Aplikasi Kode Huffman dalam Kompresi Gambar Digital JPEG

Albert Logianto - 13514046
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514046@std.stei.itb.ac.id

***Abstrak*—JPEG (Joint Photographic Experts Group) adalah suatu jenis file gambar digital yang bertipe *bitmap*. Jadi, pada awalnya file yang memuat foto digital memiliki ukuran yang besar sehingga tidak efisien dalam penyimpanannya di dalam suatu media. Dengan adanya format JPEG ini, file gambar tersebut dapat dikompresi terlebih menjadi ukuran yang kecil. Format JPEG biasanya digunakan dalam hasil jepretan kamera digital, tingkat kompresi dari hasil jepretan tersebut dapat disesuaikan dengan kualitas gambar yang diinginkan dan sisa tempat pada memori.**

***Keywords*— JPEG, Kode Huffman, kompresi, gambar, digital.**

I. PENDAHULUAN

Di zaman dimana teknologi sudah berkembang begitu pesat ini, hampir seluruh aspek pada kehidupan kita berkaitan dengan dunia digital. Penggunaan kamera yang berjenis analog sudah lama ditinggalkan, bahkan pada setiap *handphone* pun pasti memiliki kamera digital yang siap sedia digunakan kapanpun. Salah satu kendala di zaman yang serba digital ini adalah kapasitas memori yang sering tidak cukup untuk menampung *file-file* dari *user*, walaupun sebenarnya sistem berbasis *cloud* sudah banyak ditemukan. Masih banyak pengguna yang tidak mengetahui adanya sistem penyimpanan berbasis *cloud*, beberapa pengguna yang sudah mengetahuinya pun masih menemukan kendala lain yaitu kecepatan internet yang masih belum memadai, sehingga pengguna masih bergantung pada kapasitas memori lokal.

Berdasarkan kendala-kendala yang sudah dijelaskan di atas, oleh karena itu efisiensi dalam penyimpanan di memori tentu menjadi suatu poin penting di dunia yang serba digital ini. Salah satu contohnya adalah fotografer yang dapat saja memotret banyak gambar sehingga daripada membeli memori cadangan, tentu lebih baik jika ukuran dari foto hasil jepretan tersebut dapat dibuat lebih kecil dengan tetap terjaganya kualitas foto yang diambil. Salah satu format foto yang umum digunakan adalah JPEG, format ini umum digunakan karena memiliki tingkat kompresi yang baik sehingga dapat menghemat tempat pada memori, dan kualitas foto tetap terjaga, serta pemrosesan gambar dari hasil jepretan kamera menjadi format JPEG juga membutuhkan waktu yang singkat. Format JPEG ini juga umum digunakan dalam halaman-halaman web karena ukuran filenya serta kualitas yang cukup baik. Salah satu keunggulan yang dimiliki oleh JPEG adalah dapat dibaca oleh hampir semua program-program komputer dan hampir semua *Operating System* seperti Windows, Linux, MacOS dan sebagainya.

JPEG yang kepanjangannya adalah *Joint Photographic Experts Group* mulai dikembangkan oleh sejak tahun 1986 dan mulai diimplementasikan pada tahun 1996. JPEG sendiri adalah format gambar yang bertipe *bitmap* yang artinya JPEG menyimpan data warna (direpresentasikan oleh bytes) dari setiap *pixel* yang ada.

Jadi, secara dasarnya, kompresi dari JPEG dapat menggunakan kode huffman dengan memperhitungkan frekuensi kemunculan warna yang ada. Dengan menggunakan kode huffman, file gambar yang besar dapat dibuat menjadi jauh lebih kecil.

