Penerapan Pohon Biner Huffman Pada Kompresi Citra

Alvin Andhika Zulen (13507037)
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha No 10 Bandung, 40132
e-mail: alvin_az@students.itb.ac.id, if17037@students.if.itb.ac.id

Abstrak – *Makalah ini membahas tentang kompresi citra dengan menggunakan metode pohon biner Huffman. Metode ini biasanya digunakan untuk kompresi file teks. Dalam makalah ini, metode ini dikembangkan menjadi algoritma untuk kompresi citra. Hasil yang diperoleh menunjukan bahwa algoritma ini memiliki nilai rasio kompresi di atas satu jika jumlah warna yang terdapat dalam citra tidak lebih dari 100 warna. Sedangkan kualitas citra hasil dekompresi memiliki kualitas yang sama dengan citra aslinya.*

Kata Kunci*: kompresi, dekompresi, citra, pohon biner, kode Huffman, rasio kompresi, kualitas*

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat telah menjadi peran yang sangat penting untuk pertukaran informasi yang cepat. Kecepatan pengiriman informasi dalam bentuk perpaduan teks, suara dan gambar secara real-time akan menjadi bagian utama dalam pertukaran informasi. Kecepatan pengiriman ini sangat bergantung kepada ukuran dari informasi tersebut

Salah satu solusi untuk masalah di atas adalah dengan melakukan pemampatan (kompresi) data teks, suara, dan citra sebelum ditransmisikan dan kemudian penerima akan merekonstruksinya kembali menjadi data aslinya (dekompresi).

Dari materi perkuliahan, metode kompresi dengan pohon biner Huffman hanya dikenal untuk kompresi file berupa teks saja, Makalah ini menguraikan aplikasi metode Huffman untuk kompresi citra. Permasalahan utama yang akan dibahas dalam makalah ini adalah :

- Apakah metode Huffman dapat diterapkan untuk kompresi citra ?
- Seberapa besar rasio kompresi yang dapat dihasilkan ?
- Apakah metode Huffman ini dapat merekonstruksi kembali data citra sesuai dengan data aslinya ?

2. DASAR TEORI

2.1 Citra

Citra, menurut kamus Webster, adalah *suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda.* Sedangkan dari sudut pandang matematis, citra merupakan *fungsi kontinu dari intensitas cahaya pada bidang dua dimensi.*

[1] Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat:

1. optik, berupa foto
2. analog, berupa sinyal video
3. digital, yang dapat langsung disimpan pada media penyimpanan magnetik

Citra juga dapat dikelompokkan menjadi :

1. Cita tampak : foto, gambar
2. Citra tidak tampak : data gambar dalam file, citra yang direpresentasikan dalam fungsi matematis

2.2 Kompresi

Kompresi adalah proses pengubahan sekumpulan data menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat penyimpanan data dan waktu untuk transmisi data[2]. Jika data tersebut hendak dipergunakan kembali, maka harus dilakukan dekompresi, yaitu pengubahan kode-kode menjadi data awal.

Saat ini, telah banyak tersedia algoritma untuk melakukan kompresi, antara lain: Huffman, LIFO, LZHUF, LZ77 dan variannya (LZ78, LZW, GZIP), Dynamic Markov Compression (DMC), Block-Sorting Lossless, Run-Length, Shannon-Fano, Arithmetic, PPM (Prediction by Partial Matching), Burrows-Wheeler Block Sorting, dan Half Byte. Namun, ada beberapa faktor yang dijadikan pertimbangan dalam memilih algoritma yang akan digunakan dalam kompresi data, yaitu :

1. Sumber daya yang dibutuhkan (Memori, kecepatan PC)
2. Kecepatan kompresi
3. Ukuran hasil kompresi
4. Besarnya redudansi
5. Ketepatan hasil dekompresi
6. Kompleksitas algoritma

Berdasarkan output hasil kompresi, metode kompresi dapat dikelompokkan menjadi dua, yaitu :

1. **Kompresi Loseless**
   Hasil dekompresi dari data hasil kompresi akan tepat sama persis dengan data sebelum kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip

   Kelemahan dari metode ini adalah ratio kompresi yang rendah. Rasio dapat dihitung dengan persamaan :

   $$Rasio = \left(1 - \frac{Ukuran\ kompresi}{Ukuran\ asli}\right) x\ 100\ \% \quad \ldots (!)$$

   Teknik ini digunakan jika dibutuhkan data dimana setelah dikompresi harus dapat didekompresi lagi dan menghasilkan data yang tepat sama dengan data asli. Contoh pada data teks, data program/biner, dan beberapa image random seperti GIF dan PNG.

