COMPRESSING JPEG USING HUFFMAN CODE

Andika Kurniawan Susilo – NIM : 13506104
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail :andika_k_s@students.itb.ac.id

Abstrak

Makalah ini membahas tentang tipe penyimpanan file dalam bentuk JPEG dengan menggunakan teknik HUFFMAN CODE.Sebagian besar foto digital merupakan images full-color natural maupun dari benda hidup, dapat diartikan bahwa di dalam gambar tersebut terdapat seluruh komponen images (one luminances and color channels) yang kesemuanya mempunyai isi dengan frekuensi tinggi dan rendah. Sehingga sebagian besar foto digital menggunakan chroma subsampling yang membuat proses ekstraksi sedikit lebih rumit, untuk mengatasi hal itu maka digunakanlah huffman code. Sedangkan untuk proses penyimpanan, prosesnya dibagi menjadi 3 bagian, Decoding, Encoding, Convert to Binary.

Kata Kunci : HUFFMAN CODE, JPEG decoder , Binary Tree , Small Image, Solid color, Grayscale, No chroma subsampling , adding a stuff byte, Luminance, Chrominance

Pendahuluan

Apakah itu Huffman Coding?
Huffman Coding adalah metode yang digunakan untuk mengambil simbol dan mengkodekannya dengan variabel panjang kode yang dibuat menurut stastitical probabilities. Simbol yang sering muncul akan dikodekan dengan kode yang sama dan yang hanya mengambil tempat beberapa bits saja, sedangkan simbol yang jarang digunakan direpresentasikan oleh simbol yang membutuhkan lebih banyak bit untuk dikodekan. Huffman Code yang terdiri dari n x m x 3A dapat mengurangi tempat penyimpanan yang seharusnya dibutuhkan. Nilai yang dihsilkan akan mempunyai rata rata:

pi log2 pi bits (1)

Dimana pi adalah probabilitas daro intensitas nilai i, dimana i bernilai diantara 0 sampai 225

JPEG file terdiri dari 4 Huffman tabel yang menghubungkan antara panjang kode (yang berada diantara 1 sampai 16 bits) dan nilai/isi dari kode (yang mempunyai panjang 8 bits). Dalam pembuatan tabel secara umum melibatkan penghitungan seberapa sering kode tersebut muncul. (DCT code word) dalam sebuah gambar, sehingga harus mengalokasikan string bits. Tetapi sebagian besar encoder menggunakan huffman tabel untuk merepresentasikan standar JPEG. Sehingga untuk mengoptimalkan huffman tabel digunakanlah binary tree,
contoh-contoh dari JPEG images :

Grayscale – bayangan dari warna hitam ke putih/ diantara 2 warna putih dan hitam.
Solid color– membuat semua isi dari pixel 8x8 block mempunyai warna yang sama, disini tidak ada komponen AC .
No chroma subsampling – membuat scan data extraction lebih simpel: Y, Cb, Cr, Y, Cb, Cr, etc.
Small Image - Total image size adalah 16x8 = dua MCUs atau blocks. .

Untuk perbandingannya, original image (16 pixels x 8 pixels) mengandung total 128 pixels dengan 8 bits per channel (RGB). Untukk uncompressed image size mempunyai besar 384 bytes (128 pixels x 8 bits/channel x 3 channels x 1 byte/8bits). Sebetulnya dalam format GIF akan dihasilkan lebih banyak compression dengan panjang 22 bytes, tetapi tidak dapat digunakan untuk gambar organik.

Table 1 . Perbandingan Beberapa Jenis Tempat Penyimpanan Gambar

FileFormat	Total Size	Overhead Size	Image Content Size
BMP (Uncompressed)	440 Bytes	56 Bytes	384 Bytes
JPEG	653 Bytes	644 Bytes	9 Bytes
JPEG(Optimized)	304 Bytes	297 Bytes	7 Bytes
GIF	60 Bytes	38Bytes	22 Bytes