II. DASAR TEORI

A. Pohon Biner

Pohon adalah suatu graf tak-berarah terhubung dan tidak mengandung sirkuit di dalamnya. Pohon Biner adalah suatu pohon *n-ary* dimana n=2, y`3ang artinya masing-masing simpul memiliki maksimal 2 buah anak. Pohon Biner ini adalah pohon terurut *(ordered tree)* karena urutan anak dianggap penting serta dibedakannya antara anak kiri *(left child)* dan anak kanan *(right child).*

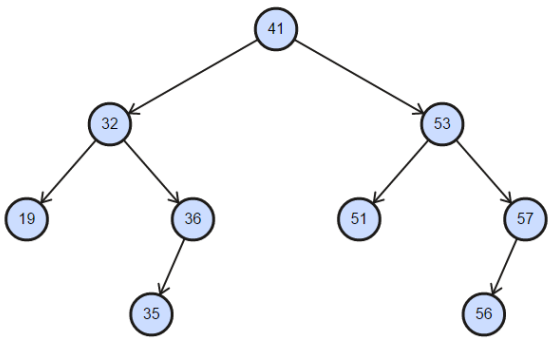


Diagram 1. Contoh pohon biner
http://btv.melezinek.cz/binary-search-tree.html

B. Kode Huffman

Kode Huffman ditemukan oleh David A. Huffman pada tahun 1952 saat berkuliah di MIT, saat itu dia ditugaskan oleh profesornya untuk menemukan kode biner yang paling efisien, Huffman mengalahkan profesornya dalam membuat kode biner yang paling efisien dan akhirnya ditemukanlah kode Huffman. Kode Huffman adalah suatu kode prefiks yang menggunakan aplikasi dari pohon biner, kode ini umum digunakan dalam kompresi data yang bersifat *lossless* yang artinya data sebelum dan sesudah proses kompresi tidak mengalami perubahan apapun. Berbeda dengan kompresi yang bersifat *lossy* yang artinya rekonstruksi dari data yang sudah dikompresi tidak sepenuhnya sama dengan data original namun mendekatinya. Kode Huffman memanfaatkan probabilitas dari kemunculan dari tiap-tiap simbol, salah satu aplikasi dari kode Huffman adalah untuk mengompresi teks yang berbasis kode ASCII. Dalam ASCII, setiap karakter dikodekan dalam 8 bit biner sehingga setiap karakter besarnya 1 byte. Contoh dari kode ASCII untuk beberapa karakter ditunjukkan sebagai berikut.

Simbol	Kode ASCII
A	0100 0001
B	0100 0010
C	0100 0011
D	0100 0100

Table 1. Kode ASCII

Sehingga untuk string ‘ABCAACD’ jika diimplementasikan dalam kode ASCII menjadi: *01000001010000100100001101000001010000010100001101000100*. Sehingga dibutuhkan 56 bit (8 byte) untuk merepresentasikan 7 karakter. Untuk mengompresi data string tersebut, pertama-tama kita buat terlebih dahulu tabel frekuensi dari karakter tersebut.

Simbol	Frekuensi	Peluang
A	3	3/7
B	1	1/7
C	2	2/7
D	1	1/7

Table 2. Tabel frekuensi dari string 'ABCAACD'

Setelah mendapatkan frekuensi dari setiap kemunculan simbol tersebut, untuk membuat kode Huffman dari *string* tersebut, langkah-langkahnya adalah sebagai berikut:

1. Pilihlah dua simbol dengan peluang atau frekuensi yang paling kecil, sebagai contoh dari *string* ‘ABCAACD’, simbol B dan D adalah simbol yang peluangnya terkecil, sehingga gabungkanlah kedua simbol tersebut menjadi suatu simpul orang tua baru dengan simbol baru BD dengan peluangnya 1/7+1/7 = 2/7. Simbol ini diperlakukan seperti simbol-simbol sebelumnya dan digunakan untuk mencari peluang simbol terkecil selanjutnya.
2. Lalu, pilihlah dua simbol dengan peluang terkecil selanjutnya, simbol baru BD juga diperhitungkan. Sebagai contoh ini, kedua simbol tersebut adalah BD dan C, yaitu masing-masing peluangnya adalah 2/7. Sehingga seperti langkah 1, dihasilkanlah simbol baru CBD dengan peluang 4/7.
3. Ulangilah langkah 1 sampai 2 sampai seluruh simbol telah habis dan membentuk simbol ABCD yang memuat semua simbol yang ada. Untuk memastikan apakah pohon Huffman tersebut benar, simbol ABCD haruslah memiliki peluang sebesar 1 (7/7).

