BAB II

LANDASAN TEORI

2.1 Kompresi Data

Kompresi data bukan merupakan cabang ilmu komputer yang ada semenjak komputer pertama kali diciptakan. Metode ini muncul ketika dunia komputer telah berkembang sedemikian rupa sehingga mampu menangani sumber-sumber data yang berukuran besar. Karena hardware yang digunakan untuk menangani pemrosesan data kurang mampu mengikuti perkembangan jumlah data yang demikian besar, maka diperlukan suatu metode yang dapat mengolah data sedemikian rupa sehingga data tersebut dapat disimpan atau ditransfer secara lebih cepat dengan seminimal mungkin jumlah data yang hilang atau berubah.

Kompresi data adalah proses mengubah suatu *input* data menjadi data lain dengan format berbeda dan ukuran yang lebih kecil (Solomon, 2000). Suatu definisi menurut *Scorer* : "Kompresi data adalah proses pengkodean dari suatu data untuk mengurangi kebutuhan akan media penyimpanan."

Jadi, dalam suatu proses kompresi dibutuhkan *input* berupa data (disebut juga *original data*) dalam bentuk *File*, dan setelah melalui proses pengkodean dengan algoritma penghitungan tertentu akan menghasilkan output berupa data terkompres.

[1] Salomon, David, Data Compression: The Complete Reference: Ney York: Springer Verlag Inc, 2th Edition, 2000

Dalam bidang kompresi citra dikenal dua jenis kompresi berdasarkan sifat hasil keluarannya, yaitu:

- *Lossy*

  Adalah jenis kompresi dimana ada informasi yang hilang, sehingga data sebelum dikompres dan sesudah didekompres tidak sama persis. Biasanya dipakai dalam kompresi *file* bertipe gambar atau audio. Misal : JPEG, MP3.

- *Lossless*

  Adalah jenis kompresi dimana tidak ada informasi yang hilang, sehingga data sebelum dikompres dan sesudah didekompres sama persis.

Tujuan dari kompresi data adalah untuk mengurangi kebutuhan akan media penyimpanan, sehingga *file* yang berisi data asli bisa dihapus dari media penyimpanan, dan sebagai gantinya yang disimpan adalah data yang telah terkompres, yang biasanya ukurannya lebih kecil daripada data asli.

2.2 Kompresi citra dengan metode Lossy

Yang penting dari suatu citra adalah tampilannya saat dilihat langsung oleh mata manusia. Oleh karena itu suatu citra dapat dikompres secara *lossy* dengan mengijinkan terjadinya sedikit perubahan pada warna, dimana hal ini tidak akan berpengaruh besar pada kualitas citra dan citra masih bagus saat dilihat.

Kompresi citra secara *lossy* dapat dilakukan dengan beberapa cara. Secara umum, algoritmanya adalah sebagai berikut :

1. *Block dividing*
2. Konversi RGB (*Red Green Blue* atau Merah Hijau Biru) ke YIQ (*Luminance Hue Saturation* atau kecerahan Corak warna Kejenuhan)
3. Menggunakan transformasi
4. Menggunakan kuantisasi
5. Pengkodean dengan algoritma kompresi *lossless*

2.3 Citra *Bitmap*

Citra Bitmap berasal dari bahasa Inggris *bit-mapped image* yang berarti *image* yang dipetakan ke dalam bit. Jadi citra Bitmap adalah kumpulan bit-bit yang dipetakan sedemikian rupa sehingga dapat merepresentasikan suatu citra agar dapat disimpan oleh perangkat keras komputer dalam bentuk *file*. Bit-bit dalam suatu citra Bitmap dipisah-pisahkan menjadi banyak kumpulan bit yang masing-masing mewakili satu piksel (*picture element – pixel*).

Suatu citra Bitmap memiliki 2 komponen penting yaitu resolusi dan warna. Resolusi adalah nilai baris × kolom dari matriks citra Bitmap. Semakin besar resolusi yang digunakan untuk merepresentasikan suatu citra Bitmap, semakin halus pula citra tersebut ditampilkan. Komponen kedua yaitu warna. Komponen ini sangat tergantung pada perangkat keras video (*VGA card*) yang digunakan. Untuk menampilkan setiap piksel pada layar monitor, perangkat keras harus membandingkan warna pada citra dengan kode warna dalam memori perangkat keras video.

