JURNAL TEKNOLOGI, Edisi No.2, Tahun XV, Juni 2001, 181-188. ISSN 0215-1685

Kompresi Gambar Dengan Metode Fractal Quadtree Partitioning

Dadang Gunawan dan Better Pasaribu
Jurusan Elektro Fakultas Teknik Unniversitas Indonesia
Kampus UI Depok, Depok – 16424.
guna@eng.ui.ac.id, wasp@eng.ui.ac.id

Abstrak

Makalah ini membahas aplikasi fractal pada kompresi gambar dengan metode quadtree partitioning sebagai penentu pasangan domain-range terbaik. Untuk mempelajari lebih lanjut mengenai metode ini dilakukan simulasi dengan mengamati beberapa parameter seperti toleransi, tingkat quadtree partitioning dan iterasi yang digunakan.

Analisa dilakukan terhadap unjuk kerja (meliputi PSNR dan rasio kompresi) dari masing-masing parameter. Dari analisa dapat ditentukan berapa tingkat quadtree partitioning yang diperlukan dan toleransi yang digunakan pada proses kompresi dan jumlah iterasi yang diperlukan untuk menghasilkan sebuah gambar yang baik pada proses dekomposisi.

Abstract

This paper discuss fractal application on image compression using the method of quadtree partitioning as a determiner of the best domain-range. In order to study further on this matter, simulation is done with some parameters, i.e a variety of tolerance, quadtree partitioning level and number of iteration.

Analysis is done toward performance (PSNR and compression ratio) of each parameters. From the analysis can be determined some levels of quadtree partitioning, tolerance and number of iteration use to produced a good image in the decompression process.

Keyword : fractal, compression, affine transform, iterated function system (IFS), quadtree partitioning, tolerance, partitioned iterated function system (PIFS).

1. Pendahuluan

Penyimpanan, manipulasi, dan transmisi gambar dalam bentuk tidak terkompresi akan menimbulkan masalah; baik masalah dalam penyimpanan (membutuhkan ruang penyimpanan yang besar) maupun masalah transmisi jika gambar tersebut dikirimkan (volume data yang dikirimkan besar, bandwidth yang lebar dan waktu yang lama dalam proses pengirimannya).

Untuk mengatasi masalah tersebut diperlukan metode kompresi gambar, yang dapat menghasilkan suatu gambar yang memiliki jumlah bit atau data yang lebih kecil dibandingkan dengan gambar aslinya tanpa menyebabkan penurunan kualitas yang berarti.

Kebanyakan gambar berisi data atau pixel yang sama (*redundancy*), yang dapat hilangkan dengan tidak mengurangi nilai informasi dari gambar sehingga mata manusia tidak dapat membedakan ada atau tidak adanya informasi yang dihilangkan dari gambar.

Banyak metode kompresi gambar yang telah dikenal, salah satunya adalah metode transformasi affine. Metode kompresi yang didasarkan pada transformasi affine ini atau yang lebih lazim disebut dengan algoritma fractal telah mendapat perhatian khusus. Hal ini disebabkan karena kompresi gambar dengan metode fractal ini dapat menghasilkan rasio kompresi yang tinggi dan kualitas gambar hasil kompresi yang baik.

2. Definisi Fraktal

2.1. Umum

Suatu *image* pada umunya direpresentasikan oleh suatu *array* dua dimensi, umunya direpresentasikan oleh suatu array dua dimensi, dimana elemen-elemennya yang berupa *pixel* (*picture element*) membawa informasi yang terdapat pada gambar [3] – [6]. *Image* sering kali membawa informasi yang bersifat *redundancy*, sehingga untuk menyimpannya diperlukan kapasitas yang besar dan jika *image* ditransmisikan akan memakai lebar pita yang besar.

2.2. Apa itu Fractal ?

Fractal adalah sebuah gambar geometri yang bagian kecil dari gambar tersebut menyerupai seluruh gambar (*self similar*). Gambar yang *self-similar* ini banyak ditemukan, contoh yang paling baik adalah garis pantai Koch dan segitiga Sierpinski (lihat pada Gambar 1).

