Pemanfaatan Algoritma JPEG untuk Kompresi Gambar Digital

Jose Galbraith Hasintongan 13519022
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13519022@std.stei.itb.ac.id

***Abstract*—Pada zaman sekarang, manusia memikirkan banyak cara untuk mewujudkan hidup yang lebih praktis dikarenakan keperluan atau kebutuhan yang beragam. Teknologi yang diciptakan manusia pada zaman sekarang lebih mudah untuk diakses oleh banyak orang dari kalangan apapun. Namun, teknologi juga memiliki keterbatasan dalam hal kapasitas penyimpanan sehingga dibutuhkannya inovasi teknologi yang dapat mewujudkan keefektifan dalam penggunaan *memory* atau untuk sekadar mempercepat pengunduhan *file*. Salah satu inovasi teknologi yang banyak digunakan sekarang adalah kompresi *file* yang dapat memadatkan suatu *file* sehingga ukurannya dapat diperkecil. Matematika diskrit sangat berperan besar dalam inovasi teknologi ini. Penggunaan pengodean Huffman yang memanfaatkan pohon biner dapat menjadi solusi dari masalah ini mengingat zaman sekarang pekerjaan dan pendidikan banyak memanfaatkan pengiriman *file.* Algoritma JPEG menggunakan Huffman Coding untuk mengompresi gambar berwarna.**

***Keywords*—Huffman Coding, Keefektifan, Kapasitas. Pohon Biner, JPEG**

I. PENDAHULUAN

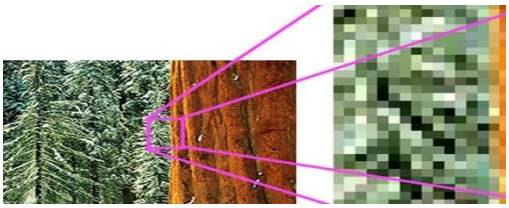
Kebutuhan manusia dalam dunia pekerjaan atau pendidikan semakin beragam. Manusia menciptakan berbagai inovasi teknologi untuk mempermudah mereka dalam mengerjakan tugas mereka. Sayangnya, teknologi, seperti komputer dan koneksi internet, memiliki keterbatasannya tersendiri dalam menjalankan sesuatu hal tergantung dari spesifkasi alat-alat tersebut. Alih-alih menciptakan atau menggunakan spesifikasi komputer atau jaringan internet yang lebih baik, yang pastinya mahal, kita dapat memanfaatkan suatu algoritma yang dapat mengatasi masalah tersebut tanpa perlu *hardware* yang mahal. Dalam bidang matematika diskrit, ada pengodean Huffman yang bisa mengatasi kompresi *file* sehingga ukuran *file* dapat diperkecil sehingga menghemat penggunaan *memory* komputer dan pengunggahan atau pengunduhan yang lebih cepat karena ukuran yang lebih kecil.

Selain menghemat *memory*, kompresi *file* juga dapat digunakan untuk alasan keamanan. Format *file* yang telah dikompres memiliki banyak nama tetapi yang sering digunakan adalah .zip, .rar, .7z, .tar. *Software* untuk kompresi *file* seperti WinZip bisa membantu dalam mengamankan *file* dengan cara mengenkripsi data sebelum diunggah ke penyimpanan *cloud.*

Pada makalah ini tidak akan dibahas lebih jauh mengenai kompresi *file* untuk alasan keamanan atau enkripsi. Penulis akan membahas pemanfaatan pengodean Huffman pada algoritma JPEG untuk mengompresi gambar.

.

II. LANDASAN TEORI



Gambar 1 Gambar yang terdiri dari piksel
(sumber : www.kompas.com)

A. Pengertian Gambar Digital

Sebenarnya, gambar digital merupakan gambar nyata yang direpresentasikan dengan sekumpulan angka yang dapat disimpan dan diolah oleh komputer. Penerjemahan gambar menjadi angka melibatkan pembagian gambar menjadi titik-titik kecil yang disebut sebagai piksel. Komputer merekam angka-angka yang memuat berbagai informasi seperti intensitas cahaya atau warna. Angka-angka disusun dalam baris dan kolom yang sesuai dengan posisi vertikal dan hortizontal sehingga membentuk gambar.