File Bitmap dapat menampung citra yang berukuran 1, 4, 8, atau 24 bit untuk setiap pikselnya. Citra 1, 4, dan 8 bit memerlukan suatu *palette* (palet) yang berisi kode untuk tiap warna, sedangkan citra berukuran 24 bit merupakan warna dasar (*true color*)[1]. Dalam program ini, citra yang dipakai adalah yang berukuran 24 bit, dimana masing-masing piksel memiliki format warna RGB.

2.3.1 Format warna RGB

Standar bagi format warna RGB (*Red Green Blue* atau Merah Hijau Biru) ditetapkan oleh ILC (*International Lighting Commission*) pada tahun 1931. Suatu piksel yang berukuran 24 bit terdiri dari 3 × 1 *byte* (masing-masing 8 bit) yang masing-masing mewakili komponen warna *Red*, *Green*, dan *Blue* yang merupakan tiga warna primer. Nilai setiap *byte* berada pada rentang 0 - 255, dengan tipe *unsigned* (hanya bisa menerima nilai positif.).

Teknik mengkompres satu *file* citra adalah seperti mengkompres tiga citra secara terpisah. Tiap nilai RGB terlebih dahulu dikompres setiap komponennya. Pertama warna merah kemudian warna hijau dan akhirnya warna biru.

2.3.2 Format Warna YIQ

Format warna YIQ dikembangkan oleh *National Television Systems Committe* (NTSC), dengan tujuan untuk meningkatkan efisiensi dalam penyimpanan serta transmisi. Format warna YIQ terdiri dari tiga atribut warna Y, I, dan Q. Y merupakan *Luminance* (kecerahan), sedangkan I dan Q merupakan

---

[1] Agustinus Nalwan, *Pengolahan Gambar secara Digital.* Jakarta: PT Elex Media Komputindo, 1997, Hal: 116

komponen warna tambahan yang biasanya bernilai kecil. I mewakili *Hue* (corak warna) sedangkan Q mewakili *Saturation* (kejenuhan warna).

Ketiga nilai YIQ bisa diperoleh dari transformasi nilai RGB, dengan perhitungan[2] :

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \quad ...(2\text{-}1)$$

$$I = 0.596\,R - 0.274\,G - 0.312\,B \quad ...(2\text{-}2)$$

$$Q = 0.211\,R - 0.523\,G + 0.312\,B \quad ...(2\text{-}3)$$

Sedangkan untuk mentransformasi nilai YIQ kembali ke RGB digunakan rumus:

$$R = 0.990\,Y + 0.967\,I + 0.605\,Q \quad ...(2\text{-}4)$$

$$G = 1.003\,Y - 0.276\,I - 0.642\,Q \quad ...(2\text{-}5)$$

$$B = 1.011\,Y - 1.116\,I + 1.720\,Q \quad ...(2\text{-}6)$$

2.4 Block Dividing

Dalam implementasi metode Hadamard, suatu citra BMP digunakan sebagai *input*. Citra tersebut dianggap sebagai suatu matriks dengan jumlah kolom sama dengan lebar citra dan jumlah kolom sama dengan tinggi citra. Masing-masing nilai R, G, dan B dari setiap piksel citra tersebut akan dihitung sendiri-sendiri sehingga kemudian diperoleh 3 buah matriks, masing-masing dengan kolom x dan baris y. Metode ini disebut pendekatan matriks. Dalam proses Transformasi, terlebih dulu perlu dilakukan *Block Dividing*, atau pembagian

