KOMPRESI CITRA MENGGUNAKAN TRANSFORMASI WAVELET

Yuyun Wahyuni Abasi[1], Yeffry Handoko Putra[2], Mira Kania Sabariah[3]
[1)] Jurusan Teknik Informatika (10199090)
[2)] Dosen Jurusan Teknik Komputer
[3)] Dosen Jurusan Teknik Informatika
Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia

ABSTRAK

Sebagai bentuk representasi data, kendala yang dihadapi ketika menggunakan citra digital adalah pada besarnya volume data yang dibutuhkan untuk merepresentasikan citra tersebut. Untuk itu dibutuhkan suatu teknik yang dapat mengecilkan volume data, teknik ini dinamakan kompresi. Pada skripsi ini dipilih metode transformasi *wavelet* sebagai salah satu metode yang bisa digunakan untuk pemampatan data citra. *Wavelet* yang digunakan adalah salah satu *wavelet* yang sangat terkenal yaitu *Daubechies D4 wavelet transform*, dengan empat koefisien *scaling function*, dan empat koefisien *wavelet function*. Selanjutnya diimplementasikan dengan menggunakan perangkat lunak MATLAB 6.0 sebagai alat bantu pemrograman untuk melihat pengaruh kualitas pada citra yang ditransformasi.

I. PENDAHULUAN

Salah satu metode pemampatan data citra adalah metode transformasi. Sekitar 20 tahun yang lalu, suatu "alat" ampuh yang dapat dipakai untuk memroses sinyal (suara ataupun citra) telah ditemukan. Alat tersebut dikenal dengan nama *wavelet* (secara harfiah berarti "gelombang kecil") yang menyaingi alat lama yakni transformasi Fourier dengan berbagai modifikasinya. Dengan ditemukannya *wavelet*, dunia pemrosesan sinyal dan teknologi digital berkembang kian pesat.

Tugas akhir ini meninjau metoda pemampatan citra dengan menggunakan *wavelet transform*. Kemampuan *wavelet* dalam merepresentasikan sinyal dalam waktu-frekuensi sekaligus dimanfaatkan untuk tujuan pengkodean dan pemampatan data citra. sebagai induk *wavelet* memiliki sub-sub metode yang tergabung dalam *wavelet family*, diantaranya adalah Daubechies D4 *wavelet transform*.

II. LANDASAN TEORI

2.1 Daubechies D4 Wavelet Transform dan Inverse-nya

Filter wavelet yang dipilih adalah salah satu *wavelet* yang sangat terkenal yaitu **Daubechies**. Daubechies adalah *filter wavelet* yang optimum digunakan untuk pemampatan data citra. Daubechies *wavelet transform* dinamakan menurut pencipta / penemunya, yaitu seorang ahli matematika bernama **Ingrid Daubechies**.

Fungsi *wavelet* Daubechies D4 yang digunakan ini dinyatakan dalam bentuk matriks yang memiliki empat koefisien *scaling function*, dan empat koefisien *wavelet function*.
Koefisien *scaling function* adalah :

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}},\quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}},\quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}},\quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

Koefisien *wavelet function* :

$$g_0 = h_3,\quad g_1 = -h_2,\quad g_2 = h_1,\quad g_3 = -h_0$$

nilai-nilainya ditunjukkan pada transform matrix berikut :

$$\begin{bmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & h_0 & h_1 \\ g_2 & g_3 & 0 & 0 & 0 & 0 & g_0 & g_1 \end{bmatrix}$$

Gambar 2.1. Daubechies D4 Wavelet Transform Matrix

$$\begin{bmatrix} h_2 & g_2 & h_0 & g_0 & 0 & 0 & 0 & 0 \\ h_3 & g_3 & h_1 & g_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2 & g_2 & h_0 & g_0 & 0 & 0 \\ 0 & 0 & h_3 & g_3 & h_1 & g_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_2 & g_2 & h_0 & g_0 \\ 0 & 0 & 0 & 0 & h_3 & g_3 & h_1 & g_1 \\ h_0 & g_0 & 0 & 0 & 0 & 0 & h_2 & g_2 \\ h_1 & g_3 & 0 & 0 & 0 & 0 & h_3 & g_3 \end{bmatrix}$$

