KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0

SKRIPSI

Disusun Oleh

BARA FIRMANA BUDIONO

04. 12. 716

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2011

LEMBAR PERSETUJUAN

KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Komputer dan Informatika Strata Satu (S-1)

Disusun oleh :

BARA FIRMANA BUDIONO

04. 12. 716

Mengetahui,

Ketua Jurusan Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 101 880 0189

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
I Komang Somawirata, ST, MT.	**Irmalia Suryani Faradisa, ST, MT.**
NIP. Y. 103 010 0361	**NIP. P. 103 000 0365**

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2011

ABSTRAK

Nama : Bara firmana Budiono, NIM : 04.12.716, Jurusan : Teknik Komputer dan Informatika S-1, Fakultas Teknologi Industri, Institut Teknologi Nasional Malang, Judul Skripsi : " *KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0*"

Dosen Pembimbing : I I Komang Somawirata,ST, MT.

II Irmalia Suryani Faradisa,ST ,MT.

Kompresi citra yang dibuat menggunakan metode Huffman dengan delphi adalah pengembangan dari teknik kompresi huffman, karena sebelumnya teknik kompresi ini hanya diaplikasikan berupa media text sehingga penulis mengaplikasikan dalam bentuk media citra menggunakan bahasa pemrograman delphi.

Algoritma metode Huffman adalah salah satu algoritma kompresi untuk citra digital. Metode kompresi Huffman menggunakan prinsip bahwa nilai derajat keabuan yang sering muncul di dalam citra akan dikodekan dengan jumlah bit yang lebih sedikit sedangkan nilai keabuan yang frekuensi kemunculannya sedikit akan dikodekan dengan jumlah bit yang lebih panjang.

Citra dengan ukuran dimensi 122 x 124 dengan ukuran 16206 bit dapat dikompres menjadi 9554 dengan menggunakan rumus Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100% sehingga rasio kompresi yang didapatkan adalah 100% - (9554 / 16206) x 100% = 41.046%.

Kata kunci : *Citra, Kompresi, algoritma Huffman, pohon Huffman, Delphi 7.0*

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, atas karunia yang telah dilimpahkanNya schingga penulis dapat menyelesaikan tugas akhir dengan judul KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0.

Selanjutnya pada kesempatan ini penulis juga menyampaikan rasa terima kasih yang sebesar-besarnya kepada pihak–pihak yang telah banyak membantu penulis selama penyusunan tugas akhir, diantaranya :

1. Bapak Ir. Yusuf Ismail Nahkoda, MT selaku Ketua Jurusan Teknik Elektro S-1 ITN Malang.
2. Bapak Dr. Aryuanto Soetedjo, ST, MT selaku Sekertaris Jurusan Teknik Elektro S-1 ITN Malang dan pengusul serta penyedia ruang Skripsi.
3. Bapak I Komang Somawirata, ST, MT selaku Dosen Pembimbing I
4. Ibu Irmalia Suryani Faradisa, ST, MT selaku Dosen Pembimbing II

   Terima kasih karena telah membimbing selama ini dan memberikan masukan schingga skripsi ini menjadi jauh lebih baik.
5. Bapak Sotyohadi ST selaku Dosen Penguji I dan Dosen Wali.
6. Bapak Sonny Prasetio ST, MT selaku Dosen Penguji II
7. Kedua orang tua yang telah memberikan dukungan untuk selalu berdoa, berusaha dan nasehat yang telah diberikan sampai saat ini.
8. Seluruh dosen dan pegawai ITN Kampus 2 Malang.
9. Saudara-saudaraku di Malang, Surabaya dan Amerika. Terima kasih atas semua doa, dukungan dan bantuannya selama ini.

10. Selly Fajrina Yuliane, Joyce Khayrina dan Bheby Louise Jane Alexandra Bolton terima kasih untuk doa, inspirasi, masukan, semangat, dukungan, motivasi serta kesabaran yang tiada habis.
11. Mbak Trismi terima kasih buat tempatnya.
12. Semua teman-teman ITN Kampus 2 Malang dari angkatan 2004-2009
13. Semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini yang tidak bisa penulis sebutkan satu persatu..

Tentunya laporan tugas akhir ini masih memiliki banyak kekurangan dan kelemahan dalam penyusunannya. Oleh karena itu penulis mengharapkan kritik dan saran yang membangun demi kesempurnaan penyusunan tugas akhir ini. Besar harapan penulis laporan tugas akhir ini dapat bermanfaat bagi semua pihak.

Malang, September 2011

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Data atau informasi tidak hanya disajikan dalam bentuk teks, tetapi juga dapat berupa gambar, *audio* (bunyi, suara, musik) dan video. Keempat macam data atau informasi ini sering disebut dengan multimedia. Era informasi teknologi saat ini tidak dapat dipisahkan dari multimedia. Situs web *(website)* di internet dibuat semenarik mungkin dengan menyertakan visualisasi berupa gambar atau video yang dapat diputar [1].

Citra *(image)* sebagai salah satu komponen multimedia memegang peranan penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya akan informasi.

Pada umumnya representasi citra digital membutuhkan memori yang besar. Semakin besar ukuran citra tentu semakin besar pula memori yang dibutuhkannya. Pada sisi lain, kebanyakan citra mengandung duplikasi data. Duplikasi data pada citra dapat berarti dua hal. Pertama, besar kemungkinan suatu *pixel* dengan *pixel* tetangganya memiliki intensitas yang sama, sehingga penyimpanan setiap *pixel* memboroskan tempat. Kedua, citra banyak mengandung bagian *(region)* yang sama, sehingga bagian yang sama ini tidak perlu dikodekan berulang kali karena redundan .

Saat ini, kebanyakan aplikasi menginginkan representasi citra dengan kebutuhan memori yang sesedikit mungkin. Kompresi citra *(image compression)* bertujuan meminimalkan kebutuhan memori untuk merepresentasikan citra digital. Prinsip umum yang digunakan dalam kompresi citra adalah mengurangi duplikasi data di dalam citra schingga memori yang dibutuhkan untuk merepresentasikan citra menjadi lebih sedikit dari pada representasi citra semula.

Kompresi citra memberikan manfaat yang sangat besar dalam industri multimedia saat ini. Salah satunya adalah pada proses pengiriman data *(data transmission)* pada saluran komunikasi data. Citra yang telah dikomprsesi membutuhkan waktu pengiriman yang lebih singkat dibandingkan dengan citra yang tidak dikompresi. Contohnya aplikasi pengiriman gambar lewat *fax*, *videoconferencing*, pengiriman data medis, pengiriman gambar dari satelit luar angkasa, pengiriman gambar via telepon genggam, *download* gambar dari

internet, dan sebagainya. Selain pada proses pengiriman data, kompresi citra juga bermanfaat pada penyimpanan data *(data storing)* di dalam media sekunder *(storage)*. Citra yang telah dikompresi membutuhkan ruang memori di dalam media *storage* yang lebih sedikit dibandingkan dengan citra yang tidak dikompresi. Contoh aplikasinya antara lain aplikasi basisdata gambar, *office automation, video storage* (seperti *Video Compact Disc)* [1].

Ada dua tipe utama kompresi citra, yaitu kompresi tipe *lossless* dan kompresi tipe *lossy*. Kompresi tipe *lossy* adalah kompresi dimana terdapat data yang hilang selama proses kompresi.

Akibatnya kualitas citra yang dihasilkan jauh lebih rendah daripada kualitas citra asli. Sementara itu, kompresi tipe *lossless* tidak menghilangkan informasi setelah proses kompresi terjadi, akibatnya kualitas citra hasil kompresi tidak menurun. Salah satu contoh dari kompresi tipe *lossless* adalah metode Huffman. Metode Huffman paling efisien dari metode lain yang sejenis karena pemetaan lain simbol dari sumber data menjadi string unik menghasilkan file *output* yang lebih kecil ketika frekuensi symbol sesuai dengan frekuensi yang digunakan untuk menghasilkan kodenya [2].

Demikian pentingnya keberadaan kompresi citra dalam bidang multimedia maka pada tugas akhir ini penulis mengambil judul "Kompresi Citra Menggunakan Metode Huffman Dengan Bahasa Pemrograman Delphi 7.0".

1.2 Rumusan Masalah

Tugas akhir ini membahas kompresi citra dengan Metode Huffman dimana penyusunannya didasarkan pada rumusan sebagai berikut:

1. Bagaimana cara kerja dari Metode Huffman sebagai teknik kompresi citra
2. Bagaimana tingkat efisiensi citra hasil kompresi menggunakan Metode Huffman.
3. Bagaimana implementasi dari Metode Huffman sebagi teknik kompresi citra dengan menggunakan bahasa pemrograman Borland Delphi 7.0.

1.3 Batasan Masalah

Batasan masalah yang ada pada penyusunan tugas akhir ini adalah sebagi berikut:

1. Citra yang digunakan adalah citra *gray scale* dalam format BMP dengan kedalaman warna 8 bit.

2. Metode Huffman diimplementasikan menggunakan Borland Delphi 7.0.

1.4 Tujuan dan Manfaat Penulisan

1.4.1 Tujuan Penulisan

Tujuan dalam penyusunan tugas akhir ini adalah sebagai berikut:

1. Untuk mengetahui dan memahami cara kerja teknik kompresi citra dengan metode Huffman.
2. Untuk mengetahui tingkat efisiensi citra hasil kompresi menggunakan Metode Huffman.
3. Untuk mengimplementasikan Metode Huffman sebagai teknik kompresi citra dengan menggunakan Borland Delphi 7.0.

1.4.2 Manfaat Penulisan

Untuk meningkatkan wawasan dan pengetahuan tentang teknik kompresi citra.

1.5 Sistematika Penulisan

Secara garis besar pembahasan masing-masing bab dari keseluruhan isi laporan tugas akhir adalah sebagai berikut:

BAB I : PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

BAB II : LANDASAN TEORI

Bab ini berisi tentang teori atau konsep dasar yang berhubungan dengan judul penulisan tugas akhir.

BAB III : DESAIN SISTEM

Bab ini berisi desain yang digunakan penulis dalam penyusunan tugas akhir ini.

BAB IV : IMPLEMENTASI

Bab ini berisi uraian pembahasan dan implementasi program kompresi citra menggunakan Borland Delphi 7.0.

BAB V : PENGUJIAN METODE HUFFMAN DENGAN METODE LOSSY

Bab ini berisi pengujian metode Huffman terhadap metode *Lossy* dan grafik kualitas citra hasil pengkompresian.

BAB VI : PENUTUP

Bab ini merupakan bab akhir yang berisi kesimpulan dan saran.

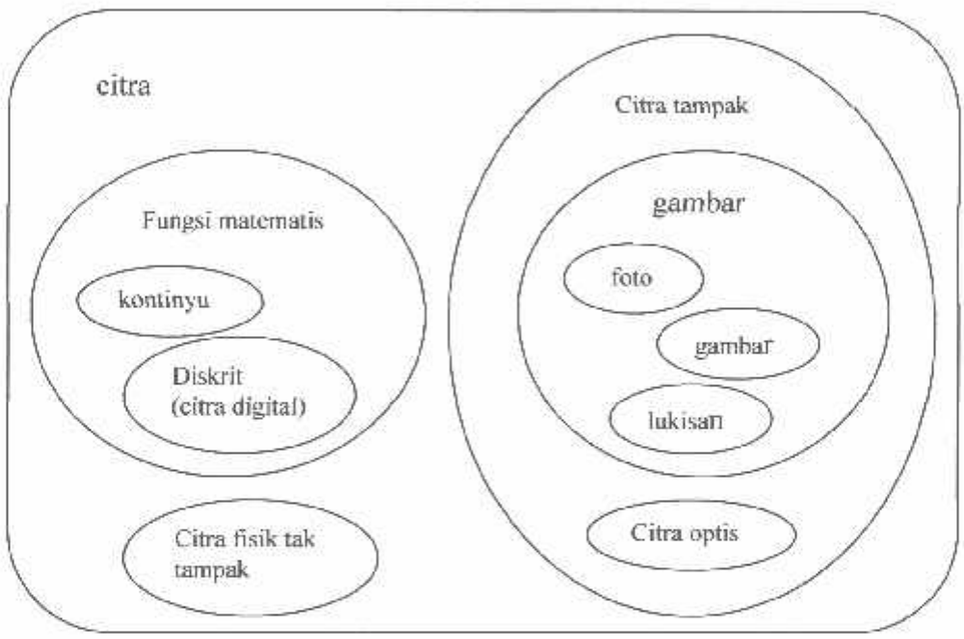
BAB II

LANDASAN TEORI

2.1 Citra

Definisi citra menurut Kamus Webster adalah "suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda". Foto sinar-X *thorax* mewakili keadaan bagian dalam tubuh seseorang, dan data dalam suatu file GIF mewakili apa yang digambarkannya.

Citra dapat dikelompokkan menjadi citra tampak dan citra tak tampak (lihat Gambar 2.1).



Gambar 2.1 Pengelompokan jenis-jenis citra (Castleman, 1996)

Contoh citra tampak dalam kehidupan sehari-hari adalah foto keluarga, lukisan, apa yang nampak di layar monitor dan televisi, serta hologram (citra optis). Sedangkan citra tak tampak misalnya data gambar dalam file (citra *digital*), dan citra yang direpresentasikam menjadi fungsi matematis. Disamping itu ada juga citra fisik tak nampak, misalnya citra distribusi panas di kulit manusia Untuk dapat dilihat mata manusia, citra tak nampak ini harus diubah

menjadi citra tampak, misalnya dengan menampilkannya di monitor, dicetak diatas kertas, dan sebagainya.

Diantara jenis-jenis citra tersebut, hanya citra *digital* yang dapat diolah menggunakan komputer. Citra jenis lain, jika hendak diolah dengan komputer, harus diubah dulu menjadi citra digital, misalnya foto dipindah *(scan)* dengan *scanner*. Kegiatan untuk mengubah informasi citra fisik non digital menjadi digital disebut sebagai pencitraan *(imaging)*.

Pada dasarnya citra yang dilihat oleh indera mata manusia merupakan sekumpulan cahaya dan dipersepsikan oleh otak manusia sebagai obyek tertentu. Suatu berkas cahaya dapat tersusun dari gelombang-gelombang dengan panjang yang berbeda-beda, yang masing-masing merepresentasikan warna yang berbeda-beda pula. Gelombang terpendek dimiliki oleh warna biru, sedangkan gelombang terpanjang dimiliki oleh warna merah. Pada dasarnya, mata manusia bekerja seperti kamera, yaitu cahaya yang masuk ke mata difokuskan oleh lensa ke retina sehingga terbentuk suatu gambar. Sesungguhnya hasil yang didapat dari fokus tersebut, merupakan pantulan citra dalam posisi terbalik. Retina mata terdiri atas kumpulan batang dan tiga macam kerucut. Batang retina peka terhadap cahaya lemah dan menghasilkan citra hitam-putih. Sedangkan kerucut peka terhadap cahaya terang dan menghasilkan persepsi tentang warna. Terdapat tiga jenis kerucut, yang masing-masing peka terhadap warna-warna yang berbeda yaitu *red* (R), *green* (G) dan *blue* (B). Pada otak manusia warna dipersepsikan dengan mengkombinasikan R-G, G-B dan B-R.

Pada suatu media atau perangkat digital yang sering dijumpai seperti kamera atau monitor, *image* atau citra digital direpresentasikan dengan beberapa karakteristik yang berbeda, antara lain ukuran citra, resolusi dan format nilainya.

2.2 Komponen Citra Digital

Setiap citra digital memiliki beberapa karakteristik, antara lain ukuran citra, resolusi, dan format lainnya. Umumnya citra digital berbentuk persegi panjang yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau piksel, sehingga ukuran citra selalu bernilai bulat.

Ukuran citra dapat juga dinyatakan secara fisik dalam satuan panjang (misalnya mm atau *inch*). Dalam hal ini tentu saja harus ada hubungan antara ukuran titik penyusun citra dengan satuan panjang. Hal tersebut dinyatakan dengan resolusi yang merupakan ukuran banyaknya titik untuk setiap satuan panjang. Biasanya satuan yang digunakan adalah dpi (*dot per inch*). Makin besar resolusi makin banyak titik yang terkandung dalam citra dengan ukuran fisik yang sama. Hal ini memberikan efek penampakan citra menjadi semakin halus.

Format citra digital ada bermacam-macam. Karena sebenarnya citra merepresentasikan informasi tertentu, sedangkan informasi tersebut dapat dinyatakan secara bervariasi, maka citra yang mewakilinya dapat muncul dalam berbagai format. Citra yang merepresentasikan informasi yang hanya bersifat biner untuk membedakan dua keadaan tentu tidak sama citra dengan informasi yang lebih kompleks sehingga memerlukan lebih banyak keadaan yang mewakilinya. Pada citra digital semua informasi tadi disimpan dalam bentuk angka, sedangkan penampilan angka tersebut biasanya dikaitkan dengan warna.

Citra digital tersusun atas titik-titik yang biasanya berbentuk persegi panjang atau bujursangkar yang secara beraturan membentuk baris-baris dan kolom. Setiap titik memiliki koordinat sesuai dengan posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 tergantung pada sistem yang digunakan. Dalam Delphi koordinat titik dalam citra dimulai dari 0. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili titik tersebut. Format nilai piksel sama dengan format citra keseluruhan. Pada kebanyakan sistem pencitraan, nilai ini biasanya berupa bilangan bulat positif juga.

2.3 Representasi Citra Digital

Komputer dapat mengolah isyarat-isyarat elektronik digital yang merupakan kumpulan sinyal biner (bernilai dua: 0 dan 1). Untuk itu citra digital harus mempunyai format tertentu yang sesuai sehingga dapat merepresentasikan obyek pencitraan dalam bentuk kombinasi data biner.

Pada kebanyakan kasus, terutama untuk keperluan penampilan secara visual, nilai data digital tersebut merepresentasikan warna dari citra yang diolah, dengan demikian format data citra digital berhubungan erat dengan warna. Format citra digital yang banyak dipakai adalah citra biner, skala keabuan, warna dan warna berindeks.

2.3.1 Citra Biner (monokrom)

Pada citra biner setiap titik bernilai 0 atau 1, masing-masing merepresentasikan warna tertentu. Contohnya adalah warna hitam bernilai 0 dan warna putih bernilai 1, pada standar citra yang ditampilkan di layar komputer, nilai biner ini berhubungan dengan ada tidaknya cahaya yang ditembakkan oleh *electron gun* yang terdapat dalam monitor komputer. Angka 0 menyatakan tidak ada cahaya, dengan demikian warna yang direpresentasikan adalah hitam. Untuk angka 1 terdapat cahaya, sehingga warna yang direpresentasikan adalah putih. Standar tersebut disebut sebagai standar citra cahaya, sedangkan standar citra tinta/cat adalah kebalikan, karena nilai biner tersebut menyatakan ada tidaknya tinta.

Setiap titik pada citra hanya membutuhkan satu bit, sehingga setiap *byte* dapat menampung informasi delapan titik.

2.3.2 Citra Skala Keabuan (*gray scale*)

Citra skala keabuan memberi kemungkinan warna yang lebih banyak daripada citra biner, karena ada nilai-nilai lain diantara nilai minimum (biasanya = 0) dan nilai maksimumnya. Banyaknya kemungkinan nilai dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Contohnya dalam skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$, dan nilai maksimumnya adalah $2^4 - 1 = 15$, sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya $2^8 - 256$, dan nilai maksimumnya $2^8 - 1 = 255$.

Format citra ini disebuat skala keabuan karena pada umumnya warna yang dipakai adalah antara hitam sebagai warna minimal dan putih sebagai warna maksimal, sehingga warna antaranya adalah abu-abu. Namun pada prakteknya warna yang dipakai tidak terbatas pada warna abu-abu, sehingga contoh yang dipilih warna minimalnya adalah putih dan warna maksimalnya adalah merah, maka semakin besar nilainya semakin besar pula intensitas warna merahnya. Beberapa buku menyebut format citra ini sebagai citra intensitas.

Jadi alasan mengapa menggunakan 8 bit adalah :

- Jika menggunakan 4 bit nilai *grey scale*nya minim diantara [0-15]
- Jika menggunakan 8 bit nilai *grey scale*nya mencukupi karena kisaran *grey scale*nya antara [0-255]
- Jika menggunakan 24 bit *true colour* (RGB), apabila salah satu dari RGB kekurangan maka RGB yang lain akan menutupi sehingga secara kasat mata citra tidak mengalami kerusakan rumusnya $R - \frac{R}{R+G+B}$ Sehingga nilainya sangat besar untuk pengkompresian citra
- Jadi keuntungan dari 8 bit adalah besarnya data penyimpanan citra 8 bit lebih kecil dari 24 bit *true colour*.

Dalam citra grey scale lebih ditekankan / difokuskan derajat keabuannya karena metode Huffman menggunakan pendekatan statistik dan prinsip metode Huffman bahwa nilai yang sering muncul dikodekan dengan jumlah bit yang lebih sedikit, sedangkan nilai yang kemunculannya sedikit akan dikodekan dengan jumlah bit yang lebih panjang.

2.3.3 Citra Warna (*true colour*)

Pada citra warna setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari tiga warna dasar, yaitu: merah, hijau, biru. Format citra ini sering disebut sebagai citra RGB (*red-green-blue*). Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit), misalnya warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB-nya adalah 255 255 0. Sedangkan warna ungu muda nilai RGB-nya adalah 150 0 150. Dengan demikian setiap titik pada citra warna membutuhkan 3 *byte*.

Jumlah kombinasi warna yang mungkin untuk format citra ini adalah 2^{24} atau lebih dari 16 juta warna, dengan demikian dianggap mencakup semua warna yang ada, inilah sebabnya format ini dinamakan *true colour*.

2.3.4 Citra Warna Berindeks

Jumlah memori yang dibutuhkan untuk format citra warna *true colour* adalah tiga kali jumlah titik yang ada dalam citra yang ditinjau. Di lain pihak pada kebanyakan kasus, jumlah warna yang ada dalam suatu citra terkadang sangat terbatas (jauh dibawah 10 juta kemungkinan warna yang ada) karena banyaknya warna dalam sebuah citra tidak mungkin melebihi banyaknya titik dalam citra itu sendiri. Untuk kasus tersebut disediakan format citra warna berindeks. Pada format ini, informasi setiap titik merupakan indeks dari suatu

tabel yang berisi informasi warna yang tersedia yang disebut palet warna (beberapa buku menyebutnya sebagai *colour map*).

Jumlah bit yang dibutuhkan oleh setiap titik pada citra bergantung pada jumlah warna yang tersedia dalam palet warna. Sebagai contoh, untuk palet berukuran 16 warna, setiap titik membutuhkan 4 bit, dan untuk palet berukuran 256 warna, setiap setiap titik membutuhkan 8 bit atau 1 byte. Palet warna merupakan bagian dari citra warna berindeks, sehingga pada saat menyimpan citra ini ke dalam file, informasi warna palet juga harus disertakan.

Keuntungan pemakaian palet warna ini adalah kita dapat dengan cepat memanipulasi warna tanpa harus mengubah informasi pada setiap titik dalam citra. Keuntungan lainnya adalah besarnya data yang diperlukan untuk menyimpan citra ini lebih kecil dibandingkan dengan citra warna *true color*.

Setting warna *display* pada Microsoft Windows biasanya memiliki beberapa pilihan sesuai dengan format citra yangh telah dijelaskan, yaitu format 16 *colour*, 256 *colour*, dan *high colour* (yang merupakan citra warna berindeks dengan ukuran palet masing-masing 4 bit, 8 bit, 16 bit), serta *true colour*.