2. **Kompresi Lossy**
   Hasil dekompresi dari data hasil kompresi tidak tepat sama persis, tetapi persepsi terhadap semantik data tetap sama. Contoh: Mp3, streaming media, JPEG, MPEG, dan WMA.

   Kelebihan dari metode ini adalah ukuran file lebih kecil disbanding loseless namun masih tetap memenuhi syarat untuk digunakan.

   Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.

2.3 Kompresi Citra

Semakin besar ukuran sebuah citra, semakin besar memori yang dibutuhkan. Namun, kebanyakan citra mengandung duplikasi data, yaitu :

- Suatu piksel memiliki intensitas yang sama dengan piksel tetangganya, sehingga penyimpanan setiap piksel memboroskan memori
- Mengandung *region* yang sama, sehingga bagian yang sama ini tidak perlu dikodekan berulang kali

Kompresi citra bertujuan untuk meminimalkan kebutuhan memori untuk merepresentasikan citra digital dengan mengurangi duplikasi data di dalam citra sehingga memori yang dibutuhkan menjadi lebih sedikit. Hal ini nantinya akan berpengaruh pada waktu pengiriman citra yang menjadi lebih singkat.

Metode kompresi citra yang diharapkan yaitu :

- **Proses kompresi dan dekompresi citra cepat.**
  Proses kompresi : Citra dalam representasi tidak mampat dikodekan dengan representasi yang meminimumkan memori. Citra terkompresi disimpan dalam file dengan format tertentu.
  Proses dekompresi : Citra yang sudah dikompresi dikembalikan lagi menjadi representasi yang tidak mampat dan disimpan dalam format tidak mampat
- **Memori yang dibutuhkan seminimal mungkin**
  Ukuran memori hasil kompresi bergantung kepada metode kompresi yang digunakan dan citra itu sendiri. Citra yang mengadung banyak elemen duplikasi akan dikompresi dengan memori lebih sedikit.
- **Kualitas citra hasil kompresi bagus**
  Informasi yang hilang akibat kompresi seharusnya seminimal mungkin sehingga kualitas hasil kompresi bagus.
  Mengukur kualitas hasil kompresi dengan PSNR (*Peak signal- to-noise ratio)*

  $$PSNR = 20\ x\ log\ \frac{b}{rms} \ldots\ldots\ldots (2)$$

  $$rms = \sqrt{\frac{1}{tinggi\ x\ lebar} \sum_{i=1}^{N} \sum_{j=1}^{M} (fij - f'ij)^2} \quad \ldots\ldots (3)$$

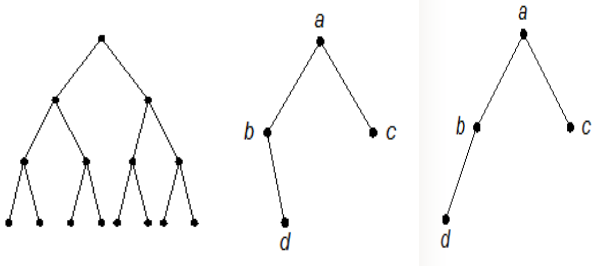
  Keterangan :
  b = sinyal terbesar (pada citra hitam putih, b = 255)
  rms = akar pangkat dua dari selisih antara citra semula dan citra hasil kompresi
  f = nilai piksel citra semula
  f' = nilai piksel citra hasil kompresi
- **Proses transfer dan penyimpanan mudah**

Pendekatan yang dapat digunakan untuk kompresi citra antara lain [3] **:**

1. **Pendekatan statistik (*statistical compression*)**
   Kompresi citra berdasarkan pada derajat keabuan (*gray level*) dari piksel-piksel dalam keseluruhan *image*.
2. **Pendekatan ruang (*spatial compression*)**
   Kompresi citra yang memiliki kelompok-kelompok piksel berderajat keabuan yang sama
3. **Pendekatan kuantisasi (*quantizing compression*)**
   Kompresi citra dengan mereduksi jumlah derajat keabuan yang ada pada citra
4. **Pendekatan fraktal (*fractal compression*)**
   Kompresi citra berdasarkan *fractal generating function*
5. **Pendekatan transformasi wavelet (*wavelet compression*)**