Gambar 2.1. Daubechies D4 Inverse Wavelet Transform Matrix

2.2. Citra digital

Citra adalah suatu fungsi dua dimensi yang terbentuk dari suatu penglihatan dalam suatu *scene*, misalnya foto. *Scene* adalah basis dari citra. Kita mengenal dua perbedaan citra yaitu citra analog dan citra digital. Citra analog merupakan citra hasil "tangkapan" lensa kamera terhadap *scene*, dan di-*scan* vertikal dan horisontal oleh *imager* kamera. Citra digital direpresentasikan sebagai sebuah matriks yang masing-masing elemennya merepresentasikan nilai intensitas. Jika I adalah matriks dua dimensi, I(x,y) adalah nilai intensitas yang sesuai pada posisi baris x dan kolom y pada matriks tersebut. Titik-titik di tempat *image* di-*sampling* disebut *picture elements*, atau sering dikenal sebagai piksel *(pixel)* atau dot.

2.3. RGB dan YCbCr

Citra digital dengan kedalaman warna 24 bit atau biasa disebut *truecolor image* merupakan kombinasi dari tiga komponen warna RGB ; R *(Red* / merah), G (*Green* / hijau), dan B (*Blue* / biru). Masing-masing komponen terdiri dari 8 bit, sehingga untuk 3 komponen warna menggunakan 24 bit atau sama dengan 3 *byte* per piksel.

Komponen warna YCbCr sebenarnya lebih luas dikenal dalam dunia video digital. Pada dasarnya ini merupakan pemisahan warna berdasarkan komponen kecerahan atau *luminance*, dan pemisahan berdasarkan komponen warna atau *chrominance*. Dalam format ini, komponen kecerahan *luminance* disimpan sebagai satu komponen Y, dan komponen warna *chrominance* disimpan sebagai dua komponen yang berbeda yaitu Cb dan Cr. [8].

Konversi warna RGB ke warna YCbCr adalah untuk mendapatkan komponen Y atau lebih dikenal dengan *grayscale* (abu-abu) yaitu citra dengan derajat keabuan dari 0 hingga 255, dimana 0 untuk merepresentasikan warna hitam dan 255 untuk merepresentasikan warna putih.

Karena mata manusia lebih peka terhadap warna *luminance* (Y) dari pada warna *chrominance* (Cb dan Cr), maka komponen Cb dan Cr tidak diikutsertakan dalam proses kompresi. Jadi komponen Y adalah citra *grayscale* yang akan dikompres, selanjutnya komponen Cb dan Cr akan diikutkan kembali pada proses dekompresi untuk membentuk kembali warna RGB. Warna YCbCr dapat diperoleh dengan mentransformasikan RGB, di dalam MATLAB perintah untuk mengkonversi warna dari RGB ke YCbCr adalah :

```
% convert RGB image to YCbCr image
citraYCbCr =  rgb2ycbcr(citra);
```

Untuk membentuk kembali warna RGB menggunakan :

```
% convert YCbCr to RGB
citraRGB = ycbcr2rgb(citraYCbCr);
```

2.4. Thresholding

Thresholding atau penentuan batas, merupakan salah satu teknik untuk memroses pengurangan jumlah bit yang diperlukan untuk menyimpan data gambar. Proses ini diterapkan pada keluaran hasil transformasi. Langkah yang dilakukan adalah dengan menentukan suatu nilai (batas) yang disebut *'threshold'*, nilai ini yang nantinya dipakai untuk menentukan koefisien yang akan diambil dari matriks hasil transformasi. Misalnya teknik-A, teknik-B, dan teknik-C adalah 3 teknik yang digunakan.

1. **Teknik-A** menggunakan 2 nilai *threshold* yang didapatkan dari pemisahan *threshold* pada baris ganjil dan *threshold* pada baris genap, persamaannya :

$$thr1 = \frac{\text{Nilai-nilai pada baris ganjil}}{\text{Jumlah elemen baris ganjil}}$$

$$thr2 = \frac{\text{Nilai-nilai pada baris genap}}{\text{Jumlah elemen baris genap}}$$

keterangan :
thr1 = nilai threshold baris ganjil
thr2 = nilai threshold baris genap

Nilai *threshold* baris ganjil merupakan rata-rata dari elemen pada baris ganjil, dan *threshold* baris genap adalah rata-rata nilai dari elemen pada baris genap. Nilai-nilai pada baris ganjil yang lebih kecil dari *threshold* baris ganjil dianggap nol, dan nilai pada baris genap yang lebih kecil dari *threshold* baris genap, juga dinolkan. Sehingga menghasilkan matriks baru yang berisi nilai-nilai terbesar di atas rata-rata.