2.4 Digitasi Citra

Agar dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dari fungsi kontinu menjadi nilai-nilai diskrit disebut digitasi. Citra yang dihasilkan inilah yang disebut citra digital (*digital image*). Pada umumnya citra digital berbentuk empat persegi panjang, dan dimensi ukurannya dinyatakan sebagai tinggi x lebar (lebar x panjang).

Citra digital yang tingginya N, lebarnya M, dan memiliki L derajat kcabuan dapat dianggap sebagai fungsi

$$f(x,y) \begin{cases} 0 \le x \le M \\ 0 \le x \le N \\ 0 < x < L \end{cases}$$

Citra digital yang berukuran N x M lazim dinyatakan dengan matrik yang berukuran N baris dan M kolom sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M) \\ f(1,0) & f1,1) & \dots & f(1,M) \\ . & . & . & . \\ . & . & . & . \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Indeks baris (i) dan indeks kolom(j) menyatakan suatu koordinat titik pada citra, sedangkan f(i,j) merupakan intensitas (derajat keabuan pada titik (i,j).

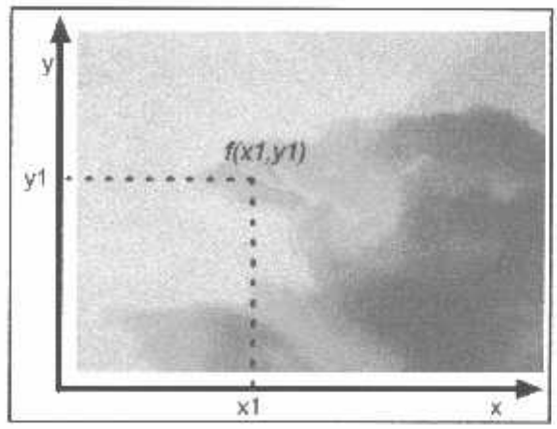
Masing-Masing elemen pada citra digital (berarti elemen matrik) disebut *image element* , *picture element* atau *pixel*. Jadi citra yang berukuran N x M mempunyai NM buah *pixel*. Sebagai contoh, misalkan sebuah citra berukuran 256 x 256 *pixel* dan direpresentasikan secara numerik dengan matrik yang terdiri dari 256 buah baris (di-indeks dari 0 sampai 255) dan 256 buah kolom (di-indeks dari 0 sampai 255) seperti contoh berikut:

$$\begin{bmatrix} 0 & 134 & 145 & \dots & \dots & 231 \\ 0 & 167 & 201 & \dots & \dots & 197 \\ 220 & 187 & 189 & \dots & \dots & 120 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 221 & 219 & 210 & \dots & \dots & 156 \end{bmatrix}$$

Pixel pertama pada koordinat (0,0) mempunyai nilai intensitas 0 yang berari warna *pixel* tersebut hitam, *pixel* kedua pada koordinat (0,1) mempunyai intensitas 134 yang berari warnanya antara hitam dan putih, dan seterusnya.

2.5 Pengolahan Citra

Citra digital dapat didefinisikan sebagai fungsi dua variabel, *f(x,y)*, dimana *x* dan *y* adalah koordinat spasial dan nilai *f(x,y)* adalah intensitas citra pada koordinat tersebut, hal tersebut diilustrasikan dalam Gambar 2.2.

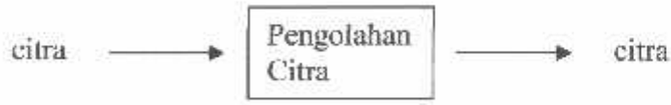


Gambar 2.2 Citra Digital

Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*Red, Green, Blue* - RGB).

Sebuah citra diubah ke bentuk digital agar dapat disimpan dalam memori komputer atau media lain. Proses mengubah citra ke bentuk digital bisa dilakukan dengan beberapa perangkat, misalnya *scanner*, kamera digital, dan *handycam*. Ketika sebuah citra sudah diubah ke dalam bentuk digital (selanjutnya disebut citra digital), bermacam-macam proses pengolahan citra dapat diperlakukan terhadap citra tersebut.

Pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik dari pada citra masukan. Termasuk ke dalam bidang ini juga adalah kompresi citra [3].



Gambar 2.3 Pengolahan citra

Umumnya, operasi-operasi pada pengolahan citra diterapkan pada citra bila [4]:

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur.

3. Sebagian citra perlu digabung dengan bagian citra yang lain.

Operasi pengolahan citra dapat diklasifikasikan dalam berbagai jenis sebagai berikut:

a. Perbaikan kualitas citra *(image enhancement)*

Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat dalam citra lebih ditonjolkan. Contoh-contoh operasi perbaikan citra:

- Perbaikan kontras gelap/terang
- Perbaikan tepian objek *(edge enhancement)*
- Penajaman *(sharpening)*
- Pemberian warna semu *(pseudocoloring)*
- Penapisan derau *(noise filtering)*

b. Pemugaran citra *(image restoration)*

Operasi ini bertujuan menghilangkan atau meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya pada pemugaran citra penyebab degradasi gambar diketahui. Contoh-contoh operasi pemugaran citra:

- Penghilangan kesamaran *(deblurring)*
- Penghilangan derau *(noise)*

c. Pemampatan citra *(image compression)*

Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan citra adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus.

d. Segmentasi citra *(image segmentation)*

Operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

e. Pengorakan citra *(image analysis)*

Operasi ini bertujuan menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadangkala

diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh-contoh operasi pengorakan citra :

- Pendeteksian tepi objek *(edge detection)*
- Ekstraksi batas *(boundary)*
- Representasi daerah *(region)*

f. Rekonstruksi citra *(image reconstruction)*

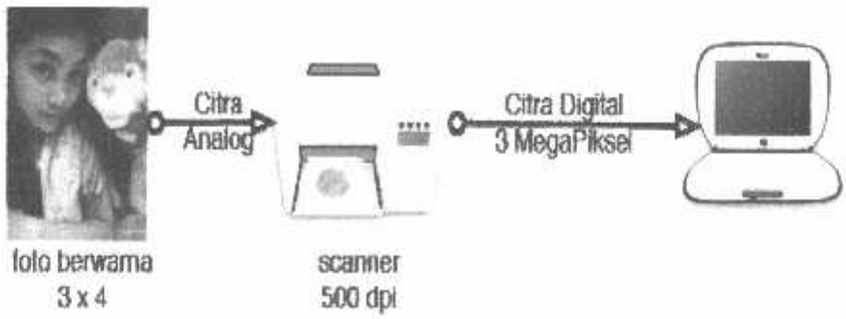
Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunaka dalam bidang medis. Misalnya beberapa foto *rontgen* dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh.

2.6 Kompresi Citra

Representas citra digital membutuhkan memori yang besar. Semakin besar ukuran citra akan mengakibatkan semakin besar pula memori yang diperlukan Kompresi Citra (pemampatan citra) adalah aplikasi kompresi data yang dilakukan terhadap citra digital dengan tujuan untuk mengurangi redundansi dari data-data yang terdapat dalam citra sehingga dapat disimpan atau ditransmisikan secara efisien[5].

Kompresi citra bertujuan untuk meminimalkan jumlah bit yang diperlukan untuk merepresentasikan citra. Apabila sebuah foto berwarna berukuran 3 inci x 4 inci diubah ke bentuk digital dengan tingkat resolusi sebesar 500 *dot per inch* (dpi), maka diperlukan 3 x 4 x 500 x 500 = 3.000.000 dot (piksel). Setiap piksel terdiri dari 3 *byte* dimana masing-masing *byte* merepresentasikan warna merah, hijau, dan biru. sehingga citra digital tersebut memerlukan *volume* penyimpanan sebesar 3.000.000 x 3 *byte* +1080 = 9.001.080 *byte* setelah ditambahkan jumlah *byte* yang diperlukan untuk menyimpan *format* (*header*) citra.

Citra yang belum dikompres disebut citra mentah (*raw image)*. Sementara citra hasil kompresi disebut citra terkompresi(*compressed image)*. Proses pengiriman dan penyimpanan citra tersebut diilustrasikan dalam Gambar 2.3.



Gambar 2.4 Proses Konversi citra analog ke citra digital

Parameter-parameter citra yang penting dalam proses kompresi diantaranya adalah sebagai berikut:

1. Resolusi

   Resolusi citra menyatakan ukuran panjang kali lebar dari sebuah citra. Resolusi citra biasanya dinyatakan dalam satuan piksel. Semakin tinggi resolusi sebuah citra, semakin baik kualitas citra tersebut. Namun, tingginya resolusi menyebabkan semakin banyaknya jumlah bit yang diperlukan untuk menyimpan dan mentransmisikan data citra tersebut.

2. Kedalaman bit

   Kedalaman bit menyatakan jumlah bit yang dipelukan untuk mrepresentasikan tiap piksel citra pada sebuah *frame*. Kedalaman bit biasanya dinyatakan dalam satuan *bit*/piksel. Semakin banyak jumlah *bit* yang digunakan untuk merepresentasikan sebuah citra, maka semakin baik kualitas citra tersebut.

3. Konsep redundansi

   Redundansi merupakan suatu keadaan dimana representasi suatu elemen data tidak bernilai signifikan dalam merepresentasikan keseluruhan data. Keadaan ini menyebabkan data keseluruhan dapat direpresentasikan secara lebih kompak dengan cara menghilangkan representasi dari sebuah elemen data yang redundan. Redundansi yang terdapat pada citra statik adalah redundansi spasial.

Dalam proses kompresi (pemampatan) citra terdapat dua proses utama yaitu sebagai berikut:

1. Pemampatan citra *(image compression)*

   Pada proses, ini citra dalam representasi tidak mampat dikodekan dengan representasi yang meminimumkan kebutuhan memori.

2. Penirmampatan citra *(image decompression)*

   Pada proses ini, citra yang sudah dimampatkan harus dapat dikembalikan lagi *(decoding)* menjadi representasi yang tidak mampat. Proses ini diperlukan jika citra tersebut akan ditampilkan ke layar atau disimpan ke dalam arsip dengan format tidak mampat.

Saat ini sudah banyak ditemukan metode-metode pemampatan citra. Kriteria yang digunakan dalam mengukur metode pemampatan citra adalah [6]:

1. Waktu pemampatan dan penirmampatan *(decompression)*

   Waktu pemampatan dan penirmampatan sebaiknya cepat. Ada metode pemampatan yang waktu pemampatannya lama, namun waktu penirmampatannya cepat. Ada pula metode yang waktu pemampatannya cepat tetapi waktu penirmampatannya lambat. Tetapi ada pula metode yang waktu pemampatan dan penirmampatannya cepat atau keduanya lambat.

2. Kebutuhan memori

   Memori yang dibutuhkan untuk merepresentasikan citra seharusnya berkurang secara berarti. Ada metode yang berhasil memampatkan dengan presentase yang besar, ada pula yang kecil. Pada beberapa metode, ukuran memori hasil pemampatan bergantung pada citra itu sendiri. Citra yang mengandung banyak elemen duplikasi (misalnya citra langit cerah tanpa awan, citra lantai keramik) umumnya berhasil dimampatkan dengan memori yang lebih sedikit dibandingkan dengan memampatkan citra yang mengandung banyak objek (misalnya citra pemandangan alam).

3. Kualitas pemampatan *(fidelity)*

   Informasi yang hilang akibat pemampatan seharusnya seminimal mungkin sehingga kualitas hasil pemampatan tetap dipertahankan. Kualitas pemampatan dengan kebutuhan memori biasanya berbanding terbalik. Kualitas pemampatan yang bagus umumnya dicapai pada proses

pada pemampatan yang menghasilkan pengurangan memori yang tidak begitu besar, demikian pula sebaliknya. Dengan kata lain, ada timbal balik *(trade off)* antara kualitas citra dengan ukuran hasil pemampatan.

Kualitas sebuah citra bersifat subyektif dan relatif, bergantung pada pengamatan orang yang menilainya. Seseorang dapat saja mengatakan kurang bagus, jelek, dan sebagainya.

4. Format keluaran

   Format citra hasil pemampatan sebaiknya cocok untuk pengiriman dan penyimpanan data. Pembacaan citra bergantung pada bagaimana citra tersebut direpresentasikan.

Ada empat jenis pendekatan dalam pemampatan citra adalah sebagai berikut:

1. Pendekatan statistik

   Pemampatan citra didasarkan pada frekuensi kemunculan derajat keabuan *pixel* di dalam seluruh bagian gambar. Contoh metode: *Huffman Coding.*

2. Pendekatan ruang

   Pemampatan citra didasarkan pada hubungan spasial antara *pixel-pixel* di dalam suatu kelompok yang memiliki derajat keabuan yang sama di dalam suatu daerah di dalam gambar. Contoh metode: *Run-Length Encoding.*

3. Pendekatan kuantisasi

   Pemampatan citra dilakukan dengan mengurangi derajat keabuan yang tersedia. Contoh metode: metode pemampatan kuantisasi.

4. Pendekatan *fractal*

   Pemampatan didasarkan pada kenyataan bahwa kemiripan bagian-bagian di dalam citra dapat dieksploitasi dengan suatu matriks transformasi. Contoh: *Fractal Image Compression.*

Metode pemampatan citra dapat diklasifikasikan ke dalam dua kelompok besar yaitu:

1. Metode *lossless*

   Metode *lossless* selalu menghasilkan citra hasil penirmampatan yang tepat sama dengan citra semula, *pixel* per *pixel.* Tidak ada informasi yang hilang akibat pemampatan. Contoh metode *lossless* adalah metode *Huffman.* Rasio kompresi citra dihitung dengan rumus:

$$\text{Rasio kompresi} = 100\% - \left(\frac{\text{ukuran citra hasil kompresi}}{\text{ukuran citra semula}}\right) \times 100\%$$

Metode *lossless* cocok untuk memampatkan citra yang mengandung informasi penting yang tidak boleh rusak akibat pemampatan. Misalnya memampatkan gambar hasil diagnosa medis.

2. Metode *lossy*

   Metode *lossy* menghasilkan citra hasil pemampatan yang hampir sama dengan citra semula. Ada informasi yang hilang akibat pemampatan, tetapi dapat ditolerir oleh persepsi mata. Mata tidak dapat membedakan perubahan kecil pada gambar. Metode pemampatan *lossy* menghasilkan nisbah pemampatan yang tinggi daripada metode *lossless*.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi *file input*) menjadi sekumpulan *codeword*, metode kompresi terbagi menjadi dua kelompok, yaitu :

1. Metode statik

   Menggunakan peta kode yang selalu sama. Metode ini membutuhkan dua fase (*two-pass*) yaitu fase pertama untuk menghitung probabilitas kemunculan tiap simbol/karakter dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan ditransmisikan. Contoh: algoritma Huffman statik.

2. Metode dinamik (adaptif) : menggunakan peta kode yang dapat berubah dari waktu ke waktu. Metode ini disebut adaptif karena peta kode mampu beradaptasi terhadap perubahan karakteristik isi file selama proses kompresi berlangsung. Metode ini bersifat *onepass*, karena hanya diperlukan satu kali pembacaan terhadap isi file. Contoh: algoritma LZW dan DMC.

Berdasarkan teknik pengkodean/pengubahan simbol yang digunakan, metode kompresi dapat dibagi ke dalam tiga kategori, yaitu :

1. Metode *symbolwise* : menghitung peluang kemunculan dari tiap symbol dalam file input, lalu mengkodekan satu simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek

dibandingkan simbol yang lebih jarang muncul, contoh: algoritma Huffman.

2. Metode *dictionary* : menggantikan karakter/fragmen dalam file input dengan indeks lokasi dari karakter/fragmen tersebut dalam sebuah kamus (*dictionary*), contoh: algoritma LZW.
3. Metode *predictive* : menggunakan model *finite-context* atau *finite-state* untuk memprediksi distribusi probabilitas dari simbol-simbol selanjutnya; contoh: algoritma DMC.

2.7 Metode Huffman

Dalam ilmu komputer dan teori informasi, kode Huffman adalah algoritma pengkodean untuk kompresi data *lossless*. Istilah ini merujuk kepada penggunaan tabel kode yang memiliki panjang bervariasi (*variable length code*) dimana tabel kode tersebut diturunkan dengan cara tertentu berdasarkan prakiraan probabilitas kemunculan setiap nilai dalam sumber data. Metode ini ditemukan oleh David A. Huffman ketika ia melakukan studi Ph.D di MIT. Kode ini dipublikasikan pada tahun 1952 pada tulisannya yang berjudul "*A Method for the Constuction of Minimim-Redudancy Codes*". Huffman menjadi anggota fakultas MIT begitu ia lulus, dan di kemudian hari menjadi salah satu pendiri Departemen Ilmu Komputer Universitas California, Santa Cruz, yang sekarang menjadi bagian dari Baskin School of Engineering[7].

Kode Huffman menggunakan metode spesifik untuk merepresentasikan setiap symbol, menghasilkan prefix-free code (*string* dari bit representasi sebuah simbol tidak pernah menjadi prefix (awalan) dari sebuah simbol lain). Yang merepresentasikan karakter yang lebih sering muncul dengan *bit string* yang lebih pendek daripada karakter yang jarang muncul dalam suatu sumber data.

Kompresi Huffman adalah metode paling efisien dari metode lain yang sejenis karena pemetaan lain simbol dari sumber data menjadi string unik menghasilkan file *output* yang lebih kecil ketika frekuensi simbol sesuai dengan frekuensi yang digunakan untuk menghasilkan kodenya.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (isi data yang diinputkan) menjadi sekumpulan *codeword*, algoritma Huffman termasuk kedalam kelas algoritma yang menggunakan metode statik. Metode statik adalah metode yang selalu menggunakan peta kode yang sama, metode ini membutuhkan dua fase (*two-pass*): fase pertama untuk menghitung probabilitas

kemunculan tiap simbol dan menentukan peta kodenya, dan fase kedua untuk mengubah pesan menjadi kumpulan kode yang akan di taransmisikan. Sedangkan berdasarkan teknik pengkodean simbol yang digunakan, algoritma Huffman menggunakan metode *symbolwise*. Metoda *symbolwise* adalah metode yang menghitung peluang kemunculan dari setiap simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang jarang muncul.

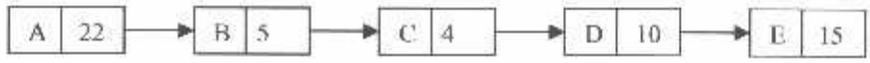
Kode Huffman pada dasarnya merupakan kode prefiks (*prefix code*). Kode prefiks adalah himpunan yang berisi sekumpulan kode biner, dimana pada kode prefik ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Kode prefiks biasanya direpresentasikan sebagai pohon biner yang diberikan nilai atau label. Untuk cabang kiri pada pohon biner diberi label 0, sedangkan pada cabang kanan pada pohon biner diberi label 1. Rangkaian bit yang terbentuk pada setiap lintasan dari akar ke daun merupakan kode prefiks untuk karakter yang berpadanan. Pohon biner ini biasa disebut pohon Huffman.

Kompresi data yang dihasilkan pada algoritma pengkodean Huffman ini dicapai dengan mengkodekan data berdasarkan pada frekuensi kemunculannya. Struktur data yang digunakan untuk mengkodekan data adalah suatu *weighted binary tree*, atau pohon Huffman (*Huffmantree*). Suatu pohon Huffman memiliki ciri-ciri unikn sebagai berikut:

1. Pohon Huffman harus merupakan suatu binary tree.
2. Pada pohon Huffman ini elemen yang paling sering muncul dalam *data stream* berada di puncak pohon sedangkan elemen yang paling jarang muncul berada di dasar pohon.
3. Tiap cabang kiri dari pohon Huffman diberi tanda sebagai nilai nol, dan tiap cabang kanan dari pohon diberi tanda sebagai nilai satu (atau sebaliknya).

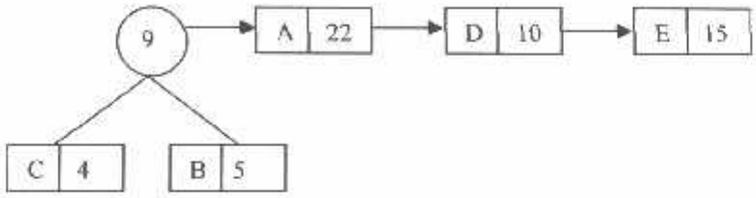
Langkah pembentukan pohon Huffman adalah sebagai berikut:

1. Dari semua data yang diketahui, lengkap dengan frekuensi masing-masing data dibentuk satu nilai yang berhubungan. Sebagai contoh dalam gambar 2.5 terdapat lima komponen dengan masing-masing frekuensinya



Gambar 2.5 Lima komponen dan Frekuensinya

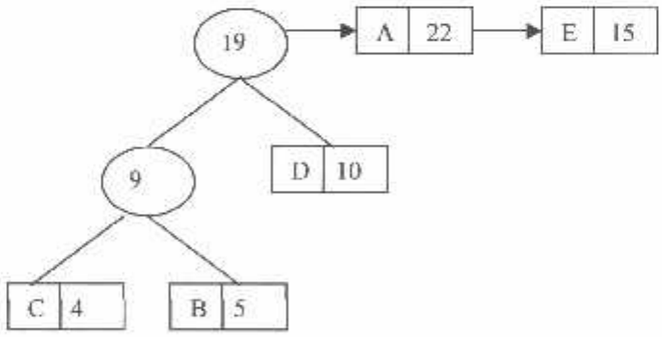
2. Memilih dua buah nilai yang mempunyai frekuensi paling kecil



Gambar 2.6 Proses pelepasan Frekuensi

Selanjutnya kedua komponen tersebut kita lepas dari nilai yang berhubungan. Dari kedua komponen tersebut kita susun sebuah pohon yang akarnya berisi jumlah frekuensi kedua komponen yang dilepas dari senarai berantai tersebut. Selanjutnya akar pohon tersebut kita sisipkan sebagai awal dari nilai yang berhubungan.

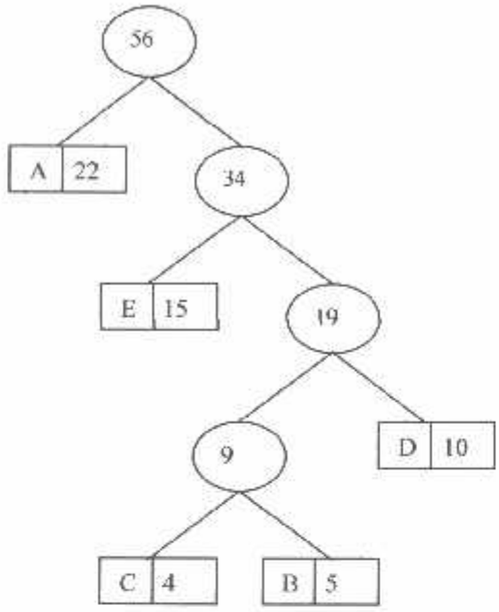
3. Langkah tersebut diulang sehingga akan diperoleh seperti di bawah ini



Gambar 2.7 Proses pelepasan Selanjutnya

4. Ulangi kembali langkah diatas sampai komponen nilai yang berhubungan tinggal sebuah yaitu akar pohon secara keseluruhan. Dari setiap dua komponen yang akan dibentuk pohon, komponen yang

mempunyai frekuensi lebih kecil dipasang sebagai cabang kiri dan komponen yang lebih besar dipasang di cabang kanan seperti dalam gambar 2.8.



Gambar 2.8 Proses Terbentuknya Pohon Huffman

Metode pemampatan Huffman menggunakam prinsip bahwa nilai (atau derajat) keabuan yang sering muncul di dalam citra akan dikodekan dengan jumlah bit yang lebih sedikit sedangkan nilai keabuan yang frekuensi kemunculannya sedikit dikodekan dengan jumlah bit yang lebih panjang.