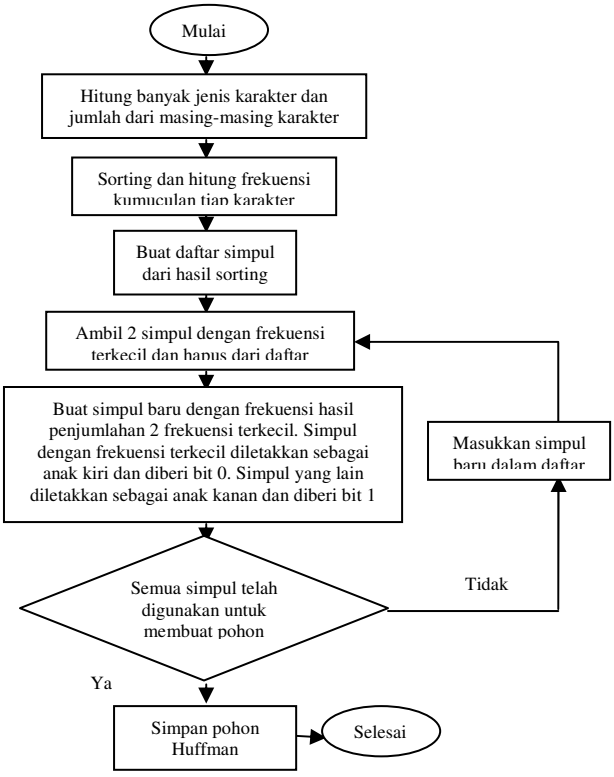
2.4 Pohon Biner Huffman

Pohon adalah *graf tak-berarah terhubung yang tidak mengandung sirkuit* Sedangkan pohon biner adalah *pohon berakar dimana setiap simpul cabanhnya mempunyai paling banyak dua buah anak*. Pohob biner teratur adalah *pohon biner dimana setiap simpul cabangnya mempunyai dua buah anak.* [4]



Gambar 1. Beberapa contoh Pohon Biner

Salah satu aplikasi dari pohon biner adalah Kode Huffman. Skema pengodean Huffman dikembangkan oleh David A. Huffman pada tahun 1952, Skema pengodean Huffman merupakan metode kompresi *Lossless* Kompresi data yang dihasilkan pada algoritma pengodean Huffman dicapai dengan mengodekan data berdasarkan pada frekuensi kemunculannya. Struktur data yang terbentuk pada algoritma pengodean Huffman adalah pohon biner berbobot . Algoritma dari skema pengodean Huffman dapat dilihat pada bagan alir di bawah ini :



Gambar 2. Bagan Alir Pembentukan Pohon Biner Huffman

Dari pohon biner Huffman, diperoleh bahwa data dengan frekuensi kemunculan terbesar memiliki jumlah bit terkescil pada kode Huffman.

Setelah terbentuk pohon biner Huffman, data asli diganti dengan kode bit berdasarkan pohon biner. Proses ini dinamakan *encoding*. Langkah untuk *encoding* suatu string biner adalah :

a. Tentukan karakter yang akan di-*encoding*
b. Mulai dari akar, baca setiap bit yang ada pada cabang yang bersesuaian sampai ditemukan daun dimana karakter itu berada
c. Karakter tersebut dikodekan dengan kode Huffman yang diperoleh
d. Ulangi langkah 2 dan 3 sapai seluruh karakter di-*encoding*

Setiap data yang telah mengalami kompresi tentu harus dapat merekonstruksi kembali data tersebut sesuai dengan aslinya, atau yang disebut dengan dekompresi. Ada 2 cara penguraian kode Huffman, yaitu:

1. Menggunakan Pohon Huffman

Langkah – langkah yang dilakukan dalam penguraian kode menggunakan pohon Huffman adalah :

a. Baca bit pertama dari string biner masukan
b. Lakukan traversal pada pohon Huffman mulai dari akar sesuai dengan bit yang dibaca. Jika bit yang dibaca adalah 0, baca anak kiri, tetapi jika bit yang dibaca adalah 1, baca anak kanan.
c. Jika anak dari pohon bukan daun, baca bit berikutnya dari string biner
d. Traversal hingga ditemukan daun.
e. Pada daun tersebut, simbol ditemukan dan proses penguraian kode selesai.
f. Proses penguraian kode dilakukan hingga seluruh string biner masukan diproses.

2. Menggunakan Tabel Kode Huffman

Cara kedua untuk menguraikan kode Huffman adalah dengan menggunakan tabel kode Huffman. Oleh karena kode Huffman disusun menggunakan kode awalah (*prefix code*), dapat dipastikan bahwa sebuah kode untuk sebuah simbol/karakter tidak boleh menjadi awalan dari simbol yang lain. Oleh karena itu, string biner yang berisi hasil enkripsi dapat dipisahkan dengan mudah berdasarkan setiap rangkaian bitnya untuk diuraikan menjadi data semula. Yang perlu dilakukan hanyalah melihat setiap rangkaian bit yang ditemukan dalam string biner hasil enkripsi di dalam tabel kode Huffman.

.

Algoritma Huffman mempunyai kompleksitas waktu T(n)=O(n log n). Hal ini karena dalam melakukan sekali proses iterasi pada saat penggabungan dua buah pohon yang mempunyai frekuensi terkecil pada sebuah akar membutuhkan waktu O(log n), dan proses itu dilakukan berkali-kali sampai hanya tersisa satu buah pohon Huffman, yang berarti dilakukan sebanyak n kali[5].

3. HASIL DAN PEMBAHASAN

Beberapa metode kompresi citra telah dikembangkan seperti *Block-coding*, CDT, *Wavelet transform,* dan

lainnya. Metode kompresi citra terus dikembangkan dengan tujuan untuk mengkompres hingga sekecil mungkin data citra, tetapi pada saat rekonstruksi diharapkan tidak satupun data citra yang hilang.

Berdasarkan pada landasan pikiran di atas serta hasil yang diperoleh dari metode Huffman dimana teks hasil dekompresi yang diperoleh 100% sama dengan teks aslinya, maka akan diteliti sejauh mana metode Huffman dapat digunakan untuk kompresi data berupa citra.

Secara fisik, sebuah citra merupakan representasi objek-objek, baik dalam keadaan diam atau bergerak, pada suatu *support* fisik seperti kertas, monitor, atau lainnya. Secara matematis, sebuah citra dinyatakan sebagai sebuah fungsi matematis dua dimensi 2Df(x,y) atau tiga dimensi 3Df(x,y,z). Dimana x dan y menyatakan posisi koordinat 2D (atau 3D), sedangkan f menyatakan nilai intensitas (kecerahan) atau menyatakan warna pada setiap posisi [x,y] (atau [x,y,z]) .

Sebuah citra digital dalam sebuah komputer dinyatakan dalam bentuk matriks 2D, dimana elemen matriks disebut piksel dan nilai dari setiap elemen matriksnya menyatakan intensitas atau warna. Bila matriks ini mewakili sebuah citra *gray-level*, maka nilai elemen matriks (piksel) menyatakan tingkat keabuan citra. Namun, bila matriks ini mewakili sebuah citra berwarna, maka nilai elemen matriks menyatakan warna. Setiap piksel dalam sebuah citra yang dikode dalam 8 bit, berarti citra tersebut memiliki 256 tingkat keabuan atau memiliki 256 warna.
Contoh sebuah citra dapat dilihat pada gambar di bawah ini.

Gambar 3. Citra 2D

Misalkan citra tersebut dinyatakan dalam matriks berikut, dimana setiap elemen matriks (piksel) menyatakan warna

$$\begin{pmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 200 & 200 & 200 & 100 \\ 250 & 100 & 200 & 100 & 250 \end{pmatrix}$$

Metode Huffman termasuk metode *lossless compression*. Pengkodean citra berdasarkan pada derajat keabuan (*gray level*) atau tingkat warna dari piksel-piksel dalam keseluruhan *image.* Dengan kata lain, metode Huffman termasuk dalam pendekatan statistikal (*statistical* compression) dalam kompresi citra. Warna atau derajat keabuan yang sering muncul di dalam citra akan dikodekan dengan jumlah bit yang lebih sedikit sedangkan nilai yang frekuensi kemunculannya sedikit dikodekan dengan jumlah bit yang lebih panjang