2. Pada **teknik-B** nilai *threshold* yang digunakan hanya 1 nilai, yang merupakan nilai rata-rata dari setiap elemen pada matriks, dengan persamaan:

$$Threshold = \frac{Jumlah}{m * n}$$

keterangan :
jumlah = hasil penjumlahan semua elemen pada matriks.
m * n = baris kali kolom.

Misalnya A = matriks hasil transformasi 8*8

$$A = \begin{bmatrix} 230 & 230 & 230 & 230 & 230 & 230 & 230 & 230 \\ 6 & 6 & 6 & 6 & 6 & 6 & 5 & 6 \\ 314 & 320 & 316 & 316 & 318 & 318 & 319 & 308 \\ 17 & 35 & 26 & 24 & 30 & 29 & 28 & -3 \\ 353 & 316 & 329 & 333 & 321 & 319 & 297 & 279 \\ 19 & 4 & -11 & -12 & -9 & -5 & -11 & -13 \\ 213 & 215 & 242 & 279 & 307 & 322 & 297 & 280 \\ 10 & 9 & 4 & 14 & 25 & 62 & 5 & 9 \end{bmatrix}$$

Jumlah = elemen (1,1) + elemen (1,2) + … + elemen n = 9404

m * n = 64

Threshold = 146,93

3. Untuk **teknik-C** menggunakan satu teknik dengan memanfaatkan hasil transformasi *wavelet* yaitu nilai pada baris ganjil adalah besar dan nilai-nilai detail yang terletak pada baris genap sangat kecil. Jadi nilai pada baris genap semuanya dibuat nol. Sehingga menghasilkan matriks dengan nilai nol berurutan pada baris-baris genap.

$$\begin{array}{l} \text{Baris 1} \\ \text{Baris 2} \\ \\ \\ \\ \\ \\ \text{Baris 8} \end{array} \begin{bmatrix} 230 & 230 & 230 & 230 & 230 & 230 & 230 & 230 \\ 6 & 6 & 6 & 6 & 6 & 6 & 5 & 6 \\ 314 & 320 & 316 & 316 & 318 & 318 & 319 & 308 \\ 17 & 35 & 26 & 24 & 30 & 29 & 28 & -3 \\ 353 & 316 & 329 & 333 & 321 & 319 & 297 & 279 \\ 19 & 4 & -11 & -12 & -9 & -5 & -11 & -13 \\ 213 & 215 & 242 & 279 & 307 & 322 & 297 & 280 \\ 10 & 9 & 4 & 14 & 25 & 62 & 5 & 9 \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} 230 & 230 & 230 & 230 & 230 & 230 & 230 & 230 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 314 & 320 & 316 & 316 & 318 & 318 & 319 & 308 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 353 & 316 & 329 & 333 & 321 & 319 & 297 & 279 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 213 & 215 & 242 & 279 & 307 & 322 & 297 & 280 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.5. Encoding dan Decoding

Encoding atau pengkodean digunakan untuk merepresentasikan data dalam bentuk lain sehingga kuantitas data dengan representasi yang baru ini lebih sedikit apabila dibandingkan dengan kuantitas data aslinya, dan proses *decoding* (dekode) adalah pembalikan dari *encoding*.

Prinsip pengkodean yang digunakan adalah ***RLE*** *(Run Length Encoding)* yaitu proses serangkaian simbol yang berurutan dikodekan menjadi sebuah kode yang terdiri dari simbol tersebut dan jumlah perulangannya. [4, 7].
Contoh hasil *thresholding* pada potongan citra 8*8 diuraikan menjadi vektor baris ukuran 1 * (m*n) :

230 230 230 230 230 230 230 230 0 0 0 0 0 0 0 0 314 320 316 316 318 318 319 308 0 0 0 0 0 0 0 0 353 316 329 333 321 319 297 279 0 0 0 0 0 0 0 0 213 215 242 279 307 322 297 280 0 0 0 0 0 0 0 0