B. Jenis Gambar Digital

Sebuah gambar dapat memiliki tiga warna, yaitu RGB (Red, Green, Blue) atau empat warna, yaitu CMYK (Cyan, Magenta, Yellow Black).

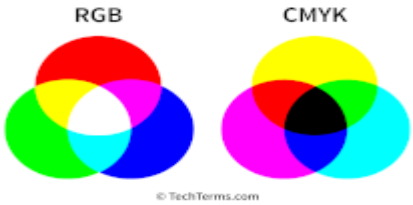
1. Red, Green, Blue

   Mengombinasi merah, hijau, dan biru adalah metode standard dalam produksi gambar berwarna pada layar TV, monitor komputer, dan layer *smartphone*. Model warna RGB adalah model "*additive*". Ketika tiap warna dikombinasikan secara merata maka akan menciptakan cahaya berwarna putih. Ketika tiap 0& dari tiap warna dikombinasikan, maka tidak ada cahaya yang dibentuk sehingga menciptakan warna hitam. Jumlah warna yang dapat dibentuk oleh RGB tergantung dari seberapa besar kemungkinan nilai yang dapat digunakan untuk merah, hijau, dan biru. Ini dikenal sebagai "*color depth*" yang diukur dalam bit. *Color depth* yang paling umum adalah 24-bit color. *Color depth* ini dikenal sebagai *"true color*". *Color depth* ini mendukung 8 bit untuk setiap warna dari

tiga warna tersebut. Dengan begitu, ia menghasilkan 2^8 atau 256 kemungkinan untuk merah, hijau, dan biru. Total kemungkinan warna yang dapat dihasilkan adalah 256 x 256 x 256 = 16.777.216 total kemungkinan.

2. Cyan, Magenta, Yellow, Black

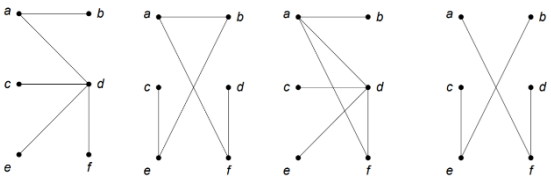
CMYK bersifat *subtractive* yang artinya warna dapat menjadi lebih gelap ketika warna-warna digabungkan. Pengombinasian secara merata dari warna-warna tersebut dapat menghasilkan warna hitam. Meskipun begitu, warna hitam yang murni sulit untuk diciptakan dari menggabungkan warna-warna tersebut akibat kotoran pada tinta sehingga tinta warna hitam (Black 'K') disatukan dengan ketiga warna lainnya.



Gambar 2 Ilustrasi model warna RGB dan CMYK
(sumber: techterms.com)

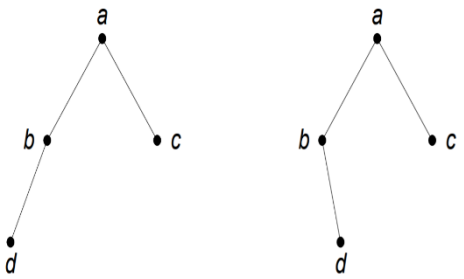
C. Pohon dan Pohon Biner

Pohon merupakan graf tak-berarah terhubung yang tidak mengandung sirkuit. Pohon tidak memiliki sirkuit.



Gambar 3 Ilustrasi pohon
(sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/)

Pohon biner adalah pohon n-ary dengan n – 2. Setiap simpul dalam pohon biner mempunyai paling banyak 2 buah anak. Anak dari simpul ini dibedakan sesuai letaknya menjadi anak kiri dan anak kanan.