Table 2 . Format Pengkodean

Section		1	2	3	4	5	6
Component	Y		Cb		Cr		
AC / DC		DC	AC	DC	AC	DC	AC

Table 3 . Hex String

1111	1100	1111	1111	1110	0010	1010	1111	1110	1111	1111	0011	0001	0101	0111	1111
0	AC		DC		AC		DC		AC		DC		AC		0

II. **Decoding**

Untuk membantu dalam pencegahan data corruption, JPEG standar menggunakan JPEG markers untuk dimunculkan dalam huffman-coded scan data segment. Tetapi JPEG decoder harus membaca untuk setiap maker (yang mempunyai nilai 0xFF byte, yang diikuti oleh non zero byte). Sedangkan jika menggunakan Huffman Code hanya perlu membaca oxFF byte, lalu dalam penulisannya nanti akan diikuti oleh 0x00, proses ini dinamakan adding a stuff byte. Untuk tujuan extraction, akan dilakukan replacement untuk setiap padding bytes (0xFF00 dengan 0xFF).

Contoh data yang telah dibaca

```
0270h: CA DA EA FA FF DA 00 0C 03 01 00 02 11 03 11 00
0280h: 3F 00 FC FF 00 E2 AF EF F3 15 7F FF D9
```

Untuk setiap MCU, tanpa chroma subsampling, data akan dikodekan dengan format seperti pada tabel 2.

Seperti yang sudah diketahui isi dari image terdiri dari 3 componen (Y, Cb, Cr). Dengan setiap komponen, tempat penyimpanannya selalu ada di dalam DC value dan akan diikuti oleh 63 AC values seperti pada tabel 2.
Hex string yang dihasilkan dari pengkodean ini dapat ditampilkan dalam binary seperti pada tabel 3.

Block yang pertama dan yang terakhir mempunyai nilai 0 yang diubah dalam binary.
Sedangkan untuk block yang lainnya adalah DC yang selalu diapit oleh AC yang telah diubah ke dalam binary dengan nilai maksimum 63.

III. **Encoding**

Table 4 - Huffman - Luminance (Y) - DC

3 bits	000	04
	001	05
	010	03
	011	02
	100	06
	101	01
	110	00 (End of Block)
4 bits	1110	07
5 bits	1111 0	08
6 bits	1111 10	09
7 bits	1111 110	0A

Table 5 - Huffman - Luminance (Y) - AC

Length	Bits	Code
2 bits	00	01
	01	02
3 bits	100	03
4 bits	1010	11
	1011	04
	1100	00 (End of Block)
5 bits	1101 0	05
	1101 1	21
	1110 0	12
6 bits	1110 10	31
	1110 11	41
	...	...
12 bits	1111 1111 0011	F0 (ZRL)
	...	...
	...	
16 bits	1111 1111 1111 1110	FA

Table 7 - Huffman - Chrominance (Cb & Cr) - AC

Length	Bits	Code
2 bits	00	01
	01	00 (End of Block)
3 bits	100	02
	101	11
4 bits	1100	03
5 bits	1101 0	04
	1101 1	21
	1110 00	12
	1110 01	31
	1110 10	41
	...	...
	1111 1100 0	F0 (ZRL)
	...	...
16 bits	...	
	1111 1111 1111 1110	FA

Table 6 - Huffman - Chrominance (Cb & Cr) - DC

Length	Bits	Code
2 bits	00	01
	01	00 (End of Block)
3 bits	100	02
	101	03
	1100	04
	1101	05
	1110	06
5 bits	1111 0	07
6 bits	1111 10	08
7 bits	1111 110	09
8 bits	1111 1110	0A
9 bits	1111 1111 0	0B

Table 8 - Huffman DC Value Encoding

DC Code	Size	Additional Bits		DC Value	
01	1	0	1	-1	1
02	2	00,01	10,11	-3,-2	2,3
03	3	000,001,010,011	100,101,110,111	-7,-6,-5,-4	4,5,6,7
04	4	0000,...,0111	1000,...,1111	-15,...,-8	8,...,15
05	5	0 0000,...	...,1 1111	-31,...,-16	16,...,31
06	6	00 0000,...	...,11 1111	-63,...,-32	32,...,63
07	7	000 0000,...	...,111 1111	-127,...,-64	64,...,127
08	8	0000 0000,...	...,1111 1111	-255,...,-128	128,...,255
09	9	0 0000 0000,...	...,1 1111 1111	-511,...,-256	256,...,511
0A	10	00 0000 0000,...	...,11 1111 1111	-1023,...,-512	512,...,1023
0B	11	000 0000 0000,...	...,111 1111 1111	-2047,...,-1024	1024,...,2047

Perlu diketahui bahwa representasi dari DHT di dalam JPEG hanya berupa list dari Length dan Code Value, seperti pada tabel 4, 5,6, dan 7. pada tabel 8 ditampilkan perubahan pada data DC kedalam signed decimal equivalent. Tabel ini dimulai dari nilai di dalam DC lalu besar DC Code dan besar bits yang akan ditampilkan.