Hasil ini memudahkan proses *RLE* karena banyak nilai sama yang berurutan, sehingga apabila dikodekan, bit yang diperlukan untuk menyimpan data tersebut lebih sedikit. Data dibaca sebagai nilai dan jumlah perulangannya. Dikodekan menjadi :

230 8 0 8 314 320 316 2 318 2 319 308 0 8 353 316 329 333 321 319 297 279 0 8 213 215 242 279 307 322 297 280 0 8

III. PERANCANGAN SISTEM

3.1. Metode Perancangan

Metode-metode yang digunakan di dalam sistem untuk kompresi dan dekompresi :
1. Konversi RGB ke YCbCr dan pembalikannya
2. Wavelet Transform dan Inverse Wavelet Transform
3. Thresholding
4. Pengkodean : RLE, scan lines
5. Dekode : De-RLE, De-scanning

3.2. Gambaran Umum Sistem

Secara umum lima proses utama yang dilakukan di dalam sistem adalah :

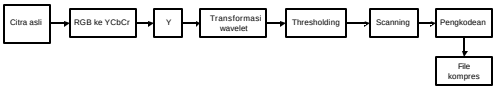


Gambar 3.1. Bagan Umum Sistem

Lebih rincinya tahap pengolahan ini dibagi dua, yaitu kompresi dan dekompresi. Proses utama yang dilakukan sistem adalah mengolah citra masukan menjadi *file* kompres, kemudian mengembalikan lagi ke representasi semula dengan proses dekompresi.
a. Tahap kompresi
b. Tahap dekompresi

3.2.1.Tahap Kompresi

Proses kompresi dilakukan dengan mentransformasikan citra masukan. Diagram kompresinya adalah sebagai berikut :



Gambar 3.2. Tahap Kompresi

Langkah - langkah kompresi :
1. Citra masukan adalah citra RGB dengan format *.bmp
2. Citra dikonversi ke bentuk YCbCr
3. Proses transformasi wavelet
4. Proses thresholding
5. Proses scanning
6. Proses pengkodean
7. File disimpan sebagai file kompres

3.2.1.1.Citra masukan

Gambar 3.2 mengilustrasikan aliran proses kompresi di dalam sistem. Langkah pertama adalah citra dengan format *.bmp *(uncompressed image)* sebagai citra masukan yang akan diproses. Citra ini adalah citra RGB dengan 3 komponen warna R *(Red)*, G *(Green)*, dan B *(Blue)*.

Di dalam program menggunakan *function* imread untuk membaca file grafik :

```
>> citra = imread('colour.bmp');
%citra adalah nama variabel untuk
%menyimpan matriks citra yang dibaca,
%sebagai contoh citra masukan adalah
%'colour.bmp'.
```

3.2.1.2.Konversi RGB ke YCbCr

Di dalam program menggunakan perintah :

```
%convert RGB image to YCbCr image
>> citraYCbCr = rgb2ycbcr(citra)
%hasil konversi berupa matriks dengan 3
%komponen warna Y, Cb, dan Cr.
```

3.2.1.3.Transformasi wavelet

Pada proses transformasi *wavelet*, dilakukan operasi perkalian matriks koefisien Daubechies D4 dengan matriks citra *gray* (komponen Y). Karakteristik dari Daubechies D4 *wavelet transform* yaitu memiliki empat nilai koefisien *scaling function* yaitu h0, h1, h2, h3 dan empat koefisien *wavelet function* g0, g1, g2 ,g3.
h0 = 0.4830, h1= 0.8365, h2 = 0.2241, h3 = -0.1294
g0 = -0.1294, g1 = -0.2241, g2 = 0.8365, g3 = -0.4830