Gambar 4 Ilustrasi pohon
(sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/)

III. KOMPRESI

A. Kompresi Gambar

Dikutip dari whydomath.org, gambar digital yang kita simpan sebagai suatu *file* sebenarnya disimpan sebagai string panjang dari suatu bit yang terdiri dari 0 dan 1. Tiap piksel pada gambar berukuran 1 byte. 1 byte terdiri dari 8 bit. Jika dimensi sebuah piksel adalah M x N piksel, maka dibutuhkan MN byte atau 8MN bit untuk menyimpan gambar. Ada 256 bytes yang berbeda dan tiap byte memiliki representasi basis 2-nya. Setiap representasi panjangnya adalah 8 bit. Sehingga, untuk kamera dapat menyimpan gambar, kamera harus mempunyai string bit sebagai tambahan pada *dictionary* yang mendeskripsikan representasi bit dari tiap byte.

Untuk mengompresi, kita hanya perlu mengubah representasi bit dari tiap byte sehingga *dictionary* baru yang menghasilkan string bit yang lebih pendek yang dibutuhkan untuk menyimpang gambar.

B. Pengodean Huffman

Ide dari Huffman adalah mengidentifikasi piksel yang sering muncul pada gambar dan menetapkan representasi bit yang pendek. Piksel yang jarang muncul pada gambar ditetapkan sebagai representasi bit panjang. Misalnya kita ingin menyimpan kata sassafras pada *disk.* Setiap huruf pada suatu kata ditetapkan angka ASCII sehingga kata sassafras menempati baris pertama pada gambar digital.

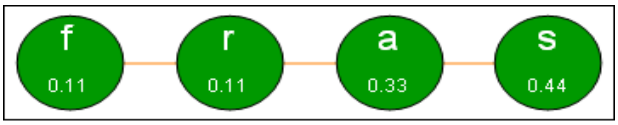
Letter	ASCII	Binary	Frequency	Relative Frequency
s	115	01110011	4	4/9
a	97	01100001	3	1/3
f	102	01100110	1	1/9
r	114	01110010	1	1/9

Gambar 5 Representasi ASCII dan biner tiap huruf
(sumber : whydomath.org)

Kata sassafras terdiri dari 9 karakter yang tiap karakternya membutuhkan 8 bit sehingga dibutuhkan 72 bit untuk menyimpan kata.

Algoritma Huffman

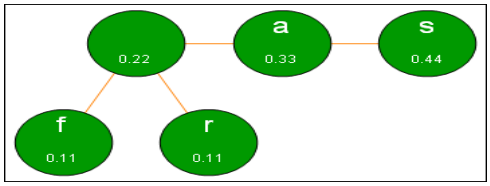
Pertama, kita perlu mengelompokkan karakter unik berdasarkan frekuensi relative, terkecil ke terbesar. Karakter dan frekuensi relatif ditempatkan pada simpul yang berhubungan dengan sisi.



Gambar 6 Ilustrasi simpul dari karakter yang telah diurutkan berdasarkan frekuensi relatif dan sisi
(sumber: whydomath.org)

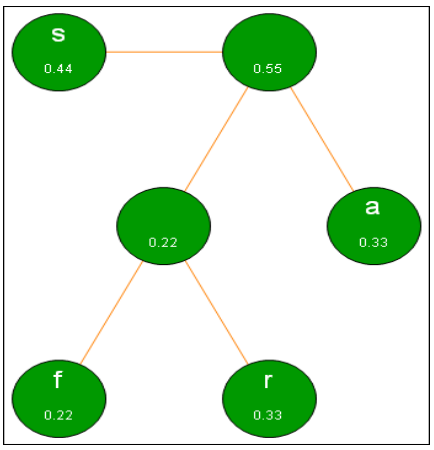
Kedua, kita perlu mengambil 2 simpul terkiri dan menjumlahkan frekuensinya sehingga mendapatkan 0,22. Dari penjumlahan ini, kita membuat simpul baru dengan frekuensi

0,22 yang memiliki dua 2 simpul anak (simpul f dan r). Setelah itu, kita kelompokkan bagian atas simpul.