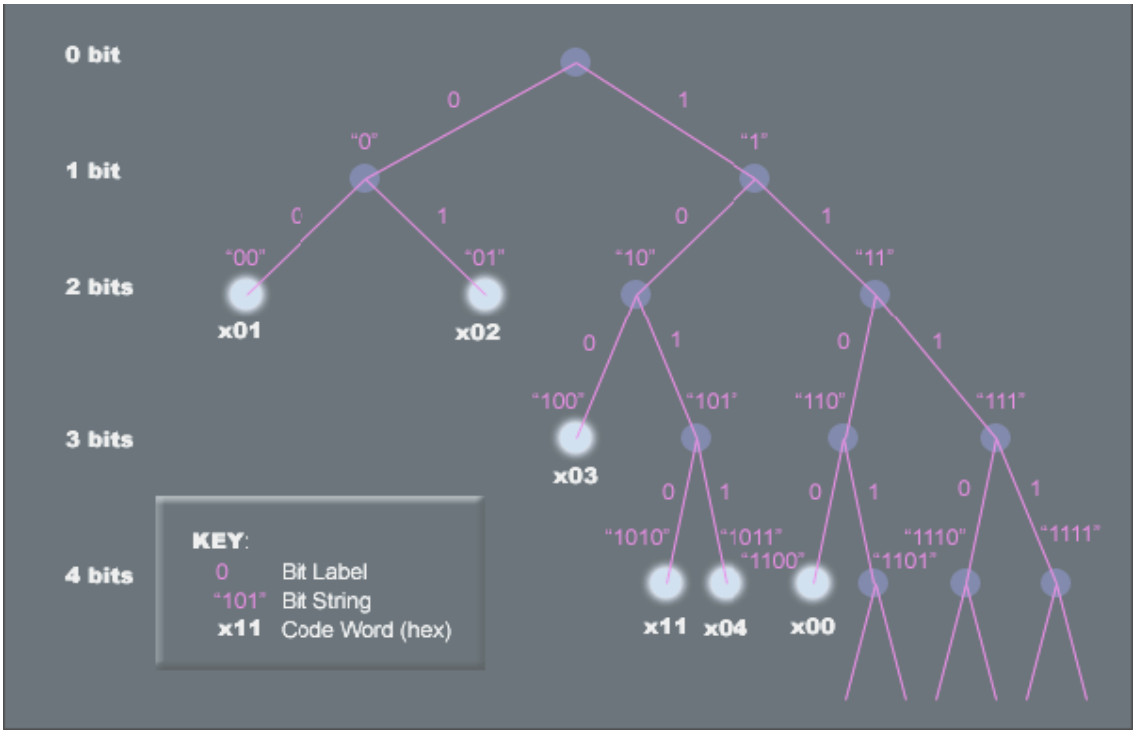
DHT table yand ada di dalam JPEG hanya untuk mendefinisikan penomoran dari setiap bit string length, yang diikuti oleh semua code words. Sedangkan untuk merepresentasikan setiap kode bukanlah tugas dari DHT melainkan JPEG Decoder.

```
Class = 1 (AC Table)
Destination ID = 0
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 11 04 00
  Codes of length 05 bits (003 total): 05 21 12
  Codes of length 06 bits (002 total): 31 41
  Codes of length 07 bits (004 total): 51 06 13 61
  Codes of length 08 bits (002 total): 22 71
  Codes of length 09 bits (006 total): 81 14 32 91 A1 07
  Codes of length 10 bits (007 total): 15 B1 42 23 C1 52 D1
  Codes of length 11 bits (003 total): E1 33 16
  Codes of length 12 bits (004 total): 62 F0 24 72
  Codes of length 13 bits (002 total): 82 F1
  Codes of length 14 bits (006 total): 25 43 34 53 92 A2
  Codes of length 15 bits (002 total): B2 63
  Codes of length 16 bits (115 total): 73 C2 35 44 27 93 A3 B3 36 17 54 64 74 C3 D2 E2
                                       08 26 83 09 0A 18 19 84 94 45 46 A4 B4 56 D3 55
                                       28 1A F2 E3 F3 C4 D4 E4 F4 65 75 85 95 A5 B5 C5
                                       D5 E5 F5 66
```

