Pembuatan Aplikasi Kompresi Image dengan Metode *Fast Discrete Cosine Transform*

Liliana, Andy Febrico Bintoro, Iwan Njoto Sandjaja
Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
lilian@petra.ac.id

Abstract

The development of digital imaging on a fast digital cameras require greater storage utilization for a given image. The resulting quality is better, but also create new problems which the amount of storage required.

With greater storage, data exchange is also increasingly difficult. The facilities provided were usually limit the storage to exchange data, even without the restriction would be quite troublesome in the exchange of data. It required a compression method which can minimize the use of storage but also still maintain image resolution and quality image.

Compression is a way to overcome this problem. Image compression is generally divided into two kinds, lossless compression and lossy compression. The method is usually used in standard JPEG is a lossy compression that is by eliminating some image information and exploit weaknesses in the tolerance of the human eye to color.

Programs that are created using lossy image compression and after the test successfully met the standard JPEG, fast in terms of computation and produces good output. This program uses computational algorithms and optimization fDCT other so that the computing speed is faster than usual. Tests conducted by IJG and the results made programs more quickly in terms of computation, while the resulting quality of compression is inversely proportional to the desired data.

Keywords:

Compression, Image Processing, DCT, fDCT, YUV, Discrete Cosine Transform, Quantization, Digital Camera.

1. LATAR BELAKANG

Perkembangan *digital imaging* pada kamera digital yang cepat membutuhkan semakin besarnya pemanfaatan *storage* untuk suatu *image*. Dengan kamera digital 10 *Megapixel*, dapat dihasilkan *image* dengan resolusi 3648 x 2736 *pixel*. Satu *image* menyimpan informasi warna yang berupa RGB sehingga jika masing-masing disimpan dalam *byte*, maka jumlah minimal *byte* yang dibutuhkan adalah 3648 x 2736 x 3 yang berarti 28,56 *Megabyte*. Jika gambar ini dimasukkan dalam format *image* yang berupa BMP, maka hasilnya akan lebih besar lagi karena ditambahkan berbagai informasi. Kualitas yang dihasilkan semakin baik, tetapi membutuhkan *storage* yang semakin besar.

Dengan *storage* yang semakin besar, pertukaran data juga semakin sulit, terutama pertukaran data dengan menggunakan *device* yang berkapasitas kecil. *Device* yang ada biasanya membatasi *storage* untuk melakukan pertukaran data. Untuk itu diperlukan metode kompresi yang dapat meminimalkan penggunaan *storage* tetapi juga tetap menjaga resolusi *image* dan kualitas *image* tersebut.

Kompresi *image* secara umum dibedakan menjadi dua macam, *lossless compression* dan *lossy compression*. Metode yang biasanya dipakai dalam standard JPEG adalah *lossy compression* yaitu dengan menghilangkan sebagian informasi gambar dan memanfaatkan kelemahan dalam ketidakpekaan mata manusia dalam mengenali gradasi warna.

2. JPEG DAN KOMPRESI IMAGE PADA JPEG

Standard format *image compressed* yang paling umum saat ini adalah JPEG. Standard ini dibuat oleh sebuah group dari *International Organization for Standardization* (ISO) yang secara informal disebut Joint Photographic Experts Group dan CCITT yang mengembangkan standard *photographic image* untuk telepon, radio, televisi, dan media komunikasi elektronik lainnya[1].

Sistem visual manusia mempunyai beberapa keterbatasan, yang dimanfaatkan JPEG untuk mendapatkan kompresi dengan *rate* yang tinggi. Sistem mata-otak manusia tidak dapat melihat baik secara detil. Jika ada banyak perubahan yang terjadi dalam beberapa *pixel*, *image* terlihat dalam *high spatial frequency* (perubahan dalam *color model* asli, dalam hal ini RGB) yang merupakan perubahan dalam bidang (x,y). Keterbatasan ini lebih terlihat pada *image* bewarna daripada hitam-putih. Karena itu, informasi *image* di JPEG *decimated* (secara parsial diturunkan atau dibuat rata-rata) dan kemudian blok-blok dari *image* ini direpresentasikan dalam *spatial frequency domain* (u dan v yang merupakan hasil konversi *color model*) dan bukan dalam (x,y). Kemudian kecepatan perubahan dalam x dan y dievaluasi, dari rendah ke tinggi, dan *image* yang baru dibentuk dengan mengelompokkan koefisien *weights* dari kecepatan ini. Dengan melakukan pembualatan pada nilai w*eights*, didapatkan kompresi dengan rate tinggi sekaligus presisi data yang hilang. Oleh karena itu, metode ini termasuk *lossy.*

Sampai saat ini JPEG merupakan format kompresi yang paling baik untuk *image*. Kelemahannya adalah metode kompresi dan dekompresinya lambat, standar masih terus berkembang, banyak implementasi yang tidak *compatible* karena ada banyaknya pilihan yang ditawarkan dalam standardnya, dan dalam implementasi awal ada banyak perbedaan algoritma pembuatan.

Meskipun JPEG masih merupakan *standard* kompresi untuk *image*, tetapi perkembangannya sangat lambat karena perkembangan *image compression* saat ini lebih dititik beratkan pada JPEG2000 yang berhasil mengatasi beberapa kelemahan JPEG[1].

2.1. Image Compression

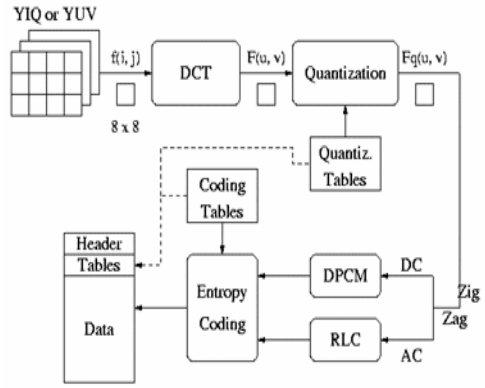
Image compression adalah suatu metode untuk memperkecil ukuran suatu *image* yang biasanya dimaksudkan untuk menghemat media penyimpanan [2]- [4].

Compression dikategorikan ke dalam dua bagian besar, yaitu *lossless compression* dan *lossy compression* [2],-[4]. Penggunaan dari *lossless compression* adalah untuk data yang sangat penting, yang tidak mengijinkan adanya perubahan dalam data. *Lossy compression* digunakan untuk data yang bisa dihilangkan sebagian informasinya karena dianggap tidak terlalu penting dan tidak banyak berpengaruh terhadap sistem keseluruhan.

Dalam *image compression*, biasanya digunakan *lossy compression* karena untuk membuat ukuran yang kecil diperlukan untuk menghilangkan atau mengubah sebagian informasi tentang warna yang ada kemudian dalam merangkainya kembali informasi yang hilang tersebut tidak akan kembali [1], [2]. *Image compression* berani menggunakan metode *lossy* karena mata manusia cenderung menganggap sama warna yang RGB-nya hanya beda kecil. Misal dalam suatu pixel R asli nilainya 200, kemudian diubah nilainya menjadi 170, maka mata manusia belum tentu dapat mengenali perubahannya[1].

2.2. JPEG Image Compression

- JPEG menggunakan *lossy image compression*. Adapun langkah-langkah utama dalam membuat kompresi ini adalah seperti tampak pada gambar 1.



Gambar 1. *Block Diagram* Kompresi JPEG [1]

2.3. DCT

DCT merupakan teknik transformasi yang paling banyak digunakan yang dapat melakukan dekorelasi dari *input signal* dalam *data independent* [5]-[8]. DCT mengubah array data intensitas menjadi array data frekuensi untuk mengetahui seberapa cepat persebaran intensitas[1],[5]-[8]. JPEG menggunakan DCT untuk mengolah tiap 8x8 *pixel* data. Untuk *image* dengan banyak komponen warna (YUV misalnya), maka DCT diterapkan pada 8x8 pixel untuk setiap komponen (dalam hal ini pada masing-masing Y, U, dan V)

Persamaan dasar 2D DCT adalah sebagai berikut:

$$F_{x,y} = \frac{C(x)C(y)}{4}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} f_{i,j}\cos\left[\frac{(2i+1)x\pi}{2N}\right]\cos\left[\frac{(2j+1)y\pi}{2M}\right] \quad (1)$$

Persamaan 2D DCT untuk 8x8 standar JPEG didefiinisikan M = N = 8, sehingga persamaan DCT untuk JPEG adalah persamaan 2 sedangkan proses inversnya menggunakan persamaan 3:

$$F(u,v) = \frac{C(u)\,C(v)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} f(i,j)$$

(2)

$$\tilde{f}(i,j) = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C(u)\,C(v)}{4} \cos\frac{(2i+1)u\pi}{16} \cos\frac{(2j+1)v\pi}{16} F(u,v)$$

(3)

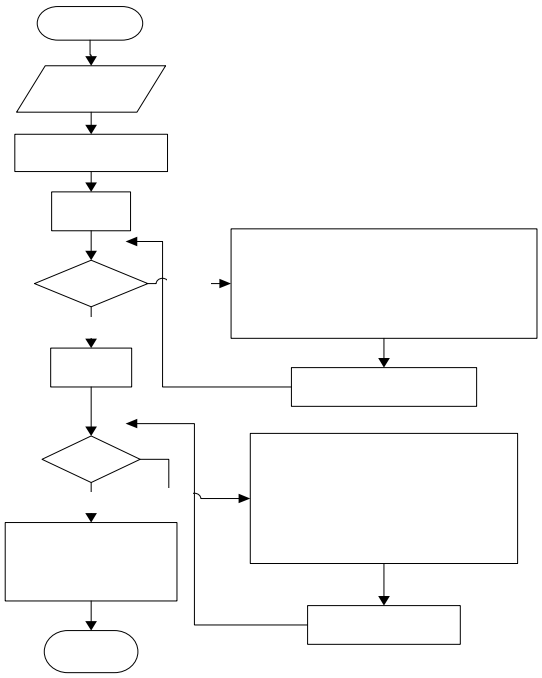
Dimana:

$$C(n) = \begin{cases} \frac{1}{\sqrt{2}}, & n = 0 \\ 1, & n \neq 0 \end{cases} \quad ; \quad n = x, y$$

(4)

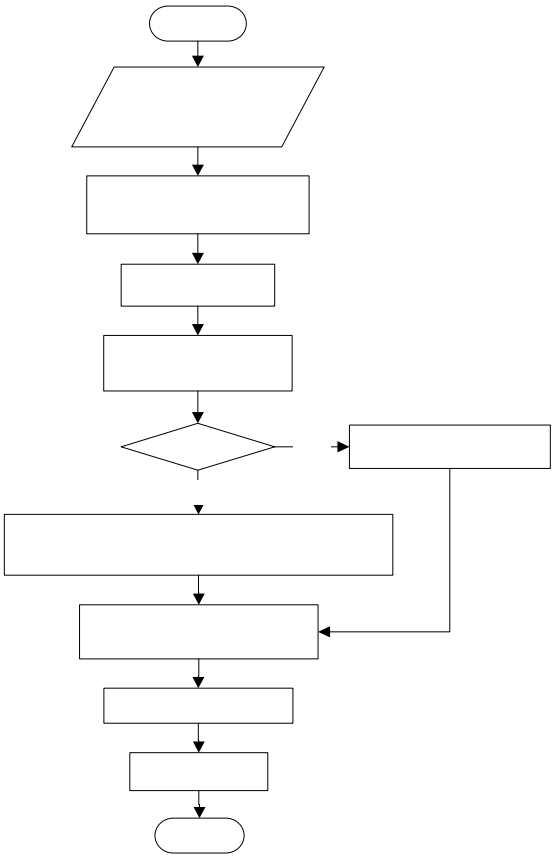
3. *FLOWCHART*

2.4. *Flowchart* Fast Discrete Cosine Transform



Gambar 2. *Flowchart* proses DCT

2.5. *Flowchart* Proses Encoding



Gambar 3. *Flowchart* proses *encoding*

Proses ini membutuhkan *input* berupa data *image* ukuran 8 x 8, tabel kuantisasi *standard* (fdTable), *standard variable* untuk DC, tabel Huffman untuk DC (HTDC) dan AC (HTAC). Kemudian dilakukan proses fDCT dan hasilnya dimasukkan dalam *zigzag encoding* untuk selanjutnya ditulis dalam file berdasarkan standard JPEG.

4. UJI KUALITAS DAN KECEPATAN

Implementasi perangkat lunak yang dikembangkan diuji dengan cara dibandingkan dengan perangkat lunak dengan menggunakan metode yang lain. Perbandingan meliputi dalam kecepatan, rasio kompresi, dan SNR. Aplikasi lain yang ingin diuji adalah IrfanView, tetapi karena bahasa pemrograman yang digunakan berbeda dan IrfanView menggunakan *library* lain untuk melakukan kompresi, maka diputuskan untuk menggunakan *library* yang digunakan IrfanView dalam proses kompresinya. *Library* yang digunakan IrfanView dalam proses kompresinya adalah *Independent JPEG Group* (yang untuk selanjutnya disingkat IJG). IJG pada mulanya ditulis menggunakan bahasa C tetapi kemudian

Start

Data, fdTable

DataOffset = 0

I = 0

dikembangkan oleh BitMiracle menggunakan bahasa C#. Akhirnya, diputuskan untuk menggunakan *library* BitMiracle sebagai aplikasi pembanding pengganti IrfanView dan sama-sama ditulis dalam bahasa yang sama sehingga pembandingan akan lebih setara.

Pengujian dilakukan pada kelompok gambar beresolusi rendah dan resolusi tinggi. Gambar beresolusi kecil adalah gambar yang memiliki resolusi di bawah 600x600 *pixel*. Gambar beresolusi besar adalah gambar yang memiliki resolusi di atas 1600x1600 *pixel*.

Pengujian berikutnya dilakukan berdasarkan warna dominan pada gambar. Warna dominan gambar yang akan diuji meliputi gambar yang memiliki kecenderungan warna merah, hijau, biru, dan warna bervariasi.

Seluruh pengujian dilakukan menggunakan Notebook HP Pavilion dv4 Core i5 M430 2.27 GHz dengan RAM 3GB. Pengujian dilakukan dengan kualitas 50% pada masing-masing implementasi perangkat lunak.

Hasil pengujian terhadap gambar berdimensi kecil ditampilkan dalam Tabel 1 sedangkan hasil pengujian menggunakan *library* IJG ditampilkan pada Tabel 2.

Dari Tabel 1 dan 2 terlihat bahwa perbandingan waktu antara program kami dengan aplikasi BitMiracle sangat jauh berbeda. Program kami jauh lebih baik dalam hal waktu. Rasio dan SNR yang dihasilkan berbanding terbalik, semakin besar rasionya maka semakin kecil perbedaan gambar yang dihasilkan dengan gambar aslinya.

Hasil pengujian program yang dibuat menggunakan gambar berdimensi besar ditampilkan dalam Tabel 3, sedangkan hasil pengujian menggunakan *library* IJG ditampilkan pada Tabel 4. Dari kedua table tersebut data yang kami peroleh tetap sama dengan hasil pengujian pada perbandingan antara kedua perangkat lunak pada gambar beresolusi kecil.