Gambar 1 Gambar segitiga Sierpinski [1].

2.3. Transformasi Linier Affine

Transformasi yang digunakan untuk membentuk segitiga Sierpinski, garis pantai Koch dan bentuk-bentuk *fractal* yang lainnya adalah menggunakan transformasi *affine*.

Transformasi *affine* adalah transformasi linier yang terdiri dari 4 (empat) operasi dasar, yang meliputi:

1. Operasi penskalaan (*scaling*)
2. Operasi rotasi (*rotation*)
3. Operasi translasi (*translation*)
4. Operasi refleksi (*reflection*).

Karena gambar atau *image* yang akan dibahas adalah bentuk dua dimensi, maka gambar tersebut harus berada dalam sistem sumbu x dan sumbu y. Setiap gambar terdiri dari *pixel-pixel* dan setiap *pixel* dianggap sebagai titik.

Jika sebuah titik P merepresentasikan suatu pixel, maka representasi pada bidang gambar (X,Y) diberikan oleh suatu hubungan sebagai $P(x,y)$. Suatu *linier mapping* F adalah transformasi terhadap setiap titik P pada bidang datar tersebut diberikan oleh hubungan $F(P) = F(P_1) + F(P_2) + \cdots + F(P_n)$ [2], dimana *linier mapping* F (transformasi) direpresentasikan oleh matrik [2]:

$$F = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = F(P) = F\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \quad (2)$$

terlihat bahwa untuk transformasi linier ini ditentukan oleh 4 (empat) koefisien, yaitu a, b, c, dan d. Untuk mempermudah mendapatkan koefisien itu maka:

$a = r \cos \theta$, $b = -s \sin \varphi$, $c = r \sin \theta$, dan $d = s \cos \varphi$ dengan :

r = faktor reduksi terhadap sumbu x

s = faktor reduksi terhadap sumbu y

θ = sudut rotasi

$\varphi = \theta$ untuk normal rotasi,

$\varphi = \theta + \pi$, untuk refleksi.

Suatu transformasi setidaknya diikuti dengan perpindahan tempat (translasi). Maka transformasi linier *affine* (w) adalah suatu gabungan antara linier *mapping* dan translasi. Jika persamaan (1) di atas (F) adalah linier *mapping* dan Q adalah suatu titik translasi, maka bentuk pemetaan (*mapping*) yang baru adalah :

$$w(P) = F(P) + Q \quad (3)$$

dimana P merepresentasikan semua titik pada suatu bidang datar, F adalah suatu matrik seperti pada persamaan (1) dan Q adalah pasangan koordinat (x,y). Jika Q direpresentasikan oleh koordinat (e,f) maka transformasi linier affine ditentukan oleh enam buah parameter sebagai berikut [2] :

$$w = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \quad (4)$$

jika $P=(x,y)$ dan $w(p)=(u,v)$ maka persamaan (2-4) dapat dituliskan menjadi :

$$\begin{bmatrix} u \\ v \end{bmatrix} = w(P) = w\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \quad (5)$$

atau

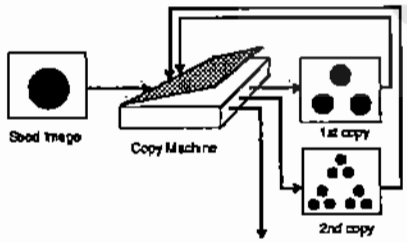
$$u = ax + by + e \quad (6)$$

$$v = cx + dy + f \quad (7)$$

$$w(x,y) = (ax + by + e, cx + dy + f) \quad (8)$$

2.4. Kompresi Gambar Dengan Metode Fractal

Kompresi gambar dengan metode fractal dapat dianggap sebagai sebuah mesin *photocopy* (Gambar 2) dengan tipe yang special yaitu mereduksi sebuah gambar masukan menjadi setengah kali aslinya dan kemudian diperbanyak sebanyak tiga kali. Apa yang akan terjadi, apabila *output* dari mesin *photocopy* tersebut dimasukkan kembali sebagai *input*? Akan dapat dilihat bahwa hasil yang akan didapat adalah seperti pada Gambar 3, apabila gambar masukannya berupa huruf "A".