IV. **Convert to Binary**

Banyak cara untuk menampilkan binary strings ke dalam decoder dan kebanyakan tidak optimal performansinya dan implementasinya sulit untuk dipelajari. Huffman Binary Tree adalah salah satu cara terbaik untuk JPEG Decoder. Binary Tree adalah susunan dari akar dan mempunyai cabang kanan dan kiri.

Lalu bagaimana membuat binary bit strings untuk setiap kode?
Pembuatan Binary Tree dimulai dari pembuatan cabang baru dan menaruh kode didalam daun. Level 0 berisi kode yang mempunyai besar 1 bit, level 1 berisi kode yang mempunyai besar 2 bit dan seterusnya.

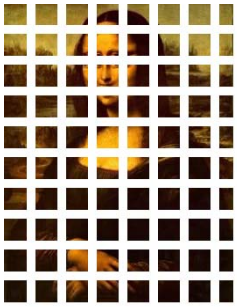


Setelah Binary Tree selesai dibuat , dapat dibaca kembali setiap kode yang telah ditulis dalam setiap akar dan daun. Misalnya code word 0x04 dikodekan dengan bit string 1011, untuk mengaksesnya dapat melalui tahap berikut: level satu kita ambil nilai 1, level 2 kita ambil nilai 0, dan seterusnya.

```
Codes of length 02 bits:
00 = 01
01 = 02
Codes of length 03 bits:
100 = 03
Codes of length 04 bits:
1010 = 11
1011 = 04
1100 = 00 (EOB)
Codes of length 05 bits:
11010 = 05
11011 = 21
11100 = 12
Codes of length 06 bits:
111010 = 31
111011 = 41
Codes of length 07 bits:
1111000 = 51
...
Codes of length 15 bits:
111111111000100 = B2
111111111000101 = 63
Codes of length 16 bits:
1111111110001100 = 73
1111111110001101 = C2
...
1111111111111100 = DA
1111111111111101 = EA
1111111111111110 = FA
```

Contoh gambar dalam bentuk penyimpanan JPEG:

Gambar2. Original



Gambar3. Gambar dipecah dalam beberapa block

Gambar3. Hasil dari JPEG

V. KESIMPULAN

1. Gambar dapat disimpan dalam beberapa jenis tipe file seperti GIF, JPEG, BMP, dll.
2. Untuk gambar dengan jenis natural dan organik yang mempunyai variasi warna yang banyak tipe file untuk tempat penyimpanan terbaik adalah dalam bentuk JPEG.
3. File JPEG dapat dioptimalisasi agar lebih kecil ukurannya dengan menggunakan Huffman Code.
4. Dalam proses penyimpanan, file JPEG prosesnya dibagi menjadi 3 bagian, Decoding, Encoding, Convert to Binary.

VI. DAFTAR PUSTAKA

[1] Pigeon , Steven. *Hu_man Coding*.Universit´e de Montr´eal

[2] Ellis, Dan(2006).Columbia University Dept of Electrical Engineering. http://www.ee.columbia.edu/~dpwe/e6820/.Tanggal akses: 1 Januari 2007 pukul 22:00.

[3] Munir, Rinaldi. (2004). *Bahan Kuliah IF2153 Matematika Diskrit. Departemen Teknik Informatika,* Institut TeknologiBandung.

[4] Ammeraal, Leendert. (1996). *Algorithms and data structure in C++*. New York, NY: John Wiley & Sons.

[5] Berghel, Hal and Roach, David. (1996). *An extension of Ukkonen's enhanced dynamic programming ASM algorithm*.http://www.acm.org/~hlb/publications/asm/asm.html

[6] Lelewer, Debra A. and Daniel S. Hirschberg. "Data Compression". AMC Computing Survey. Vol. 19, No. 3. September 1987.

[7] Nelson, Mark. *The Data Compression Book*. M & T Publishing Inc.: NY. 1992.

[8] Nelson, Mark. (1996*). Priority Queues and the STL*. Dr. *Dobb's journal*. Retrieved on June 16, 2003 at: