Penerapan Kode Huffman dalam Proses Kompresi dan Dekompresi Format Gambar JPEG

Abel Stanley (13517068)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517068@std.stei.itb.ac.id

***Abstract*—Data digital sudah terintegrasi secara luas dan mendalam di kehidupan manusia. Kalimat "sebuah gambar bermakna ribuan kata" sering diutarakan dan menunjukkan betapa pentingnya kehadiran gambar dalam dunia maya. JPEG merupakan format gambar digital yang sangat dominan di internet karena kemampuan kompresi yang tinggi tanpa mengorbankan kualitas gambar secara signifikan. Makalah ini akan membahas mengenai peran Kode Huffman (*Huffman's Coding)* dalam proses kompresi dan dekompresi pada format *file* gambar JPEG.**

***Kata Kunci*— JPEG, Kode Huffman (*Huffman's Coding)*, Kompresi, Dekompresi.**

1. PENDAHULUAN

Gambar adalah sinyal dimensi dua yang diproses oleh indera penglihatan manusia yang merupakan perpaduan antara titik, garis, bidang dan warna yang berguna untuk mencitrakan sesuatu. Sinyal gambar yang mewakilkan gambar yang kita lihat umumnya bersifat analog. Namun, pada komputer dan perangkat elektronik lainnya, agar gambar dapat disimpan dan dioperasikan maka harus dikonversi menjadi bentuk digital, yaitu dengan representasi bilangan biner.

Pada umumnya, gambar adalah sebuah *array* dimensi 2 yang terdiri dari banyak pixel. Pixel berasal dari istilah *PEL (picture element)* yang merupakan satuan terkecil dari sebuah gambar digital atau grafis lainnya yang dapat ditampilkan oleh sebuah layar digital. Sebuah pixel dapat berupa sebuah titik atau persegi dari sebuah gambar raster yang dapat diukur oleh satuan DPI.

Di tahap perkembangan teknologi digital pada zaman ini, hampir semua media digital, baik aplikasi, website, video, ataupun *videogame* menggunakan gambar dalam mencerminkan dan mengekspresikan diri. Oleh karena itu, kualitas dari suatu gambar harus dijaga dengan baik agar informasi dan kesan yang ingin disampaikan tidak hilang atau berubah. Namun kenyataanya, ukuran data dari suatu gambar yang mentah umumnya besar. Hal ini akan mempersulit proses penyimpanan pada suatu *storage device* dan proses pengiriman data, baik dengan internet ataupun proses perpindahan data secara *hard drive*. Oleh karena itu, kompresi gambar sangat dibutuhkan untuk mengurangi redundansi data.

Kompresi/pemampatan data adalah proses yang dilakukan untuk mereduksi kuantitas data yang digunakan untuk mewakilkan suatu *file*, gambar, atau video. Pengurangan kuantitas data dapat berupa pengurangan jumlah *bits* yang dipakai untuk menyimpan sebuah medium digital.

Ada 2 jenis kompresi yaitu:

1. *Lossless Compression*: Teknik kompresi yang mempertahankan semua informasi dari sumber. Contoh format *file* yang menerapkan *lossless compression* adalah: Zip, RAR, NCW, FLAC, ALAC, HD-AAC, WMA, dan PNG.
2. *Lossy Compression*: Teknik kompresi yang mengorbankan sebagian informasi demi pengurangan ukuran data. Contoh format *file* yang menerapkan *lossy compression* adalah: MP3, MP4, AAC, WMA, Ogg dan JPEG (JPEG bisa bersifat *lossless* juga).

Kedua jenis kompresi bermanfaat untuk situasi yang berbeda. Untuk berkas teks umumnya digunakan *Lossless* karena setiap informasi yang diwakilkan oleh huruf bersifat penting. Namun untuk gambar dan video, *Lossy* dapat digunakan berhubung bahwa setiap pixel tidak sesignifikan sebuah huruf pada berkas teks.

Contoh format berkas gambar yang tidak terkompresi adalah BMP (bitmap) dan tiff. Kedua format tersebut membutuhkan banyak informasi untuk menyimpan sebuah gambar karena mereka membutuhkan 32 bits untuk menyandi warna-warna pada setiap pixel. Berbeda untuk JPEG karena format *file* JPEG dapat melalui proses kompresi tiga tahap yang akan mengurangi ukuran *file* secara signifikan.

JPEG umumnya digunakan untuk gambar-gambar dengan *continuous tone*. Kompresi JPEG menganut algoritma yang menganalisis persepsi seorang manusia. Kompresi ini akan menghapus detil-detil dan perbedaan yang tak kasat mata. Penghapusan detil-detil tersebut bersifat *irreversible* sehingga proses tersebut tegolong *lossy compression*.

Kode Huffman dapat digunakan dalam kompresi lebih lanjut yaitu dalam mengurangi redundansi penyimpanan domain frekuensi setelah kompresi *lossy*. Kompresi dengan Kode Huffman bersifat *lossless* sehingga kualitas gambar tidak akan menurun. Selain itu, Kode Huffman juga dapat digunakan untuk melakukan proses dekompresi JPEG agar didapatkan gambar kembali.

Makalah ini akan membahas lebih lanjut mengenai penerapan Kode Huffman dalam proses kompresi dan dekompresi JPEG.

2. KOMPRESI DATA

2.1. Cara Kerja Teknik-Teknik Kompresi Data Gambar Umum

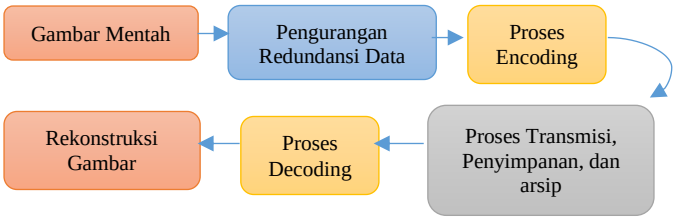
Tujuan utama dari berbagai algoritma kompresi adalah untuk mengurangi redundansi dan data-data yang tidak relevan pada gambar. Proses pertama yang akan dilakukan adalah konversi dari gambar mentah dari representasi domain spasial menjadi tipe representasi data biner yang disebut bitstream. Kuantitas data dari bitstream ini akan lebih kecil daripada kuantitas data milik gambar mentah, sehingga terjadi Kompresi ukuran data.

Redundansi yang akan dilacak dan eliminasi saat proses kompresi adalah :

1. Redundansi kode: Target yang dicari adalah kode-kode yang tidak efisien dalam segi banyak kata.
2. Redundansi Inter-pixel: Target didapatkan dari hasil analisis korelasi antara pixel-pixel dari suatu gambar.
3. Redundansi *Pyscho-visual*: Target adalah data yang tidak penting dan dapat diabaikan oleh indera penglihatan manusia

Setelah proses kompresi dilakukan, akan dilakukan penyandian *(encoding)* untuk mengubah informasi data yang terkompresi mejadi sebuah string biner yang disebut bitstream. Bitstream inilah yang akan disimpan atau ditransmisikan. Ketika gambar ingin diamati, proses selanjutnya yang harus dilakukan adalah proses *decode/decompression* oleh sebuah *decoder* yang akan mengubah bitstream tersebut kembali menjadi representasi domain spasial gambar. Dengan demikian, gambar dapat disimpan dengan ukuran yang kecil dan dapat direkonstruksi kembali sesuai keinginan.

Flowchart dari sebuah proses kompresi gambar yang umum adalah sebagai berikut: [7]



Flowchart 1, Alur proses kompresi dan dekompresi umum

2.2. Pentingnya Proses Kompresi Data

Gambar 1, Kampus ITB (412 pixel x 324 pixel)
(Sumber : tempo.co)

Rumus yang digunakan dalam menghitung ukuran gambar adalah :

$$Pixel\ Panjang\ \times Pixel\ Tinggi\ \times Kedalaman\ bit = Ukuran\ Gambar$$

Keterangan :
- Ukuran Gambar dalam bits.

Persamaan 1, Ukuran Suatu *File* Gambar

Sebagai contoh kasus nyata, gambar diatas dengan ukuran 412 pixel x 324 pixel x 24 bit RGB (16.7 juta warna) pada keadaan mentah akan membutuhkan tempat sebesar 3203712 bits atau 0.3819 MB untuk disimpan.

Jika disimulasikan proses transfer data melewati sebuah jaringan internet, untuk menghitung waktu transmisi, dapat digunakan rumus *transfer rate* berikut :

$$\text{Waktu Transfer} = \frac{\text{Ukuran Data}}{\text{Kecepatan Transfer Data}}$$

Persamaan 2, Waktu Transfer Data

Maka untuk melakukan proses transmisi gambar diatas melalui koneksi internet yang berkecepatan 64 Kbit/s akan memakan waktu sebesar 48,8848 detik.

Kompresi data umumnya diukur dengan rasio Kompresi data atau penghematan data yang didapat melalui rumus:

$$Rasio\ Kompresi\ Data\ = \frac{Ukuran\ Data\ Awal}{Ukuran\ Data\ Terkompresi}$$

Persamaan 3, Rasio Kompresi Data

$$Penghematan\ Data = \left(1 - \frac{Ukuran\ Data\ Terkompresi}{Ukuran\ Data\ Awal}\right) \times 100\%$$

Persamaan 4, Penghematan Data

Jika gambar tersebut dikompresi dengan rasio Kompresi 10:1, tempat penyimpanan yang dibutuhkan akan berkurang menjadi hanya 38.19 KB saja dan waktu transmisi yang dibutuhkan akan menjadi sekitar 4.8848 detik. Ini menunjukkan peningkatan efisiensi waktu sekitar 90.01%. Hal ini sangat berharga, terutama untuk gambar-gambar yang berukuran besar.

Waktu yang dibutuhkan untuk melakukan kompresi sangatlah singkat. Diperkirakan bahwa untuk melakukan kompresi pada 7 buah gambar yang masing-masing berukuran 1 MB akan memakan waktu yang lebih singkat dari waktu yang dibutuhkan untuk mengirim gambar mentah pada suatu koneksi internet yang umum.

3. KODE HUFFMAN

3.1. Apa itu Kode Huffman?

Kode Huffman (Huffman code) adalah sebuah tipe kode prefix optimal yang umum digunakan untuk *lossless compression* pada sebuah data.

3.2. Sejarah Penemuan Kode Huffman

Algoritma ini dikembangkan oleh David A. Huffman waktu ia melaksanakan pendidikan doktoral bidang sains di

Massachusetts Institute of Technology. Pada tahun 1951, beliau dan rekan-rekan mata kuliah teori informasi diberikan tugas untuk menulis sebuah *paper* untuk menemukan kode biner yang paling efisien. Beliau tidak dapat menemukan kode apapun untuk dijadikan jawaban yang menurutnya paling benar, dan sudah hampir ingin menyerah hingga ia mendapatkan ide untuk menggunakan *frequency-sorted binary tree* (pohon biner dengan pengurutan berdasarkan frekuensi kemunculan). Dengan ide ini, beliau mampu membuktikan bahwa metode tersebut adalah metode yang paling efisien.