Algoritma metode Huffman untuk kompresi citra yaitu:
1. Buat data citra yang berupa matriks tersebut menjadi vektor
2. Tentukan frekuensi kemunculan tiap warna atau derajat keabuan
3. Urutkan secara menaik warna atau nilai keabuan berdasarkan frekuensi kemunculannya atau peluang kemunculan piksel dalam citra Setiap nilai dinyatakan sebagai pohon bersimpul tunggal dan setiap simpul di-*assign* dengan frekuensi kemunculan nilai tersebut.
4. Gabungkan 2 buah pohon yang mempunyai frekuensi kemunculan paling kecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua pohon penyusunnya. Frekuensi dengan nilai lebih kecil diletakkan di sisi kiri dan diberi bobot 0 sedangkan sisi kanan diberi bobot 1.
5. Ulangi langkah 4 sampai tersisa 1 pohon biner.
6. Telusuri pohon biner dari akar ke daun. Barisan bobot sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan atau warna yang bersesuaian.
7. Mengganti data dengan kode Huffman yang bersesuaian
8. Menyimpan data lebar citra, tinggi citra, kode bit untuk tiap nilai, data warna yang terdapat di dalam citra, dan data citra yang sudah dikodekan ke dalam file hasil kompresi.

Untuk mengembalikan data citra terkompres menjadi data citra aslinya, diperlukan suatu algoritma dekompresi yang merupakan kebalikan dari algoritma kompresi. Berikut ini adalah langkah-langkah untuk mengembalikan data citra yang sudah dikodekan menjadi data citra semula :

1. Baca file hasil kompresi dan data-datanya dimasukkan ke variabel yang sesuai yaitu variabel ukuran citra, variabel kode bit data, dan variabel warna
2. Baca data kode bit per bit dari kiri ke kanan dan dicocokkan dengan kode Huffman dari data warna yang didapat. Demikian

seterusnya konversi dilakukan hingga data terakhir.

3. Rekonstruksi citra dengan menggunakan data ukuran citra, berarti data pixel berbentuk 1D dipenggal baris dan kolom sesuai ukuran citra.

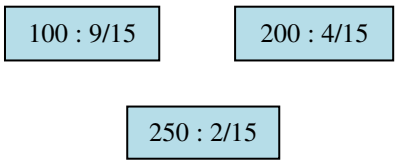
Sebagai contoh penerapan kompresi citra dengan metode Huffman, dapat dilihat pada dua contoh berikut :

Contoh 1 :
Terdapat citra dengan representasi matriks sebagai berikut. Tiap piksel dikodekan dengan 8-bit warna.

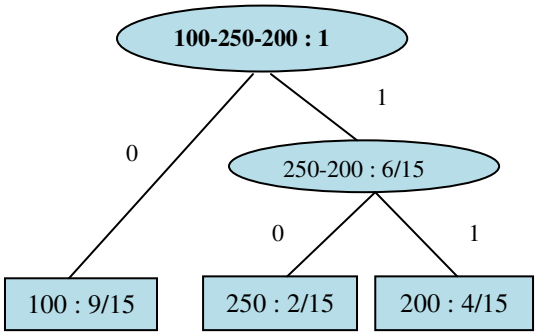
$$\begin{pmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 200 & 200 & 200 & 100 \\ 250 & 100 & 200 & 100 & 250 \end{pmatrix}$$

Akan dilakukan kompresi terhadap data citra tersebut. Langkah yang dilakukan adalah :

1. Bentuk vektor dari data citra yang berupa matriks
   [100 100 100 100 100 100 100 100 100 200 200 200 200 250 250]
   Jumlah piksel = 15
2. Peluang kemunculan warna pada citra :
   100 = 9/15
   200 = 4/15
   250 = 2/15
3. Simpul pohon biner :



4. Pembentukan pohon Biner Huffman



Tabel 1. Tabel Kode Huffman untuk contoh

Warna	Peluang kemunculan	Kode Huffman
100	9 / 15	0
200	4 / 15	11
250	2 / 15	10

5. Konversi data
   [100 100 100 100 100 100 100 100 100 200 200 200 200 250 250] menjadi :

   000000000111111111010
6. Penyimpanan data pada citra
   - Ukuran : 5 x 3
   - Kode Huffman
   - Data warna