$$\begin{bmatrix} 0.4830 & 0.8365 & 0.2241 & -0.1294 & 0 & 0 & 0 & 0 \\ -0.1294 & -0.2241 & 0.8365 & -0.4830 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4830 & 0.8365 & 0.2241 & -0.1294 & 0 & 0 \\ 0 & 0 & -0.1294 & -0.2241 & 0.8365 & -0.4830 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4830 & 0.8365 & 0.2241 & -0.1294 \\ 0 & 0 & 0 & 0 & -0.1294 & -0.2241 & 0.8365 & -0.4830 \\ 0.2241 & -0.1294 & 0 & 0 & 0 & 0 & 0.4830 & 0.8365 \\ 0.8365 & -0.4830 & 0 & 0 & 0 & 0 & -0.1294 & -0.2241 \end{bmatrix} * \begin{bmatrix} 216 & 216 & 216 & 216 & 216 & 216 & 216 & 216 \\ 130 & 130 & 130 & 130 & 130 & 130 & 130 & 130 \\ 206 & 206 & 206 & 206 & 206 & 206 & 206 & 206 \\ 226 & 227 & 226 & 226 & 227 & 227 & 228 & 226 \\ 253 & 253 & 253 & 253 & 253 & 250 & 240 & 203 \\ 242 & 205 & 224 & 229 & 215 & 213 & 196 & 198 \\ 196 & 177 & 181 & 197 & 210 & 215 & 200 & 186 \\ 121 & 135 & 164 & 193 & 214 & 216 & 214 & 202 \end{bmatrix}$$

Gambar 3.10. Perkalian Matriks Wavelet dengan Matriks Citra

Di dalam program digunakan perintah :

```
>> hasil = matriks * Y
%matriks adalah nama variabel yang
%menyimpan matriks transformasi wavelet
%dan Y adalah citra grayscale hasil
%konversi dari RGB.
```

Matriks keluarannya :

$$\begin{bmatrix} 230 & 230 & 230 & 230 & 230 & 230 & 230 & 230 \\ 6 & 6 & 6 & 6 & 6 & 6 & 5 & 6 \\ 314 & 320 & 316 & 316 & 318 & 318 & 319 & 308 \\ 17 & 35 & 26 & 24 & 30 & 29 & 28 & -3 \\ 353 & 316 & 329 & 333 & 321 & 319 & 297 & 279 \\ 19 & 4 & -11 & -12 & -9 & -5 & -11 & -13 \\ 213 & 215 & 242 & 279 & 307 & 322 & 297 & 280 \\ 10 & 9 & 4 & 14 & 25 & 62 & 5 & 9 \end{bmatrix}$$

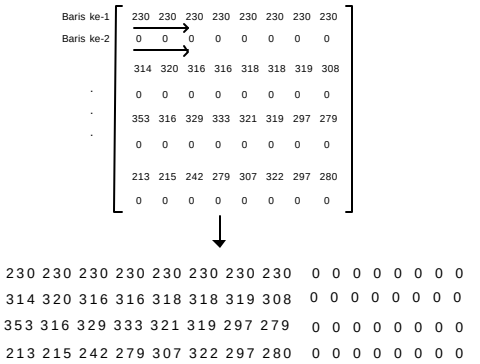
Gambar 3.11. Matriks Keluaran Proses Transformasi

3.2.1.4.Thresholding

Proses *thresholding* dilakukan terhadap citra hasil proses transformasi. Proses ini adalah untuk mengurangi jumlah data yang disimpan sehingga dapat menghemat bit untuk penyimpanan atau transmisi. Teknik-A, teknik-B, dan teknik-C Ketiga teknik digunakan di dalam sistem untuk melihat perbandingan kualitas citra hasil dekompresi.

3.2.1.5.Scan lines

Proses berikut adalah *scan lines*, atau pembacaan secara baris perbaris, mulai dari elemen pada pojok kiri atas hingga elemen pada pojok kanan bawah. [4]. Penerapannya adalah hasil *thresholding* dibaca perbaris mulai baris 1, 2, ... sampai baris terakhir, lalu ditulis sebagai vektor ukuran 1 * (m*n) atau matriks satu baris.



Gambar 3.15. Scanning

3.2.1.6.Pengkodean

Proses selanjutnya yaitu pengkodean menggunakan prinsip *RLE (Run Length Encoding)*. Pengkodean digunakan untuk merepresentasikan data dalam representasi yang lain sehingga kuantitas data dengan representasi yang baru ini lebih sedikit apabila dibandingkan dengan kuantitas data aslinya. Sebaliknya proses *decoding* adalah mengembalikan data hasil *encoding* ke representasi yang semula.