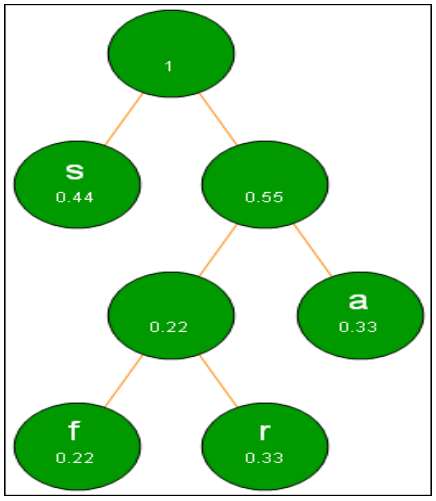
Gambar 7 Ilustrasi dari tahap 2
(sumber: whydomath.org)

Ketiga, mirip seperti tahap kedua, jumlahkan frekuensi dari dua simpul terkiri dan buat simpul baru yang memiliki dua anak. Kelompokkan simpul bagian atas. Perlu diperhatikan bahwa simpul yang baru memiliki frekuensi yang lebih besar sehingga bergeser ke kiri.



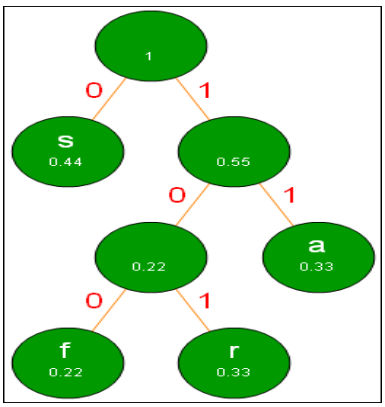
Gambar 8 Ilustrasi dari tahap 3
(sumber: whydomath.org)

Keempat, ulangi 2 tahap terakhir. Ketika sudah menghasilkan simpul sebesar 1 maka sudah selesai.



Gambar 9 Ilustrasi dari tahap 4
(sumber: whydomath.org)

Kelima, tandai semua sisi yang condong ke kiri dengan 0 dan 1 untuk yang condong ke kanan.



Gambar 10 Ilustrasi dari tahap 5
(sumber: whydomath.org)

Sekarang, kita bisa membaca representasi bit terbaru untuk tiap karakter dari pohon Huffman. Untuk mendapatkan representasi tiap karakter, kita hanya perlu mengumpulkan 0 dan 1 yang kita lalui dari atas hingga ke karakter yang diinginkan.

Letter	Binary	Huffman
s	01110011	0
a	01100001	11
f	01100110	100
r	01110010	101

(A)

0 11 0 0 11 100 101 11 0
s a s s a f r a s

(B)

Gambar 11 (A) Representasi baru tiap karakter dan Gambar 12 (B) Representasi kata sassafras yang baru
(sumber: whydomath.org)

Representasi bit terbaru untuk kata sassafras hanya memerlukan 16 bit sedangkan yang awal membutuhkan 72 bit. Dengan ini, kita telah menghemat banyak ruang pada tempat penyimpanan. Rasio compresi direpresentasikan dengan *bits per pixel* sehingga kompresi dari kata sassafras adalah 16/9 = 1,77 bpp.

Pengodean Huffman dan gambar digital.

Gambar 13 Ilustrasi
(sumber : whydomath.org)

Misal Gambar 11.0 berukuran 512 x 768 piksel sehingga totalnya 393.216 piksel atau 3.145.728 bit. Jika kita menggunakan pengodean Huffman untuk gambar ini maka hasil

dari kompresi adalah 7,714 bit sehingga berukuran 3.033.236 bit. Tentu saja pengurangan masih belum signifikan, Dari sini, kita tahu bahwa pengodean Huffman lebih efektif ketika digunakan secara langsung pada gambar digital, akan lebih efektif ketika gambar terdiri dari intensitas berbeda yang lebih sedikit.

IV. ALGORITMA JPEG

A. Pendahuluan JPEG

JPEG (Joint Photographic Experts Group) merupakan suatu salah satu motode kompresi. JPEG menjadi suatu standard pada tahun 1992. JPEG merupakan metode *lossy compression*. JPEG menggunakan versi pengodean Huffman yang lebih rumit untuk melaksanakan pengodean. Dikutip dari whydomath.org, ada 4 tahapan dasar pada algoritma – preproses, transformasi, kuantisasi, dan pengodean.

B. *Lossy Compression* dan *Losless Compression*

Lossy Compression adalah metode kompresi yang ketika proses kompresi, ia kehilangan data dan data tidak bisa di dibuat ulang menjadi gambar awalnya.

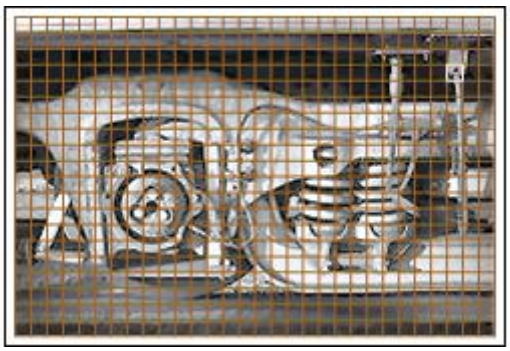
Lossless Compression adalah metode kompresi yang tidak kehilangan data saat proses kompresinya dan bisa dibuat ulang menjadi gambar awal.

C. Tahapan Kompresi

1. Preproses

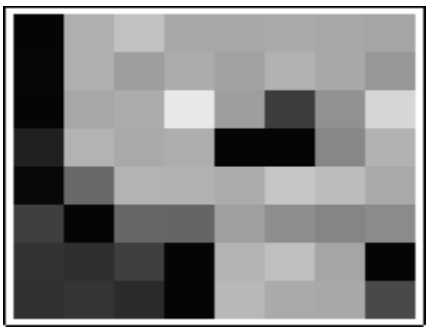
   Dikutip dari whydomath.org, untuk dapat mengompres gambar berwarna dalam format JPG, pertama kta harus mengonversi RGB *channels* ke YCbCr *space*. Setelah itu, semua kita lakukan ke saluran Y, Cb, dan Cr untuk gambar *grayscale*.

   Partisi gambar menjadi blok dengan ukuran 8x8 piksel.



Gambar 14 Ilustrasi partisi
(sumber : whydomath.org)

Pada Gambar 14.0 terdapat sebuah gambar dengan ukuran 160 x 240 = (8x20) x (8x30). Jumlah total dari blok adalah 600 blok. Pada kolom 28 baris 4 terdapat piksel yang diberi warna cerah. Kita menggunakan elemen pada matriks untuk mengilustrasikan transformasi matematika dan langkah kuantisasi.



Gambar 15 Blok contoh yang diperbesar
(sumber : whydomath.org)

5	176	193	168	168	170	167	165
6	176	158	172	162	177	168	151
5	167	172	232	158	61	145	214
33	179	169	174	5	5	135	178
8	104	180	178	172	197	188	169
63	5	102	101	160	142	133	139
51	47	63	5	180	191	165	5
49	53	43	5	184	170	168	74

Pixel intensities of the block row 4, block column 28.