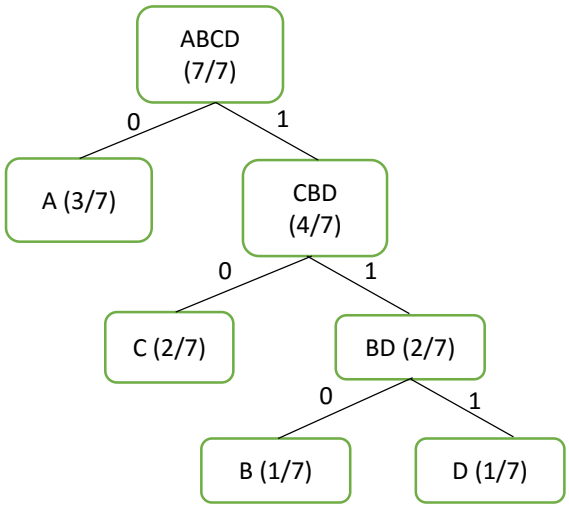


Diagram 2. Pohon Huffman untuk string 'ABCAACD'

Jadi dari pohon Huffman tersebut didapatkanlah kode huffmannya sebagai berikut:

Simbol	Kode Huffman
A	0
B	110
C	10
D	111

Table 3. Kode Huffman untuk string 'ABCAACD'

Setelah didapatkannya kode Huffman, maka dapat disusun string ‘ABCAACD’ menjadi kode biner *0110100010111* Panjang kode Huffman yang didapatkan adalah 13 bit. Sehingga rasio yang didapatkan adalah 13/56 = 23.2%. Dapat dilihat juga bahwa simbol dengan kemunculan yang paling tinggi memiliki kode Huffman yang paling pendek yaitu simbol A dengan panjang kodenya 1 saja. Sedangkan simbol dengan kemunculannya paling sedikit memiliki kode Huffman yang paling panjang yaitu simbol B dan D yang memiliki panjang kode sebesar 3.

C. Run-length Encoding

Run-length Encoding (RLE) adalah suatu bentuk dasar dalam kompresi data yang bersifat *lossless.* Sebuah nilai dari data yang sama dan berulang dapat direpresentasikan dengan nilai dari data tersebut serta jumlah kemunculannya. Metode *RLE* ini sangat efektif untuk data yang mengandung banyak nilai yang sama dan berurutan. Sebagai contoh, *string* 'aaaabbbbccccddd' dapat direpresentasikan menjadi '5a4b3c3d'. Dengan metode ini, string yang awalnya memiliki panjang 15 bisa diperpendek menjadi panjangnya 8. Panjang stringnya menjadi sekitar 50% dari awalnya. Metode ini cukup efektif digunakan dalam kompresi JPEG dan mesin faks.

D. JPEG (*Joint Photographic Experts Group*)

JPEG adalah suatu tipe format gambar digital yang bertipe *bitmap*. Suatu format gambar JPEG terdiri atas beberapa segmen dan masing-masing segmen tersebut ditandai dengan adanya suatu *marker*. Setiap *marker* didahului oleh byte 0xFF lalu dengan byte yang menandakan jenis dari *marker* tersebut. Berikut adalah tabel daftar *marker* tersebut.