[2] Michael S. Lew. op. cit., Hal: 45

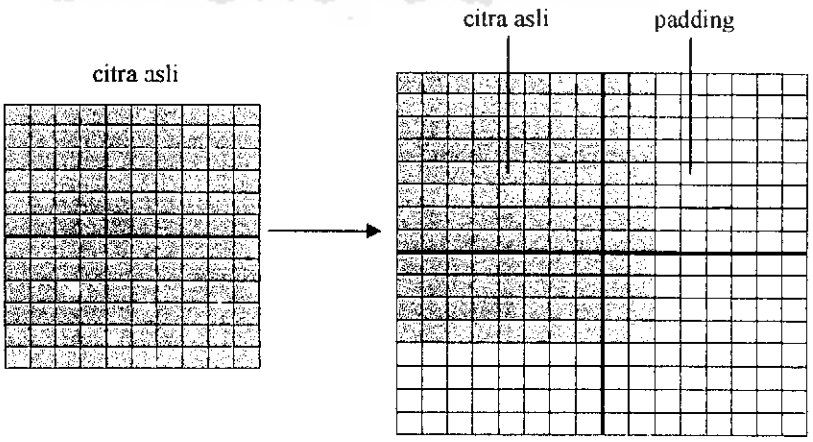
matriks citra menjadi sub-sub matriks yang berukuran lebih kecil agar lebih mudah diproses. Syarat sub-matriks yang harus dipenuhi antara lain :

1. Jumlah baris dan kolom sama (Matriks Bujursangkar).
2. Jumlah baris atau kolom merupakan bilangan 2^n, misal 2, 4, 8, 16, dst.

Ada 2 ukuran matriks yang sering dipakai, yaitu matriks ukuran 4 × 4 dan matriks ukuran 8 × 8. Ukuran *Block* yang dipakai dapat menentukan lamanya proses, berhubungan dengan jumlah operasi matematik yang dilakukan, serta menentukan kualitas citra yang dihasilkan.

Pada saat dilakukan *block dividing*, ada kemungkinan terdapat sisa baris atau kolom yang jumlahnya kurang dari ukuran block, dalam kasus ini perlu dilakukan penambahan beberapa baris atau kolom tambahan dengan nilai sama dengan baris atau kolom matriks yang terakhir sekian kali sampai ukuran block tersebut dapat terpenuhi. Baris-baris atau kolom-kolom tambahan ini disebut *padding*.

Contoh : Sebuah citra berukuran 10 × 12 piksel. Akan dilakukan block dividing dengan ukuran 8 × 8.



Gambar 2.1 Citra sebelum dan sesudah ditambah padding

2.5 Metode Transformasi Hadamard

Metode transformasi Hadamard merupakan suatu metode transformasi dengan menggunakan matriks bujursangkar yang berisikan hanya 1 dan -1 yang memiliki dua atau lebih kolom atau baris yang terletak berhadapan dimana setengahnya bagiannya memiliki tanda yang sama dan setengah bagian lainnya memiliki tanda yang berlawanan.

Suatu matriks Hadamard berukuran n x n (H_n) harus mempunyai kotak putih sebanyak n(n-1)/2 (bernilai -1) dan kotak hitam sebanyak n(n+1)/2 (bernilai 1). Suatu definisi tentang matriks Hadamard diberikan oleh persamaan berikut ini:

$$H_n H_n^T = nI_n \qquad ...(2\text{-}7)$$

dimana I_n merupakan matrik identitas.

Untuk orde 2 maka matriks transformasi H_2 adalah

$$H_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad ...(2\text{-}8)$$

sedangkan untuk orde yang lebih besar maka matriks transformasi H_n dapat dicari dengan rumus [3]

$$H_n = \frac{1}{\sqrt{n}}\begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix} \qquad ...(2\text{-}9)$$

[3] Kenneth R. Castleman, Digital Image Processing, Prentice Hall, 1996

Matriks dengan ukuran 4 × 4 dan 8 × 8 yang digunakan dalam penelitian ini adalah

$$H_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$H_8 = \frac{1}{2\sqrt{2}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

2.6 Kuantisasi

Kuantisasi digunakan untuk mengkonversi nilai numerik menjadi nilai lain yang lebih kecil sebagai representasi dari nilai itu sendiri. Kuantisasi dipakai dalam kompresi *lossy*.