Alogoritma metode Huffman adalah sebagai berikut:

1. Urutkan secara menaik *(ascending order)* nilai-nilai keabuan berdasarkan frekuensi kemunculannya (atau berdasarkan peluang kemunculan, p_k, yaitu frekuensi kemunculan (n_k) dibagi dengan jumlah *pixel* di dalam gambar (n). Setiap nilai keabuan dinyatakan sebagai pohon bersimpul tunggal. Setiap simpul di-*assign* dengan frekuensi kemunculan nilai keabuan tersebut.
2. Gabung dua pohon yang mempunyai frekuensi kemunculan paling kecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua pohon penyusunnya.
3. Ulangi langkah 2 sampai tersisa hanya satu pohon biner.

Agar pemilihan dua pohon yang akan digabungkan berlangsung cepat, maka semua pohon yang ada selalu terurut menaik berdasarkan frekuensi.

4. Beri label setiap sisi pada pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1.

   Simpul-simpul daun pada pohon biner menyatakan nilai keabuan yang terdapat di dalam citra semula.

5. Telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan yang sesuai.

2.8 Gambaran Umum Borland Delphi 7.0

Delphi merupakan program aplikasi visual database berbasis *Windows*, yaitu suatu perangkat lunak berbasis pemograman Pascal yang dikembangkan oleh *Borland Coorporation.* Perangkat lunak ini sangat terkenal dilingkungan pengembang aplikasi karena mudah untuk dipelajari dan dapat digunakan untuk menangani berbagai hal. Berbagai aplikasi dapat dibuat menggunakan Delphi termasuk aplikasi untuk mengolah teks, angka, database, dan aplikasi web.

Beberapa kelebihan Delphi khususnya jika dibandingkan dengan bahasa pemrograman lain adalah :

a. Pengolahan *memory* lebih mudah.
b. Aplikasi dapat dikembangkan diatas berbagai macam sistem operasi, misalnya Windows, Unix dan Linux.
c. Tipe data yang lebih lengkap.
d. Pada umumnya aplikasi yang dikembangkan dengan Delphi akan berjalan lebih cepat daripada aplikasi yang dikembangakn dengan Visual Basic.
e. Kode-kode dalam bahasa perakit dapat diselipkan dalam Delphi.

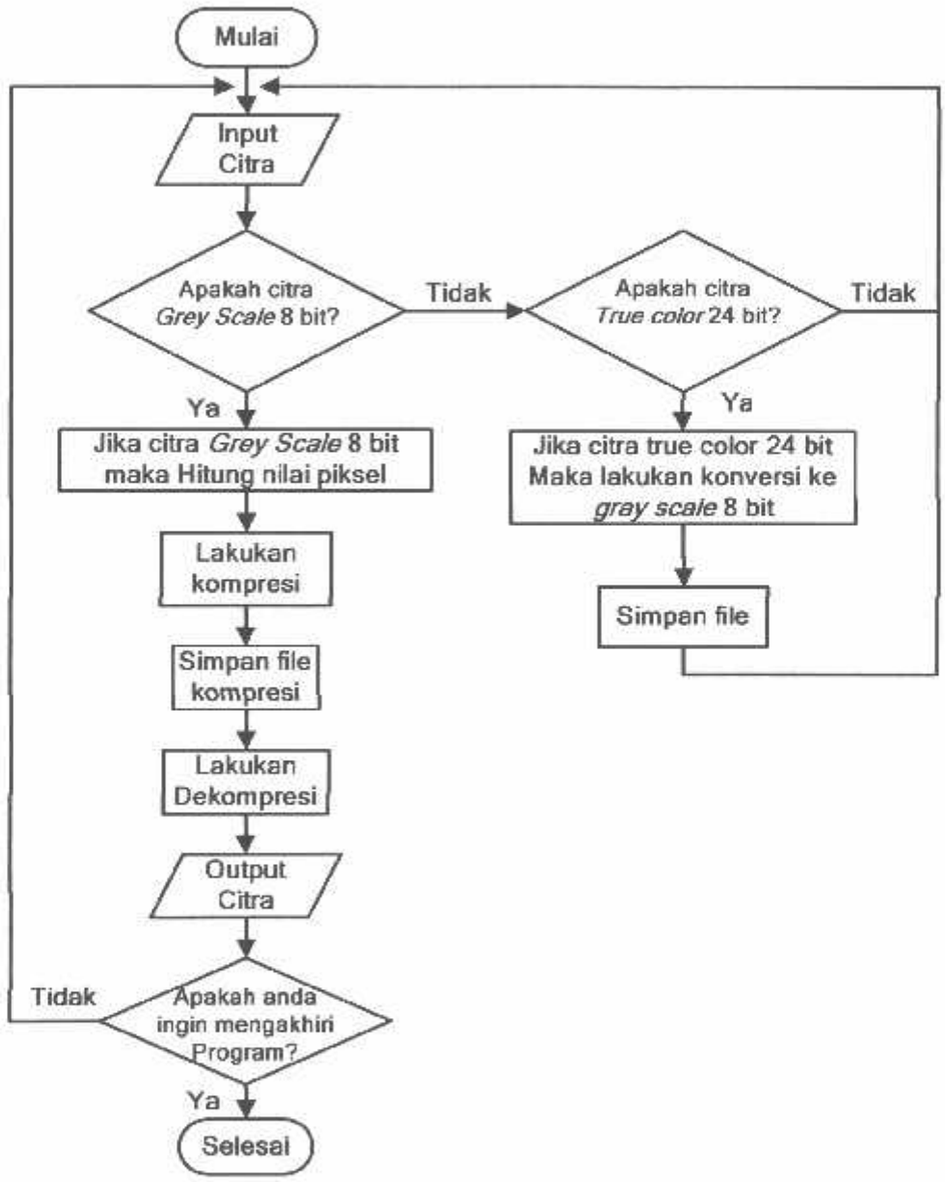
Delphi tidak menyediakan secara khusus rutin-rutin untuk pengolahan citra, oleh karena itu perlu dibuat sendiri program untuk mengolah citra, yaitu melalui komponen *TImage* yang terdapat pada palet komponen *Additional*.

BAB III

DESAIN SISTEM

3.1 Teknik Kompresi

Teknik kompresi pada citra *gray scale* 8 bit secara garis besar dapat digambarkan seperti dalam gambar 3.1.

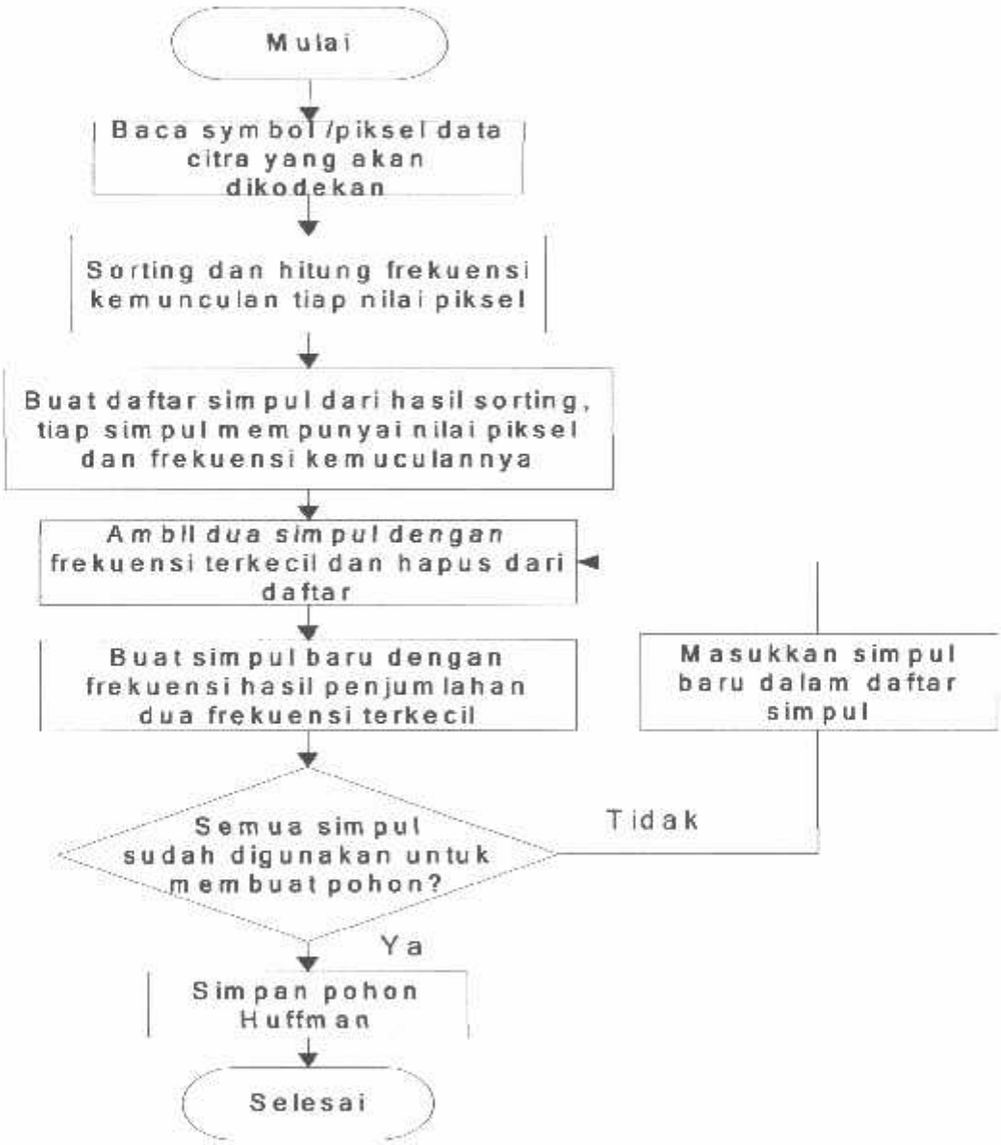


Gambar 3.1 *Flowchart* Teknik Kompresi

Dari citra *gray scale* 8 bit diambil nilai pikselnya dan dihitung frekuensi kemunculan masing-masing nilai piksel tersebut. Kemudian dilakukan proses kompresi dengan metode Huffman. Hasil file kompresi tersebut disimpan beserta kode Huffmannya. Untuk melakukan proses dekompresi tersebut diambil dari file hasil kompresi dengan mengembalikan kode Huffmannya. Dari file hasil dekompresi akan dihasilkan citra yang sama dengan cira sebelum kompresi.

3.2 Pohon Huffman

Flowchart Proses pembentukan pohon Huffman dapat ditunjukkan dalam 3.2.

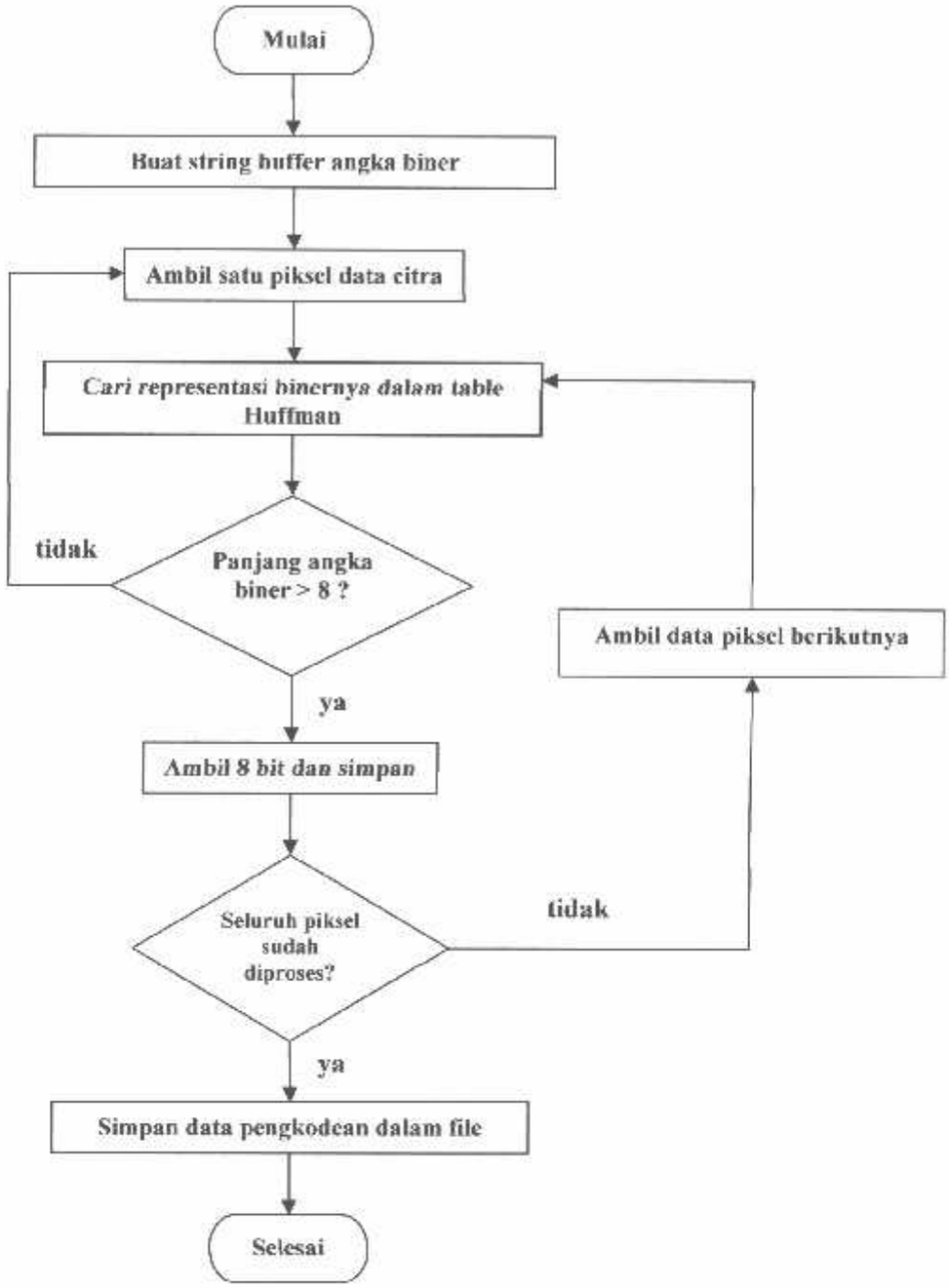


Gambar 3.2 *Flowchart* pohon Huffman

Pohon Huffman dibuat dengan menggunakan algoritma pengkodean Huffman di atas. Dari data piksel dihitung frekuensi kemunculannya. Tiap data dianggap satu simpul yang mempunyai dua nilai data: nilai keabuan dan frekuensi kemunculannya. Dua simpul dengan frekuensi terkecil dari daftar diambil dan frekuensinya dijumlahkan. Hasil penjumlahan kedua frekuensi ini menjadi frekuensi baru untuk sebuah simpul baru yang mempunyai dua cabang kedua simpul tadi (cabang kiri dan kanan). Simpul baru ini selanjutnya dimasukkan dalam daftar untuk dilakukan pengurutan simpul, penjumlahan dua simpul dengan frekuensi terkecil, dan pembuatan simpul baru. Proses ini berlanjut sampai semua simpul telah masuk ke dalam pohon dan frekuensi akar (*root*) dari pohon akan merupakan hasil penjumlahan semua frekuensi yang ada.

3.3 Tabel Huffman

Setelah dibentuk pohon Huffman maka dilakukan pembentukan tabel Huffman yang digunakan untuk menyimpan hasil dari pengkodean Huffman. Dalam gambar 3.3 ditunjukkan *flowchart* metode Huffman.



Gambar 3.3 *Flowchart* tabel Huffman

BAB IV
PEMBAHASAN DAN IMPLEMENTASI

4.1 Pembahasan

4.1.1 Komponen TImage pada Delphi

Untuk menampilkan citra yang akan dikompres menggunakan komponen TImage yang terdapat pada palet komponen *Additional* pada Delphi. Komponen ini memiliki properti *Picture* yang digunakan untuk menyimpan data citra. Citra yang akan ditampilkan diambil pada saat program dijalankan dengan menggunakan *procedure LoadFromFile*. Properti yang ada pada picture antara lain:

1. *Height*, berisi nilai tinggi citra.
2. *Width*, berisi nilai lebar citra.
3. *Bitmap*, berisi data format dan piksel citra.

Dalam Delphi, informasi format citra terdapat pada subproperti Bitmap, yaitu PixelFormat. Dengan membaca nilai PixelFormat dapat diketahui cara penyimpanan piksel dalam memori. Sehingga mempermudah dalam pemrograman. Untuk citra *gray scale* 8 bit mempunyai nilai PixelFormat pf8bit, setiap piksel disimpan dalam ukuran 1 byte (8 bit). Pf8bit ini direpresentasikan sebagai citra berindeks 8 bit dengan komponen palet warna merah, hijau dan biru yang bernilai sama, sehingga menampilkan warna keabuan dari hitam sampai putih.

4.1.2 Struktur Data

Struktur data yang digunakan pada program kompresi ini menggunakan pointer yang elemennya berupa rekaman (*record*). Ada dua pointer yang digunakan yaitu pointer untuk pembentukan pohon Huffman dan pointer untuk pembentukan kode Huffman.

1. Deklarasi pointer untuk pohon huffman

```
type
 Phuffinfo=^Thuffinfo;
 THuffInfo=record
    kiri:phuffinfo;
```

```
    kanan:phuffinfo;
    piksel: array[0..255] of byte;
    jml_kode:integer;
    frek:integer;
    sudah:boolean;
  end;
```

Pada deklarasi diatas `Phuffinfo` menyatakan data yang bertipe pointer. `Thuffinfo` merupakan nilai data yang ditunjuk oleh pointer `Phuffinfo`. Simpul-simpul yang akan dibentuk pada pohon Huffman dinyatakan dalam `Thuffinfo`. `Thuffinfo` mempunyai elemen sebuah data berupa record (rekaman) yang berisi:

`Kiri` : bertipe pointer yang menyatakan cabang kiri dari sebuah pohon Huffman.

`Kanan` : bertipe pointer yang menyatakan cabang kanan dari sebuah pohon Huffman.

`Piksel` : bertipe array yang menyatakan nilai piksel dari citra yang akan dikompresi.

`jml_kode`: bertipe integer yang menyatakan jumlah kode dalam akar (root) sebuah pohon huffman.

`frek` : bertipe integer yang menyatakan frekuensi kemunculan nilai piksel.

`sudah`: bertipe boolean yang menyatakan apakah nilai piksel yang mempunyai frek > 0 sudah dimasukkan dalam pohon Huffman.

2. Deklarasi pointer untuk kode Huffman

```
type
PhuffCode=^THuffCode;
 THuffcode=record
  Ada:boolean;
  kode:array[0..255] of byte;
  panjangKode:integer;
 end;
```

Pada deklarasi diatas `PhuffCode` menyatakan data yang bertipe pointer. `THuffCode` merupakan nilai data yang ditunjuk oleh pointer `PhuffCode`. Simpul-simpul kode yang akan dibentuk pada kode

Huffman dinyatakan dalam `THuffCode`. `THuffCode` mempunyai elemen sebuah data berupa rekaman (*record*) yang berisi:

`Ada`: bertipe boolean yang menyatakan bahwa apakah nilai piksel dari citra sudah dikodekan

`kode`: bertipe array yang menyatakan nilai piksel dari kode yang sudah dibentuk

`panjang Kode`: bertipe integer yang menyatakan panjang dari nilai piksel yang sudah dikodekan

4.1.3 Teknik Kompresi

Prinsip dari metode Huffman pada citra digital adalah mengkodekan setiap nilai keabuan dengan rangkaian bit 0 dan 1, di mana simbol yang memiliki frekuensi lebih sedikit memiliki rangkaian bit yang lebih panjang daripada simbol yang memiliki frekuensi banyak dalam data. Kode yang dihasilkan adalah kode awalan, yaitu himpunan kode sedemikian sehingga tidak ada anggota kumpulan yang merupakan awalan dari anggota yang lain. Misalnya, himpunan A{00,010,0111} adalah kode awalan, namun himpunan B{00,010,001} bukan kode awalan, karena '00' adalah prefiks dari '001'.

Langkah-langkah yang dilakukan pada proses kompresi adalah sebagai berikut:

1. Menghitung frekuensi kemunculan dari setiap nilai keabuan (piksel) citra yang akan dikompres

   Mula-mula periksa apakah format citra pada Picture adalah keabuan (pf8bit). Jika benar, maka sebelum mencacah banyaknya piksel untuk setiap kebuan dilakukan inisialisasi terhadap variabel `charlist.frek`. Kemudian untuk setiap baris sampai setinggi citra dan setiap kolom sampai selebar citra, naikkan 1 nilai `charlist.frek` yang sesuai dengan keabuan piksel yang sedang ditinjau. Berikut ini listing dari menghitung jumlah frekuensi tiap nilai keabuan:

```
if( fcitra.Image1.Picture.bitmap.PixelFormat= pf8Bit)
then begin
    for i := 0 to 255 do
      begin
        charlist[i].frek:= 0;
      end;
    x:=0;
    for i := 0 to fcitra.Image1.Picture.Height-1 do
      begin
        PC := fcitra.Image1.Picture.Bitmap.ScanLine[i];
        for j := 0 to fcitra.Image1.Picture.Width-1 do
          begin
            inBuffer[x]:=PC[j];
            inc(charlist[inBuffer[x]].frek);
            inc(x);
          end;
      end;
```

2. Pembentukan Pohon Huffman

Kode Huffman pada dasarnya merupakan kode prefiks (*prefix code*). Kode prefiks adalah himpunan yang berisi sekumpulan kode biner, dimana pada kode prefik ini tidak ada kode biner yang menjadi awal bagi kode biner yang lain. Kode prefiks biasanya direpresentasikan sebagai pohon biner yang diberikan nilai atau label. Untuk cabang kiri pada pohon biner diberi label 0, sedangkan pada cabang kanan pada pohon biner diberi label 1. Rangkaian bit yang terbentuk pada setiap lintasan dari akar ke daun merupakan kode prefiks untuk karakter yang berpadanan. Pohon biner ini biasa disebut pohon Huffman.

Langkah-Langkah pembentukan pohon Huffman adalah sebagai berikut:

1. Pada pembentukan pohon Huffman digunakan dua pointer yaitu kanan (sebagai penunjuk cabang kanan dari pohon Huffman) dan kiri (sebagai penunjuk cabang kiri dari pohon huffman). Selain itu ada juga dua variabel bertipe pointer yaitu `pinfo1` dan `pinfo2` yang digunakan masing-masing sebagai simpul dari setiap cabang.
2. Pembentukan pohon Huffman dilakukan dengan membuat simpul (`charlist`) yang mempunyai nilai data yaitu nilai piksel dan frekuensi kemunculannya (`charlist.frek`).
3. Dicari simpul yang memiliki frekuensi terkecil pertama, kemudian disimpan dalam `pinfo1.frek`. Selanjutnya mencari simpul yang

memiliki frekuensi terkecil kedua dan disimpan dalam pinfo2.frek.