Gambar 2 Ilustrasi sistem kompresi sebuah huruf A [2].

Gambar 3 Proses pembentukan dekompresi sebuah huruf A.

Pada Gambar 2 di atas mengilustrasikan suatu kunci dari kompresi pada gambar dengan metode *fractal*. Sekarang, proses apa yang terjadi di dalam mesin *photocopy* tersebut? Di dalam mesin tersebut, terjadi suatu transformasi, yang dinamakan transformasi *affine*, yang didefinisikan seperti pada persamaan (4). Transformasi *affine* ini mempunyai kemampuan untuk *skewing*, *stretching*, *scaling* dan *translation*. Apabila diberikan beberapa nilai parameter dari transformasi *affine*, maka dari parameter tersebut akan dapat dihasilkan gambar, dimana gambar yang dihasilkan adalah gambar yang *self-similar* yang dinamakan *fractal*.

2.4.1. Iterated Function System

Karakteristik dari mesin di atas dijelaskan dengan sebuah model matematika yang dinamakan dengan *Iterated Function System* (IFS). Sebuah IFS terdiri dari sekumpulan transformasi *contractive affine* yang merupakan fungsi pemetaan seperti persamaan di bawah ini [1] :

$$w(\bullet) = \bigcup_{i=1}^{n} w_i(\bullet) \quad (9)$$

Dengan kata lain, apabila diberikan sebuah gambar masukan f_0, kemudian transformasi *affine* diterapkan kepada gambar masukan tersebut secara berulang-ulang, maka akan didapat: $f_1 = w(f_0)$ sebagai hasil yang pertama, $f_2 = w(f_1) = w(w(f_0)) = w^2(f_0)$ sebagai hasil yang kedua, dan apabila diteruskan sampai n kali, makaakan dihasilkan sebuah gambar yang memenuhi persamaan [1]:

$$|w| = f_\infty = \lim_{x \to \infty} w^n(f_0) \quad (10)$$

2.4.2. Self-Similarity Pada Gambar

Untuk mengetahui apakah *w contractive* maka harus didefinisikan sebuah jarak antara dua gambar. Jarak ini didefinisikan dengan persamaan [1]:

$$\delta(f,g) = \sup_{(x,y) \in P} |f(x,y) - g(x,y)| \quad (11)$$

dimana

- f dan g adalah merupakan nilai tingkat keabuan *pixel* untuk gambar *grayscale*.

- P adalah ruang atau *space* dari gambar
- x,y adalah merupakan letak koordinat dari tiap *pixel* yang menyusun suatu gambar.

Jarak ini akan memberikan informasi pada posisi mana *(x,y)* gambar *f* dan *g* sangat berbeda ataupun hampir sama (*self similar*).

Tidak semua gambar yang ada benar-benar *self similar*), contohnya gambar wajah dari Lenna, gambar ini bukan merupakan tipe gambar yang *self similar* seperti gambar segitiga Sierpinski. Tetapi dalam gambar ini (Lenna) dapat ditentukan bagian yang *self similar* dari gambar seperti diperlihatkan pada Gambar 4.

Gambar 4 Bagian *self similar* dari *test image* Lenna [1]

Gambar 4 ini menunjukkan contoh bagian dari gambar Lenna yang *similar* pada skala yang berbeda, bagian bahu dari gambar menunjukkan suatu kesamaan pada skala yang berbeda dengan bagian yang merupakan refleksi dari topi pada cermin adalah sama (setelah proses transformasi) dengan bagian dari topi. Perbedaan utama antara jenis *self similarity* dalam gambar segitiga Sierpinski (Gambar 3) dan gambar Lenna (Gambar 4) adalah bahwa segitiga Sierpinski dibentuk dari keseluruhan segitiga yang kemudian dikodekan ke dalam transformasi *affine*, sedangkan gambar Lenna dibentuk dari sebagian gambar yang layak untuk mewakili gambar. Bagian gambar ini tidak sepenuhnya sama, hal ini berarti bahwa gambar yang dikodekan sebagai satu bentuk transformasi bukanlah merupakan bentuk sepenuhnya sama dengan gambar aslinya.