Gambar 16 Intensitas piksel baris 4 kolom 28
(sumber : whydomath.org)

Langkah terakhir pada preproses adalah dengan mengurangi intensitas piksel tiap blok sebesar 127. Langkah ini memusatkan intensitas nilai 0 dan dilakukan untuk menyederhankan matematika dari langkah-langkah transfomasi dan kuantisasi

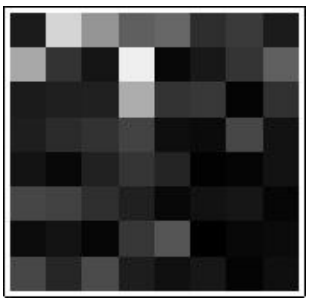
-122	49	66	41	41	43	40	38
-121	49	31	45	35	50	41	24
-122	40	45	105	31	-66	18	87
-94	52	42	47	-122	-122	8	51
-119	-23	53	51	45	70	61	42
-64	-122	-25	-26	33	15	6	12
-76	-80	-64	-122	53	64	38	-122
-78	-74	-84	-122	57	43	41	-53

Pixel intensity values less 127 in block row 4, block column 28.

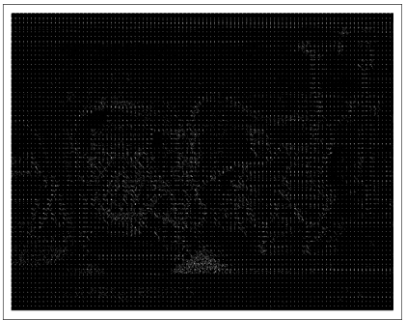
Gambar 17 Nilai intensitas setelah dikurangi 127
(sumber : whydomath.org)

2. Transformasi

   JPEG *Image Compression Standard* bergantung pada Discrete Cosine Transformation (DCT) untuk mentransformasi gambar. DCT merupakan produk $C = U\ B\ U^T$. B adalah blok berukuarn 8 x 8 dari gambar preproses dan U adalah matriks special 8 x 8. DCT cenderung mendorong banyak informasi intensitas tinggi dalam blox 8 x 8 ke bagian atas kiri C dengan nilai sisa di C mengambil nilai yang cenderung lebih kecil.



Gambar 18 Contoh DCT dari suatu blok
(sumber : whydomath.org)



Gambar 19 DCT sudah diaplikasikan pada tiap blok
(sumber : whydomath.org)

Kebanyakkan dari intensitas dalam gambar transformasi cosine bentuknya kecil.

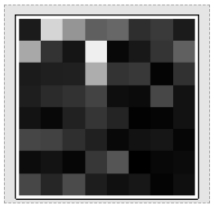
-27.500	-213.468	-149.608	-95.281	-103.750	-46.946	-58.717	27.226
168.229	51.611	-21.544	-239.520	-8.238	-24.495	-52.657	-96.621
-27.198	-31.236	-32.278	173.389	-51.141	-56.942	4.002	49.143
30.184	-43.070	-50.473	67.134	-14.115	11.139	71.010	18.039
19.500	8.460	33.589	-53.113	-36.750	2.918	-5.795	-18.387
-70.593	66.878	47.441	-32.614	-8.195	18.132	-22.994	6.631
12.078	-19.127	6.252	-55.157	85.586	-0.603	8.028	11.212
71.152	-38.373	-75.924	29.294	-16.451	-23.436	-4.213	15.624

DCT values in block row 4, block column 28.

Gambar 20 Nilai DCT pada blok baris 4 kolom 28
(sumber : whydomath.org)

Nilai pada matriks di atas sudah dibulatkan ke 3 digit. DCT menyebabkan munculnya angka-angka besar yang berdekatan dengan nilai 0.

3. Kuantisasi
   Elemen yang dekat 0 akan dikonversikan ke – dan elemen lain akan diperkecil sehingga nilainya lebih dekat ke 0. Tahap DCT sepenuhnya *invertible.* Kita bisa mengembalikan B dengan komputasi B = U^T C U. Ketika memperkecil nilai, sangat mungkin untuk mengembalikannya. Namun, mengonversi nilai kecil ke 0 dan membulatkan semua nilai yang terkuantisasi bukanlah langkah yang dapat kembali. Kita akan kehilangan gambar awal. Kita menggunakan kuantisasi untuk mendapatkan bilangan bulat dan mengonversikan sejumlah besar nilai menjadi 0. Setelah terkuantisasi, pengodean Huffman bisa digunakan secara efektif.