4. Dari dua simpul dengan frekuensi terkecil tadi dijumlahkan frekuensinya.
5. Hasil penjumlahan kedua frekuensi ini menjadi frekuensi baru untuk sebuah simpul baru yaitu charlist.frek yang mempunyai dua cabang kedua simpul tadi (pinfo1 dan pinfo2). Untuk pinfo1 ditetapkan sebagai cabang kiri dan pinfo2 ditetapkan sebagai cabang kanan. Simpul baru tersebut juga memiliki data berupa jumlah dari nilai piksel yaitu charlist.jml_kode.
6. Simpul baru ini selanjutnya dimasukkan dalam daftar untuk dilakukan pencarian frekuensi terkecil, dan pembuatan simpul baru.
7. Proses ini berlanjut sampai semua simpul telah masuk ke dalam pohon dan frekuensi akar (*root*) dari pohon akan merupakan hasil penjumlahan semua frekuensi yang ada.. Berikut ini listing dari pembentukan pohon Huffman.

```
procedure pohonHuffman;
var
 i,cnt,tmp:integer;
 pinfo1,pinfo2:PhuffInfo;
begin
 pinfo1:=nil;
 pinfo2:=nil;
 cnt:=255;
 while true do
  begin
   tmp:=maxint;
   for i:=0 to cnt do
    begin
    if (charlist[i].frek<tmp) and (charlist[i].
frek > 0) and(charlist[i].ticked=false) then
      begin
       pinfo1: @charlist[i];
       tmp: pinfo1.frek;
      end;
    end;
   if pinfo1=nil then
    break;
   pinfo1.ticked:=true;
   tmp:=maxint;
   for i:=0 to cnt do
    begin
     if (charlist[i].frek<tmp) and (charlist[i].
frek > 0) and (charlist[i].ticked=false) then
      begin
       pinfo2: @charlist[i];
       tmp:=pinfo2.frek;
```

```
        end;
       end;
      if pinfo2=nil then
       break;
      pinfo2.ticked:=true;
      inc(cnt);
      charlist[cnt].frek:=pinfo1.frek+pinfo2.frek;
    charlist[cnt].jml_kode:=pinfo1.jml_kode+pinfo2.
    jml_kode;
      charlist[cnt].kiri:=pinfo1;
      charlist[cnt].kanan:=pinfo2;
      pinfo1:=nil;
      pinfo2:=nil;
     end;
    rootnode:=@charlist[cnt];
   end;
```

3. Pengkodean

Pemberian kode biner pada pohon Huffman dilakukan dengan memberikan kode pada tiap cabang pada pohon Huffman. Berikut ini langkah-langkah dalam pemberian kode pada pohon Huffman:

1. Ambil nilai piksel yang akan dikodekan
2. Dari setiap nilai keabuan yang akan dikodekan telusuri mulai dari akar. Apabila pada cabang kiri terdapat nilai piksel yang sedang ditinjau, maka berikan kode "0". Dan apabila terdapat pada cabang kanan maka berikan simbol "1".
3. Setiap cabang yang telah ditelusuri, tetapkan sebagai akar.
4. Ulangi langkah diatas sampai ditemukan daun. Beriukut ini listing dari pemberian kode Huffman.

```
procedure kodeHuffman;
 var
  i,j:integer;
  tmpnode:phuffinfo;
  flag:integer;
 begin
  for i:=0 to 255 do
   begin
    flag:=-1;
    tmpnode:=rootnode;
    while tmpnode.kiri<>nil do
     begin
      for j:=0 to tmpnode.kiri.jml_kode-1 do
       if tmpnode.kiri.piksel[j]= i then
        begin
         flag:=0;
         tmpnode:=tmpnode.kiri;
         break;
        end;
       if flag=-1 then
```

```
        begin
         for j:=0 to tmpnode.kanan.jml_kode-1 do
          if tmpnode.kanan.piksel[j]= i then
           begin
            flag:=1;
            tmpnode:=tmpnode.kanan;
            break;
           end;
         end;
       if flag=-1 then
        break;
       huffcodes[i].ada:=true;

        huffcodes[i].kode[huffcodes[i].panjangKode]:=flag;

huffcodes[i].panjangKode:= huffcodes[i].panjangKode+1;
       huffcodes[i].char:=i;
       flag:=-1;
      end;
   end;
end;
```

4. Pembuatan tabel Huffman

Tabel huffman digunakan untuk menyimpan data-data hasil dari pengkodean huffman. Data-data disini disimpan dalam variabel array. Deklarasi dari variabel array tersebut adalah table: array[0..1495] of byte; Langkah-langkah dalam pembuatan tabel huffman adalah sebagi berikut:

1. Ambil satu nilai keabuan yang akan disimpan.
2. Indeks pertama diisi untuk menyimpan nilai keabuan. Misalkan table[0]:=25.
3. Indeks kedua diisi untuk menyimpan hasil panjang kode huffman dari nilai keabuan yang sedang ditinjau.
4. Indeks ketiga diisi untuk menyimpan hasil kode huffman. Karena hasil dari panjang kode huffman bervariasi maka semua kode disimpan dalam 8 bit. Berikut ini listing dari pembentukan tabel huffman:

```
procedure TabelHuffman;
var
 i,j,k:integer;
 tmpcode:byte;
 bit:integer;
begin
k:=0;
 bit:=0;
 tmpcode:=0;
 for i:=0 to 255 do
  begin
   if huffcodes[i].ada=false then
```

```
        continue;
       table[k]:=i;
       table[k+1]: huffcodes[i].panjangKode;
       k:=k+2;
       for j:=0 to huffcodes[i].panjangKode-1 do
        begin
         tmpcode:=tmpcode shl 1 or
    huffcodes[i].kode[j];
         bit:=bit+1;
         if bit=8 then
          begin
           table[k]:=tmpcode;
           k:=k+1;
           bit:=0;
           tmpcode:=0;
          end;
        end;
       if bit>0 then
        begin
         tmpcode: tmpcode shl (8-bit);
         table[k]:=tmpcode;
         k:=k+1;
         bit:=0;
         tmpcode:=0;
        end;
      end;
     Jumtabel:=k;
    end;
```

Untuk setiap nilai keabuan (i) dari 0 sampai 255, diperiksa nilai keabuan yang mempunyai kode huffman. Apabilai nilai keabuan tersebut memiliki kode huffman maka variabel tabel disisi:

`k:=0;`

`table[k]:=i` (tabel ke 0 diisi nilai kebauan)

`table[k+1]:= huffcodes[i].panjangKode` (tabel selanjutnya diisi panjang kode huffman nilai keabuan tersebut).

Untuk jumlah bit dari 0 sampai panjang kode huffman disimpan sebesar 8 bit dalam `tabel[k+2]`. Yaitu dengan cara menggeser 1 bit ke kiri sampai sepanjang 8 bit. Tabel berikutnya diisi dengan data-data dari nilai keabuan selanjutnya dengan proses yang sama. Tabel sejumlah k akan disimpan pada prosedur selanjutnya yaitu penyimpanan hasil kompresi.

5. Penyimpanan hasil kompresi

Kode hasil proses huffman harus disimpan ke dalam memori komputer . Berikut ini listing dari penyimpanan hasil kompresi.

```
procedure SimpanFile;
var
x,y, i,j,k:integer;
tmpcode:byte;
bit:integer;
bufcount:integer;
PC:PByteArray;
begin
k:=0;
tmpcode:=0;
bit:=0;
h:= fcitra.Image1.Picture.Height;
w:=fcitra.Image1.Picture.Width;
ofile.seek(1,sefrombeginning);
ofile.write(Jumtabel,sizeof(Jumtabel));
ofile.write(table,Jumtabel);
ofile.write(h,sizeof(h));
ofile.write(hh,h);
ofile.write(w,sizeof(w));
ofile.write(ww,w);
ifile.seek(0,soFromBeginning);
for i:=0 to byk do
begin
for j:=0 to huffcodes[inbuffer[i]].panjangKode-1
 do
begin
tmpcode:=(tmpcode shl 1) or
huffcodes[inbuffer[i]].kode[j];
inc(bit);
if bit=8 then
begin
outbuffer[k]:=tmpcode;
inc(k);
bit:=0;
tmpcode:=0;
end;
if k=byk then
begin
ofile.write(outbuffer,byk);
k:=0;
end;
end;
end;
if bit>0 then
begin
tmpcode:=tmpcode shl (8-bit);
outbuffer[k]:=tmpcode;
k:=k+1;
end;

if k>0 then
ofile.write(outbuffer,k);
ofile.seek(0,sofrombeginning);
ofile.write(bit,1);
end;
```

Pada proses penyimpanan, tinggi dan lebar citra yang dikompres juga ikut disimpan ditambah dengan jumlah tabel pada prosedur tabel Huffman. Untuk tinggi citra disimpan dalam variabel *h*, dan lebar

citra disimpan dengan variabel *w*. Banyaknya titik piksel dari citra yang dikompres disimpan dengan variabel *byk*. Dari nilai piksel sejumlah *byk* disimpan masing-masing kode huffman sejumlah 8 bit dalam variabel *outbuffer*.

6. Perhitungan Hasil Kompresi

   Untuk mengetahui tingkat efisiensi ukuran file hasil kompresi diketahui dari besarnya nilai rasio kompresi. Semakin besar rasio kompresi maka semakin besar tingkat efisiensi ukuran file hasil kompresi. Rasio kompresi citra dihitung dengan rumus:

$$\text{Rasio kompresi} = 100\% - \left(\frac{\text{ukuran file hasil kompresi}}{\text{ukuran citra semula}} \right) \times 100\%$$

4.1.4 Dekompresi

Langkah- langkah yang dilakukan pada proses dekompresi adalah sebagai berikut:

1. Mengembalikan tabel Huffman
2. Mengembalikan pohon Huffman

   Langkah-langkah yang dilakukan untuk mengembalikan pohon Huffman adalah sebagi beriku:

   1. Baca sebuah nilai kebauan
   2. Mulai dari akar untuk setiap bit pada langkah 1, lakukan traversal pada cabang yang bersesuaian.
   4. Ulangi langkah 1, 2 dan 3 sampai bertemu daun. Kodekan rangkaian bit yang telah dibaca dengan karakter di daun.
   5. Ulangi dari langkah 1 sampai semua nilai keabuan habis.
3. Merubah file kompresi dalam bentuk citra

   Setelah kode Huffman dapat dikembalikan ke nilai semula (nilai piksel citra masukan) maka dari nilai-nilai itu diubah kedalam bentuk matrik untuk dilakukan penggambaran per titik piksel pada *image canvas*. Berikut ini listing dari penggambaran pada *image canvas*

```
z:=0;
    form1.image1.ClientHeight:=m;
    form1.image1.ClientWidth:=n;
    for i:= 0 to m-1  do
```

```
            for j:=0 to n-1   do
              begin
                 tmp:=OutBuffer[z];
form1.image1.Canvas.Pixels[j,i]:=RGB(tmp,tmp,tmp);
                 z:=z+1;
              end;
```

File hasil kompresi menyimpan tinggi dan lebar citra yang dikompres. Dari tinggi dan lebar citra masukan tersebut digunakan sebagai tinggi dan lebar citra hasil. Kemudian dilakukan dengan pengubahan citra ke *gray scale* 8 bit. Sehingga akan dihasilkan citra yang sama seperti citra sebelum dikompres.

4.1.5 Contoh kasus

Misalkan sebuah citra *gray scale* yang berukuran 155x153 dengan 8 derajat keabuan seperti dalam gambar 4.1 di bawah ini. Maka ukuran citra tersebut adalah 155 x 153 x 8 = 189720 bit.

Gambar 4.1 Contoh Citra

Dalam gambar 4.1 diatas adalah citra yang akan dikompresi dengan metode Huffman. Berikut ini adalah langkah-langkah untuk melakukan kompresi terhadap citra diatas dengan metode Huffman:

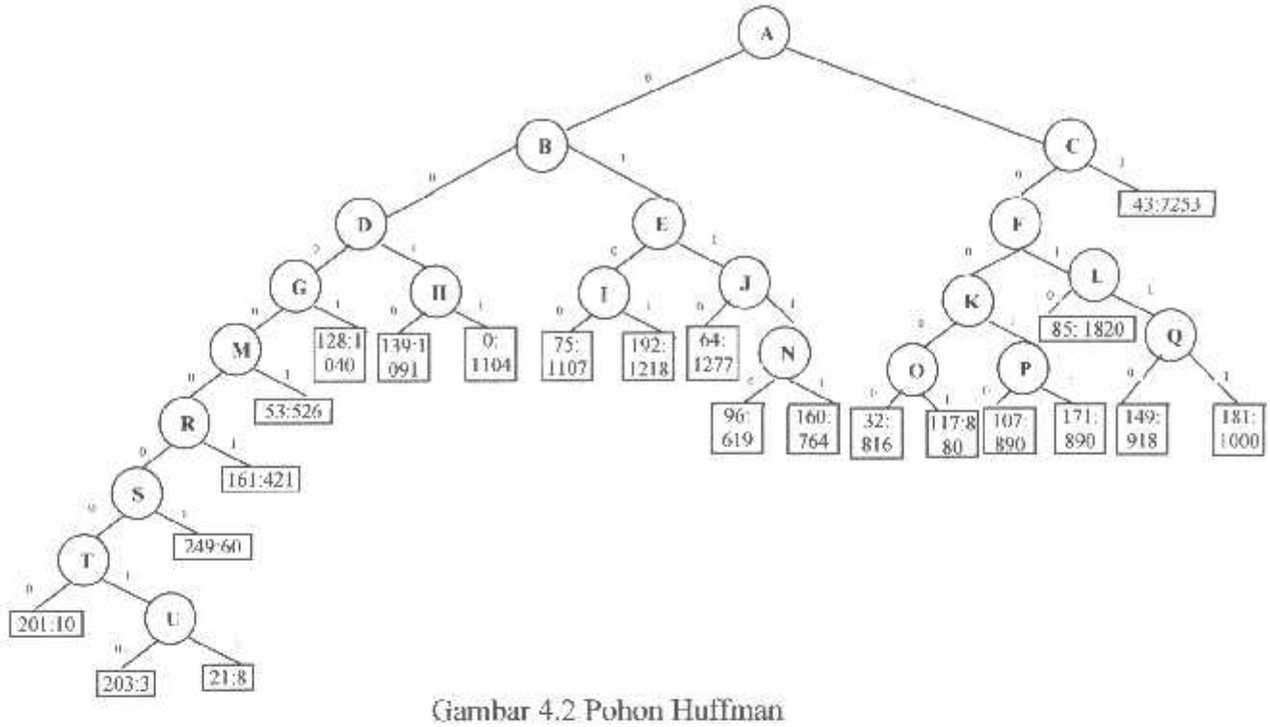
1. Hitung frekuensi dari tiap nilai keabuan untuk setiap baris sampai setinggi citra dan setiap kolom sampai setinggi citra. Sehingga akan diperoleh frekuensi setiap nilai keabuan (piksel) seperti dalam tabel 4.1 dibawah ini.

Tabel 4.1 Tabel Frekuensi Contoh Citra

Keabuan	Frekuensi
0	1104
21	8
32	816
43	7253
53	526

64	1277
75	1107
85	1820
96	619
107	890
117	880
128	1040
139	1091
149	918
160	764
161	421
171	890
181	1000
192	1218
201	10
203	3
249	60

2. Setiap nilai piksel merupakan sebuah simpul yang mempunyai data nilai keabuan dan jumlah frekuensi.
3. Dari simpul-simpul tersebut, cari simpul yang memiliki jumlah frekuensi terkecil pertama. Kemudian cari simpul yang mempunyai jumlah frekuensi terkecil kedua. Dari kedua simpul tadi jumlahkan frekuensinya. Hasil penjumlahan dari kedua simpul tadi menjadi sebuah simpul baru yang memiliki dua cabang kedua simpul tadi. Simpul yang memiliki jumlah frekuensi terkecil pertama menjadi cabang kiri dan simpul yang memiliki jumlah frekuensi terkecil kedua menjadi cabang kanan.
4. Simpul baru ini selanjutnya dimasukkan daftar dalam beserta kedua cabang untuk dilakukan pencarian frekuensi terkecil, dan pembuatan simpul baru.
5. Proses ini berlanjut sampai semua simpul telah masuk ke dalam pohon dan frekuensi akar (*root*) dari pohon akan merupakan hasil penjumlahan semua frekuensi yang ada.
6. Setelah terbentuk pohon Huffman dilakukan penelusuran terhadap pohon Huffman dengan memberikan kode "0" untuk setiap cabaang kiri dan "1" untuk setiap cabang kanan.
7. Pohon Huffman yang terbentuk ditunjukkan dalam gambar 4.2.



Gambar 4.2 Pohon Huffman

Tabel berikut ini berisi keterangan pada setiap akar dan cabang pada pohon Huffman diatas.

Tabel 4.2 Tabel Keterangan pohon Huffman

Huruf	Simpul	Jumlah Frekuensi
A	201 203 21 249 161 53 128 139 0 75 192 64 96 160 32 117 107 171 85 149 181 43	23715
B	201 203 21 249 161 53 128 139 0 75 192 64 96 160	9248
C	32 117 107 171 85 149 181 43	14467
D	201 203 21 249 161 53 128 139 0	4263
E	75 192 64 96 160	4985
F	32 117 107 171 85 149 181	7214
G	201 203 21 249 161 53 128	2068
H	139 0	2195
I	75 192	2325
J	64 96 160	2660
K	32 117 107 171	3476
L	85 149 181	3738
M	201 203 21 249 161 53	1028
N	96 160	1383
O	32 117	1696
P	107 171	1780
Q	149 181	1918
R	201 203 21 249 161	502

S	201 203 21 249	81
T	201 203 21	21
U	203 21	11

8. Kemudian dilakukan penelusuran pada pohon Huffman. Dimulai dari akar, penelusuran menurun didapatkan simpul tanpa cabang (daun).
9. Begitu didapatkan daun, maka nilai piksel dimasukkan dalam tabel beserta angka-angka bit ("0" atau "1") dari cabang-cabang yang dilalui untuk sampai pada daun ini.
10. Maka diperoleh kode Huffman dari masing-masing nilai kebuan seperti dalam tabel 4.3 di bawah ini:

Tabel 4.3 Tabel Kode Huffman

Keabuan	Kode Huffman	Panjang kode
0	0011	4
21	000000011	9
32	10000	5
43	11	2
53	00001	5
64	0110	4
75	0100	4
85	1010	4
96	01110	5
107	10010	5
117	10001	5
128	0001	4
139	0010	4
149	10110	5
160	01111	5
161	000001	6
171	10011	5
181	10111	5
192	0101	4
201	00000000	8
203	000000010	9
249	0000001	7

Dari tabel kode Huffman diatas dapat dilihat bahwa setiap nilai kebauan yang memiliki frekuensi dalam jumlah besar akan dikodekan dengan jumlah bit yang lebih sedikit. Sedangkan nilai keabuan yang memiliki jumlah frekuensi sedikit akan dikodekan dengan jumlah bit yang lebih panjang.

Kode Huffman dari masing-masing nilai keabuan tersebut disimpan dalam memori komputer. Sebagai contoh dari tabel Huffman diatas akan menyimpan nilai keabuan 0 yang memiliki kode huffman 0011. Maka proses penyimpanannya adalah sebagai berikut:

```
Tmpcode:=0
        k:=0;
        table[k]:=0   (tabel ke 0 diisi nilai kebauan 0)
table[k+1]:= 4    (tabel ke 1 disii panjang kode huffman)
```

untuk menyimpan 8 bit dilakukan proses sebagai berikut

```
tmpcode:=tmpcode shl 1 or huffcodes[i].kode
bit:=bit+
if bit=8 then
table[k+2]:=tmpcode;
if bit>0 then
begin
 tmpcode:=tmpcode shl (8-bit);
 table[k]: tmpcode;
end
```

11. Simpan kode-kode Huffman tersebut harus ke dalam memori komputer untuk mempermudah dalam proses dekompresi.
12. Lakukan proses perhitungan hasil kompresi dengan cara mengalikan panjang kode dengan jumlah frekuensi setiap nilai keabuan. Kemudian jumlahkan semua hasil perkalian tersebut. Dalam tabel 4.4 berikut ini adalah tabel hasil perhitungannya:

Tabel 4.4 Tabel Perhitungan Kompresi

Keabuan	Frekuensi	Panjang kode	Frekuensi * panjang kode
0	1104	4	4416
21	8	9	72
32	816	5	4080
43	7253	2	14506
53	526	5	2630
64	1277	4	5108
75	1107	4	4428
85	1820	4	7280
96	619	5	3095
107	890	5	4450
117	880	5	4400
128	1040	4	4160
139	1091	4	4364
149	918	5	4590

160	764	5	3820
161	421	6	2526
171	890	5	4450
181	1000	5	5000
192	1218	4	4872
201	10	8	80
203	3	9	27
249	60	7	420
		jumlah	88774

Maka ukuran citra setelah pemampatan adalah 88774 bit. Jadi kebutuhan memori telah dapat dikurangi dari 189720 bit menjadi 88774 bit.

Sehingga Rasio Kompresi adalah (100 % - $\frac{88774}{189720}$ X 100%) = 53.21 %)

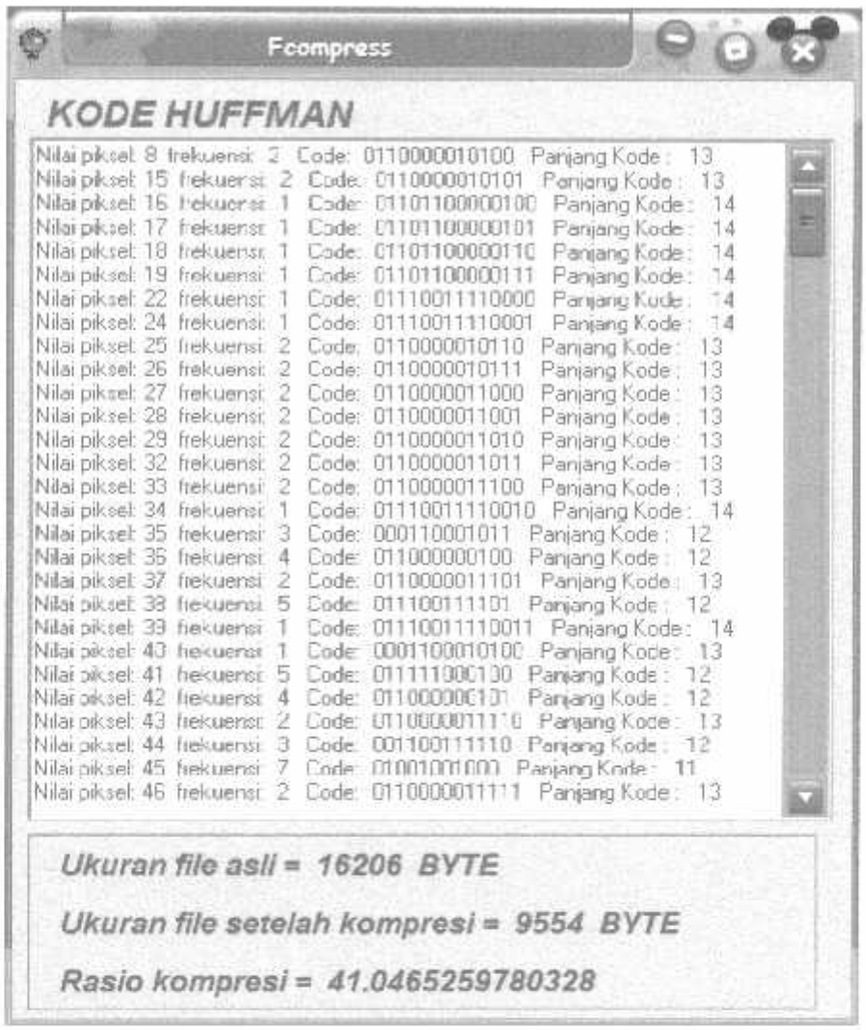
4.1.6 Pengujian

Untuk uji coba ini digunakan empat buah citra grayscale dalam format bmp.

a. Citra A dengan dimensi 122 x 124 dengan ukuran 16206 byte.