Tabel 1. Pengujian gambar berdimensi kecil menggunakan aplikasi

File Name	Dimension	Original Size	Toro Compressor			
			Time	50% Compression Size	Ratio	SNR
Small Red						
apple_red.png	512x512	93433 bytes	0.2180125 s	27735 bytes	0.296844	1.419479
4.2.01.tiff	512x512	786572 bytes	0.2808005 s	42577 bytes	0.05413	0.801857
4.2.04.tiff	512x512	786572 bytes	0.2964005 s	48989 bytes	0.062282	0.829507
Small Green						
hd_green.png	512x512	96822 bytes	0.2140123 s	27689 bytes	0.285978	1.267844
grass.bmp	512x512	786486 bytes	0.2500143 s	140565 bytes	0.178725	0.552011
leaf.png	512x512	434778 bytes	0.2320133 s	58056 bytes	0.13353	0.506991
Small Blue						
2.1.03.tiff	512x512	786572 bytes	0.2850163 s	46789 bytes	0.059485	0.838998
2.1.09.tiff	512x512	786572 bytes	0.2720156 s	43247 bytes	0.054982	1.212545
2.1.10.tiff	512x512	786572 bytes	0.2840163 s	74194 bytes	0.094326	1.168777
Small Colorful						
4.2.03.tiff	512x512	786572 bytes	0.3120005 s	104661 bytes	0.13306	0.702055
4.2.07.tiff	512x512	786572 bytes	0.3120005 s	55832 bytes	0.070981	0.653358
colorful_512x512_clean.bmp	512x512	263224 bytes	0.2390137 s	98087 bytes	0.372637	0.435058

Tabel 2. Pengujian gambar berdimensi kecil menggunakan BitMiracle

File Name	Dimension	Original Size	BitMiracle (IJG for C#)			
			Time	50% Compression Size	Ratio	SNR
Small Red						
apple_red.png	512x512	93433 bytes	1.5020859 s	18433 bytes	0.197286	0.909184
4.2.01.tiff	512x512	786572 bytes	1.6300932 s	20335 bytes	0.025853	0.597532
4.2.04.tiff	512x512	786572 bytes	1.6170925 s	24329 bytes	0.03093	0.687307
Small Green						
hd_green.png	512x512	96822 bytes	1.6470942 s	18142 bytes	0.187375	0.794786
grass.bmp	512x512	786486 bytes	1.7270988 s	78056 bytes	0.099247	0.533538
leaf.png	512x512	434778 bytes	1.521087 s	27705 bytes	0.063722	0.5461
Small Blue						
2.1.03.tiff	512x512	786572 bytes	1.6380936 s	21155 bytes	0.026895	0.692282
2.1.09.tiff	512x512	786572 bytes	1.6390937 s	19441 bytes	0.024716	0.979759
2.1.10.tiff	512x512	786572 bytes	1.6580949 s	35960 bytes	0.045717	1.036902
Small Colorful						
4.2.03.tiff	512x512	786572 bytes	1.6440941 s	50694 bytes	0.064449	0.661532
4.2.07.tiff	512x512	786572 bytes	1.6710956 s	26287 bytes	0.03342	0.568298
colorful_512x512_clean.bmp	512x512	263224 bytes	1.7180983 s	48261 bytes	0.183346	0.406838

Tabel 3. Pengujian gambar berdimensi besar menggunakan aplikasi

File Name	Dimension	Original Size	Toro Compressor			
			Time	50% Compression Size	Ratio	SNR
Large Red						
red car.bmp	2304x1728	11943990 bytes	3.4591978 s	497735 bytes	0.041672	1.278186
red.bmp	2742x1802	14826910 bytes	5.2663012 s	472650 bytes	0.031878	2.632665
postvinyl_duchamp_red.bmp	2362x1772	12559992 bytes	4.3282476 s	271098 bytes	0.021584	5.68732
Large Green						
big_tree.bmp	6088x4550	83101254 bytes	29.016 s	5647602 bytes	0.06796	0.234378
wild-dog-08.bmp	2464x1648	12182070 bytes	3.3831935 s	1003041 bytes	0.082337	0.58613
leaves_iso_1600.bmp	3008x2000	18048054 bytes	6.3804112 s	1601701 bytes	0.088746	0.376356
Large Blue						
vroman.bmp	3850x2975	11460778 bytes	12.9757422 s	4248275 bytes	0.370679	1.105983
blue_bass.bmp	1700x2339	11928954 bytes	4.4712558 s	1064505 bytes	0.089237	1.179006
MOJA 2004.bmp	2450x1821	13388046 bytes	5.8113324 s	1454525 bytes	0.108644	1.219056
Large Colorful						
ManyStory.bmp	2396x1863	13391298 bytes	4.7472716 s	653087 bytes	0.04877	16.95664
balloons.bmp	2272x1704	11614518 bytes	3.6122066 s	604027 bytes	0.052006	0.789503
test_pattern6.bmp	2456x2058	15163398 bytes	6.2833594 s	382874 bytes	0.02525	7.193621