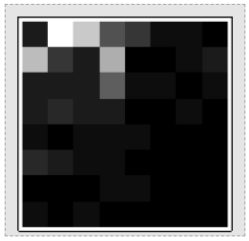


-27.500	-213.468	-149.608	-95.281	-103.750	-46.946	-58.717	27.226
168.229	51.611	-21.544	-239.520	-8.238	-24.495	-52.657	-96.621
-27.198	-31.236	-32.278	173.389	-51.141	-56.942	4.002	49.143
30.184	-43.070	-50.473	67.134	-14.115	11.139	71.010	18.039
19.500	8.460	33.589	-53.113	-36.750	2.918	-5.795	-18.387
-70.593	66.878	47.441	-32.614	-8.195	18.132	-22.994	6.631
12.078	-19.127	6.252	-55.157	85.586	-0.603	8.028	11.212
71.152	-38.373	-75.924	29.294	-16.451	-23.436	-4.213	15.624

DCT values in block row 4, block column 28.

DCT of the example block.

Gambar 20 Nilai DCT pada blok baris 4 kolom 28
(sumber : whydomath.org)

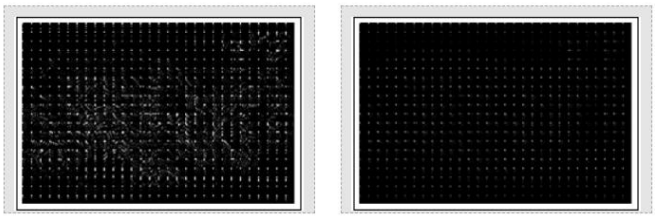


-2	-19	-15	-6	-4	-1	-1	0
14	4	-2	-13	0	0	-1	-2
-2	-2	-2	7	-1	-1	0	1
2	-3	-2	2	0	0	1	0
1	0	1	-1	-1	0	0	0
-3	2	1	-1	0	0	0	0
0	0	0	-1	1	0	0	0
1	0	-1	0	0	0	0	0

DCT values in block row 4, block column 28.

Quantized DCT of the example block.

Gambar 21 Nilai hasil kuantisasi pada blok baris 4 kolom 28
(sumber : whydomath.org)

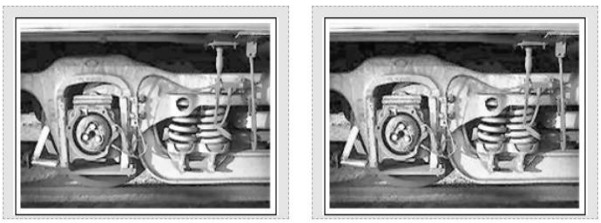


Gambar 22 Gambar hasil transformasi (kiri) dan gambar yang terkuantisasi.
(sumber : whydomath.org)

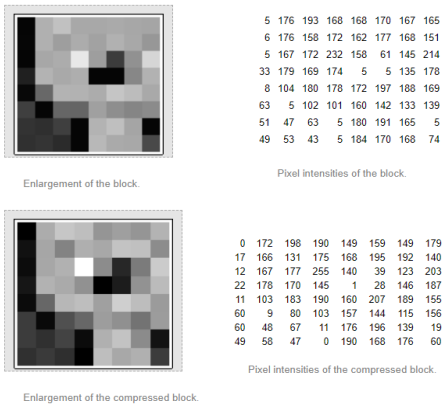
4. *Encoding* / Pengodean
   Pada tahap ini digunakanlah algoritma Huffman. Gambar awal memiliki dimensi 160 x 240 sehingga total bit adalah160 x 240 x 8 = 307.200 bit yang harus disimpan di *disk*. Jika kita menerapkan pengodean Huffman pada versi gambar yang tertransformasi dan terkuantisasi, kita hanya perlu 85.143 bit untuk penyimpanan. Rasio kompresi adalah 2,217 bpp yang artinya telah menghemat 70 % dari besaran awal.