Dari penemuan beliau, Ia telah melampaui professor-nya, Robert M. Fano, yang sedang berusaha untuk mengembangkan kode yang serupa.

Pada tahun 1952, Ia menerbitkan sebuah jurnal yang berjudul "A Method for the Construction of Minimum-Redundancy Codes" untuk memperkenalkan algoritma ini kepada dunia.

3.3. Karakteristik Kode Huffman

Algoritma Huffman merupakan salah satu jenis algoritma yang bersifat *entropy encoding*. *Entropy encoding* merupakan skema Kompresi data yang bersifat *lossless* dan bebas dari karakteristik-karakteristik khusus dari sebuah medium. [10]

Hasil dari algoritma Huffman adalah sebuah tabel kode dengan panjang yang bervariasi yang disusun berdasarkan probabilitas statistika untuk menyandikan simbol-simbol dari *file* sumber. Simbol dapat berupa karakter-karakter dari sebuah *file* teks, bytes, atau koefisien DCT. Algoritma Huffman akan memperhitungkan probabilitas atau frekuensi dari peristiwa kemunculan untuk setiap nilai yang mungkin dari kumpulan simbol pada berkas sumber untuk menghasilkan suatu tabel kode.

3.4. Teknik Dasar

Sesuai dengan karakteristik dari algoritma *entropy encoding*, prinsip penting yang dianut oleh Kode Huffman adalah:

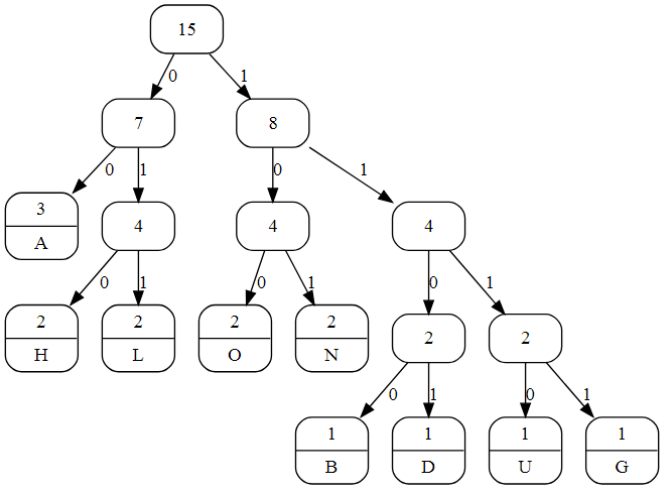
1. Simbol yang sering muncul akan memiliki panjang kode yang lebih pendek dibandingkan dengan kode yang jarang muncul.
2. Dua simbol yang memiliki frekuensi kemunculan yang identik atau berdekatan mungkin memiliki panjang kode yang sama.
3. Proses pembangunan Kode Huffman dimulai dengan pengambilan dua simbol yang memiliki frekuensi kemunculan yang paling rendah lalu akan digabungkan. Proses ini dilakukan berulang-ulang hingga tersisa satu simbol.

Proses pembangunan Kode Huffman dimulai dari proses menggabungkan dua simbol yang memiliki frekuensi kemunculan paling rendah. Proses ini akan diulang terus-menerus sampai hanya tersisa satu simbol. Representasi yang umum digunakan adalah pohon biner. Simbol yang memiliki frekuensi kemunculan terendah akan diletakkan pada bagian paling bawah pohon (daun), dan simbol yang memiliki frekuensi kemunculan tertinggi akan berada diatas pohon (akar).

Setelah pohon Huffman berhasil dibuat, setiap simbol sekarang dapat ditentukan kode baru yang sudah terkompresi. Cara penentuan kode suatu simbol adalah dengan menelusuri pohon Huffman hasil mulai dari akar-nya (paling atas) dan turun kebawah sampai ke daun yang mengandung simbol yang bersangkutan. Setiap kali anak kiri diakses maka string kode yang akan mewakilkan simbol tersebut akan dikonkatenasi dengan karakter biner '0'. Sebaliknya, jika anak kanan yang diakses, maka string hasil akan dikonkatenasi dengan karakter biner '1'. Dengan demikian, hasil string yang didapat setelah mencapai daun yang berisi simbol merupakan kode baru hasil algoritma Huffman yang akan mewakili simbol tersebut.

Dengan menggunakan Kode Huffman ini, panjang rata-rata dari suatu kode akan berkurang sehingga terjadi Kompresi data.

Berikut adalah contoh penerapan Kode Huffman pada sebuah contoh string yaitu: "HALOHALOBANDUNG".



Gambar 2, Pohon Huffman dari string "HALOHALOBANDUNG"

Berikut adalah tabel yang berisi dari panjang kode setiap karakter yang digunakan pada pohon di gambar 2.

Karakter	Frekuensi Relatif	Kode	Bits
A	20%	00	2
H	13.33%	010	3
L	13.33%	011	3
O	13.33%	100	3
N	13.33%	101	3
B	6.67%	1100	4
D	6.67%	1101	4
U	6.67%	1110	4
G	6.67%	1111	4

Tabel 1, Tabel Hasil Kode Huffman dari Gambar 2

Dari hasil Kode Huffman, string "HALOHALOBANDUNG" dapat disimpan dengan kode:

010000111000100001110011000010111011110101111 1

String hasil dari Kode Huffman akan memiliki data yang berjumlah 46 bits. Bandingkan dengan besar data sebelum kompresi, yaitu berjumlah 15 char yang masing-masing 1 byte (ASCII *encoding)*. Maka, untuk mendapatkan ukuran dalam bits akan dikalikan dengan 8. Total ukuran teks awal tanpa kompresi adalah:

$$Ukuran\ Awal\ String\ = 15 \times 8 = 120\ bits$$

Rasio kompresi pada kasus ini adalah:

$$Rasio\ Pemampatan\ Data\ = \frac{120}{46} = \ 2.6087 : 1$$

Persentase besar data yang dihemat adalah:

$$Penghematan\ Data\ = \left(1 - \frac{46}{120}\right) \times 100\% = \ 61.67\%$$

Dengan demikian, dapat dilihat bahwa Kode Huffman dapat melakukan kompresi data yang cukup signifikan tanpa mengorbankan informasi apapun (*lossless compression).*

4. JPEG

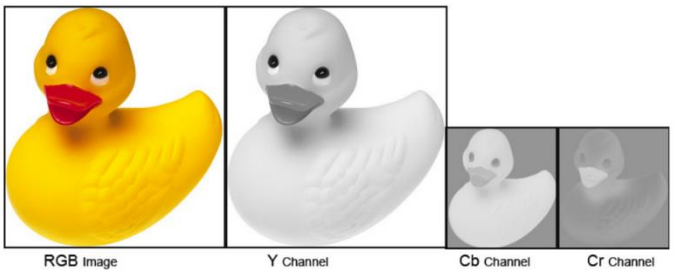
JPEG adalah format gambar yang sudah bertahun-tahun menjadi format yang paling umum digunakan internet. JPEG distandardisasi pada tahun 1994 dan mampu mempertahankan kualitas gambar yang cukup tinggi dengan ukuran *file* yang kecil. Kunci dari teknik kompresi format JPEG adalah dengan melakukan eliminasi pada informasi visual yang tidak kasat mata. Dapat dikatakan bahwa teknik kompresi ini menjadikan kelemahan dan keterbatasan persepsi indera mata manusia menjadi aset untuk mereduksi ukuran gambar.

Kompresi yang dilakukan pada format gambar ini bersifat *lossy* dan *irreversible.* Ketika informasi yang dianggap tidak penting menurut algoritma dibuang, maka informasi tersebut akan hilang selamanya. Kelemahan lainnya adalah JPEG tidak mendukung transparansi layaknya PNG.

4.1. Penyandian JPEG (*Encoding*)

Ketika JPEG dilalui oleh sebuah *encoder*, maka akan dilakukan pemisahan komponen RGB gambar menjadi tiga komponen yaitu:

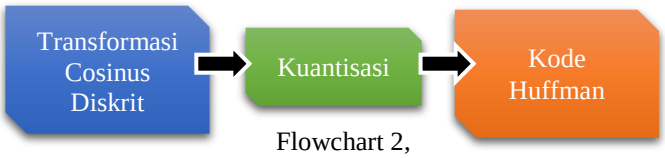
a. Monokroma (*luminance)*
   Kuantitas cahaya yang diterima.
b. Kroma biru (Cb)
   Kuantitas sinyal kroma perbedaan warna biru.
c. Kroma merah (Cr)
   Kuantitas sinyal kroma perbedaan warna merah.



Gambar 3, Komponen Y, Cb, dan Cr dari JPEG.
Sumber : http://www.robertstocker.co.uk

5. PENERAPAN KODE HUFFMAN DALAM KOMPRESI GAMBAR JPEG

Flowchart yang mewakili proses kompresi gambar JPEG adalah sebagai berikut:



Flowchart 2,
Flowchart Proses Kompresi JPEG

5.1. Transformasi Cosinus Diskrit

Transformasi ini sering disebut DCT (*Discrete Cosine Transformation*), adalah transformasi diskrit yang mengubah domain spasial dari suatu gambar menjadi domain frekuensinya. Domain spasial mengandung digit-digit yang melambangkan intensitas dari setiap *channel* pada suatu pixel. Domain Frekuensi berisi perubahan intensitas dari sebuah pixel relatif terhadap pixel selanjutnya.

Secara logika, domain frekuensi akan mengandung jumlah informasi yang lebih sedikit daripada domain spasial. Hal ini nyata dari kenyataan bahwa pada gambar-gambar yang diambil dari dunia nyata, perubahan intensitas pada suatu pixel dengan tetangganya adalah kasus yang sangat jarang. Umumnya mereka memiliki gradien yang mudah ditebak dan diprediksi.

DCT dapat mewakilkan domain frekuensi dari suatu gambar sebagai suatu kombinasi dari fungsi cosinus pada arah sumbu x dan y pada system koordinat kartesius.

Persamaan untuk mengkomputasi DCT dari elemen ke-i dan ke-j dari suatu gambar adalah:

$$D(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cos\left[\frac{(2x+1)i\pi}{2N}\right] \cos\left[\frac{(2y+1)j\pi}{2N}\right]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{if } u > 0 \end{cases}$$

Persamaan 4, Persamaan Matriks DCT Umum

Keterangan :

- *p(x,y)* adalah element ke-(x,y) dari suatu gambar yang diwakilkan oleh matriks *p*.
- N adalah ukuran dari blok dimana akan diterapkan DCT.

Proses untuk menerapkan DCT pada gambar untuk mendapatkan matriks frekuensi adalah :

1. Gambar akan dipisah-pisahkan menjadi JPEG Minimum Coded Unit (MCU), yaitu 8x8 blok pixel.
2. DCT akan diterapkan secara terstruktur mulai dari kiri ke kanan dan atas ke bawah.

Karena pada kasus kompresi JPEG, MCU memiliki ukuran 8x8 pixel, maka N adalah 8 dan *range* dari x dan y adalah 0 sampai 7. Maka rumus DCT untuk setiap MCU pada gambar JPEG akan menjadi:

$$D(i,j) = \frac{1}{4} C(i)C(j) \sum_{x=0}^{7} \sum_{y=0}^{7} p(x,y) \cos\left[\frac{(2x+1)i\pi}{16}\right] \cos\left[\frac{(2y+1)j\pi}{16}\right]$$

Persamaan 5, Persamaan Matriks DCT bagi MCU JPEG

Jika dianalisis secara matematika, karena DCT menggunakan fungsi cosinus, maka matriks hasil akan bergantung berat pada frekuensi horizontal, diagonal, dan vertikal.

Matriks DCT dari suatu MCU dapat ditemukan dengan menggunakan ketentuan:

Akan didapatkan matriks:

$$C = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix}$$

Matriks 1, Matriks DCT untuk MCU JPEG

Matriks ini dapat digunakan untuk menentukan domain frekuensi dari suatu domain spasial yang dimiliki gambar tertentu dengan persamaan:

$$D = CMC'$$

Persamaan 5, Persamaan Transformasi Cosinus Diskrit

Perlu diperhatikan bahwa :

1. Nilai pixel dari suatu gambar hitam-putih memiliki range 0-255 dengan setiap *increment* bernilai 1. Hitam akan diwakilkan oleh nilai 0 dan putih akan diwakilkan oleh nilai 255.
2. DCT hanya dapat diterapkan pada nilai pixel yang memiliki range -128 hingga 127. Oleh karena itu, setiap nilai pada matriks yang ingin diterapkan DCT harus dikurangkan dengan 128 terlebih dahulu.

Hasil dari sebuah transformasi consinus diskrit akan menghasilkan sebuah matriks yang memiliki 2 komponen, yaitu komponen DC (*Direct Current)* dan komponen AC (*Alternating Current)*.

Pada suatu MCU, komponen DC adalah nilai rata-rata dari 64 nilai pixel pada matriks MCU. Komponen DC akan berada di bagian ujung kiri-atas pada matriks hasil DCT.

Komponen AC adalah 63 nilai pixel yang menyusun matriks hasil DCT selain komponen DC. Nilai komponen AC tidak bergantung pada nilai rata-rata pixel.

5.2. Kuantisasi

Setelah proses DCT, gambar akan diwakilkan sebagai domain frekuensi yang sangat detil. Di proses ini akan dilakukan pembulatan frekuensi-frekuensi yang tidak dapat dipersepsi oleh mata manusia (seperti perubahan warna-warna yang sangat cerah atau redup) menjadi 0. Disinilah terjadi *lossy compression*.

Untuk melakukan kuantisasi, diperlukan matriks kuantisasi. Matriks kuantisasi standar adalah matriks kuantisasi dengan level 50 yang telah dioptimisasi untuk menghasilkan *file* gambar dengan tingkat kompresi tinggi dan kualitas gambar yang masih baik. Matriks standar kuantiasi adalah:

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Matriks 2, Matriks Standar Kuantisasi

Hasil dari kuantisasi didapat dengan membagi setiap elemen pada matriks hasil DCT yaitu D dengan nilai elemen di matriks kuantisasi yang sesuai menurut indeks i dan j. Perhatikan bahwa pembagian ini bersifat integer division, sehingga akan dilakukan pembulatan ke bilangan bulat yang paling dekat.

Semakin tinggi nilai elemen pada matriks kuantisasi, maka akan semakin rendah frekuensi dari gambar yang berarti kualitas gambar yang lebih buruk. Berikut perbedaan hasil kualitas gambar yang berhubungan dengan matriks kuantisasi yang berbeda-beda.

Gambar 4, Korelasi antara matriks kuantisasi dengan kualitas gambar akhir JPEG (sebelah kanan).
Sumber : Holloway, C.

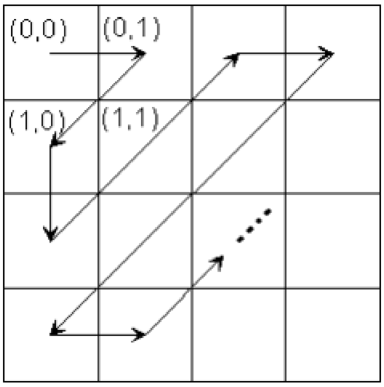
5.3. Kompresi Kode Huffman

Setelah proses DCT dan kuantisasi, matriks akan memiliki karakteristik sebagai berikut:

1. Frekuensi rendah (frekuensi yang mata manusia paling sensitif terhadapnya) akan terkumpul di bagian ujung atas kiri dari matriks.
2. Frekuensi tinggi (mata manusia tidak sensitif terhadapnya) akan terkumpul di bagian ujung bawah kanan dan sudah sebagian besar bernilai 0 akibat kuantisasi.

Algoritma kompresi JPEG mempunyai cara khusus untuk

mengakses koefisien-koefisien dari matriks. Proses pengambilan koefisien dilakukan secara zig-zag seperti gambar dibawah.



Gambar 5, Metode Pengambilan Koefisien dari Matriks

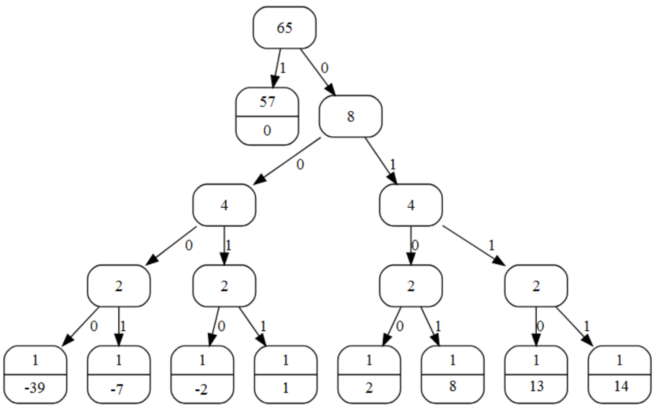
Teknik pengambilan ini ditujukan agar dapat mengurutkan koefisien 0 yang berkemungkinan berulang banyak dan merupakan redundansi. Format pengambilan ini juga bertujuan untuk mempersiapkan penerapan Kode Huffman pada bitstream.

Sebagai contoh, jika koefisien yang diambil dari suatu 8x8 matriks hasil adalah berikut:

–7, 2, 13, 14, -39, -2, 1, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.

Setelah pengambilan koefisien dilakukan, maka hasil tersebut akan disandikan menjadi suatu bitstream. Selanjutnya akan diterapkan Kode Huffman pada bitstream tersebut. Karena dapat dilihat banyak nilai 0 yang muncul, Kode Huffman adalah teknik kompresi yang tepat karena akan menyandi koefisien 0 yang berulang banyak menjadi kode yang memiliki jumlah bit yang kecil.

Dari contoh string nilai pixel diatas akan didapatkan pohon Huffman sebagai berikut:



Gambar 6, Pohon Huffman dari List Nilai Pixel Contoh

Karakter	Frekuensi Relatif	Kode	Bits
-39	1.53846%	0000	4
-7	1.53846%	0001	4
-2	1.53846%	0010	4
1	1.53846%	0011	4
2	1.53846%	0100	4
8	1.53846%	0101	4
13	1.53846%	0110	4
14	1.53846%	0111	4
0	87.6923%	1	1

Tabel 2, Tabel Hasil Kode Huffman dari Gambar 6

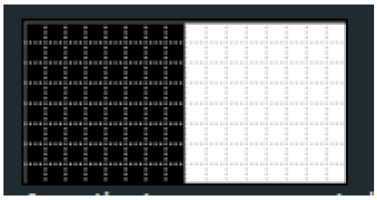
Besar ukuran string awal jika disimpan dalam *format* ASCII akan membutuhkan 520 bits. Dengan menggunakan Kode Huffman, nilai pixel dapat disimpan hanya dengan 89 bits.

Kompresi dengan Kode Huffman bersifat *lossless*, berbeda dengan pada bagian kuantisasi yang bersifat *lossy*. Hal ini berarti bahwa kualitas gambar akan tetap terjaga namun kuantitas data dapat dimampatkan lebih lagi.

6. Penerapan Kode Huffman dalam Dekompresi/Rekonstruksi JPEG

Setelah proses kompresi pada suatu JPEG telah dilakukan, bitstream yang merupakan hasilnya sudah siap disimpan atau ditransmisikan. Namun, agar dapat melihat representasi visual dari bitstream tersebut harus dilewatkan pada *decoder* untuk melalui proses dekripsi agar didapatkan gambar yang diinginkan kembali. Kode Huffman juga digunakan sebagai salah satu algoritma pada proses dekompresi pada suatu *file* JPEG yang telah melalui proses kompresi.

Sebagai contoh untuk dianalisis, akan digunakan gambar yang sangat sederhana, yaitu berukuran 8x16 pixel dan hanya mengandung dua warna yang saling kontras, yaitu hitam dan putih seperti gambar berikut:



Gambar 7, Contoh Gambar
Sumber : impulseadvanture.com

Terlihat bahwa gambar di atas terdiri dari 2 buah blok MCU, setiap blok berukuran 8x8 pixel.

Langkah pertama yang akan diambil adalah menganalisa hexdump dari datastream *file* JPEG yang terkompresi. Akan dicari nilai hex yang menandakan *Start of Scan*.

Gambar 8, Hexdump dari Gambar JPEG Contoh

Start of Scan dapat dikenali dengan nilai hex yaitu 0xFFDA. Selanjutnya akan dicari penanda *End of Image* dengan hex value 0xFFD9 yang menunjukkan akhir dari pembacaan. Data hasil *scan* yang relevan dalam proses dekripsi Huffman ini hanyalah yang ditandakan berwarna biru tua, yaitu:

FC FF 00 E2 AF EF F3 15 7 (9 bytes)

Namun perlu diperhatikan bahwa pada format JPEG ada

sebuah penanda format yang unik bagi JPEG yaitu 0xFF yang diikuti 0x00 (stuff byte). Untuk proses ekstraksi, hex 0x00 dapat ditiadakan, sehingga hexstream yang perlu dianalisis adalah:

FC FF E2 AF EF F3 15 7F (8 bytes)

Umumnya untuk gambar JPEG di dunia nyata, gambar memiliki 4 komponen yaitu Y1, Y2, Cb, Cr. Namun pada kasus gambar tanpa variasi warna ini maka komponen yang dimiliki hanya Y, Cb, Cr.

Untuk proses kedepan, hexstream lebih baik diubah menjadi bitstream terlebih dahulu:

1111 1100 1111 1111 1110 0010 1010 1111 1110 1111 1111 0011 0001 0101 0111 1111

Dalam setiap *file* JPEG yang telah terenkripsi, akan tersimpan satu hingga empat buah tabel Huffman yang mengatur pemetaan nilai kode dengan jumlah panjang kode. Dalam kehidupan nyata, sangatlah lebih praktis untuk menggunakan *software* seperti *JPEGsnoop* yang dapat mengekstraksi kode Huffman dari *file* JPEG. Dari gambar contoh didapatkan empat tabel Huffman yang mewakili pemetaan bagi setiap komponen gambar.

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

Tabel 3, Tabel Huffman Luminance(Y)-DC

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

Tabel 4, Tabel Huffman Luminance(Y)-AC

Length	Bits	Code
2 bits	00 01	01 00 (End of Block)
3 bits	100 101	02 03
4 bits	1100 1101 1110	04 05 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

Tabel 5, Tabel Huffman Chrominance(Cb&Cr)-DC

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

Tabel 6, Tabel Huffman Chrominance (Cb&Cr)-AC

Sebelum memasuki tahap selanjutnya, menurut algoritma dekompresi, ketika didapat pembacaan yang sukses terhadap tabel Huffman di bagian DC, maka harus dilakukan proses konversi dari DC Code menjadi DC Value. Konversi ini dilakukan dengan bantuan tabel referensi *Huffman DC Value Encoding*.

DC Code	Size	Additional Bits		DC Value	
00	0			0	
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

Tabel 7, Tabel Referensi *Huffman DC Value Encoding*

Dengan demikian, tahap dekompresi dapat dimulai. Karena gambar terdiri dari 2 blok MCU, maka analisis akan dijalankan sebanyak dua kali.

Bit string awal:
1111 1100 1111 1111 1110 0010 1010 1111 1110 1111 1111 0011 0001 0101 0111 1111

1. Block 1 – Luminance
   a. Luminance (Y) – DC
      Dengan referensi tabel 3, dapat dilihat bahwa string 1111 110 dapat ditranslasikan sebagai kode x0A. Dengan demikian, pembacaan sukses. Pembacaan yang sukses pada bagian DC harus melalui tahap referensi dengan tabel 7. Hex x0A melambangkan bahwa ada 10 bits selanjutnya yang akan mewakilkan nilai *signed* dari komponen DC. 10 bits selanjutnya adalah 0 1111 1111 1. Menurut tabel referensi, biner *signed* tersebut akan menjadi DC Value yang bernilai desimal -512. Nilai kode DC tersebut disimpan.

   - String tersisa: 110 0010 1010 1111 1110 1111 1111 0011 0001 0101 0111 1111

   b. Luminance (Y)- AC
      Terbaca string 1100 yang merupakan EOB (*End of Block)* menurut tabel 4. Pembacaan diberhentikan.
   - String tersisa: 010 1010 1111 1110 1111 1111 0011 0001 0101 0111 1111

   c. Chrominance (Cb)- DC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 5. Pembacaan diberhentikan.
   - String tersisa: 0 1010 1111 1110 1111 1111 0011 0001 0101 0111 1111

   d. Chrominance(Cb)-AC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 6. Pembacaan diberhentikan.
   - String tersisa: 010 1111 1110 1111 1111 0011 0001 0101 0111 1111

   e. Chrominance(Cr)-DC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 6. Pembacaan diberhentikan.
   - String tersisa: 0 1111 1110 1111 1111 0011 0001 0101 0111 1111

   f. Chrominance(Cr)-DC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 6. Pembacaan diberhentikan.
   - String tersisa: 0 1111 1110 1111 1111 0011 0001 0101 0111 1111

2. Block 2
   a. Luminance (Y) – DC
      Terbaca string 111 1110. Kode Hex adalah 0xA, sehingga akan diambil kembil 10 bit untuk dikonversi menjadi Code Value menurut tabel 7. 1111 1111 00 adalah bernilai +1020. Nilai kode DC tersebut disimpan.
   - String tersisa: 11 0001 0101 0111 1111

   g. Luminance (Y)- AC
      Terbaca string 1100 yang merupakan EOB (*End of Block)*menurut tabel 4. Pembacaan diberhentikan.
   - String tersisa: 01 0101 0111 1111

   h. Chrominance (Cb)- DC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 5. Pembacaan diberhentikan.
   - String tersisa: 0101 0111 1111

   i. Chrominance(Cb)-AC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 6. Pembacaan diberhentikan.
   - String tersisa: 01 0111 1111

   j. Chrominance(Cr)-DC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 6. Pembacaan diberhentikan.
   - String tersisa: 0111 1111

   k. Chrominance(Cr)-DC
      Terbaca string 01 yang merupakan EOB (*End of Block)* menurut tabel 6. Pembacaan diberhentikan.
   - String tersisa: 11 1111

3. Sisa
   String yang tersisa adalah 11 1111. Karena setiap data scan harus berhenti pada batas akhir byte, maka bits yang tersisa akan dibuang.

Hasil pembacaan data scan didapat dua nilai DC yaitu -512 dari blok 1 dan + 1020 dari blok 2. Nilai-nilai tersebut bersifat relatif. Jadi, nilai *luminance* dari blok 2 (+1020) sebenarnya relative pada nilai *luminance* bawaan dari blok 1, sehingga nilai absolutnya adalah:

$$-512 + 1080 = +508$$

Maka hasil yang didapat menjadi:

Blok	Nilai Absolut
1	Y = -512
2	Y = +508

Tabel 8, Tabel Hasil *Decode* Bitstream JPEG Gambar 7

Analisis dari nilai didapat adalah:
1. Karena *channel* Cb dan Cr bernilai 0, maka gambar adalah bersifat *grayscale*.
2. Karena setiap komponen AC adalah 0, maka tidak ada perubahan frekuensi pada gambar. Bisa disimpulkan bahwa gambar bersifat *monochrome*.