Gambar 4.3 Citra A

Dalam Gambar 4.4 di bawah ini menunjukkan nilai piksel dari citra, jumlah frekuensi tiap piksel, kode Huffman tiap nilai piksel serta panjang dari kode Huffman. Dari gambar terebut dapat dilihat bahwa frekuensi terbesar berada pada nilai piksel 255 dengan jumlah frekuensi 7694, sehingga citra tersebut cenderung ke warna putih Berdasarkan prinsip Huffman bahwa setiap nilai piksel yang mempunyai frekuensi terbesar akan dikodekan dengan jumlah bit yang pendek. Maka nilai piksel 255 dikodekan dengan panjang 1 bit. Hal ini dapat memperkecil ukturan file yang cukup berarti. File hasil kompresi sebesar 9554 dengan rasio kompresi sebesar 41,046.



Gambar 4.4 Kode Huffman Citra A

b. Citra B dimensi 256 x 256 dengan ukuran 66614 byte

Gambar 4.5 Citra B

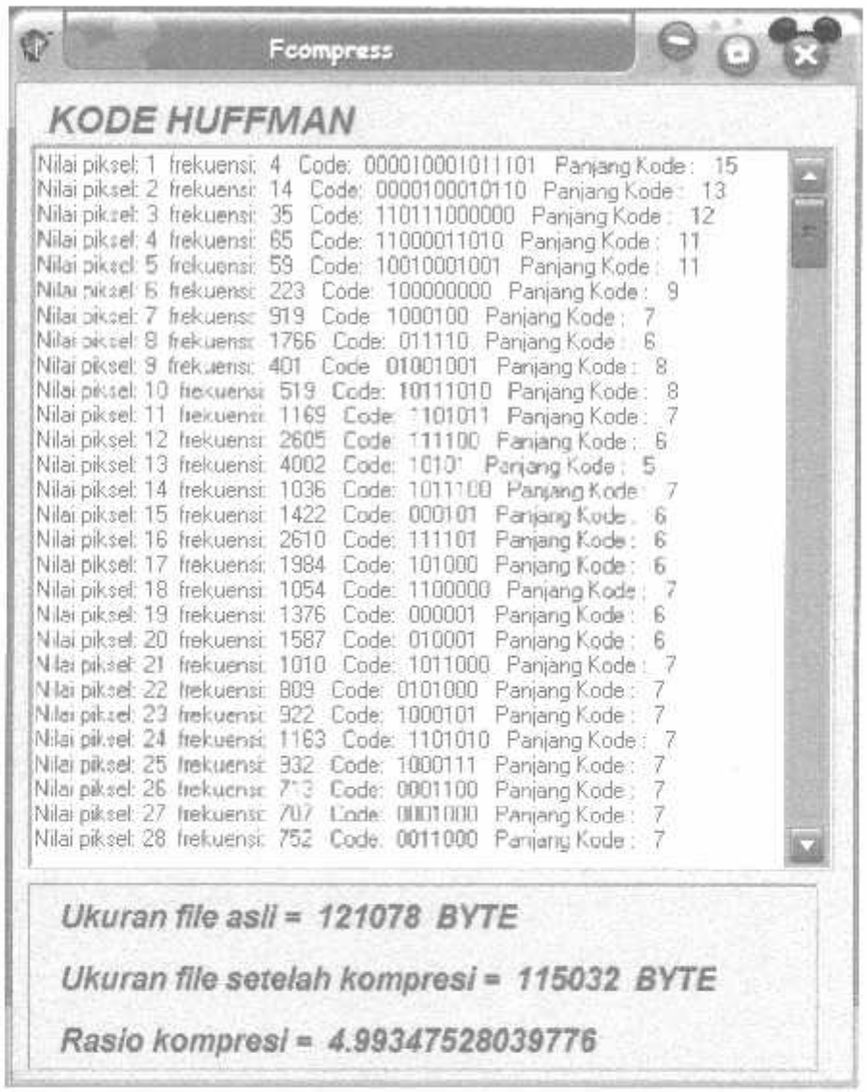
Tampilan dalam gambar 4.6 di bawah ini menunjukkan nilai piksel dari citra, jumlah frekuensi tiap piksel, kode Huffman tiap nilai piksel serta panjang dari kode Huffman. Dari gambar tersebut dapat dilihat bahwa jumlah frekuensi antara satu piksel dengan piksel yang lain tidak cukup besar. Panjang kode Huffman yang dihasilkan rata-rata hampir sama. Sehingga file hasil kompresi hanya menghasilkan rasio kompresi sebesar 11,239 byte.

Gambar 4.6 Kode Huffman Citra B

c. Citra C dimensi 300 x 400 dengan ukuran 121078 byte

Gambar 4.7 Citra C

Tampilan dalam gambar 4.8 di bawah ini menunjukkan nilai piksel dari citra, jumlah frekuensi tiap piksel, kode Huffman tiap nilai piksel serta panjang dari kode Huffman. Dalam gambar 4.8 tersebut dapat dilihat bahwa jumlah frekuensi antara satu piksel dengan piksel yang lain tidak cukup besar. Persebaran warna antara hitam dan putih lebih merata. Panjang kode Huffman yang dihasilkan rata-rata hampir sama. Sehingga file hasil kompresi hanya menghasilkan rasio kompresi sebesar 4.993.



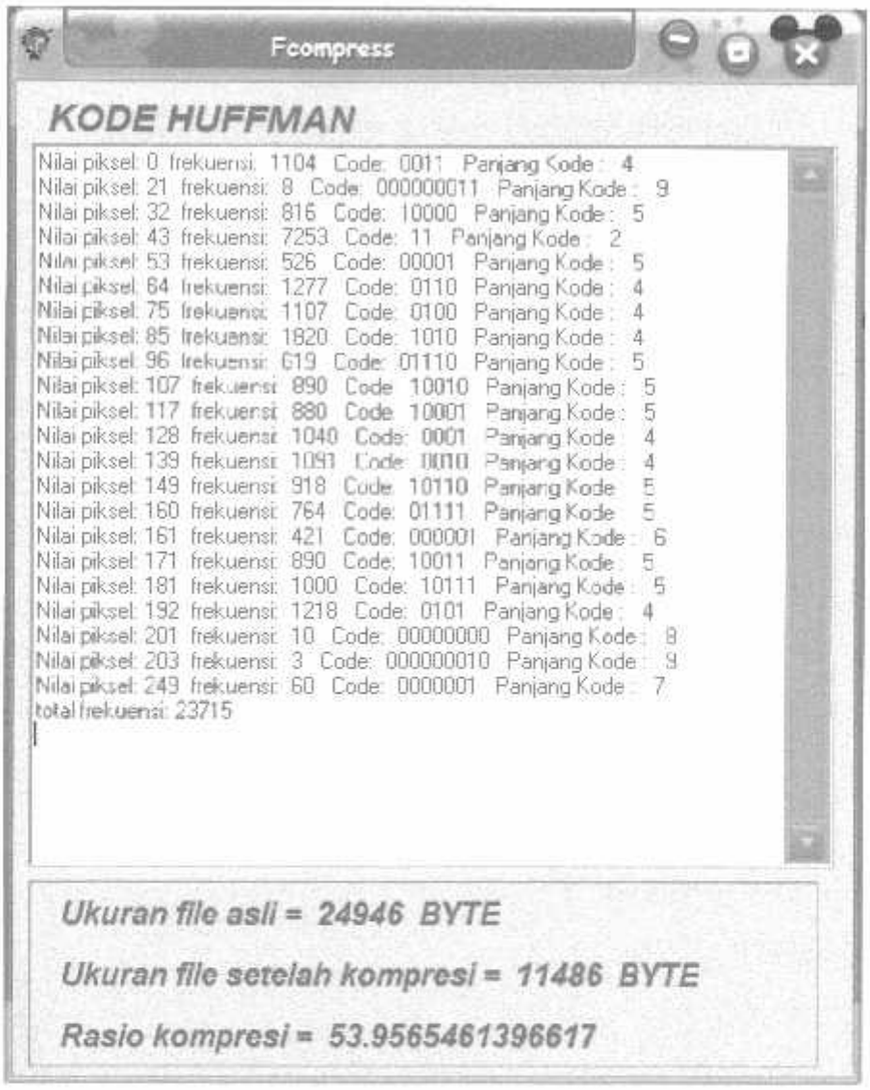
Gambar 4.8 Kode Huffman Citra C

d. Citra D dimensi 155 x 153 dengan ukuran 24946 byte

Gambar 4.9 Citra D

Tampilan dalam gambar 4.10 di bawah ini menunjukkan nilai piksel dari citra, jumlah frekuensi tiap piksel, kode Huffman tiap nilai piksel serta

panjang dari kode Huffman. Dalam gambar 4.10 tersebut dapat dilihat bahwa frekuensi terbesar berada pada nilai piksel 43 dengan jumlah frekuensi 7253. Jumlah frekuensi cukup berbeda jauh dengan frekuensi nilai piksel lainnya. Sehingga rasio kompresi yang dihasilkan cukup besar yaitu 53,956.

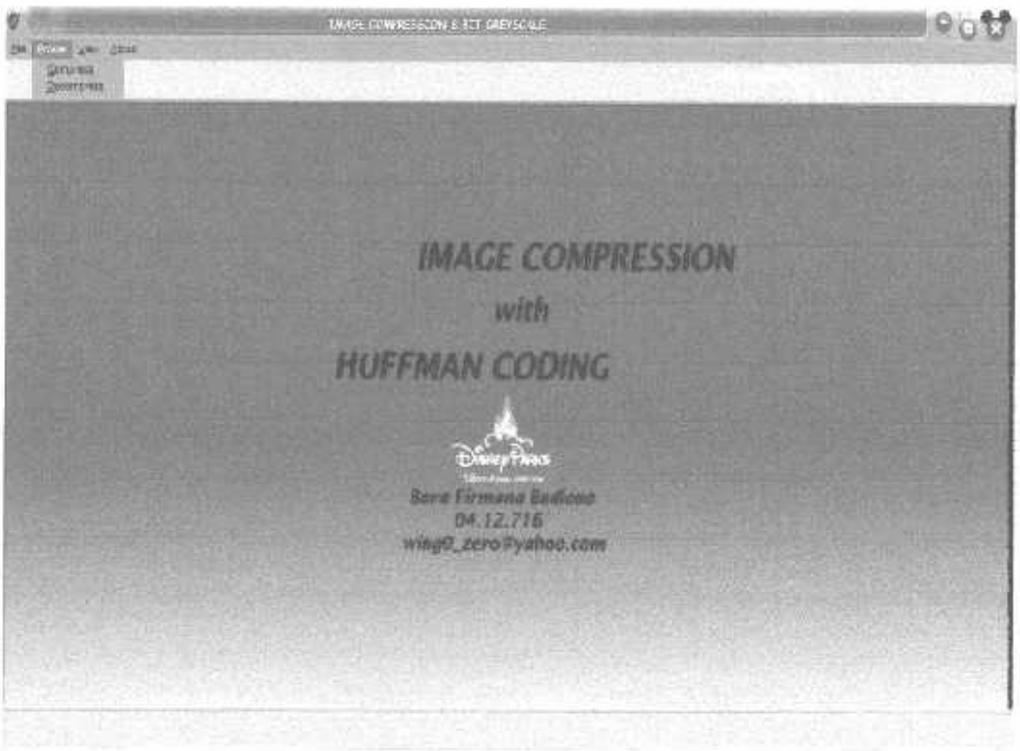


Gambar 4.10 Kode Huffman Citra D

4.2 Implementasi

4.2.1 Tampilan Awal

Implementasi dari program kompresi citra ini menggunakan Borlan Delphi 7. Tampilan awal ketika menjalankan programa adalah seperti dalam gambar 4.11 di bawah ini.



Gambar 4.11 Tampilan Form Kompresi dan Dekompresi

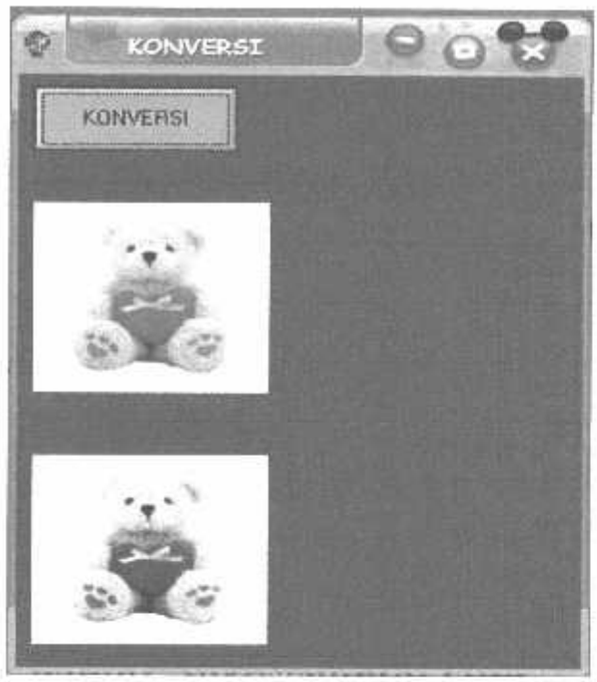
Pada awal tampilan tersebut terdapat menu yaitu:

1. File

   Berisi sub menu yang terdiri dari Open, save dan exit. Sub menu open digunakan untuk membuka citra yang akan dikompresi. Sub menu save digunakan untuk menyimpan kode Huffman. Dan sub menu exit digunakan untuk keluar dari program.

2. Proses

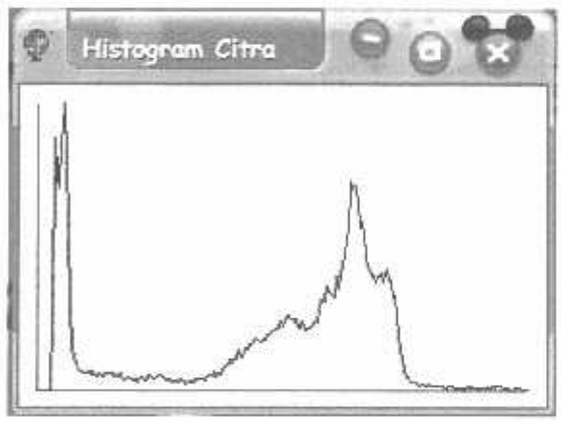
   Di dalam menu proses terdapat sub menu yaitu kompresi dan dekompresi. Sub menu dekompresi digunakan untuk mengkompresi citra masukan. Dan sub menu dekompresi digunakan untuk mendekompresi citra hasil kompresi. Citra yang dapat dikompres hanya citra *gray scale* dengan kedalaman 8 bit. Apabila citra yang dibuka adalah citra *true colour* maka di dalam menu ini terdapat program untuk mengkonversi citra *true colour* (warna) menjadi citra *gray scale*. Citra hasil konversi disimpan, kemudian dapat dibuka lagi untuk dilakukan prose kompresi. Dalam gambar 4.12 berikut ini tampilan form konversi warna ke *gray scale*.



Gambar 4.12 Tampilan Form Konversi

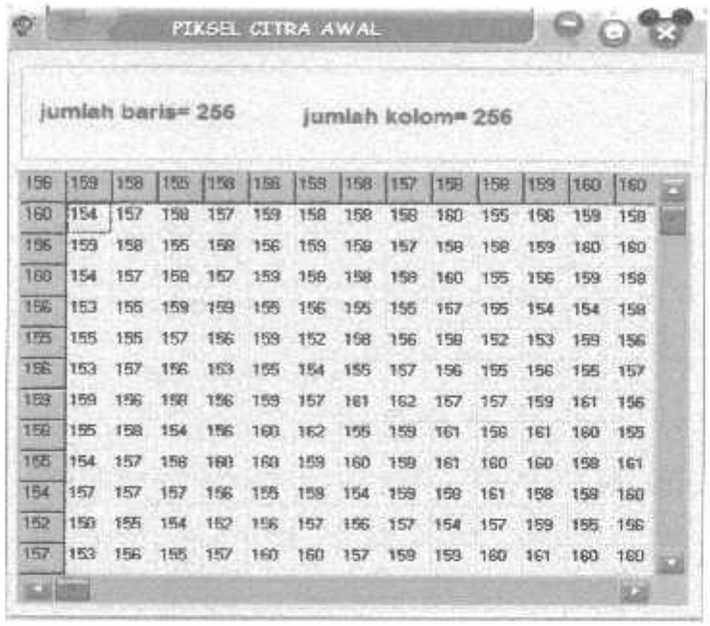
3. View

Menu ini untuk menampilkan histogram dan nilai piksel dari citra masukan. Untuk menampilkan histogram dengan memilih sub menu Histogram. Dalam gambar 4.13 berikut ini contoh tampilan dari histogram citra masukan



Gambar 4.13 Tampilan histogram citra masukan

Dan untuk menampilkan nilai piksel dari citra masukan pilih sub menu nilai piksel citra awal. Dalam gambar 4.14 berikut ini contoh tampilan dari nilai piksel citra masukan



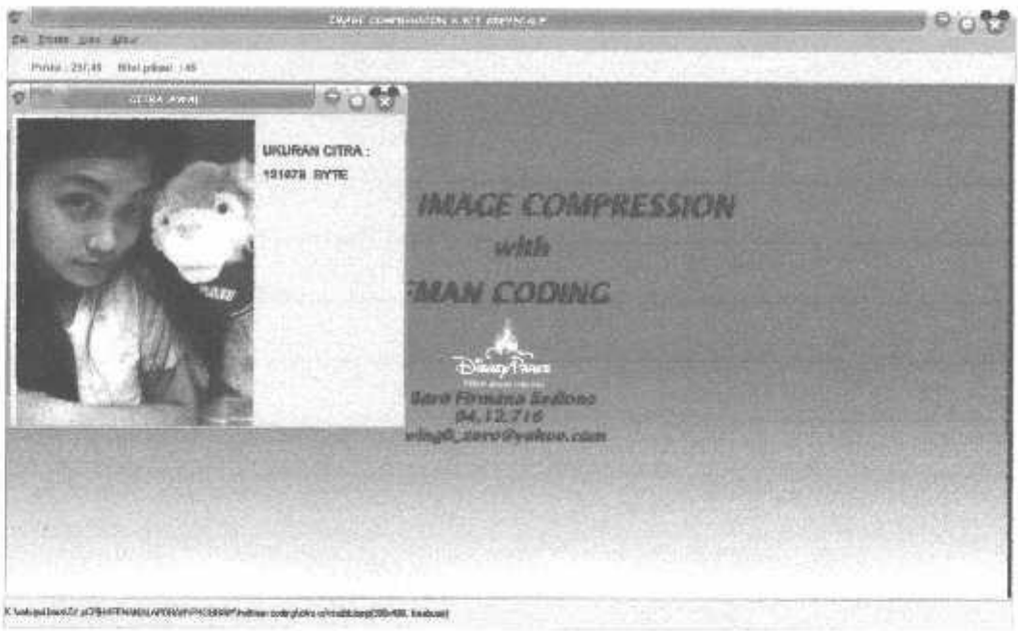
Gambar 4.14 Tampilan nilai piksel citra masukan

4. About

Menu ini berisi tentang pembuat program kompresi citra.

4.2.2 Membuka file citra

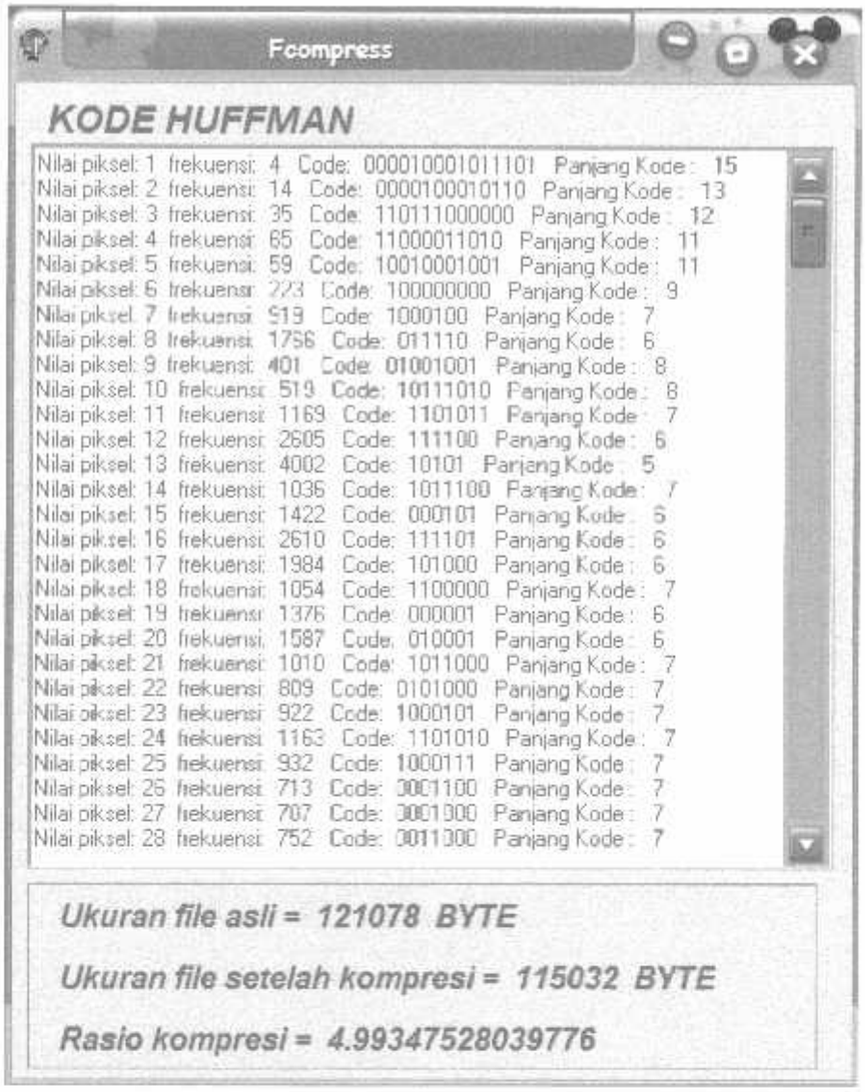
Apabila akan mengkompres suatu citra, panggil citra tersebut dengan mengklik tombol "Open" sehingga akan tampil *open picture dialog*. Format citra yang bisa dipilih hanya format BMP. Setelah memilih citra yang akan dikompres maka citra tersebut akan tampil. Pada label 1 terdapat informasi posisi dan nilai piksel ketika kita meletakkan kursor pada citra. Dan pada bagian bawah terdapat status bar yang berisi informasi ukuran dan jenis citra. Dalam gambar 4.15 berikut ini tampilan ketika citra dipanggil.



Gambar 4.15 Tampilan Citra

4.2.3 Proses Kompresi Citra

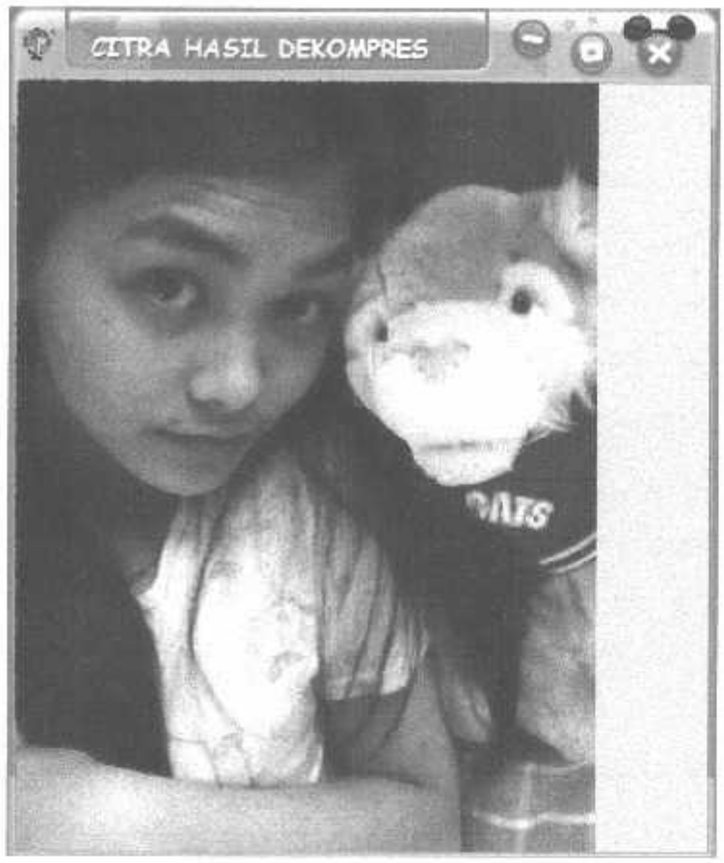
Untuk melakukan proses kompresi pada citra yang sudah ditampilkan dengan mengklik menu proses kemudian sub menu kompresi. Maka akan muncul save dialog yang digunakan untuk menyimpan file hasil kompresi.Apabila citra yang ditampilkan bukan citra 8 bit maka tidak bisa melakukan proses kompres. Setelah itu akan tampil form yang berisi hasil kode Huffman dalam gambar 4.16 dibawah ini.



Gambar 4.16 Tampilan Kode Huffman

4.2.4 Proses Dekompresi Citra

Proses dekompresi dilakukan dengan memilih menu file, kemudian sub menu dekompres. Maka akan ditampilkan *open dialog* yang digunakan untuk membuka file yang akan didekompres. Setelah memilih file yang akan didekompres maka akan muncul form hasil dekompres yang citranya tepat sama dengan file aslinya sebelum dikompres. Untuk menyimpan hasil dekompres dengan cara klik kanan pada citra hasil dekompres dapat ditunjukkan dalam gambar 4.17.



Gambar 4.17 Tampilan citra hasil dekompresi

BAB V

PENGUJIAN METODE HUFFMAN DENGAN METODE LOSSY

5.1 Percobaan

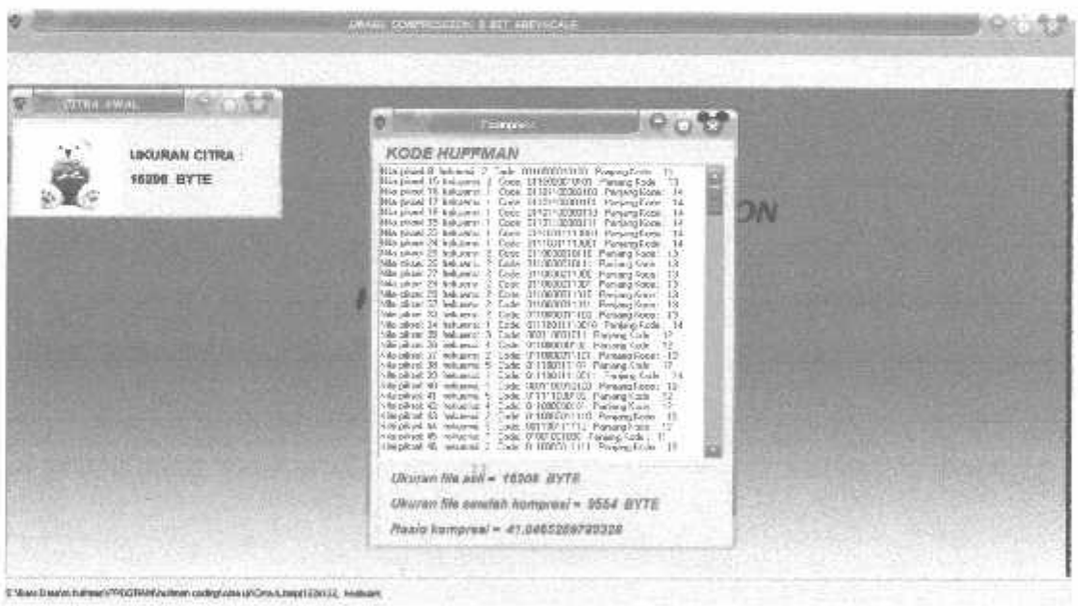
5.1.1 Perbandingan Teknik Kompresi

Untuk mengetahui bagaimana teknik pengkompresian citra berjalan efisien pada suatu citra maka perlu diadakan percobaan perbandingan dan penghitungan antara teknik kompresi citra dengan metode *lossless* menggunakan teknik *Huffman coding* dan Metode *lossy*, rasio kompresi citra dihitung dengan rumus :

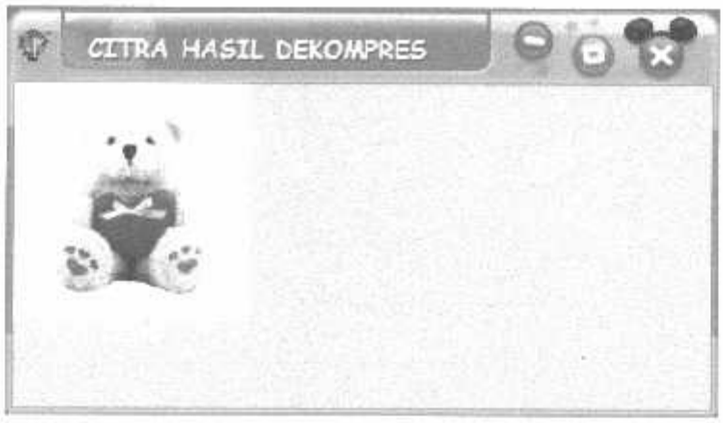
$$\text{Rasio kompresi} = 100\% - \left(\frac{\text{ukuran citra hasil kompresi}}{\text{ukuran citra semula}}\right) \times 100\%$$

Sehingga pada percobaan perbandingan bisa menemukan hasil terbaik dalam teknik pengkompresian citra dan mengetahui kelebihan serta kekurangan pada masing – masing teknik pengkompresian citra baik menggunakan metode *Lossless* dengan teknik *Huffman coding* dan metode *Lossy*.

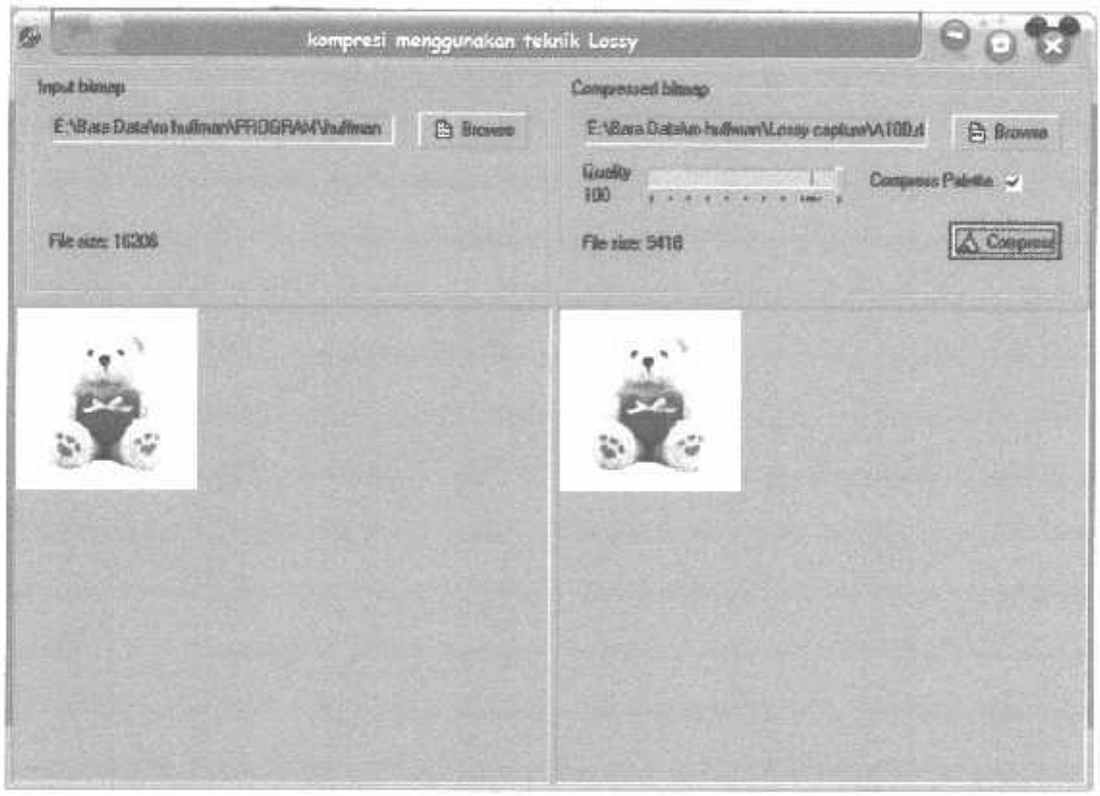
a) Citra A dengan dimensi 122 x 124 dengan ukuran 16206



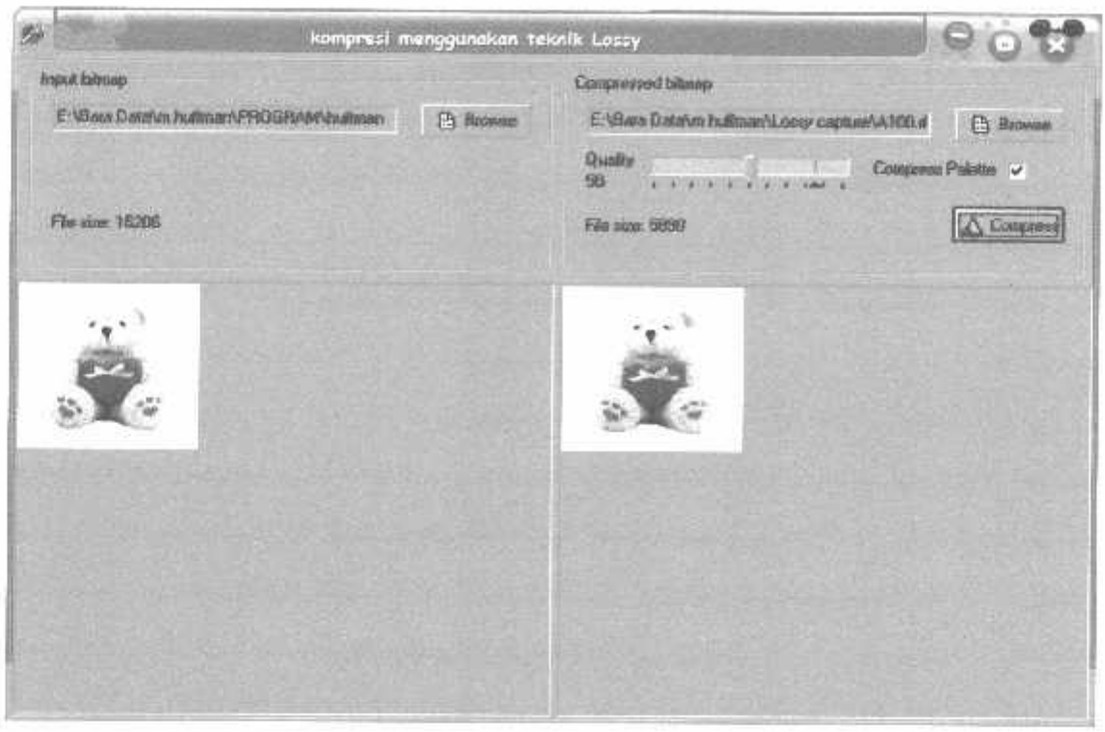
Gambar 5.1 citra A hasil kompresi menggunakan teknik *Huffman*



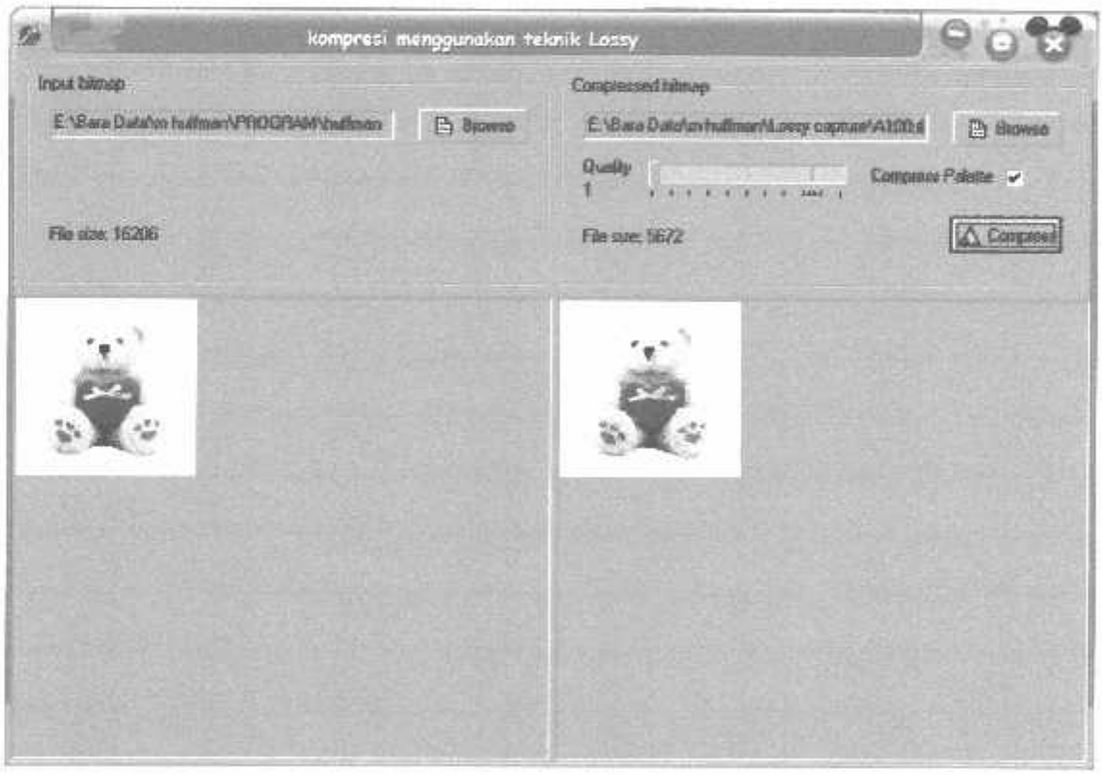
Gambar 5.2 citra A hasil dekompresi menggunakan teknik *Huffman*



Gambar 5.3 citra A menggunakan teknik *lossy* dengan kualitas citra 100%



Gambar 5.4 citra A menggunakan teknik *lossy* dengan kualitas citra 50%



Gambar 5.5 citra A menggunakan teknik *lossy* dengan kualitas citra 1%

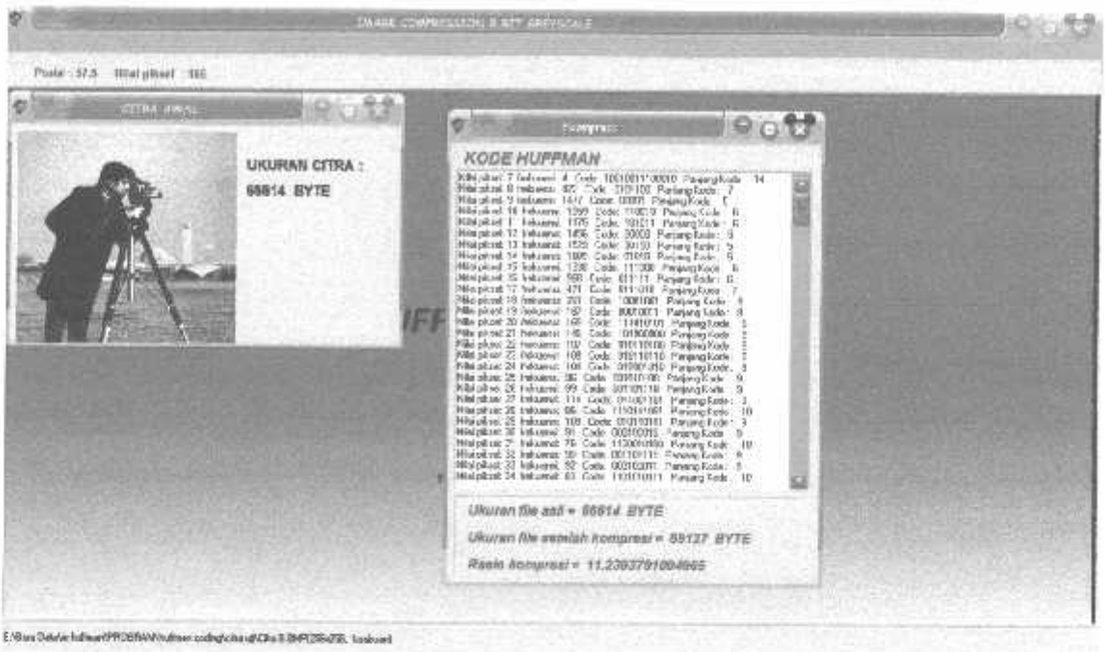
Pada percobaan menggunakan teknik *Huffman* kualitas citra yang didapatkan setelah proses kompresi dan dekompresi adalah citra tidak mengalami kerusakan pada

saat sesudah dikompresi dan dekompresi untuk ukuran hasil pengkomppresian masih terbilang efisien dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil perhitungannya adalah : *"100% - (9554 / 16206) x 100% = 41.046%"*.

Pada percobaan menggunakan teknik *lossy* dapat diukur nilai pada citra seperti pada percobaan diatas :

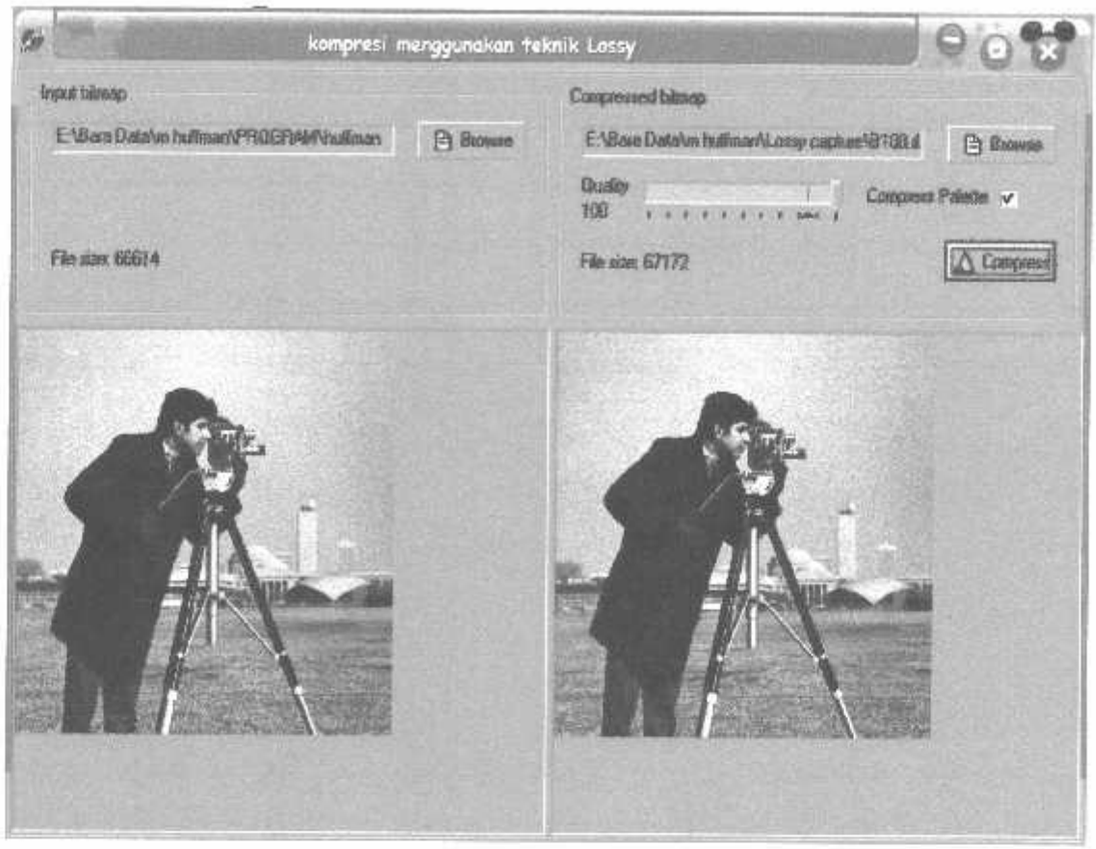
- Apabila citra Λ dikompresi dengan kualitas 100% maka nilai dari kompresi citra akan berukuran kecil dari 16206 menjadi 9418 akan tetapi pada citra tersebut akan mengalami sedikit kerusakan pada kualitasnya, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya *"100% - (9418 / 16206) x 100% = 41.885%"*.
- Apabila citra Λ dikompresi dengan kualitas citra 50% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil dari 16206 menjadi 5890, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya *"100% - (5890 / 16206) x 100% = 63.655%"*.
- Apabila citra Λ dikompresi dengan kualitas 1% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil dari 16206 menjadi 5672, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil perhitungannya *"100% - (5672 / 16206) x 100% = 65.000%"*.

b) Citra B dimensi 256 x 256 dengan ukuran 66614 byte

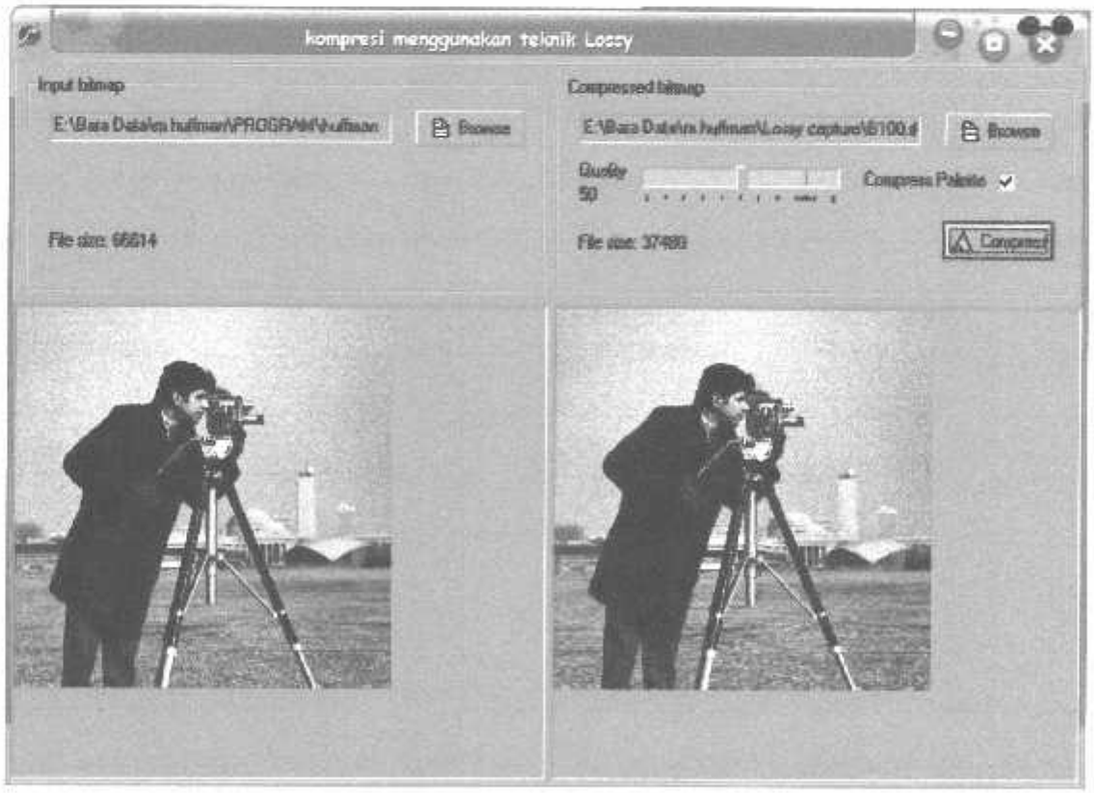


Gambar 5.6 citra B hasil kompresi menggunakan teknik *Huffman*

Gambar 5.7 citra B hasil dekompresi menggunakan teknik *Huffman*



Gambar 5.8 citra B menggunakan teknik *lossy* dengan kualitas citra 100%



Gambar 5.9 citra B menggunakan teknik *lossy* dengan kualitas citra 50%

Gambar 5.10 citra B menggunakan teknik *lossy* dengan kualitas citra 1%

Pada percobaan menggunakan teknik *Huffman* kualitas citra yang didapatkan setelah proses kompresi dan dekompresi adalah citra tidak mengalami kerusakan pada saat sesudah dikompresi dan dekompresi untuk ukuran hasil pengkompppresian masih terbilang efisien dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (59127 / 66614) x 100% = 11.239%.