Tabel 3.1. Pengkodean 1

Simbol	Nilai	Frekuensi
1	230	8
2	0	8
3	314	1
4	320	1
5	316	2
6	318	2
7	319	1
8	308	1
9	0	8
10	353	1
11	316	1
12	329	1
13	333	1
14	321	1
15	319	1
16	297	1
17	279	1
18	0	8
19	213	1
20	215	1
21	242	1
22	279	1
23	307	1
24	322	1
25	297	1
26	280	1
27	0	8

Nilai = [230 0 314 320 316 318 319 308 0 353 316 329 333 321 319 297 279 0 213 215 242 279 307 322 297 280 0] size = 1 x 27
Frekuensi = [8 8 1 1 2 2 1 1 8 1 1 1 1 1 1 1 1 8 1 1 1 1 1 1 1 8] size = 1 x 27

Jika data citra berukuran 8 x8 dengan 8 bit per piksel tersimpan sebanyak 64 *byte*, dikompres menjadi (1 x 27) + (1 x 27) = 54 *byte*, penyusutannya tidak terlalu signifikan. Pada tabel terlihat bahwa nilai dengan frekuensi 1 kali, jumlahnya lebih banyak daripada nilai yang berulang lebih dari 1 kali. Jika variabel nilai dan frekuensi disimpan maka pemampatan kurang efektif, karena jumlah data yang disimpan masih relatif banyak.

Untuk itu dibuat bentuk yang lebih ringkas dari kode tersebut, yaitu menyimpan simbol nilai yang berulang lebih dari 1 kali dan frekuensinya.

Tabel 3.1. Pengkodean 2

Simbol	Frekuensi
1	8
2	8
5	2
6	2
9	8
18	8
27	8

- cek simbol dengan frekuensi lebih dari 1.
- tulis simbol dan frekuensinya.

Kode = [1 8 2 8 5 2 6 2 9 8 18 8 27 8] size = 1 x 14
Nilai = [230 0 314 320 316 318 319 308 0 353 316 329 333 321 319 297 279 0 213 215 242 279 307 322 297 280 0] size = 1 x 27

Jadi untuk data citra ukuran 8 x 8 adalah 64 *byte*, setelah dikompres kemudian disimpan menjadi (1 x 14) + (1 x 27) = 41 *byte*. Data yang disimpan sebagai file kompres adalah elemen yang terdapat pada variabel kode dan nilai. Bentuk ini lebih sedikit membutuhkan ruang penyimpanan, karena jumlah data yang disimpan sudah berkurang.

Pada citra RGB data yang ditulis ke *file* adalah; kode, nilai, tinggi dan lebar gambar, komponen Cb dan Cr. Informasi ini diperlukan untuk proses dekompresi, sehingga dari *file* kompres tersebut dapat menghasilkan citra rekonstruksi yang baru.

3.2.2.Tahap Dekompresi

Proses dekompresi adalah tahap pembalikan dari kompresi. Diagram dekompresi digambarkan sebagai berikut :



Gambar 3.3. Tahap Dekompresi

Langkah - langkah dekompresi :
1. Load file kompres
2. dekode
3. de-scanning
4. Inverse wavelet transform
5. Konversi bentuk YCbCr ke RGB
6. Citra hasil rekonstruksi

3.2.2.1.Load file kompres

Pada proses dekompresi, kode pendek yang tersimpan sebagai *file* kompres adalah sebagai inputan untuk di-dekodekan. Perintah yang digunakan :

```
%load <filename.MAT>
%function load digunakan untuk
%memanggil file masukan,diikuti nama
%file.
```

Variabel yang ditampilkan adalah nilai-nilai yang dikodekan menjadi bentuk ringkas.

Kode = [1 8 2 8 5 2 6 2 9 8 18 8 27 8]
Nilai = [230 0 314 320 316 318 319 308 0 353 316 329 333 321 319 297 279 0 213 215 242 279 307 322 297 280 0]

3.2.2.2.Proses dekode

Pada proses dekode, nilai-nilai yang disimpan sebagai *file* kompres diuraikan kembali menjadi kode yang panjang. Elemen-elemen pada variabel kode mengandung informasi yang diperlukan untuk dekode, simbol ke-1 menunjuk pada variabel nilai dan ditulis sebanyak nilai sesudahnya, simbol ke-2 ditulis sebanyak nilai sesudahnya, dan seterusnya.