Perlu diingat bahwa nilai DC dan AC yang didapat dari Kode Huffman adalah domain frekuensi dari gambar, bukan nilai aslinya. Selanjutnya harus ditebak koefisien *gain* dari transfomrasi DCT terhadap nilai DC tersebut. RGB memiliki nilai *range* antara 0-255. Agar nilai -512 dan +508 dapat masuk ke *range* 0-255, maka harus dibagi 4, maka koefisien *gain* DCT adalah 4. Nilai yang akan dibawah ke proses selanjutnya adalah -128 dan +127.

Tahap terakhir adalah untuk melakukan konversi nilai Y, Cr, Cb menjadi nilai RGB, yaitu dengan menggunakan persamaan:

$$R = Cr * (2 - 2 * Cred) + Y$$
$$G = \frac{Y - Cb * B - Cr * R}{Cg}$$
$$B = Cb * (2 - 2 * Cb) + Y$$

$$Y = Cr * R + Cg * G + Cb * B$$

Keterangan:

- Koefisien Cr = 0.299, Cg = 0.587, Cb = 0.114
- Rumus diaatas adalah untuk RGB dengan range (-128 sampai 127). Sehingga, hasil akhir harus ditambahkan +128 untuk mendapatkan RGB dengan range (0-255).
- Cg dalam kasus ini bernilai 1 (diabaikan).

Persamaan 6, Konversi Y, Cr, Cb ke RGB

Akan didapat hasil terakhir yaitu:

Blok	RGB (Hex)	RGB (Desimal)
1	(0x00,0x00,0x00)	(0,0,0)
2	(0xFF,0xFF,0xFF)	(255,255,255)

Tabel 9, Tabel Hasil Nilai RGB JPEG Gambar 7

Dapat dilihat bahwa nilai RGB tersebut telah mewakili warna yang dimiliki oleh gambar aslinya, yaitu blok 1 berwarna hitam dan blok 2 berwarna putih.

7. KESIMPULAN

Kode Huffman dapat diterapkan pada banyak proses kompresi data, salah satunya adalah kompresi format gambar JPEG. Kompresi ukuran data gambar dengan format JPEG dilakukan dengan tiga proses yang menggabungkan teknik kompresi *lossy* dan *lossless*. Pertama, bagian informasi yang bersifat diskrit dari setiap pixel akan diubah menjadi sebuah pola dua dimensi dengan teknik Transformasi Cosinus Diskrit. Lalu, informasi-informasi dari gambar yang tidak dapat atau sulit dipersepsi oleh indera mata manusia akan dihilangkan dengan matriks kuantisasi. Terakhir, informasi yang tersisa akan dikompresi lebih lanjut menggunakan Kode Huffman. Ketika gambar ingin diamati, akan dilakukan rekonstruksi gambar dengan Tabel Huffman yang tersimpan dalam gambar.

Hasil dari ketiga proses tersebut akan memungkinkan *file* gambar JPEG untuk memiliki ukuran *file* yang kecil namun memiliki kualitas gambar yang tetap baik. Oleh karena itulah format JPEG menjadi format gambar yang dominan di internet selama bertahun-tahun.

8. UCAPAN TERIMA KASIH

Penulis pertama-tama ingin memanjatkan puji dan syukur kepada Tuhan yang Maha Esa atas anugerah dan kesempatan yang diberikan untuk menulis makalah ini. Hanya karena berkat-Nya makalah ini dapat diselesaikan tepat waktu. Selanjutnya, penulis juga ingin berterima kasih kepada Dra. Harlili M.Sc. sebagai dosen pengajar dari mata kuliah IF2120 Matematika Diskrit K-02 atas kesempatan dan pengalaman beliau perbolehkan bagi penulis untuk alami dan pelajari. Dari kesempatan ini, penulis dapat belajar banyak mengenai format-format gambar dan teknik kompresi data. Penulis juga ingin berterima kasih kepada orang tua, keluarga, dan kerabat penulis karena tanpa mereka, penulis tidak akan menjadi diri penulis yang sekarang.

9. REFERENSI

[1] Holloway, C. 2008. *JPEG Image compression: Transformation, Quantization and Encoding*, Honours Linear Algebra 2008.

[2] Huffman, D. (1952). A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the I.R.E.

[3] Khayam, A. S. 2003. *The Discrete Cosine Transform: Theory and Application. Michigan State University*, CA: Wadsworth.

[4] Mamta Sharma. (2010). Compression Using Huffman Coding. IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5.

[5] R. C. Gonzalea, R. E. Woods. 2004. *Digital Image Processing*, 2nd Ed., Prentice Hall.

[6] Singh, Manjari & Kumar, Sushil & Chouhan, Siddharth & Shrivastava, Manish. 2016. *Various Image Compression Techniques: Lossy and Lossless.* International Journal of Computer Applications.

[7] Wallace, G. K. .1991. *The JPEG Still Picture Compression Standard*. Communications of the ACM, Vol. 34, Issue 4, pp.30-44.

[8] WY, Wei. 2011. *An Introduction to Image Compression.* National Taiwan University.

[9] https://www.impulseadventure.com/photo/jpeg-huffman-coding.html, diakses pada 6 Desember 2018 pukul 19.42 WIB.

[10] https://www.computerhope.com/jargon/i/image.htm, diakses pada 7 Desember 2018 pukul 11.13 WIB.

[11] https://www.techopedia.com/definition/24012/pixel, diakses pada 7 Desember 2018 pukul 11.21 WIB.

[12] https://www.math.cuhk.edu.hk/~lmlui/dct.pdf, diakses pada 7 Desember 2018 pukul 18.37 WIB.

[13] http://www.robertstocker.co.uk/jpeg/jpeg_new_8.htm, diakses pada 8 Desember 2018 pukul 13.03 WIB.

10. 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 2018

Abel Stanley 13517068