2.4.3. Partitioned Iterated Function System

Dalam mengkodekan sebuah gambar, gambar tersebut harus lebih dahulu dibagi ke dalam bagian-bagian yang memungkinkan dilakukan transformasi atau pengkodean terhadap bentuk yang sulit ditransformasikan dengan menggunakan *Partitioned Iterated Function System* (*PIFS*). *PIFS* juga didasarkan pada transformasi *affine*, akan tetapi *PIFS* ini mentransformasikan sebuah bagian dari gambar ke bagian lain dalam gambar yang sama (yang disebut dengan Domain, *D*). Transformasi *affine* dapat mentrasformasikan tingkat keabuan dari *pixel* yang ada pada gambar. Untuk alasan ini ditambahkan suatu dimensi baru ke dalam transformasi *affine*, seperti terlihat pada persamaan (2.12) di bawah ini [2] :

$$w_i\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \qquad (12)$$

dimana:

s_i = parameter untuk mengatur kekontrasan dari gambar (*contrast*)

o_i = parameter yang mengatur *brightnessi* dari gambar

$z = f(x,y)$.

Transformasi *w* harus *contractive* ke semua arah, arah sumbu *x*, *y*, dan *z*. Transformasi tersebut akan *contractive* apabila jarak pada sumbu *z* direduksi dengan faktor lebih kecil dari 1 atau untuk nilai $s_i < 1$.

2.4.4. Pengkodean Gambar

Diberikan sebuah gambar *f* yang akan dikodekan, maksud dari dikodekan dalam hal ini adalah menemukan persamaan (9) dan (10) sehingga didapatkan *f* sebagai sebuah titik gambar (*fixed point*) dari pemetaan *w*. *Fixed point* yang didapat tersebut adalah memenuhi persamaan [1]:

$$f = w(f) = w_1(f) \cup w_2(f) \cup \ldots . w_N(f) \qquad \ldots\ldots (13)$$

Bila f dibagi menjadi N bagian yang tidak *overlapped* dari gambar dan kemudian menerapkan transformasi w_i dan mendapatkan kembali gambar f, hal ini merupakan sesuatu yang sangat ideal, karena suatu gambar tidak secara tepat disusun oleh bagian-bagian yang dapat ditransformasikan dan sesuai dengan bagian yang lain di dalam gambar, yang diharapkan adalah sebuah gambar, $'f = |w|$ dan $\delta('f,f)$ yang kecil. Atau dengan kata lain, dicari suatu transformasi w yang hasilnya (*fixed point*) bukan f, akan tetapi sebuah gambar yang dekat dengan gambar f, seperti yang diekspresikan pada persamaan (14) di bawah ini [1]:

$$f \approx \dot{f} = w(\dot{f}) \approx w(f) = w_1(f) \cup w_2(f) \cup \ldots w_N(f) \quad \ldots\ldots(14)$$

Dari persamaan ini harus dicari bagian D_i dan transformasi w_i, sehingga apabila transformasi tersebut diterapkan ke bagian gambar D_i akan didapatkan sebuah gambar yang sangat mirip (dekat) dengan bagian gambar R_i seperti ditunjukkan pada Gambar 5 di bawah ini.

Gambar 5 Proses pengkodean [3].

2.4.5. Pasangan D_i dan R_i Terbaik (mencari domain-range)

Dalam mengkompres sebuah gambar dengan metode *fractal* terlebih dahulu gambar tersebut dibagi ke dalam dua bagian, yaitu bagian yang *non-overlapping* atau yang disebut dengan *range* blok(R_i) dan bagian yang *overlapping* yang disebut dengan *domain* blok(D_i), dimana ukuran *domain* blok adalah 2 R_i. Tujuan dari proses kompresi itu sendiri adalah bagaimana mencari pasangan R_i dan D_i yang terbaik atau dengan jarak yang sekecil mungkin. Apabila diberikan sebuah gambat *grayscale* dengan ukuran 256x256 *pixel*, maka akan didapat 1024 *range* blok dengan ukuran 8x8 *pixel*, kemudian gambar dibagi menjadi *domain* blok D_i dengan ukuran 16x16 *pixel* yang dapat *overlapped*, sehingga akan didapatkan (256-16+1)*(256-16+1) = 58.081 *domain* blok. Untuk tiap R_i akan dicari sebuah D_i dengan jarak yang minimal. Ada delapan operasi simetri yang digunakan untuk mendapatkan pasangan Range-Domain yang optimal, yaitu:

1. Dirotasikan sebesar 0° (Domain blok itu sendiri
2. Dirotasikan sebesar π/2
3. Dirotasikan sebesar π radian
4. Dirotasikan sebesar -π/2
5. Direfleksikan terhadap sumbu vertikal
6. Direfleksikan terhadap sumbu horizontal
7. Direfleksikan terhadap diagonal utama (pertama)
8. Direfleksikan terhadap diagonal kedua.

Dengan adanya delapan operasi simetri ini berarti bahwa D_i yang akan dibandingkan adalah sebanyak 8*58.081 = 464.648 *domain* blok dengan 1.024 *range* blok. Untuk menemukan D_i yang paling mirip dengan gambar R_i, harus ditentukan juga s_i (*contrast*) dan *brightness* dengan menggunakan perhitungan jarak Euclidean [1]:

$$E = \sum_{i=1}^{n} \left[(s.d_i + o) - r_i \right]^2 \qquad (15)$$

dimana: d_i = intensitas *pixel* pada D_i
r_i = intensitas *pixel* pada R_i

Menghitung nilai parameter *contrast* dan *brightness*, harus menghitung jarak minimum antara *domain* blok dan *range* blok. (R) minimum terjadi pada saat turunan partial dari : $\frac{\partial E}{\partial s} = 0$ dan $\frac{\partial E}{\partial o} = 0$, dimana hal ini akan terjadi pada nilai [1] :

$$s = \frac{\left[n^2 \left(\sum_{i=1}^{n} a_i b_i \right) - \left(\sum_{i=1}^{n} a_i \right) \left(\sum_{i=1}^{n} b_i \right) \right]}{\left[n^2 \sum_{i=1}^{n} a_i - \left(\sum_{i=1}^{n} a_i \right)^2 \right]} \quad \ldots\ldots(16)$$

dan

$$o = \frac{\left[\sum_{i=1}^{n} b_i - s.\sum_{i=1}^{n} a_i\right]}{n^2} \qquad (17)$$

sehingga diperoleh :

$$R = \frac{1}{n}\left[\sum_{i=1}^{n} b_i^2 + s\left(s\sum_{i=1}^{n} a_i^2 - 2\sum_{i=1}^{n} a_i b_i + 2o\sum_{i=1}^{n} a_i\right) + o\left(on - 2\sum_{i=1}^{n} b_i\right)\right]$$

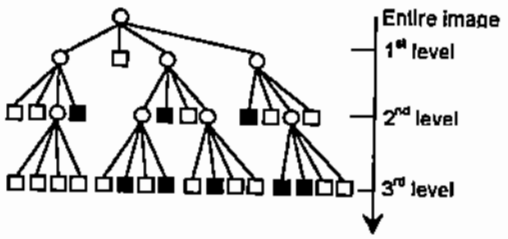
.....(18)

Dengan demikian setiap *range* blok harus dibandingkan dengan *domain* blok yang telah ditransformasikan dengan menghitung besarnya R(rms_error). Sehingga *domain* blok yang ditransformasikan dengan nilai R yang minimum akan merupakan pasangan dari *range* yang paling optimal dan pemilihan D_i dengan parameter s_i dan o_i yang sesuai akan menentukan sebuah pemetaan w_i dan apabila dilakukan sebanyak 1.024 (jumlah *range* blok) maka akan didapatkan w_1, w_2, ... w_{1024}, sehingga suatu gambar dapat dikodekan.

2.4.6. Quadtree Partitioning

Kelemahan dari metode yang diberikan pada penjelasan di atas adalah digunakannya ukuran yang tetap dari R_i dan D_i sehingga ada bagian-bagian tertentu dari gambar yang tidak dapat di-*cover* dengan baik dengan cara tersebut di atas.

Untuk mengatasi masalah tersebut digunakan suatu metode yang dinamakan dengan metode *quadtree partitioning*. Dalam metode ini, sebuah gambar segi empat (*square*) dibagi menjadi empat *sub squared* yang mempunyai ukuran yang sama, seperti diperlihatkan pada Gambar 6 di bawah ini:



Gambar 6 Proses Quadtree tiga tingkat.

Dengan proses ini, gambar tersebut dibandingkan dengan *domain* blok yang ada. Untuk setiap *range* blok akan didapatkan suatu nilai *rms_error* yang paling kecil yang memenuhi suatu nilai *threshold* tertentu atau akan didapatkan pasangan *range-domain* yang optimal sesuai dengan nilai *threshold* yang diberikan, sehingga akan didapatkan nilai transformasi w. Apabila nilai *rms_error* belum didapat, maka gambar tersebut dibagi lagi menjadi empat bagian yang sama dan proses ini akan diulangi lagi (*recursive*) sampai didapat gambar (*square*) yang cukup kecil dan didapat suatu pasangan *range-domain* yang optimal. Hal ini terjadi karena *square* yang lebih kecil akan di-*cover* lebih baik oleh *domain* blok, karena *pixel* yang berdekatan di dalam sebuah gambar cenderung sangat berdekatan. *Quadtree partitioning* ini adalah sebuah representasi dari sebuah gambar yang dapat diumpamakan seperti sebuah pohon yang mempunyai cabang, dimana setiap cabang mempunyai *square* yang merupakan bagian dari sebuah ranting gambar. Proses *quadtree* tersebut ditunjukkan seperti pada Gambar 6 di atas.

2.5. Proses Dekompresi

Setelah sebuah gambar berhasil dikompres secara *fractal*, hasil yang didapatkan adalah dalam bentuk kode yang berisi nilai-nilai transformasi w yang didapat dari pasangan *range-domain*.

Untuk mengembalikan gambar ke dalam bentuk yang dapat dilihat secara kasat mata, maka dilakukan suatu proses yang dinamakan dengan proses dekompresi. Pada proses dekompresi (rekonstruksi) ini, yang dilakukan adalah proses iterasi terhadap kode-kode yang didapatkan dari proses kompresi pada sebuah *initial image* yang dipilih secara bebas, biasanya dipilih gambar yang berwarna hitam. Proses iterasi ini akan dilakukan terus sampai didapatkan gambar yang diinginkan.

3. Metode Simulasi

Simulasi dilakukan melalui langkah-langkah berikut:

3.1. Proses Kompresi

Pada proses kompresi yang dilakukan adalah langkah-langkah sebagai berikut:

1. Gambar dibagi menjadi *domain-domain* blok dengan ukuran 16x16 *pixel*.
2. Pilih *range* blok dengan ukuran 8x8 *pixel*.
3. Pilih pasangan *range-domain* yang terbaik, dengan membandingkan setiap *rms_error* yang didapat dengan toleransi yang ditentukan untuk setiap *domaini* blok.
4. Bila *rms_error* toleransi belum terpenuhi, tiap *range* blok dibagi lagi menjadi empat *sub range* kemudian dilakukan pencarian pasangan *range-domain* yang terbaik untuk setiap *domain* yang ada.
5. Bila *rms_error* yang terbaik sudah ditemukan, kemudian disimpan sebagai hasil kompresi.

3.2. Proses Dekompresi

Pada proses dekompresi ini langkah-langkahnya adalah sebagai berikut:

1. Membaca transformasi *affine* dari *file* gambar terkompresi
2. Membuat *memory bufferi* untuk *domain* dari *range screen* (alokasi *memory*)
3. Membuat inisialisasi *screen buffer* dari *range* ke sebuah gambar inisial.
4. Dilakukan pengujian untuk tiap *domain* blok.
5. Mengganti *domain* dengan data hasil transformasi dari *range*, yang disiapkan dengan menggunakan parameter *affine* yang disimpan untuk *domain* dan dilakukan untuk setiap *domain* blok.
6. Hal ini dilakukan untuk beberapa iterasi sehingga didapatkan gambar keluaran yang baik.