Kode = [1 8 2 8 5 2 6 2 9 8 18 8 27 8]

Nilai = [230 0 314 320 316 318 319 308 0 353 316 329 333 321 319 297 279 0 213 215 242 279 307 322 297 280 0]

Sehingga menjadi bentuk kode panjang kembali :

230 230 230 230 230 230 230 230 0 0 0 0 0 0 0 0
314 320 316 316 318 318 319 308 0 0 0 0 0 0 0 0
353 316 329 333 321 319 297 279 0 0 0 0 0 0 0 0
213 215 242 279 307 322 297 280 0 0 0 0 0 0 0 0

3.2.2.3.Proses de-scanning

Pada proses de-scanning, hasil dekode dibaca kemudian dipetakan ke dalam susunan matriks seperti kondisi sebelum *scanning*.

230	230	230	230	230	230	230	230
0	0	0	0	0	0	0	0
314	320	316	316	318	318	319	308
0	0	0	0	0	0	0	0
353	316	329	333	321	319	297	279
0	0	0	0	0	0	0	0
213	215	242	279	307	322	297	280
0	0	0	0	0	0	0	0

Gambar 3.17. Matriks Hasil De-scanning

3.2.2.4.Inverse wavelet transform

Tahap berikutnya adalah mengembalikan hasil transformasi untuk merekonstruksi citra *grayscale* (komponen Y) yang diproses, dengan *inverse wavelet transform*. Sebelumnya matriks *wavelet* diinversikan terlebih dahulu dengan menggunakan *function* `inv`.

```
>> mat_inv = inv(matriks);
% mat_inv adalah nama variabel untuk
% memuat matriks inverse wavelet
% transform.
```

Matriks *inverse* ini dikalikan dengan matriks de-*scanning* untuk mendapatkan komponen Y yang baru.

0.2241	0.8365	0.4830	-0.1294	0	0	0	0
-0.1294	-0.4830	0.8365	-0.2241	0	0	0	0
0	0	0.2241	0.8365	0.4830	-0.1294	0	0
0	0	-0.1294	-0.4830	0.8365	-0.2241	0	0
0	0	0	0	0.2241	0.8365	0.4830	-0.1294
0	0	0	0	-0.1294	-0.4830	0.8365	-0.2241
0.4830	-0.1294	0	0	0	0	0.2241	0.8365
0.8365	-0.2241	0	0	0	0	-0.1294	-0.4830

*

230	230	230	230	230	230	230	230
0	0	0	0	0	0	0	0
314	320	316	316	318	318	319	308
0	0	0	0	0	0	0	0
353	316	329	333	321	319	297	279
0	0	0	0	0	0	0	0
213	215	242	279	307	322	297	280
0	0	0	0	0	0	0	0

Gambar 3.18. Perkalian Matriks Inverse Wavelet dengan Matriks De-Scanning

Menghasilkan matriks :

203	206	204	204	205	205	206	200
233	238	235	235	236	236	237	228
241	224	230	232	226	225	215	204
255	223	234	238	227	226	207	194
182	175	191	209	220	227	210	198
132	139	160	190	215	228	210	198
159	159	165	174	180	183	178	174
165	165	161	156	153	151	154	156

Gambar 3.19. Matriks Keluaran Hasil Rekonstruksi

Matrik hasil ini selanjutnya akan digabungkan lagi dengan komponen Cb dan Cr menggunakan *function* `cat`.

```
>> YCbCrnew = cat(3,Y,Cb,Cr);
```

Langkah terakhir adalah konversi ke bentuk RGB.

3.2.2.5.Konversi YCbCr ke RGB

Langkah terakhir proses dekompresi adalah mengembalikan lagi komponen warna YCbCr (dengan komponen Y yang baru) menjadi warna RGB. Untuk membentuk kembali warna RGB menggunakan *function* : `ycbcr2rgb`.

Variabel RGB sekarang membentuk citra yang baru atau citra rekonstruksi. Hasil ini kemudian ditulis ke *file* dengan *function* `imwrite`, yaitu menulis ke *file image* grafik.

```
% imwrite(var,'filename.ext');
%var adalah nama variabel yang akan
%ditulis, diikuti nama file citra dan
%ekstension-nya, yaitu berupa bmp image.
>> imwrite(RGB,'colournew.bmp')
```

Citra baru ini adalah hasil dekompresi yang sangat mendekati citra asli sebelum kompresi.