Tabel 4. Pengujian gambar berdimensi besar menggunakan BitMiracle

File Name	Dimension	Original Size	BitMiracle (IJG for C#)			
			Time	50% Compression Size	Ratio	SNR
Large Red						
red car.bmp	2304x1728	11943990 bytes	29.4796861 s	222400 bytes	0.01862	0.787975
red.bmp	2742x1802	14826910 bytes	37.2961332 s	334489 bytes	0.02256	0.972783
postvinyl_duchamp_red.bmp	2362x1772	12559992 bytes	32.0468329 s	141734 bytes	0.011285	1.845788
Large Green						
big_tree.bmp	6088x4550	83101254 bytes	170.5394996 s	2120824 bytes	0.025521	mem out
wild-dog-08.bmp	2464x1648	12182070 bytes	31.4567992 s	533545 bytes	0.043798	0.501188
leaves_iso_1600.bmp	3008x2000	18048054 bytes	40.6224713 s	782135 bytes	0.043336	0.325838
Large Blue						
vroman.bmp	3850x2975	11460778 bytes	102.4458595 s	1658501 bytes	0.144711	1.038912
blue_bass.bmp	1700x2339	11928954 bytes	32.4778576 s	599229 bytes	0.050233	1.055782
MOJA 2004.bmp	2450x1821	13388046 bytes	31.1607823 s	820271 bytes	0.061269	0.739994
Large Colorful						
ManyStory.bmp	2396x1863	13391298 bytes	31.138781 s	340892 bytes	0.025456	4.833735
balloons.bmp	2272x1704	11614518 bytes	27.3295631 s	258305 bytes	0.02224	0.557089
test_pattern6.bmp	2456x2058	15163398 bytes	35.2240147 s	179527 bytes	0.011839	1.772038

5. KESIMPULAN

Dari hasil analisis, perancangan, dan pembuatan program konversi JPEG, maka dapat diambil beberapa kesimpulan

1. Program yang dibuat telah memenuhi standard JPEG dan dapat digunakan untuk melakukan proses konversi dari jenis file umum (BMP, TIFF, TGA, dll.) ke file JPEG.
2. Menggunakan transformasi fDCT meningkatkan performa waktu jauh lebih cepat daripada menggunakan DCT dengan cara biasa
3. Melalui pengujian dapat disimpulkan bahwa program yang dibuat memerlukan waktu yang jauh lebih sedikit daripada *library* IJG yang merupakan *library* yang umum digunakan dalam pembuatan program serupa
4. Melalui pengujian dapat disimpulkan bahwa kualitas gambar yang dihasilkan berbanding terbalik dengan SNR gambar baik di program yang dibuat ataupun *library* yang digunakan sebagai pengujian karena tabel kuantisasi yang digunakan sama

DAFTAR PUSTAKA

[1] Li, Ze-Nian., Drew, M.S. (2004). *Fundamentals of multimedia*. Upper saddle river, N.J., 2004.

[2] Barni, Mauro. (2004). *Document and image compression*. Siena: Taylor and Francis.

[3] Sayood, Khalid. (2006). *Introduction to data compression third edition*. Sansome, S.F., 2006.

[4] Solomon, David. (2004). *Data compression : The complete reference*. Springer, N.Y., 2004.

[5] Blinn, James. (1993, July). "What's the deal with the DCT?". *IEEE Computer Graphics and Application*: 78-83.

[6] Cho, Nam Ik. (1991, July). "A Fast Algorithm for 2-D DCT". *IEEE Computer Graphics and Application*: 2197-2200.

[7] Shibata, Yoshiaki. (1999, March). "A Fast Degradation Free Algorithm for DCT Block Extraction in the Compressed Domain". *IEEE Computer Graphics and Application*: 3185-3188.

[8] Sundarajan, D. (2001). *The discrete cosine transform : Theory, algorithms, and application*. Rosewood danver, M.A.