3.3. Hasil Simulasi

Hasil simulasi untuk *test image* "Lenna" dengan ukuran 256 x 256 *pixels* atau 65.536 *bytes* untuk nilai *quadtree partioning* 6 dan 8 untuk berbagai nilai toleransi ditunjukan pada Tabel 1. Hasil simulasi untuk PSNR vs *quadtree partioning* dapat dilihat pada Tabel 2. Dari hasil simulasi tersebut dapat dilihat bahwa semakin tinggi tingkat quadtree partioning semakin kecil nilai kompresi rationya. Nilai toleransi semakin tinggi akan menghasilkan nilai PSNR yang tinggi pula, akan tetapi ada suatu batasan atau *trade-off* untuk nilai toleransi.

Tabel 1 Hasil simulasi *test image* "Lenna" untuk nilai toleransi dan *partitioning*.

	Partioning = 6		Partioning = 8	
Toleransi	Bytes terkompresi	Ratio Kompresi	Bytes terkompresi	Ratio Kompresi
1	10.709	6,12	39.276	1,67
2	8.498	7,71	27.241	2,40
3	7.150	9,16	20.492	3,20
4	6.537	10,02	17.107	3,83
4	6.269	10,45	15.429	4,25
6	6.039	10,85	14.329	4,57
7	5.831	11,24	13.880	4,72
8	5.476	11,96	13.581	4,82
9	5.081	12,90	13.342	4,91
10	4.765	13,75	13.189	4,97

Tabel 2 Hasil simulasi Toleransi vs PSNR untuk masing-masing *Partition Quadtree*.

Toleransi	PSNR [dB] PQ = 6	PSNR [dB] PQ = 8
1	35,55	20,32
2	36,73	26,14
3	38,11	29,31
4	39,10	32,39
5	39,77	35,04
6	40,18	36,67
7	40,35	37,63
8	40,43	38,40
9	40,50	38,40
10	40,51	38,40

4. Kesimpulan

Berdasarkan hasil uji coba dan analisa yang dilakukan, maka dapat disusun suatu kesimpulan sebagai berikut:

1. Ukuran berkas gambar terkompresi dengan menggunakan metode fraktal ini tergantung dari parameter Toleransi *(t)* dan tingkat *quadtree partitioning*. Semakin besar nilai toleransi, ukuran gambar terkompresi akan semakin kecil, dan semakin tinggi tingkat *quadtree partitioning* yang digunakan maka ukuran *file* terkompres yang didapatkan akan semakin besar. Berdasarkan hasil simulasi diperoleh untuk nilai toleransi 5 sampai 10 untuk tingkat *quadtree partitioning* = 6 akan menghasilkan ratio kompresi yang baik dan nilai PSNR yang masih dapat diterima oleh mata manusia. Sedangkan untuk nilai toleransi 5 sampai 10 untuk tingkat *quadtree partitioning* = 8 tidak memberikan nilai kompresi yang berarti.
2. Pada proses dekompresi, gambar yang dihasilkan tergantung dari parameter toleransi yang digunakan. Semakin besar jumlah toleransi yang digunakan, semakin baik gambar yang dihasilkan.

Referensi

[1] Yuval Fisher, "Fractal Image Compressor", *SIGGRAPH COURSE NOTES* 1992.

[2] Arnaud E. Jacquin, "Fractal Image Coding: A Review", *Proceeding of the IEE*, vol. 81, no. 10 pp.1451-1465, Oktober 1993.

[3] Shinhaeng Lee, "*Pararel Processing Architecture For Fractal Image Processing*", Department of Electrical and Communication Engineering, Tohoku University, Japan. 1995

[4] L. F. Anson, "Fractal Image Compression", *Byte*, pp.195-202, Oktober 1993.

[5] Carsten Frigaard, Jess Gade, Thomas T. Hemmingsen and Torben Sand, "*Image Compression Based on Fractal Theory*", Institute for Electronic Systems, Aalborg University, Denmark. Januari 1994.

[6] John Kominek, "*Advances in Fractal Compression For Multimedia Application*", University of Waterloo Technical Report, 1995.