IV. IMPLEMENTASI SISTEM

4.1. Proses Kompresi

Setelah pada tahap perancangan, interaksi yang dilakukan adalah dengan jendela *M-file* editor, maka pada proses kompilasi, *user* lebih interaktif dengan *Command Window*, jadi semua eksekusi dijalankan pada jendela *Command Window MATLAB 6.0*.

Panggil nama file kompresi untuk *running* program :

```
>> run kompresi_A
```

4.2. Proses Dekompresi

Untuk proses dekompresi, mengembalikan citra ke bentuk semula atau mendekati bentuk asli. *M-file* untuk dekompresi dijalankan pada *Command Window* dengan memanggil nama file : `>> dekompresi_A`

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada intinya proses dari sebuah kompresi data adalah pemodelan dan pengkodean. Jika pemilihan metode tepat disertai teknik pembuatan sistem yang baik, maka bobot kompresi yang dihasilkan akan efektif. Semakin kompleks teknik dan metode yang digunakan akan semakin rumit pula pembuatan sistem. Proses pembelajaran serta ketelitian menjadi faktor penting untuk menghasilkan kompresi efektif, juga pada saat memutuskan alat bantu perangkat lunak apa yang akan digunakan.

Beberapa hal yang bisa disimpulkan dari laporan ini :

1. transformasi wavelet dengan Daubechies D4 sangat baik dalam hal pemrosesan citra, karena citra hasil rekonstruksi setelah didekompresi sangat mirip citra asli meskipun sudah mengalami proses *thresholding* di dalamnya.
2. teknik-B menghasilkan file kompres terkecil dibanding dua teknik lain.
3. teknik-A menghasilkan kualitas citra rekonstruksi yang paling baik diantara ketiga teknik

5.2 Saran

Program yang dihasilkan akan lebih kompleks dan interaktif jika dikembangkan dengan membuat *GUI (Graphical User Interface)* -nya. Untuk melihat demo-demo *image processing* lainnya yang menggunakan *wavelet* dalam MATLAB, bisa menggunakan fasilitas *help reference*.

Dua hal yang penulis catat adalah masalah kompresi dan pengolahan citra. Bagi yang memiliki minat pada problematika *data compression*, maka sebagian prinsip kompresi yang digunakan dalam skripsi ini bisa dikembangkan untuk kompresi data selain citra *(image)* misalnya data teks, video, audio, dan lain-lain. Karena pada dasarnya kompresi data sangat berkaitan dengan ukuran penyimpanan.

Untuk pengolahan citra digital *(image processing)*, banyak hal yang dapat dilakukan, apalagi begitu banyak kemajuan dalam bidang grafika komputer. Program seperti ini dapat juga dikembangkan dengan menggunakan perangkat lunak yang lebih *user friendly*, untuk memroses perbaikan citra *(image restoration)*, pengenalan pinggiran *(edge)*, penghalusan citra, *blurring effect*, penajaman citra, peningkatan kualitas citra, dan masih banyak lagi.

DAFTAR PUSTAKA

[1] Burrus C. S., Gopinath R. A., and Guo H., *Introduction to Wavelets and Wavelet Transforms*, Prentice-Hall, Inc., 1998.

[2] Etter D. M., Kuncicky D. C. with Hull D., *Pengantar MATLAB 6*, PT Indeks Kelompok Gramedia, Jakarta, 2003.

[3] Hanselman D., Littlefield B., *MATLAB Bahasa Komputasi Teknis*, Andi, Yogyakarta, 2001.

[4] Salomon D., *Data Compression 2nd Edition : The Complete Reference*, 1998 Springer-Verlag New York, 2000.

[5] Strang G., Nguyen T., *Wavelets and Filter Banks*, Wellesley-Cambridge Press, 1996.

[6] Sutopo A. H., *Pengantar Grafika Komputer*, Gava Media, Yogyakarta, 2002.

[7] Yonata Y., *Singkat Tepat Jelas Kompresi Video*, PT Elex Media Komputindo, Jakarta, 2002.

[8] Panduan *Help*, Mathworks, MATLAB R12 6.0, 2000.

[9] http://www.cs.ui.ac.id/kuliah/citra/2002/kompresi2.pdf

[10] http://dsp.ee.itb.ac.id/artikel/paper-digitalmmediapdf

[11] http://www.bearcave.com/misl/misl_tech/wavelets /daubechies/

[12] http://www.kompas.com/kompas-cetak/0207/26/ dikbud/diba34.htm