Nama	*Bytes*
SOI (Start of Image)	0xFF, 0xD8
SOF0 (Start of Frame Baseline DCT)	0xFF, 0xC0
SOF2 (Start of Frame Progressive DCT)	0xFF, 0xC2
DHT (Define Huffman Table)	0xFF, 0xC4
DQT (Define Quantization Table)	0xFF, 0xDB
DRI (Define Restart Interval)	0xFF, 0xDD
SOS (Start of Scan)	0xFF, 0xDA
RSTn (Restart)	0xFF, 0xD*n* (n=0-7)
APPn (Application-specific)	0xFF, 0xE*n*
COM (Comment)	0xFF, 0xFE
EOI (End of Image)	0xFF, 0xD9

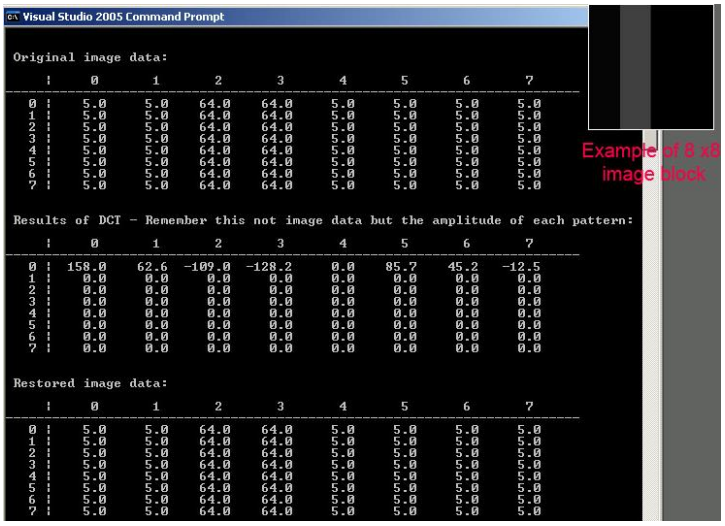
Table 4. Marker JPEG
http://www.digicamsoft.com/itu/itu-t81-36.html

Proses kompresi pada JPEG ada beberapa tahap yaitu:

1. Representasi warna pada gambar dalam RGB diubah menjadi $Y'C_BC_R$, terdiri dari satu komponen *luma* yaitu Y' yang merepresentasikan kecerahan gambar dan dua komponen *chroma* yaitu C_B dan C_R yang merepresentasikan warna dari gambar tersebut. RGB dapat diubah menjadi $Y'C_BC_R$ dengan persamaan berikut:
   Y = 0.299 R + 0.587 G + 0.114 B
   Cb = - 0.1687 R - 0.3313 G + 0.5 B + 128
   Cr = 0.5 R - 0.4187 G - 0.0813 B + 128

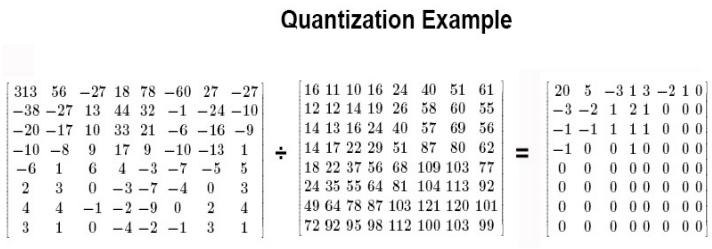
2. Resolusi dari data chroma direduksi atau *downsampling* menggunakan faktor 2 atau 3, ini dilakukan karena mata manusia tidaklah terlalu peka terhadap warna yang detil sekali.

3. Komponen *luma* dan *chroma* pada Gambar dipisah menjadi suatu matriks yang merepresentasikan 8x8 pixel. Matriks ini dilakukan suatu pemrosesan yang dinamakan *Discrete Cosine Transformation* (DCT). Proses ini menghasilkan suatu keluaran seperangkat data yang terdiri dari 64 nilai. Setiap nilai ini merepresentasikan daya (*strength*) dari masing-masing komponen frekuensi dimulai dari tingkat yang paling stabil yaitu DC atau rata-rata dari semua pixel dan pada setiap langkah selanjutnya adalah peningkatan frekuensi atau kekerapan dalam arah sumbu x dan y pada matriks tersebut. Berikut adalah contoh hasil DST.



Gambar 1. Contoh hasil DST
http://www.robertstocker.co.uk/jpeg/DCT_1.jpg

4. Langkah selanjutnya dinamakan *Quantization*, pada proses ini dilakukan kompresi yang bersifat *lossy*. Matriks 8x8 yang didapatkan dari hasil DST akan dibagi dengan suatu matriks kuantisasi 8x8, masing-masing nilai pada matriks kuantisasi dipilih untuk menjaga informasi data pada frekuensi rendah dan mereduksi komponen dengan frekuensi yang tinggi. Hal ini dilakukan karena mata manusia lebih peka dengan variasi warna atau cahaya yang lebih sedikit. Pada proses ini juga bisa ditentukan seberapa besar kompresi ingin dilakukan, semakin besar tingkat kompresinya maka detil dan kualitas dari gambar akan berkurang. Berikut adalah contoh proses kuantisasi.

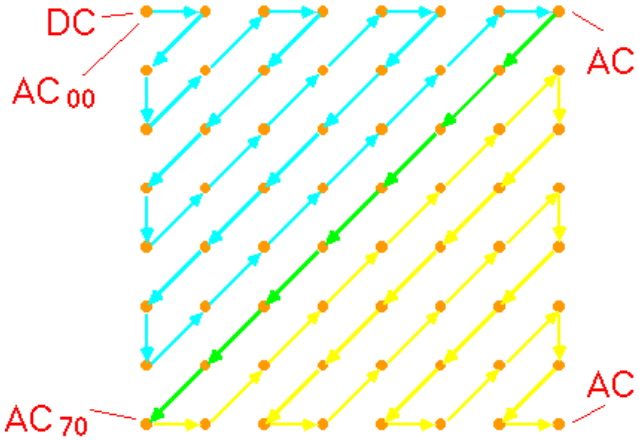


Gambar 2. Contoh hasil quantization
http://www.robertstocker.co.uk/jpeg/quan.gif

5. Langkah terakhir yaitu dilakukanlah kompresi menggunakan kode Huffman.

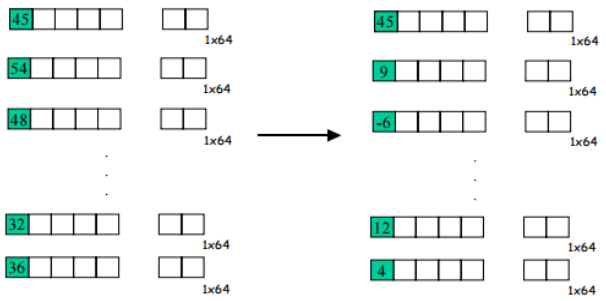
III. Aplikasi Kode Huffman dalam Kompresi JPEG

Setelah mendapatkan matriks 8x8 dari hasil kuantisasi, dapat dilihat bahwa terdapat banyak angka 0. Kode Huffman mengambil keuntungan dari ini karena kode Huffman sangat efektif dalam mengompresi data yang terdapat banyak redundansi. Pertama-tama dari matriks 8x8 tersebut dikonversi menjadi suatu vektor 1x64 dengan melakukan yang namanya *zig-zag scan*. Ilustrasinya dapat dilihat pada gambar berikut.



Gambar 3. zig-zag scan
http://xputers.informatik.uni-kl.de/faq-pages/cjpeg.gif

Zig-zag scan ini dipilih karena akan mengelompokkan frekuensi rendah pada bagian atas vektor dan frekuensi tinggi pada bagian bawah vektor. Pada bagian komponen DC, nilainya cukup besar dan berbeda-beda namun perbedaan antar blok dengan blok sebelumnya kecil. Jadi dilakukan suatu proses DPCM (*Differential Pulse Code Modulation*). Jika angka-angka yang didapatkan setelah proses DPCM ini acak dan tidak menentu, maka kompresi dengan kode Huffman akan menjadi tidak berguna.



Gambar 4. Contoh DPCM
http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M4L1.pdf

Pada komponen DC dibedakan lagi menjadi (*SIZE*, *Value*). Kode untuk *value* atau nilai dapat dilihat dari tabel berikut.