D. Proses *Inverting*

Proses inversi diawali dengan dekode kode Huffman untuk mendapatkan gambar DCT yang terkuantisasi. Untuk membatalkan proses pengecilan, elemen di tiap 8x8 blok diperbesar oleh sebesar besaran yang pantas. Pada tahap ini kita memiliki blok C' yang sebelum kuantisasi DCT awalnya C = U B U^T. Setelah itu inverse DCT dengan komputasi B'= U^T C^U untuk tiap blok. Tahap akhirnya adalah dengan menambahkan 127 untuk tiap elemen di tiap blok, Matriks hasil merupakan aproksimasi dari gambar awal.



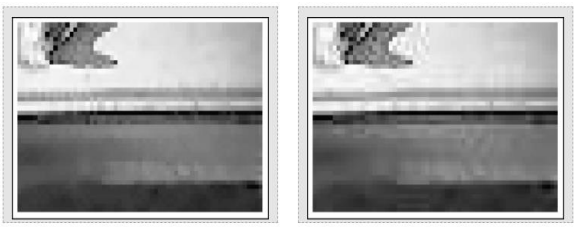
Gambar 23 Gambar awal (kiri) dan gambar yang sudah dikompres (kanan)
(sumber : whydomath.org)



Gambar 24 blok kolom 28 baris 4 awal (atas) dan blok kolom 28 baris 4 yang sudah dikompres (bawah)
(sumber : whydomath.org)

E. Masalah

Walaupun algoritma JPEG sangat efektif dalam mengompres gambar digital, ia memiliki kekurangan. Masalahnya adalah *decoupling* yang terjadi sebelum menerapkan DCT – mempartisi gambar menjadi 8 x 8 blok menghasilkan gambar yang terkompres terlihat "*blocky*".



Gambar 25 Bagian atas kanan gambar awal (kiri) dan bagian atas kanan gambar yang sudah dikompres(kanan)
(sumber : whydomath.org)

V. KESIMPULAN

Di dunia yang sangat pesat ini, kebutuhan manusia beragam. Kita memerlukan suatu inovasi agar keperluan kita dapat terpenuhi. Huffman Coding sangat membantu manusia dalam urusan kompresi *file* yang sangat dibutuhkan di dunia yang serba digital. Kompresi *file* memengaruhi kecepatan pengunggahan dan pengunduhan. Namun, yang Namanya teknologi tak lepas dari kekurangan. Pengodean Huffman pada algoritma JPEG-pun harus dikembangkan agar hasil kompresi bisa memuaskan.

VI. UCAPAN TERIMAKASIH

Penulis ingin mengucap syukur kepada Tuhan Yang Maha Esa karena berkar bimbingan-Nya makalah ini dapat selesai. Penulis juga ingin mengucapkan terima kasih kepada Dra. Harlili M.Sc. selaku pengajar Matematika Diskirt K-02. Tak lupa juga, penulis ingin mengucap terima kasih kepada Bapak Rinaldi Munir yang telah menyediakan bahan referensi untuk belajar di *website*.

REFERENSI

[1] https://www.officeorbiter.com/what-is-file-compression/ diakses pada tanggal 11 Desember 2020.

[2] https://www.kompas.com/skola/read/2020/10/13/180340869/apa-itu-gambar-digital?page=all diakses pada tanggal 11 Desember 2020.

[3] https://www.whydomath.org/node/wavlets/imagecompression.html diakses pada tanggal 11 Desember 2020.

[4] https://techterms.com diakses pada tanggal 11 Desember 2020.

[5] https://medium.com/breaktheloop/jpeg-compression-algorithm-969af03773da diakses 11 Desember 2020.

[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/ diakses tanggal 11 Desember 2020.

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, 3 Desember 2020

Jose Galbraith Hasintongan 13519022