Rasio kompresi untuk contoh 1 dihitung dengan persamaan (1) :

- **Ukuran citra sebelum kompresi**
  15 piksel x 8 bit = **120 bit**
- **Ukuran citra setelah kompresi**
  (9 x 1 bit) + (4 x 2 bit) + (2 x 2 bit) = **21 bit**
- **Rasio Kompresi**

$$Rasio = \left(1 - \frac{Ukuran\ kompresi}{Ukuran\ asli}\right) x\ 100\ \%$$

$$= \left(1 - \frac{21}{120}\right) x\ 100\ \%$$

$Rasio$ = **82.5 %**

Jika dilakukan dekompresi, citra yang diperoleh akan sama persis dengan ciitra awal (*Lossless*).

Contoh 2 :
Terdapat citra dengan ukuran 64 x 64 dengan 8 derajat keabuan (k)
Jumlah seluruh piksel (n) = 64 x 64 = 4096

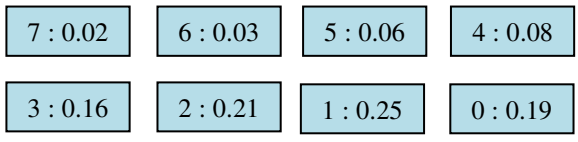
Tabel 2. Tabel Frekuensi Piksel untuk contoh 2

Derajat keabuan	Frekuensi	Peluang kemunculan
0	790	0.19
1	1023	0.25
2	850	0.21
3	656	0.16
4	329	0.08
5	245	0.06
6	122	0.03
7	81	0.02
Jumlah	4096	1.00

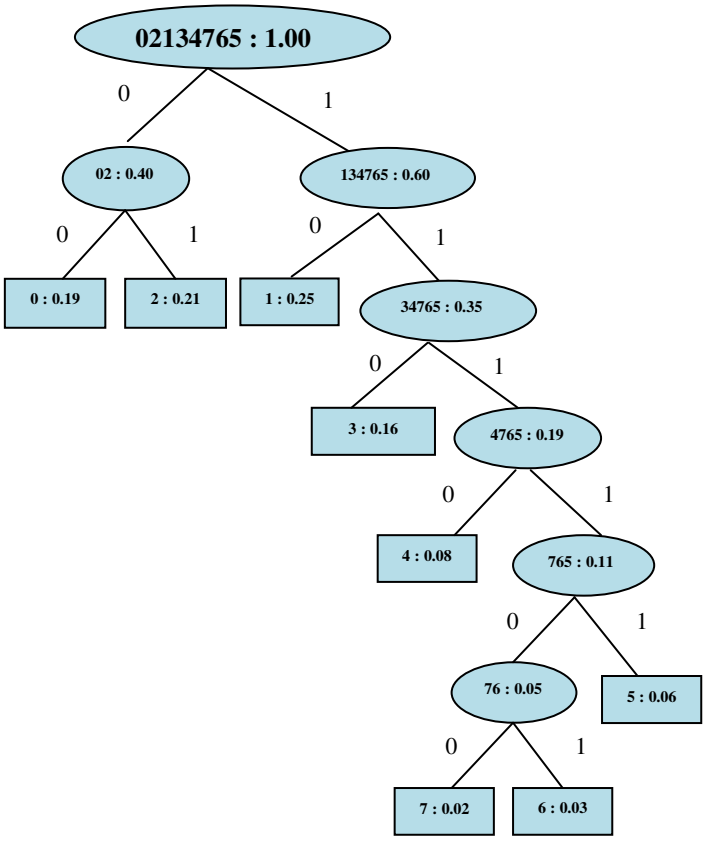
Akan dilakukan kompresi terhadap data citra tersebut. Langkah yang dilakukan adalah :

1. Data derajat keabuan citra dan peluang kemunculannya dapat dilihat pada tabel di atas.

2. Simpul pohon biner



3. Pembentukan pohon biner Huffman



Tabel 3. Tabel Kode Huffman untuk contoh 2

Derajat keabuan	Kode Huffman	Ukuran	Frekuensi
0	00	2 bit	790
1	10	2 bit	1023
2	01	2 bit	850
3	110	3 bit	656
4	1110	4 bit	329
5	11111	5 bit	245
6	111101	6 bit	122
7	111100	6 bit	81