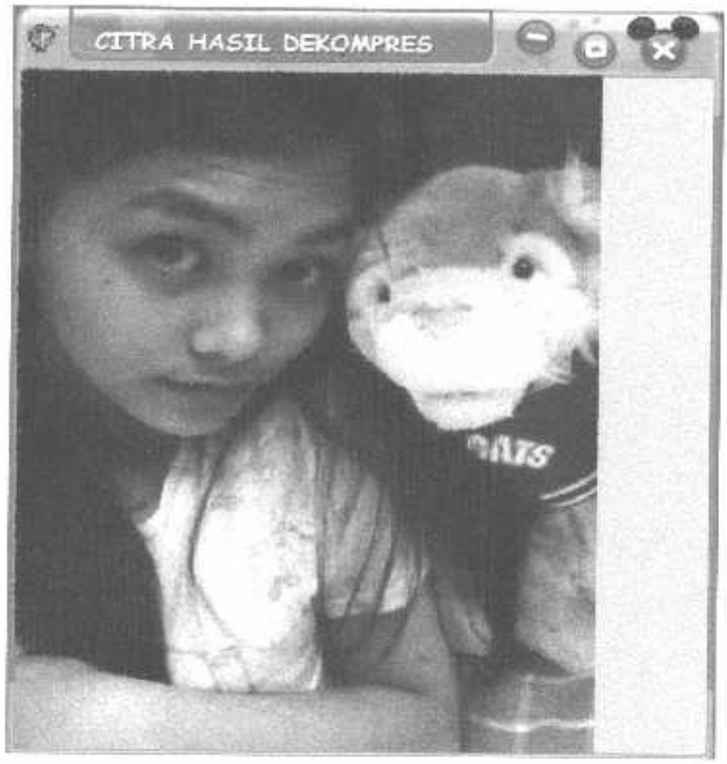
Pada percobaan menggunakan teknik *lossy* dapat diukur nilai pada citra seperti pada percobaan diatas :

- Apabila citra B dikompresi dengan kualitas 100% maka nilai dari kompresi citra akan berukuran dari 66614 menjadi 67172. pada citra tersebut akan mengalami pembengkakan (*Coding Redundancy*) pada nilai kualitasnya dikarenakan jumlah kode yang diberikan untuk menampilkan suatu *grayscale* melebihi dari apa yang dibutuhkan, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* hasil penghitungannya 100% - (67172 / 66614) x 100% = 0.837%.

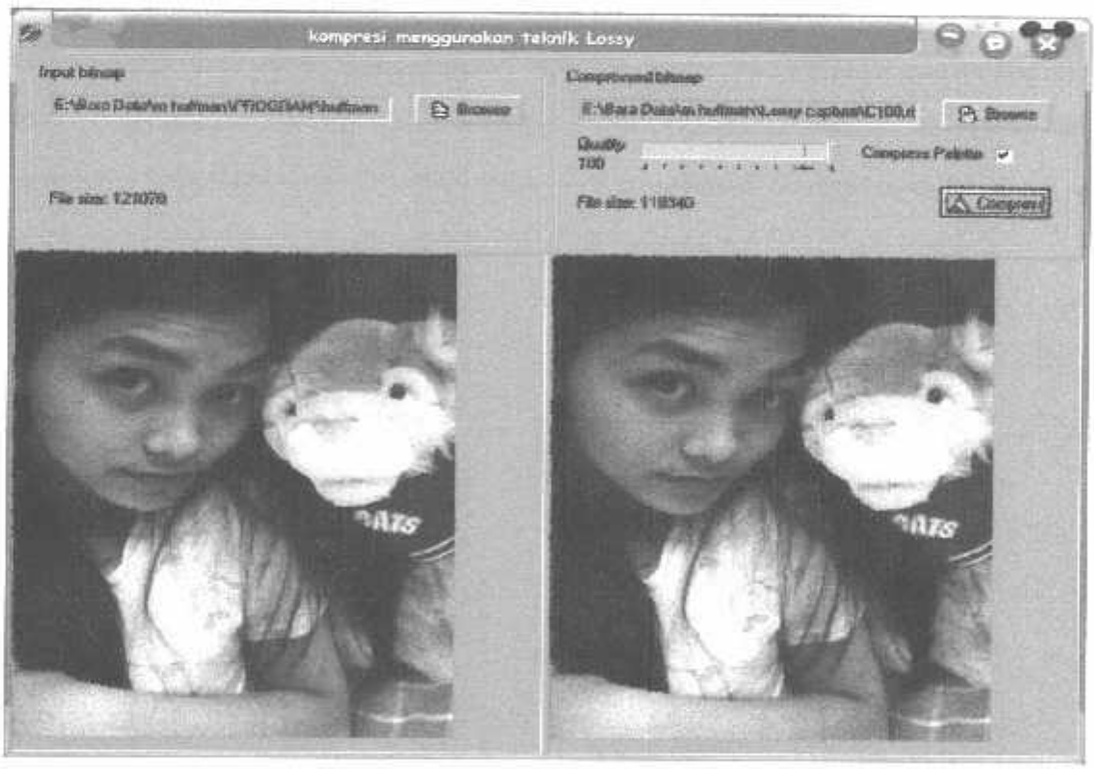
- Apabila citra B dikompresi dengan kualitas citra 50% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil, dari 66614 menjadi 37480 , dengan menggunakan rumus *"Rasio kompresi =100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (37480 / 66614) x 100% = 43.735%.
- Apabila citra B dikompresi dengan kualitas 1% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil dari 66614 menjadi 36168, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (36168 / 66614) x 100% = 45.705%.

c) Citra C dimensi 300 x 400 dengan ukuran 121078 byte

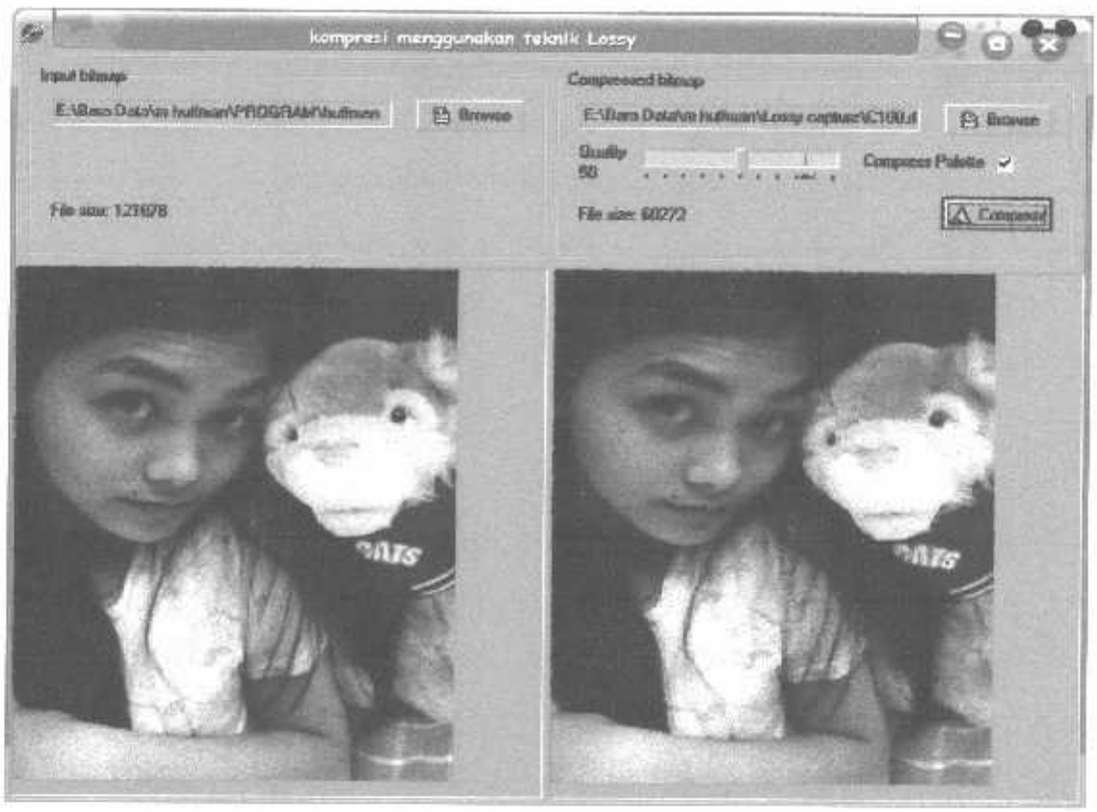
Gambar 5.11 citra C hasil kompresi menggunakan teknik *Huffman*



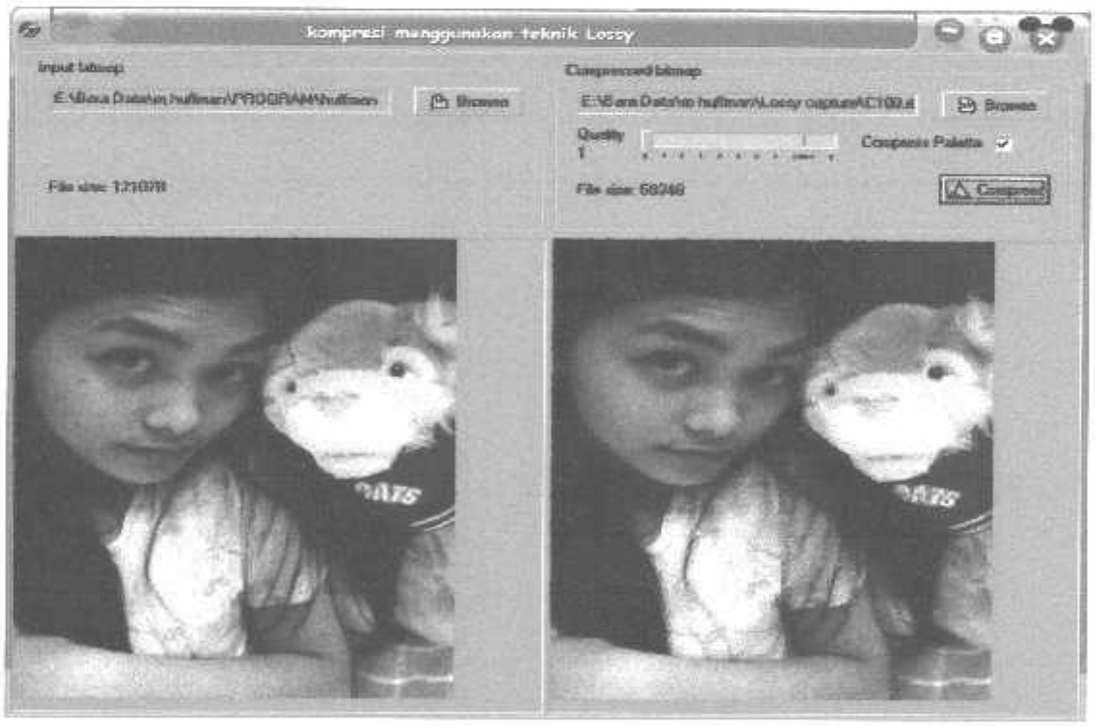
Gambar 5.12 citra C hasil dekompresi menggunakan teknik *Huffman*



Gambar 5.13 citra C menggunakan teknik *lossy* dengan kualitas citra 100%



Gambar 5.14 citra C menggunakan teknik *lossy* dengan kualitas citra 50%



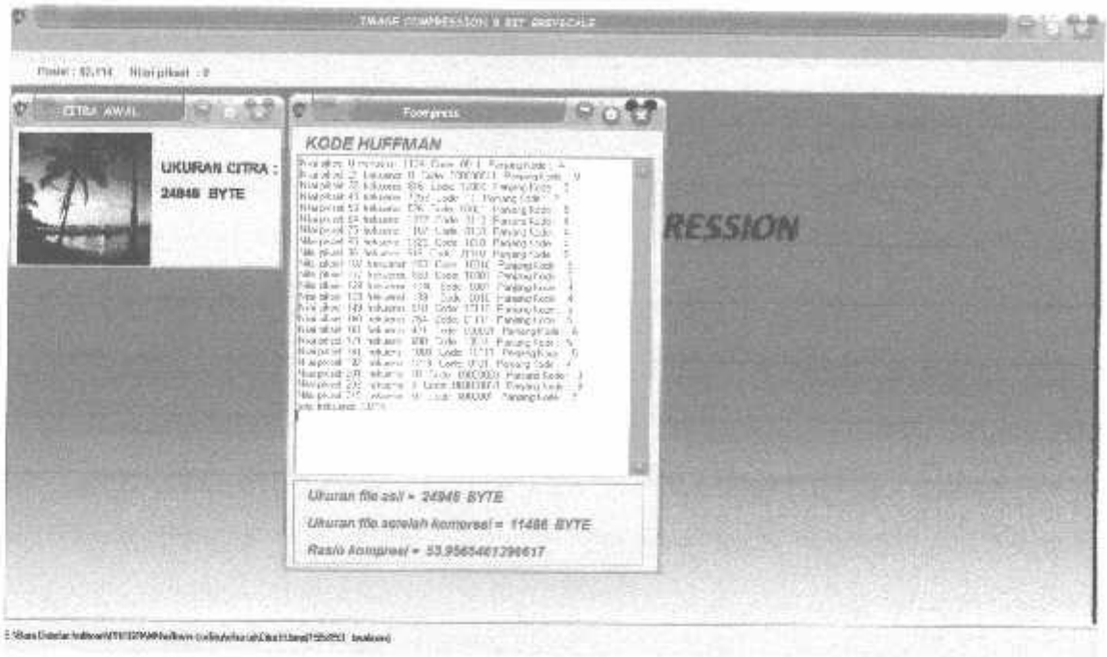
Gambar 5.15 citra C menggunakan teknik *lossy* dengan kualitas citra 1%

Pada percobaan menggunakan teknik *Huffman* kualitas citra yang didapatkan setelah proses kompresi dan dekompresi adalah citra tidak mengalami kerusakan pada saat sesudah dikompresi dan dekompresi untuk ukuran hasil pengkomppresian masih terbilang efisien, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (115032 / 121078) x 100% = 4.993%.

Pada percobaan menggunakan teknik *lossy* dapat diukur nilai pada citra seperti pada percobaan diatas:

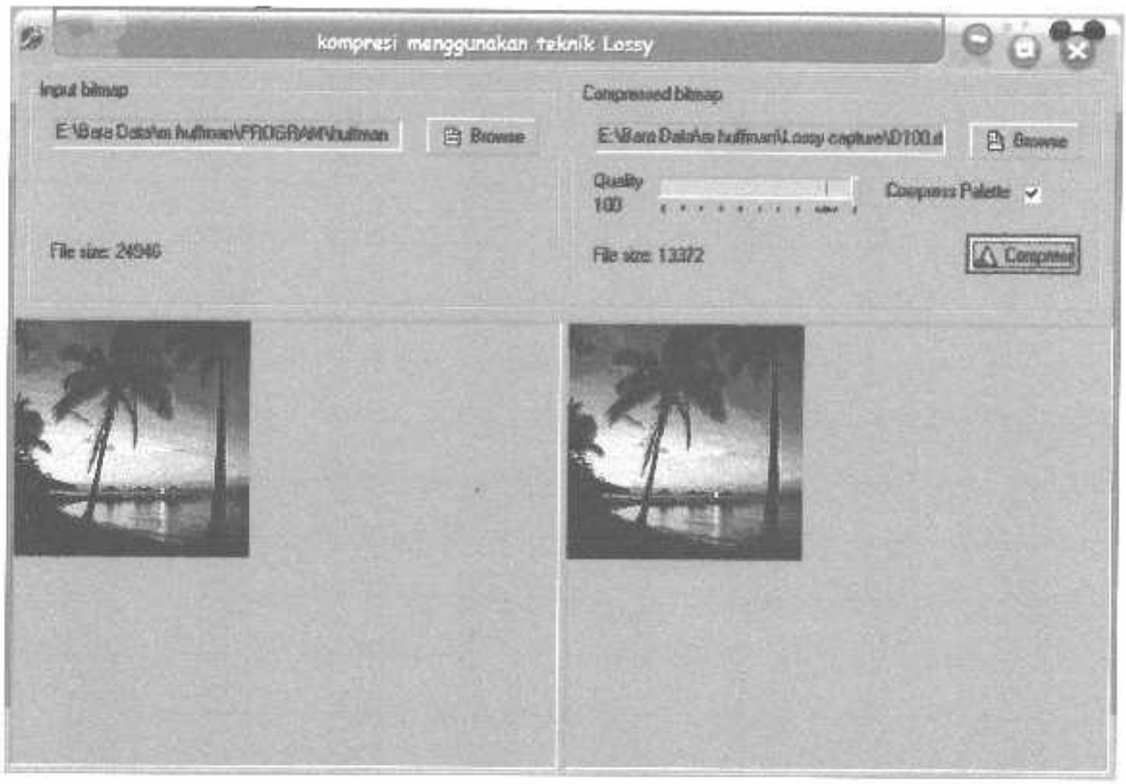
- Apabila citra C dikompresi dengan kualitas 100% maka nilai dari kompresi citra akan berukuran kecil dari 121078 menjadi 118340 pada citra tersebut tidak kerusakan pada kualitasnya, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (118340 / 121078) x 100% = 2.261%.
- Apabila citra C dikompresi dengan kualitas citra 50% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil dari 121078 menjadi 60272, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (60272 / 121078) x 100% = 50.220%.
- Apabila citra C dikompresi dengan kualitas 1% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil dari 121078 menjadi 59348, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (59348 / 121078) x 100% = 50.983%.

d) Citra D dimensi 155 x 153 dengan ukuran 24946 byte

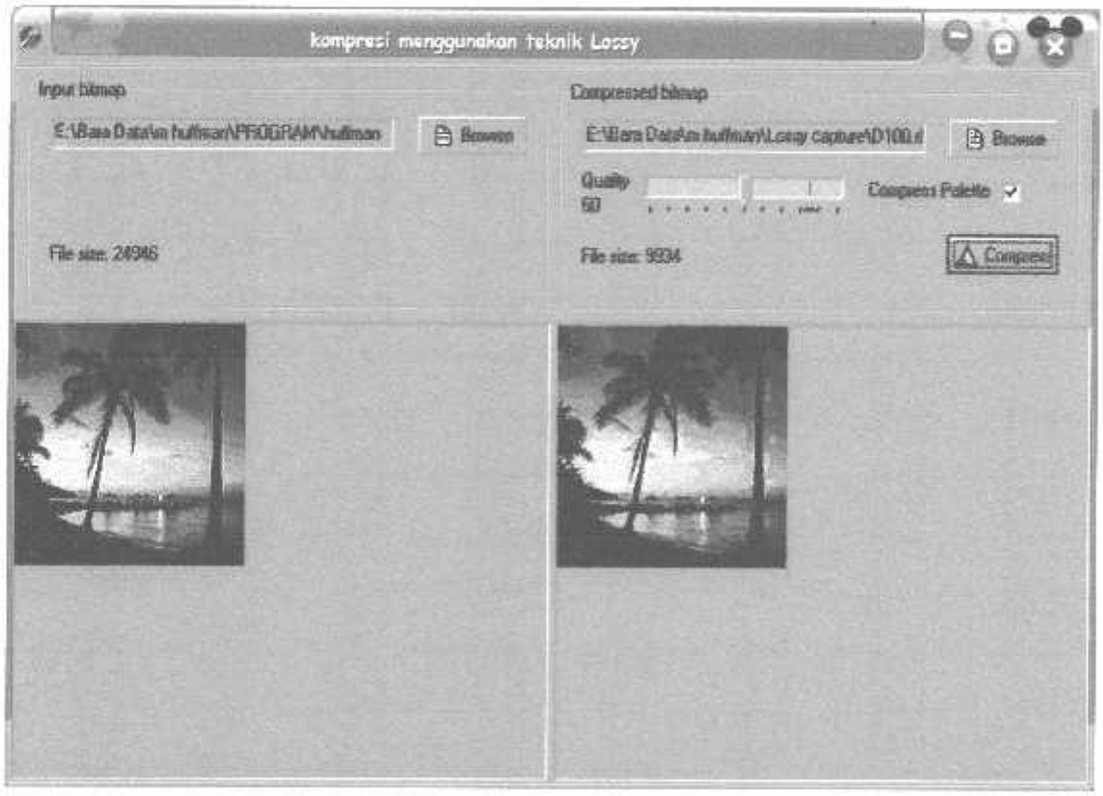


Gambar 5.16 citra D hasil kompresi menggunakan teknik *Huffman*

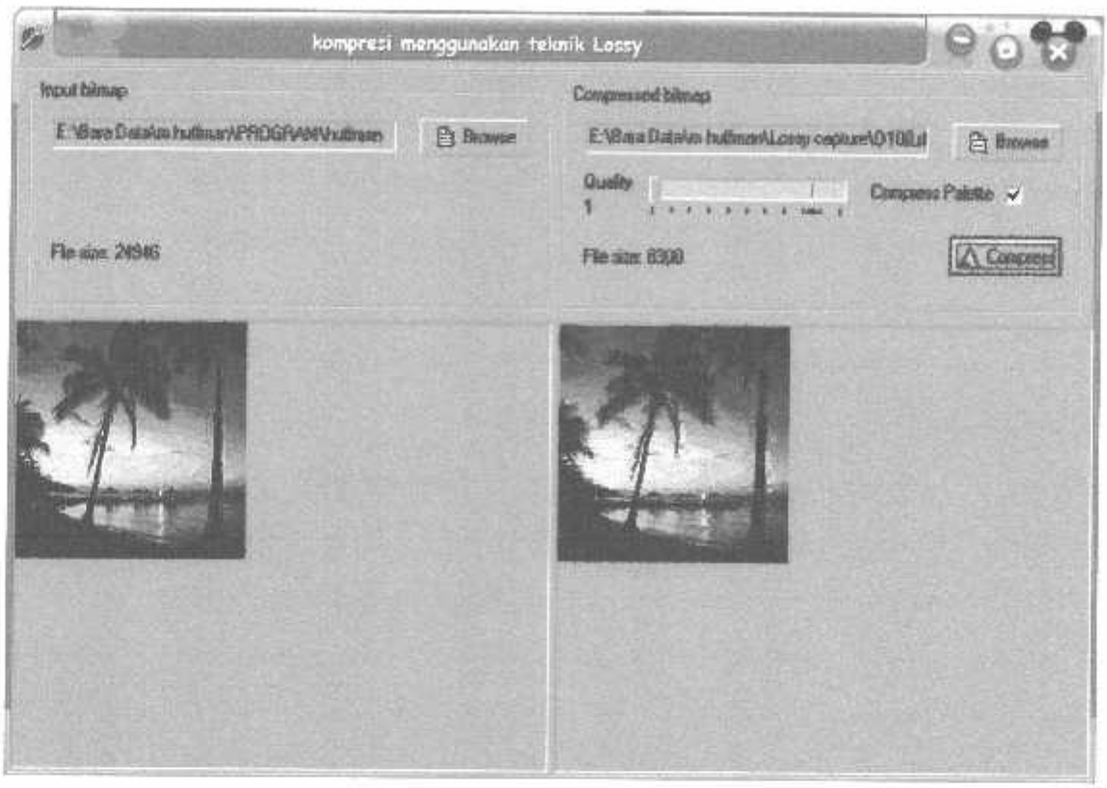
Gambar 5.17 citra D hasil dekompresi menggunakan teknik *Huffman*



Gambar 5.18 citra D menggunakan teknik *lossy* dengan kualitas citra 100%



Gambar 5.19 citra D menggunakan teknik *lossy* dengan kualitas citra 50%



Gambar 5.20 citra D menggunakan teknik *lossy* dengan kualitas citra 1%

Pada percobaan menggunakan teknik *Huffman* kualitas citra yang didapatkan setelah proses kompresi dan dekompresi adalah citra tidak mengalami kerusakan pada saat sesudah dikompresi dan dekompresi untuk ukuran hasil pengkomppresian masih terbilang efisien, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (11486 / 24946) x 100% = 53.956%.

