,c)* sedangkan interval peluang baru yang terbentuk adalah *low* = 0.6334, *high* = 0.6667.

d. Pada proses *decode*.. simbol 0.64 yapg ke empat, *p(a), p(b)* dan *p(c)* akan dihitung ulang. Karena saat ini pada *symbol occurrence pool* berisi tiga simbol, yaitu satu simbol *b* dan dua simbol *c* maka *N(a)* =0, nilai *N(b)* = I, *N(c)* = 2 setelah itu akan didapatkan *p(a)* = 116. *p(b)* = 2/6, *p(c)* = 316. Pada tahap ini 0.64 berada diantara nilai posisi *l011.* 0.6390. */ugh* · 0.6501 yang mewakili simbol *c.* maka simbol *c* dimasukkan ke dalam *symbol occurrence pool(0)* dan dinotasikan sebagai berikut :*O = {b.c,c,b}* sedangkan interval

peluang baru yang terbentuk adalah *low* = 0.6390, *high* = 0.650**I.** Pada saat ini jumlah symbol pada *symbol occurrence pool* sama dengan *window size,* maka proses *decode* dihentikan dan string yang dihasilkan adalah kumpu-lan symbol yang ada pada *Jymbol occurrence pool* yaitu string *bccb.*

Tahapan proses *decoding* menggunakan *adaptive arithmetic coding* adalah sebagai berikut :

Langkah 1 Inisialisasi *symbol occurrence pool* **O** = *{},* inisialisasi *window size* sesuai dengan *window size* yang ada pada proses *encoding* dan inisialisasi interval peluang dengan *low* = 0 dan *high* = **I.**

Langkah 2 Hitung peluang masing-masing simbol yang ada pada himpunan simbol yang ada.

Langkah 3 Hitung nilai posisi setiap simbol pada berdasarkan peluangnya masing-masing interval peluang

Langkah 4 *Decode* simbol baru dengan cara mema-sukkan nilai simbol baru tersebut ke dalam peluang interval yang ada, masuk-kan simbol asli yang memiliki nilai posisi yang mencakup nilai dari simbol yan£ di *JccoJ,·* l..c dalam *symbol occurrence pool.*

Langkah 5 Buat inten al pcluang yang baru dengan nilai *low* dan *high* berdasarkannilai posisi *low* dan *high* pada symbol asli yang terpilih.

Langkah 6 Cek apakah jumlah simbol yang ada pada *symbol occurrence pool* Ielah mencapai *window size.* Jika sudah, akhiri prosedur *decoding.* Pada tahap ini kumpulan sim-bol yang ada pada *symbol occurrence pool* adalah representasi dari kumpulan simbol asli. Jika bel urn, ulangi langkah ke 2

9.4. Analisis Pembcntukan Citra

Proses pembentukan citra menggunakan proses IFS, yaitu proses yang dilakukan berulang-ulang sesuai jumlah iterasi yang ditentukan. Untuk setiap dimasuk-kan ke dalam domain kemudian dilakukan *transformasi-affine* terhadap domain sesuai dengan simetri yang ada. Informasi koordinat domain, nilai luminansi, dan simetri *affine* untuk setiap range tersimpan pada *koefisien fractal* yang terbentuk saat proses kompresi. Hasil dari proses ini adalah terbentuknya citra sementara pada iterasi saat ini. Sesuai dengan jumlah iterasi yang ditentukan, ulangi proses IFS terhadap setiap range yang ada. Setelah iterasi selesai dilakukan akan terbentuk citra keseluruhan hasil dekompresi.

9.5. Penghitungan Kualitas Citra

Citra hasil dekompresi akan mengalami penuru-nan kualitas *(lossy)* dari citra aslinya. Untuk menghitung seberapa besar perbandingan penurunan kua-litas citra (citra asli dengan citra hasil dekompresi) maka dilakukan proses penghitungan kualitas citra menggunakan PSNR.

Pada dasarnya PSNR *(peak signal-to-noise ratio)* berguna untuk menghasilkan suatu angka yang dapat mencerminkan kualitas dari sebuah citra. Teori PSNR mengatakan bahwa semakin tinggi nilai PSNR berarti semakin baik kualitas citra hasil dekompresi. Namun dikarenakan angka-angka yang di proses pada perhi-tungan PSNR didapat dari komputasi nilai piksel citra asli dan citra hasil dekompresi, maka sesungguhnya ukuran PSNR tidaklah sesuai dengan persepsi manusia akan kualitas citra. Oleh karena itu nilai PSNR yang lebih tinggi sesungguhnya belum tentu memiliki kualitas citra yang lebih baik.

Saat ini, telah banyak penelitian mengenai cara mengukur kualitas citra berdasarkan persepsi penglihatan manusia, namun semuanya masih merupakan perhi-tungan yang masih sulit diaplikasikan. Oleh karena itu PSNR masih menjadi tolak ukur penghitungan kualitas citra dalam dunia *computer vision.* Penjelasan penghit-ungan PSNR adalah sebagai berikut. Tersedia suatu citra asli yang dilambangkan sebagai *f(i,j)* yang memiliki *NxN* piksel dan citra hasil dekompresi yang dilam-bangkan sebagai *F{ij).* Untuk menghi-tung nilai PSNR,

Jangkah pertama yang perlu dilakukan adalah menghi-tung nilai dari MSE *(mean squared err01)* Setelah itu dilakukan penghitungan *bit rate* dan MAX1. Setelah mendapatkan semua koefisien yang dibutuhkan, maka penghitungan PSNR dapat dilakukan. Adapun rumus-rumus untuk penghitungan MSE, MAX1 dan PSNR adalah sebagai berikut :

$$MSE = \frac{\sum[f(i,j) - F(t,j)]^2}{N\ x\ N}$$

$$MAX_t = 2^B - 1$$

$$PNR = 20\ log_{10}\left(\frac{MAX_1}{\sqrt{MSE}}\right)$$

Dimana:

f(i.j)	: Nilai piksel citra asli pada koordinat *i.j*
F(i.j)	: Nilai piksel citra hasil dekompresi pada koordinat *i.j*
Jumlah bit	: Jumlah bit
Jumlah_pxl	:Jumlah piksel
B	: Jumlah bit per piksel
N	: Resolusi citra
MSE	: Nilai error keseluruhan citra
MAX_1	: Nilai piksel maksimum pada citra

PSNR biasanya memiliki nilai yang berkisar diantara 20 sampai 40 dan biasanya ditulis menggu-nakan format desimal dengan 2 angka dibelakang koma dengan satuan db *(decibel)*. Namun untuk menampil-kan data yang sesungguhnya, maka dalam penelitian ini nilai PSNR ditulis dengan lengkap sesuai dengan tipe data yang digunakan dalam program yang dirancang.

10. Simpulan

Berdasarkan hasil analisis, perancangan, implementasi dan evaluasi yang telah dilakukan, dapat disimpulkan:

1. bahwa metode *adaptive quadtree* yang digunakan dalam pengompresan citra secara keseluruhan lebih memilki nilai tambah dibanding metode IFS yang lain dengan menggunakan metode PIFS sebagai perbandingan, ditinjau dari segi rasio dan waktu.
2. Ditinjau dari segi kualitas citra hasil dekompresi yang dapat dilihat pada tabel pcrbandingan dekompresi *adaptive quadtree* dan PIFS, metode *adaptive quad-tree* masih kalah dengan metode PIFS, ini disebabkan pemilihan range pada metode *adaptive quadtree* yang dibatasi dengan level maksimum dan level minimum berdasarkan nilai *threshold.* sehingga blok-blok range yang dihasikan memiliki bentuk yang tidak sama besar. Tidak seperti metode PIFS yang telah menentukan bentuk rangenya secara tetap dengan ukuran yang sama.

Namun, hal diatas menjadi nilai lebih pada metode metode *adaptive guadtree,* yaitu dari rasio pengompresan. Dikarenakan adanya minimum dan maksimum level pada metode *adaptive quadtree* maka range yang dihasilkan lebih sedikit jumlahnya dari metode PIFS sehingga data informasi yang disimpan menjadi lebih sedikit dibandingkan dengan metode PIFS. Metode *adaptive quadtree* menghasilkan rasio kompresi yang cukup tinggi yaitu 60%-70% untuk citra dengan detil yang berbeda-beda dibandingkan dengan PIFS. serta tidak menutup kemungkinan metode ini digunakan dalam pengompresan video.

11. Saran

Hendaknya penelitian ini membuka pintu baru bagi peneliti mengenai penelitian citra *fractal* yang merupakan teknologi yang tergolong sangat baru dan masih dalam tahap penelitian. Masih terbuka peluang yang cukup luas untuk menggali dan mengembangkan citra fractal.

Daftar Pustaka

[1] Sharat Chandran, S., Kar, (2002). Retrieving Faces by the PIFS Fractal Code, http://www.umiacs.umd.edu/sharat

[2] Fisher, Yuval. (1995). *Fractal Image Compression: Theory and Application.* Springer- Verlag Telos.

[3] Vales, R.C.,Woods, R.E. (2001). *Digital Image Processing.* Addison Wesley Publishing Company. Inc.
[4] Amstein, H., Ruhl, M., Saupe, D. (2001). *Region-based fractal image compression.* IEEE Transactions On Image Processing, Vol. 9, p1171-1184. IEEE.
[5] Jin, A. E. (1992). *Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations.* IEEE Transactions On Image Processing. Vol. 1, No. 1. IEEE.
[6] Mandelbrot, Benoit M. (1983), *The Fractal Geometry of Nature*, WH Freeman and Company, New York.
[7] Devaney, R and L. Keen, eds. (1989), *Chaos, Fractals and Dynamics*, Addison-Wesley, Menlo Park, CA.
[8] Oliver, Dick. (1999) *Memandang Realita dengan Fractal Vision.* Edisi Terjemahan. Andi Yogyakarta, Yogyakarta.
[9] Aterman, E., Feder, M. (1994). *Image Compression via Improved Quadtree Decomposition Algorithms.* IEEE Transactions On Image Processing, Vol. 3, No. 2. IEEE.
[10] Mandelbrot, Benoit M. (1983), *The Fractal Geometry of Nature*, WH Freeman and Company, New York.
[11] Stevens, Roger T (1990), *Advanced Fractal Programming in C.* M&T, Redwood City, CA.
[12] Chien-Cheng Tseng, Tsung-Ming Hwang, 2003), Quantum Digital Image Processing Algorithms, 16th IPPR Conference on Computer Vision, Graphics and Image Processing (CVGIP 2003)
[13] stead, Stephen. (1999). *Fractal and Wavelet Image Compression Techniques. SPIE Tutorial Texts in Optical Engineering Vol. TT40).* SPIE Press.
[14] Kamel Belloulata and Janusz Konrad, (2002), *Fractal Image Compression with Region-Based Functionality*, *Senior Member*, *IEEE*, IEEE TRANS. ON IMAGE PROCESSING, VOL. 11, NO. 4, APRIL 2002