Untuk contoh 2, akan dihitung rasio kompresi dengan persamaan (1) :

- **Ukuran citra setelah kompresi**
  (790 x 2 bit) + (1023 x 2 bit) + (850 x 2 bit) + (656 x 3 bit) + (329 x 4 bit) + (245 x 5 bit) + (122 x 6 bit) + (81 x 6 bit) = **11053 bit**
- **Ukuran citra sebelum kompresi**
  4096 piksel x 3 bit = **12288 bit**
- **Rasio Kompresi**

$$Rasio = \left(1 - \frac{Ukuran\ kompresi}{Ukuran\ asli}\right) x\ 100\ \%$$

$$= \left(1 - \frac{11053}{12288}\right) x\ 100\ \%$$

Rasio = **10.05 %**

Dari kedua contoh ini, dapat dilihat bahwa kompresi menggunakan metode pohon biner Huffman dapat menghasilkan citra terkompresi dengan rasio kompresi cukup besar. Jika dilakukan dekompresi, citra yang diperoleh akan sama persis dengan ciitra awal (*Lossless*).

Untuk mengetahui seberapa besar kemampuan metode Huffman dalam kompresi citra, akan dilakukan dua model uji coba. Pertama, menggunakan sejumlah citra dengan resolusi yang sama tetapi jumlah warna yang ada dalam citra berbeda satu sama lain. Kedua, menggunakan sejumlah citra dengan jumlah warna yang sama tetapi dengan resolusi yang berbeda[6]. Hasil pengujian dapat dilihat pada kedua tabel di bawah ini.

Tabel 4. Tabel Hasil Kompresi pada Citra dengan Resolusi 128x128 dan Setiap Pikselnya dikode dalam 8 bit warna

Jumlah warna	Citra Asli (byte)	Hasil Kompresi (byte)	Rasio kompresi
1	16384	1048	15,63
5	16384	3120	5,25
10	16384	5708	2,87
15	16384	8316	1,97
20	16384	10850	1,51
25	16384	13429	1,22
30	16384	16062	1,02
31	16384	16549	0,99
32	16384	17066	0,96
35	16384	18618	0,88

Tabel 5. Tabel Rasio Kompresi 8 warna dengan Resolusi yang Berbeda

Resolusi citra	Ukuran citra asli (dlm. Byte)	Ukuran kompresi (dlm. Byte)	Rasio kompresi
64 x 64	4096	1226	3,34
128 x 128	16386	4681	3,50
256 x 256	65536	18460	3,55
512 x 512	262144	71624	3,66
1024 x 1024	1048576	286496	3,66

Perhitungan rasio kompresi dilakukan melalui persamaan :

$$Cr = \frac{Ukuran\ file\ citra\ asil}{Ukuran\ file\ citra\ hasil\ kompresi} \quad \text{.....(4)}$$

Keterangan :
Cr = *Compression ratio*

Pada tabel 4 terlihat bahwa semakin banyak jumlah warna yang terdapat dalam sebuah citra, maka rasio kompresi semakin mengecil. Hal ini sangat sesuai dengan bentuk pohon biner Huffman. Semakin banyak jumlah warna yang muncul dalam sebuah

citra, maka pohon biner yang terbentuk memiliki cabang-cabang yang jauh lebih banyak. Hal ini dapat meyebabkan sejumlah piksel akan dikode dengan jumlah bit yang besar. Misalnya pada tabel 4, sebuah citra yang memiliki 31 macam warna di dalamnya, memiliki rasio kompresi 0.99. Ini berarti bahwa bukannya data citra terkompres tetapi justru memperbanyak jumlah data. Hal ini dapat terjadi karena pohon biner yang terbentuk memiliki 30 cabang dan dengan demikian piksel-piksel yang dikode akan memiliki jumlah bit yang bervariasi antara 1 sampai dengan 30 bit per pikselnya, sementara piksel aslinya dikode dalam 8 bit.