SIZE	Value	Code
0	0	---
1	-1,1	0,1
2	-3, -2, 2,3	00,01,10,11
3	-7,..., -4, 4,..., 7	000,..., 011, 100,...111
4	-15,..., -8, 8,..., 15	0000,..., 0111, 1000,..., 1111
.		.
.		.
11	-2047,..., -1024, 1024,... 2047	...

Table 5. Kode pada komponen DC
http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M4L1.pdf

Untuk kode ini dipakai *ones complement*, sehingga untuk angka negatif dari sebuah angka dilakukan bitwise operator. Sebagai contoh angka desimal 2 adalah 10 dan untuk -2 adalah 01 begitu juga untuk 4 adalah 100 dan 011 untuk -4.

SIZE	Code Length	Code
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

Table 6. Kode Huffman untuk Size pada komponen DC
http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M4L1.pdf

Tabel Huffman ini merupakan salah satu kode yang disarankan dalam perbedaan DC. Sebagai contoh apabila suatu komponen DC bernilai 42 dan komponen sebelumnya adalah 48, sehingga perbedaannya adalah sebesar -6. Sehingga dari komponen tersebut dapat dikodekan sebagai 100 011. Dengan 011 adalah perbedaan nilanya yaitu -6 dan sehingga didapatkan size sebesar 3 yang dikodekan 100. Hal ini dilakukan sampai semua komponen telah diproses yaitu sampai ditemukannya *End of Block*. Pada komponen AC dilakukan dengan kompresi berbeda yaitu RLE (*Run-length Encoding*), metode ini dipilih karena biasanya komponen AC mengandung banyak nilai 0 dan berurutan. Kompresi menggunakan kode Huffman lebih dipilih daripada menggunakan metode aritmetik karena metode Huffman membutuhkan waktu yang jauh lebih kecil untuk mengompresi. Kedua metode ini adalah metode kompresi yang bersifat *lossless*. Dengan kompresi menggunakan kode Huffman, besar file gambar dapat direduksi sebesar 28%.

IV. KESIMPULAN

Di zaman yang serba digital ini, diperlukan kompresi data sehingga dapat melakukan penghematan memori. Kode Huffman merupakan metode kompresi yang cukup baik karena bersifat *lossless* dan cukup baik digunakan dalam format JPEG karena redundansi data yang umum ditemukan. Proses kompresi JPEG sendiri terdapat beberapa tahapan, yaitu pengubahan dari format RGB menjadi $Y'C_BC_R$, lalu dilakukan DCT (*Discrete Cosine Transformation*) dan *Quantization*. Proses DCT dan Quantization ini adalah proses kompresi yang bersifat *lossy*. Sehingga langkah terakhir yaitu dikompresi dengan kode Huffman untuk komponen DC dan RLE (Run-length Encoding) untuk komponen AC. Kedua metode kompresi ini bersifat *lossless*. Metode Huffman ini lebih dipilih dibandingkan dengan metode kompresi aritmetik karena waktu pengompresiannya yang secara signifikan jauh lebih kecil.

REFERENSI

[1] http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M4L1.pdf diakses pada 9 Desember 2015.

[2] http://arxiv.org/ftp/arxiv/papers/1109/1109.0216.pdf diakses pada 9 Desember 2015.

[3] Munir, Rinaldi. "Diktat Kuliah IF2120 Matematika Diskrit" edisi keempat. Program Studi Teknik Infomatika STEI ITB. 2006.

[4] http://www.robertstocker.co.uk/jpeg/ diakses pada 8 Desember 2015.

[5] http://www.impulseadventure.com/photo/jpeg-huffman-coding.html diakses pada 9 Desember 2015.

[6] http://web.stanford.edu/class/ee398a/handouts/lectures/08-JPEG.pdf diakses pada 8 Desember 2015.

[7] http://jpeg.org/jpeg/ diakses pada 8 Desember 2015.

[8] http://www-i6.informatik.rwth-aachen.de/web/Misc/Coding/365/li/material/notes/Chap4/Chap4.2/Chap4.2.html diakses pada 9 Desember 2015.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 10 Desember 2015

Albert Logianto
13514046