Dalam proses kuantisasi, setiap elemen matriks citra dibagi suatu bilangan bulat pembagi, kemudian dilakukan pembulatan (round) dengan mengambil nilai bulat yang paling mendekati. Bilangan pembagi ini diperoleh dengan rumus:

$$\text{Quant}[i, j] = 1 + ((i + j) \times R) / q \qquad \ldots(2\text{–}10)$$

dimana i dan j merupakan bilangan yang mewakili letak elemen matriks.

Nilai q menentukan kualitas kuantisasi, dimana semakin kecil q kualitas semakin baik dan semakin besar q kualitas semakin menurun. Nilai R dapat ditentukan sendiri dari awal, misalnya R = 2. Apabila bilangan pembagi dimasukkan dalam suatu tabel akan tampak seperti dalam gambar berikut:

$$\begin{bmatrix} 1 & 3 & 5 & 7 & 9 & 11 & 13 & 15 \\ 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \end{bmatrix}$$

Gambar 2.2 Matriks kuantisasi

Contoh :

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \end{bmatrix} \quad \begin{bmatrix} 100 & 33 & 20 & 14 & 11 & 9 & 8 & 7 \\ 33 & 20 & 14 & 11 & 9 & 8 & 7 & 6 \\ 20 & 14 & 11 & 9 & 8 & 7 & 6 & 5 \\ 14 & 11 & 9 & 8 & 7 & 6 & 5 & 5 \\ 11 & 9 & 8 & 7 & 6 & 5 & 5 & 4 \\ 9 & 8 & 7 & 6 & 5 & 5 & 4 & 4 \\ 8 & 7 & 6 & 5 & 5 & 4 & 4 & 4 \\ 7 & 6 & 5 & 5 & 4 & 4 & 4 & 3 \end{bmatrix}$$

(a) (b)

Gambar 2.3 Contoh matriks citra (a) dan hasil kuantisasinya (b)

2.6.1 Kuantisasi Reversi

Kuantisasi reversi dilakukan dengan cara mengalikan setiap elemen matriks citra yang telah dikuantisasi sebelumnya dengan bilangan pembagi yang dipakai dalam proses kuantisasi.

$$\begin{bmatrix} 100 & 99 & 100 & 98 & 99 & 99 & 104 & 105 \\ 99 & 100 & 98 & 99 & 99 & 104 & 105 & 102 \\ 100 & 98 & 99 & 99 & 104 & 105 & 102 & 95 \\ 98 & 99 & 99 & 104 & 105 & 102 & 95 & 105 \\ 99 & 99 & 104 & 105 & 102 & 95 & 105 & 92 \\ 99 & 104 & 105 & 102 & 95 & 105 & 92 & 100 \\ 104 & 105 & 102 & 95 & 105 & 92 & 100 & 108 \\ 105 & 102 & 95 & 105 & 92 & 100 & 108 & 87 \end{bmatrix}$$

Gambar 2.4 Contoh hasil *reverse* kuantisasi

2.7 Pengkodean

Proses pengkodean diperlukan untuk menyimpan hasil kompresi kedalam file. Proses ini dilakukan menggunakan metode kompresi yang bersifat *lossless*.

2.7.1 *Huffman coding*

Metode pengkodean Huffman merupakan salah satu metode pengkodean *lossless* yang berbasis statistik. Dalam pengkodean ini dibangun sebuah susunan data yang terkait satu sama lain menyerupai pohon, sehingga disebut Pohon Huffman. Langkah awal dalam menyusun pohon Huffman adalah menyusun daftar semua simbol dalam data secara urut terbalik menurut kardinalitas masing-masing. Kardinalitas adalah jumlah munculnya suatu simbol dalam suatu deretan data.

Simbol yang paling sering muncul diberi kode paling kecil. Simbol yang jarang muncul akan diberi kode yang paling besar. Dengan demikian apabila jumlah kode yang paling sering muncul berbeda jauh dengan yang paling jarang, maka hasil pengkodean akan berukuran lebih kecil dari data asli.