Pada tabel 5, dapat dilihat bahwa rasio kompresi yang dihasilkan dari beberapa citra memiliki perbedaan yang tidak signifikan. Hal ini berarti resolusi citra dengan jumlah warna yang sama tidak banyak mempengaruhi nilai rasio kompresi. **Dengan demikian dapat dikatakan bahwa metode Huffman tidak dapat digunakan untuk kompresi citra yang memiliki jumlah warna lebih besar dari 30.**

Untuk mengatasi keterbatasan metode pohon biner Huffman dalam kompresi citra, akan dilakukan kembali kompresi kode bit data citra yang dihasilkan dari kompresi citra dengan metode Huffman. Algoritma yang digunakan untuk kompresi pada ietrasi kedua sama dengan algoritma sebelumnya. Pengaruh dari kompresi kembali kode data dapat dilihat pada tabel berikut.

Tabel 6. Rasio Kompresi Citra pada Iterasi ke dua

Jumlah warna	Citra Asli (byte)	Hasil Kompresi (byte)	Rasio kompresi
32	16384	7154	2,29
35	16384	7953	2,06
50	16384	8904	1,84
100	16384	12318	1,33
150	16384	15753	1,04
159	16384	16384	1,00
160	16384	16549	0,99
170	16384	17066	0,96

Melihat tabel di atas, maka dapat dikatakan bahwa dengan mengkompres kembali kode data citra, nilai rasio kompresi meningkat. Namun, algoritma yang dijalankan hingga iterasi ke dua ini hanya memberikan nila rasio kompresi di atas 1 untuk jumlah warna di bawah 100 warna.

Kemungkinan kompresi dapat dilanjutkan pada iterasi berikutnya, tetapi perlu diperhitungkan waktu kompresi dan dekompresi agar kedua operasi tersebut masih dalam hitungan waktu untuk *real-time*.

Gambar berikut ini memperlihatkan hasil algoritma yang dikembangkan di atas untuk sebuah citra. Gambar 4 memperlihatkan gambar berwarna asli yang dikode dalam 8 bit. Gambar 5 adalah hasil dekompresi untuk dua kali iterasi. Gambar hasil dekompresi, baik untuk satu kali maupun dua kali iterasi, menghasilkan kualitas citra yang sama dengan aslinya.

Gambar 4. Citra Asli

Gambar 5. Citra Dekompresi untuk Dua kali Iterasi (Cr = 2.104)

4. KESIMPULAN

Dari hasil penelitian yang diperoleh, dapat disimpukan bahwa :

1. Metode pohon biner Huffman dapat diterapkan untuk kompresi data citra, tidak hanya untuk kompresi file teks.
2. Resolusi citra dengan jumlah warna yang sama tidak banyak mempengaruhi nilai rasio kompresi.
3. Metode pohon biner Huffman hanya dapat diterapkan untuk citra yang memiliki jumlah warna tertentu untuk menghasilkan rasio kompresi lebih dari 1.00 Metode dapat digunakan untuk citra dengan 30 warna untuk sekali iterasi dan 100 warna untuk dua kali iterasi, sesuai dengan algoritma yang dikembangkan dalam makalah ini.
4. Kualitas citra hasil dekompresi, baik untuk sekali iterasi maupun untuk dua kali iterasi sama dengan citra aslinya. Hal ini terjadi karena data warna atau derajat keabuan

(piksel citra) hanya dikodekan dalam bentuk kode pohon biner Huffman dan rekonstruksi datanya (dekompresi) hanya tinggal mengonversikan kode pohon biner Huffman menjadi data aslinya.

DAFTAR REFERENSI

[1]*http://toba.mytoba.com/dl/Pengolahan%20Citra.pdf*
Tanggal akses: 23 Desember 2008, 19:30 WIB

[2]*http://www.informatika.org/~rinaldi/Matdis/2007-2008/Makalah/MakalahIF2153- 0708-012.pdf*
Tanggal Akses : 23 Desember 2008, 20:37 WIB

[3]*http://fivedots.coe.psu.ac.th/~montri/Teaching/image/Compression1.ppt*
Tanggal akses : 24 Desember 2008, 00:20

[4]Munir, Rinaldi. 2008. Diktat Kuliah IF2091 Struktur Diskrit. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

[5]*http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/introduction.html*
Tanggal Akses : 24 Desember 2008, 15:56 WIB

[6]*http://www.batan.go.id/ppin/lokakarya/LKSTN_10/Sarifuddin1-.pdf*
Tanggal Akses : 25 Desember 2008, 10:00 WIB