Setelah memperoleh data statistik, kemudian dibangun pohon Huffman menggunakan algoritma:

1. Pilih 2 simbol dengan kardinalitas paling kecil. Simbol ini disebut *node*.
2. Bit terakhir dari masing-masing kode untuk kedua *node* tersebut dibedakan antara satu dengan lainnya.
3. Gabung kedua *node* tadi menjadi satu node baru.
4. *Node* baru yang terbentuk di langkah 3 memiliki frekuensi kemunculan berupa jumlah dari frekuensi masing-masing node penyusunnya
5. Proses 1 s/d 4 diulang hingga masing-masing simbol memiliki kode

Setelah pohon selesai dibangun maka akan diperoleh kode Huffman yang mewakili masing-masing simbol dalam data. Berikutnya dilakukan pengkodean data dengan cara membaca data secara urut maju, kemudian mencari kode dari setiap simbol pada tabel. Data baru yang terdiri dari rangkaian kode-kode Huffman ini merupakan bentuk terkompres dari data asli yang akan disimpan ke *file* beserta dengan pohon Huffman yang telah dibangun.

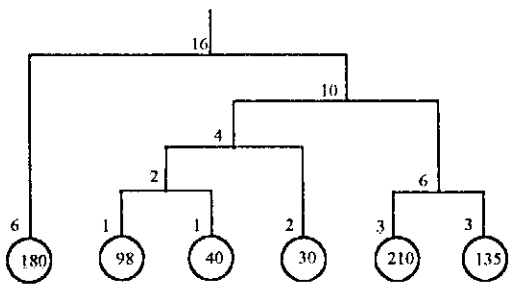
Contoh :

Data yang akan dikompres:

(210 210 210 30 180 180 180 135 135 135 40 180 180 180 30 98)

Ukuran (asli) = 16 × 8 bit = 128 bit

Simbol	Kardinalitas
180	6
135	3
210	3
30	2
98	1
40	1



Gambar 2.5 Frekuensi dan pohon huffmannya

Simbol	Kode
180	0
135	111
210	110
30	101
98	1001
40	1000

Gambar 2.6 Kode yang diperoleh

Data terkompres:

(110 110 110 101 0 0 0 111 111 111 1000 0 0 0 101 1001)

Ukuran (terkompres) = (6 × 1) + (8 × 3) + (2 × 4) = 6 + 24 + 8 = 38 bit

2.8 Ukuran kehilangan data

Untuk melakukan pengukuran tingkat kehilangan data dalam kompresi *lossy* dipakai 2 macam ukuran, yaitu *ratio* kompresi (*compression ratio*) dan *Mean Square Error*.

2.8.1 Ratio kompresi

Rumus Ratio kompresi yaitu:

$$RatioKompresi = \left(1 - \frac{UkuranFileOutput}{UkuranFileInput}\right) \times 100\% \qquad ...(2\text{-}11)$$

Ratio kompresi digunakan untuk mengukur tingkat keberhasilan kompresi berdasarkan ukuran *file* yang dihasilkan. Apabila ratio kompresi lebih besar dari 100% maka kompresi dapat disebut sebagai kompresi negatif karena ukuran *file output* lebih besar dari ukuran *file input*. Jadi *ratio* kompresi menentukan berhasil atau tidaknya suatu proses kompresi.

2.8.2 *Roud Mean Square Error* (RMSE)

Rumus *Mean Square Error* yaitu:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{(RP_{i,j}-RQ_{i,j})^2+(GP_{i,j}-GQ_{i,j})^2+(BP_{i,j}-BQ_{i,j})^2}{3}} \quad ..(2\text{-}12)$$

dimana

n : panjang citra × lebar citra

$RP_{i,j}$: Warna merah pada p*ixel* citra asli

$RQ_{i,j}$: Warna merah pada p*ixel* citra setelah dikompres

$GP_{i,j}$: Warna hijau pada p*ixel* citra asli

$GQ_{i,j}$: Warna merah pada p*ixel* citra setelah dikompres

Mean Square Error adalah rata-rata perbedaan antara setiap piksel pada citra asli dan citra terkompres. Digunakan untuk mengukur tingkat kesalahan data dalam proses kompresi citra yang bersifat *lossy*. Semakin tinggi MSE semakin tinggi pula tingkat kesalahan yang terjadi.