Pada percobaan menggunakan teknik *lossy* dapat diukur nilai pada citra seperti pada percobaan diatas:

- Apabila citra D dikompresi dengan kualitas 100% maka nilai dari kompresi citra akan berukuran kecil dari 24946 menjadi 13372 citra tersebut tidak mengalami kerusakan pada kualitasnya, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* dan hasil penghitungannya 100% - (13372 / 24946) x 100% = 46.396%.
- Apabila citra D dikompresi dengan kualitas citra 50% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi kecil dari 24946 menjadi

9934, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* maka didapat hasil penghitungannya 100% - (9934 / 24946) x 100% = 60.177%.

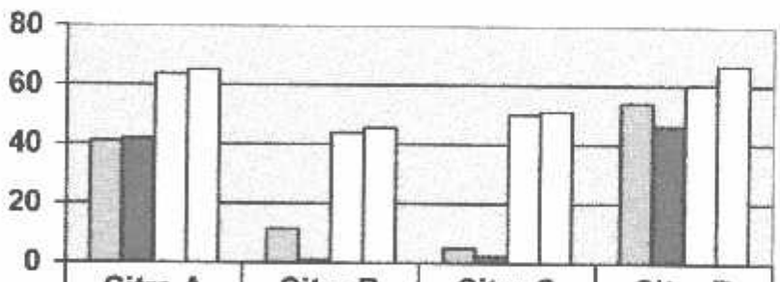
- Apabila citra D dikompresi dengan kualitas 1% maka citra akan mengalami kerusakan pada kualitasnya namun untuk nilai kompresi yang didapat dapat dikatakan ukuran citra tersebut menjadi sangat kecil dari 24946 menjadi 8300, dengan menggunakan rumus *"Rasio kompresi=100%-(ukuran citra hasil kompresi/ukuran citra semula)x100%"* didapatkan hasil penghitungannya adalah 100% - (8300 / 24946) x 100% = 66.728%.

5.2 Hasil Pengujian Metode

Hasil pengujian kedua metode diatas dapat dilihat dalam tabel 5.2 dan grafik 5.21 dibawah ini.

5.2 Tabel Hasil Perbandingan Rasio Kompresi Citra

No	Percobaan Citra	*Huffman* rasio	*Lossy* rasio pada kualitas citra 100%	*Lossy* rasio pada kualitas citra 50%	*Lossy* rasio pada kualitas citra 1%
1	Citra A dengan ukuran 16206	41.046%	41.885%	63.655%	65.000%
2	Citra B dengan ukuran 66614	11.239%	0.837%	43.735%	45.705%
3	Citra C dengan ukuran 121078	4.993%	2.261%	50.220%	50.983%
4	Citra D dengan ukuran 24946	53.956%	46.396%	60.177%	66.728%



	Citra A	Citra B	Citra C	Citra D
Huffman Rasio	41.046	11.239	4.993	53.956
Lossy Rasio 100%	41.885	0.837	2.261	46.396
Lossy Rasio 50%	63.655	43.735	50.22	60.117
Lossy Rasio 1%	65	45.705	50.983	66.728

Gambar 5.21 grafik perbandingan rasio pada kompresi citra

5.3 Kesimpulan Pengujian kedua metode :

Pada empat kali percobaan menggunakan teknik *Huffman* dan *Lossy* penulis dapat mengambil kesimpulan berupa :

- Untuk teknik kompresi *Huffman,* kualitas citra yang didapatkan setelah proses kompresi tidak mengalami kerusakan bahkan pada saat sesudah dikompresi dan dekompresi untuk ukuran hasil pengkompressian masih terbilang efektif, untuk kelebihan lainnya *Huffman* memiliki kemampuan dekompresi sehingga citra tersebut kembali keukuran semula dan tidak ada file yang hilang ataupun citra yang rusak saat pengkompresian berlangsung.
- Untuk teknik kompresi *Lossy*, kelebihan dari teknik kompresi ini yaitu kompresi bisa diperkecil dari pada teknik kompresi *Huffman* akan tetapi kekurangannya adalah semakin dikecil sebuah kompresi citra maka akan semakin kehilangan kualitas dari sebuah citra tersebut dan apabila warna hitam pada sebuah citra pada citra *greyscale* lebih mendominasi maka ukurannya akan semakin terbalik yaitu menjadi besar seperti ditampilkan dalam "Gambar 5.8 citra B menggunakan teknik *lossy* dengan kualitas citra 100% " dikarenakan pada citra tersebut akan mengalami pembengkakan (*Coding Redundancy*) pada nilai kualitasnya dikarenakan jumlah kode yang diberikan untuk menampilkan suatu *grayscale* melebihi dari apa yang dibutuhkan. kekurangan lainnya dari teknik kompresi ini adalah tidak adanya teknik dekompresi untuk mengembalikan lagi citra keukuran semula apabila sudah disimpan dalam file PC.

BAB VI

PENUTUP

6.1 Kesimpulan

Dari uraian dan pembahasan tugas akhir yang berjudul "Kompresi Citra Menggunakan Metode Huffman Dengan Bahasa Pemrograman Delphi 7.0". beberapa kesimpulan yang dapat diperoleh antara lain:.

1. Teknik kompresi citra dengan metode Huffman pada citra *gray scale* 8 bit dilakukan dengan pembuatan pohon Huffman berdasarkan frekuensi kemunculan nilai piksel. Pohon Huffman akan menghasilkan kode biner yang disimpan dalam tabel Huffman tiap 8 bit.
2. Kode biner yang dihasilkan dari Metode Huffman adalah setiap nilai piksel yang mempunyai frekuensi terbesar mempunyai jumlah bit yang pendek sedangkan nilai piksel yang mempunyai frekuensi kecil mempunyai bit yang lebih panjang..
3. Citra yang mempunyai sebaran nilai piksel tidak merata memiliki rasio kompresi yang relatif besar sedangkan citra dengan nilai piksel yang merata memiliki rasio kompresi yang lebih kecil.
4. Tingkat efisiensi memori file hasil kompresi dengan menggunakan Metode Huffman diukur dari besarnya rasio kompresi yang dihasilkan. Citra yang mempunyai sebaran nilai piksel tidak merata memiliki tingkat efisiensi lebih besar dibandingkan dengan citra dengan nilai piksel yang merata.
5. Apabila Citra berdimensi 300 x 400 dengan ukuran 121078 byte maka didapatkan hasil rasio kompresi adalah sebagai berikut :
   - Metode Huffman 100% - (115032 / 121078) x 100% = 4.993%.
   - Metode Lossy kualitas
     - 100% adalah 100% - (118340 / 121078) x 100% = 2.261%.
     - 50% adalah 100% - (60272 / 121078) x 100% = 50.220%.
     - 1% adalah 100% - (59348 / 121078) x 100% = 50.983%.

   Kualitas citra pada metode Lossy 100% kualitas citra, citra tidak mengalami kerusakan sedangkan pada kualitas 50% citra mengalami kerusakan dan pada kualitas 1% citra sangat tampak kerusakan pada saat selesai dikompresi.

6.2 Saran

1. Untuk dapat lebih melihat dan membuktikan keefisienannya, kelebihan dan kelemahan dari algoritma Huffman, perlu diadakannya sebuah penelitian yang bertujuan membandingkan seluruh algoritma kompresi dalam mengompres berbagai citra selain metode Lossy.
2. Untuk menyempurnakan program kompresi citra dengan metode Huffman pada Borland Delphi 7.0 hendaknya ditambahkan program untuk mengenali file hasil Huffman *Coding*.

DAFTAR PUSTAKA

Anonim http://en.wikipedia.org/wikipedia/Data_compression, Tanggal akses: Sabtu, 15 Maret 2010 pukul 06:15

Anonim http://home.unpar.ac.id/~integral/Volume%209/Integral%209%20No.%201/Perbandingan%20Kinerja%20Algoritma%20Kompresi.pdf, Tanggal akses: Minggu, 4 Mei 2010 pukul 20:01

Anonim http://mail.informatika.org/~rinaldi/Matdis/Makalah/MakalahIF2153-0708-075.pdf, Tanggal akses: Minggu, 4 Mei 2008 pukul 05:50

Darma Putra. 2009. "Pengolahan Citra Digital". Andi Publisher, Yogyakarta.

Darrel Hankerson, Greg A. Harris, Peter D. Johnson, Jr 2003. Information Theory and Data Compression Introduction to Second Edition. library.nu//Introduction to Information Theory and Data Compression.tanggal akses : 8-7-2011 pukul 23:00. Chapman & Hall/CRC Publication.

David Salomon, Giovanni Motta, 2010 With Contributions by David Bryant. Handbook of Data Compression Fifth Edition. Library.nu//ebooksclub.org__Handbook_of_Data_Compression__Fifth_edition. tanggal akses : 8-7-2011 pukul 19:30. Springer Publication.

Huffman Coding, http://www.en.wikipedia.org/wiki/Huffman_coding, Tanggal akses: Sabtu, 15 Maret 2010 pukul 06:28

Khalid Sayood, 2003. Lossless Compression Handbook. Library.nu//ebooksclub.org__Lossless_Compression_Handbook__Communications__Networking_and_Multimedia_. tanggal akses : 8-7-2011 pukul 20:16. Academic Press Publication.

K. S. Thyagarajan, 2011. Still Image and Video Compression with Matlab. library.nu//Still Image and Video Compression with MATLAB. tanggal akses : 8-7-2011 pukul 17:55. A JOHN WILEY & SONS, INC Publication.

Peter Wayner 1999. Data Compression for Real Programmers. library.nu//Compression Algorithms(The For Real Programmers Series).tanggal akses : 8-7-2011 pukul 22:40. Elsevier Publication.

Practical Huffman Coding, http://www.compressconsult.com/huffman/, Tanggal akses: Minggu, 4 Mei 20010 pukul 05:50

S. G. Hoggar 2006. Mathematics of Digital Images Creation, Compression, Restoration, Recognition. library.nu// Mathematics of Digital Images Creation, Compression, Restoration, Recognition. tanggal akses : 8-7-2011 pukul 18:06. Cambridge University Press Publication.

Todd R. Reed 2005. Digital Image Sequence Processing, Compression, and Analysis. library.nu//ebooksclub.org__Digital_Image_Sequence_Processing__Compression_and_Analysis.tanggal akses : 8-7-2011 pukul 21:45. CRC Press Publication.

T.Sutoyo, Edy Mulyanto, Dr. Vincent Suhartono, Oky Dwi Nurhayati dan Wijanarto. 2005. "Teori Pengolahan Citra Digital". Andi Publisher, Yogyakarta.

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No.2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI

FAKULTAS TEKNOLOGI INDUSTRI

Nama : BARA FIRMANA BUDIONO

NIM : 04. 12. 716

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Komputer dan Informatika S-1

Judul Skripsi : **KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0**

Dipertahankan di hadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Selasa

Tanggal : 9 Agustus 2011

Dengan Nilai : 85,05 (A)

Panitia Ujian Skripsi:

Ketua Majelis Penguji	Sekretaris Majelis Penguji
Ir. Yusuf Ismail Nakhoda, MT	Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP.Y.1018800189	NIP.Y.1030800417

Anggota Penguji:

Penguji I	Penguji II
Sotyohadi, ST	Sonny Prasetio, ST, MT
NIP.P.1039700309	NIP.P.1031000433

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Karanglo KM.2 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi Teknik Komputer, maka perlu adanya perbaikan skripsi untuk mahasiswa :

Nama : BARA FIRMANA BUDIONO
Nim : 04. 12. 716
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Komputer dan Informatika S-1
Masa Bimbingan : 02 April – 02 Oktober 2011
Judul Skripsi : **KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0**

No	Penguji	Tanggal	Uraian	Paraf
1.	Penguji I	09/08/2011	1. Alasan mengapa menggunakan 8 bit 2. Mengapa lebih ditekankan (difokuskan) derajat keabuan	
2.	Penguji II	09/08/2011	1. Abstrak 2. Keterangan pada gambar 3. Kesimpulan	

Disetujui:

Penguji I

Sotyohadi, ST
NIP.P.1039700309

Penguji II

Sonny Prasetio, ST, MT
NIP.P.1031000433

Mengetahui:

Dosen Pembimbing I

I Komang Somawirata, ST, MT
NIP.Y.1030100361

Dosen Pembimbing II.

Irmalia Suryani Faradisa, ST, MT
NIP. P.1030000365

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Bara Firman B.
NIM : 04.12.716.
Perbaikan meliputi :

- Alasan mengapa menggunakan 8 bit?
- Mengapa lebih diteliti/dikaji (difocuskan) derajat keabuan?

Malang, 9/8/ 2011

(Sofyohadi, ST)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : BARA FIRMANA DUDIONO ~~SONNY PRASETIO,ST.MT~~
NIM : 0412716
Perbaikan meliputi :

- Abstrak
- Keterangan pada gambar
- Kesimpulan

Malang, 9 Agustus 2008

(SONNY PRASETIO, ST.MT)

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, Km. 2
MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang bertanda tangan dibawah ini :
Nama : Bara Firmana B
NIM : 04.12.716
Semester : XIII (tiga belas)
Fakultas : Teknologi Industri
Jurusan : Teknik Elektro S-1
Konsentrasi : ~~TEKNIK ELEKTRONIKA~~
~~TEKNIK ENERGI LISTRIK~~
TEKNIK KOMPUTER DAN INFORMATIKA
Alamat : Jl. Sawojajar VII/53. Malang

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat ***SKRIPSI Tingkat Sarjana***. Untuk melengkapi permohonan tersebut, bersama kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan-persyaratan pengambilan ***SKRIPSI*** adalah sebagai berikut :

1. Telah melaksanakan semua praktikum sesuai dengan konsentrasinya (.....)
2. Telah lulus dan menyerahkan laporan Praktek Kerja (.....)
3. Telah lulus seluruh mata kuliah keahlian (MKB) sesuai konsentrasinya (.....)
4. Telah menempuh mata kuliah ≥ 134 sks dengan IPK ≥ 2 dan tidak ada nilai E (.....)
5. Telah mengikuti secara aktif kegiatan seminar skripsi yang diadakan jurusan (.....)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatianya kami ucapkan terima kasih.

Telah diteliti kebenaran data tersebut atas
Recording Teknik Elektro

(Ayu Wahkaryani)

Malang, 30-September-2010
Pemohon

(Bara Firmana B)

Ketua Jr

I Komang Somawirata, ST, MT
Irmalia Faradisa, ST, MT

PROPOSAL SKRIPSI

KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0

Diajukan untuk memenuhi persyaratan dalam memperoleh gelar Sarjana Teknik

Disusun Oleh:

Bara Pirmana Budiono 04.12.716

Diperiksa dan Disetujui Oleh

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
KONSENTRASI TEKNIK KOMPUTER & INFORMATIKA S-1

2011

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

LEMBAR PENGAJUAN JUDUL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik / Teknik Elektronika /Teknik Komputer & Informatika /Teknik Komputer / Teknik Telekomunikasi*)

1.	Nama Mahasiswa: Bara Fismana Budiono			Nim: 04.12.716
2.	Waktu Pengajuan	Tanggal: 04	Bulan: Februari	Tahun: 2011

Spesifikasi Judul (berilah tanda silang)**)

3.
- a. Sistem Tenaga Elektrik
- b. Energi & Konversi Energi
- c. Tegangan Tinggi & Pengukuran
- d. Sistem Kendali Industri
- e. Elektronika & Komponen
- f. Elektronika Digital & Komputer
- g. Elektronika Komunikasi
- h. lainnya .....

4.	Konsultasikan judul sesuai materi bidang ilmu kepada Dosen*) I Komang Somawirata, ST, MT	Ketua Jurusan Ir. Yusuf Ismail Nakhoda, MT NIP. Y. 1018800189
5.	Judul yang diajukan mahasiswa:	Kompresi citra menggunakan Metode HUFFMAN dengan bahasa pemrograman Delphi 7.0
6.	Perubahan judul yang disetujui Dosen sesuai materi bidang ilmu	.....
	Catatan:	
7.	Persetujuan Judul skripsi yang dikonsultasikan kepada Dosen materi bidang ilmu	Disetujui 201 Dosen Komentar : S.....SMT

Perhatian:
1. Formulir pengajuan ini harap dikembalikan kepada jurusan paling lambat satu minggu setelah disetujui kelompok dosen keahlian dengan dilampirkan proposal skripsi beserta persyaratan skripsi sesuai form S-1
2. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-2

INSTITUT TEKNOLOGI NASIONAL
Jl. Sigura-gura No 2
M A L A N G

Lampiran : 1 (Satu) Berkas
Pembimbing Skripsi

Kepada : **Yth. I Komang Somawirata, ST, MT.**
Dosen Institut Teknologi Nasional
M A L A N G

Yang bertanda tangan di bawah ini:

Nama : BARA FIRMANA BUDIONO
Nim : 04.12.716
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Komputer & Informatika

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing (Utama / Pendamping *), untuk penyusunan Skripsi dengan judul (proposal terlampir) :

"KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN

DENGAN BAHASA PEMROGRAMAN DELPHI 7.0"

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Skripsi Sarjana Teknik.
Demikian permohonan kami dan atas kesediaan Bapak/Ibu kami ucapkan terima kasih.

Malang, Februari 2011

Mengetahui
Ketua Jurusan Teknik Elektro S-1,

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

Hormat kami,

Bara Firmana Budiono

*) coret yang tidak perlu

Form S-3a

INSTITUT TEKNOLOGI NASIONAL
Jl. Sigura-gura No 2
M A L A N G

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa :

Nama : BARA FIRMANA BUDIONO

Nim : 04. 12. 716

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Komputer & Informatika

Dengan ini Menyatakan (bersedia / tidak bersedia *) Membimbing Skripsi dari mahasiswa tersebut, dengan judul :

"KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN

DENGAN BAHASA PEMROGRAMAN DELPHI 7.0"

Demikian surat Pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang, Februari 2011

Kami yang membuat pernyataan,

I Komang Somawirata, ST, MT.
NIP.Y. 1030100361

Catatan :
Setelah disetujui agar formulir ini
Diserahkah mahasiswa/i yang bersangkutan
Kepada Jurusan untuk diproses lebih lanjut.
***) coret yang tidak perlu**

Form S-3b

INSTITUT TEKNOLOGI NASIONAL
Jl. Sigura-gura No 2
M A L A N G

Lampiran : 1 (Satu) Berkas
Pembimbing Skripsi

Kepada : **Yth. Irmalia Suryani Faradisa, ST, MT.**
Dosen Institut Teknologi Nasional
M A L A N G

Yang bertanda tangan di bawah ini:

Nama : BARA FIRMANA BUDIONO
Nim : 04.12.716
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Komputer & Informatika

Dengan ini mengajukan permohonan, kiranya Bapak/Ibu bersedia menjadi Dosen Pembimbing (Utama / Pendamping *), untuk penyusunan Skripsi dengan judul (proposal terlampir) :

"KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN

DENGAN BAHASA PEMROGRAMAN DELPHI 7.0"

Adapun tugas tersebut sebagai salah satu syarat untuk menempuh Skripsi Sarjana Teknik.
Demikian permohonan kami dan atas kesediaan Bapak/Ibu kami ucapkan terima kasih.

Malang, Februari 2011

Mengetahui
Ketua Jurusan Teknik Elektro S-1,

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y. 1018800189

Hormat kami,

Bara Firmana Budiono

*) coret yang tidak perlu

Form S-3a

INSTITUT TEKNOLOGI NASIONAL
Jl. Sigura-gura No 2
M A L A N G

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa :

Nama : BARA FIRMANA BUDIONO

Nim : 04. 12.716

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Komputer & Informatika

Dengan ini Menyatakan (Lersedia / tidak bersedia *) Membimbing Skripsi dari mahasiswa tersebut, dengan judul :

"KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN

DENGAN BAHASA PEMROGRAMAN DELPHI 7.0"

Demikian surat Pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang, Februari 2011

Kami yang membuat pernyataan,

Irmalia Suryani Faradisa, ST, MT.
NIP.P. 1030000365

Catatan :
Setelah disetujui agar formulir ini
Diserahkah mahasiswa/i yang bersangkutan
Kepada Jurusan untuk diproses lebih lanjut.
***) coret yang tidak perlu**

Form S-3b

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/~~Teknik Elektronika~~/ Teknik Komputer & Informatika*)

1.	Nama Mahasiswa: BARA FIRMANA BUDIONO			N.m: 04.12.716
2	Keterangan Pelaksanaan	Tanggal 02-04-2011	Waktu	Tempat Ruang:

Spesifikasi Judul (berilah tanda silang)**)

3.	a. Sistem Tenaga Elektrik	e. Elektronika & Komponen
	b. Energi & Konversi Energi	(f.) Elektronika Digital & Komputer
	c. Tegangan Tinggi & Pengukuran	g. Elektronika Komunikasi
	d. Sistem Kendali Industri	h. lainnya .....

4	Judul Proposal yang diseminarkan Mahasiswa	KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian	.....
6.	Catatan:	.....

Catatan

.....

Persetujuan Judul Skripsi

Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II

7. Sama

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting) Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 14 April 2011

Nomor : ITN-222/I.TA/2/11
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Sdr./i. **I KOMANG SOMAWIRATA, ST, MT**
Dosen Institut Teknologi Nasional Malang

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: BARA FIRMANA. B
Nim	: 0412716
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik **Komputer & Informatika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu (enam) 6 bulan, terhitung mulai tanggal :

02 April 2011 s/d 02 Oktober 2011

Sebagai satu syarat untuk menempuh ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Di

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BNK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting) Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 14 April 2011

Nomor : ITN-223/I.TA/2/11
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Sdr./i. **IRMALIA S. FARADISA, ST, MT**
Dosen Institut Teknologi Nasional Malang

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: BARA FIRMANA. B
Nim	: 0412716
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik **Komputer & Informatika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu (enam) 6 bulan, terhitung mulai tanggal :

02 April 2011 s/d 02 Oktober 2011

Sebagai satu syarat untuk menempuh ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demi
ke

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo 10 KM
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : **BARA FIRMANA BUDIONO**

NIM : **04.12.716**

Masa Bimbingan : **02 April 2011 s/d 02 September 2011**

Judul Skripsi : **KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0**

No	Tanggal	Uraian	Paraf Pembimbing
1	23 April 2011	1. BAB II : Perbaikan Proses konversi citra analog ke digital 2. BAB III : Perbaikan Flowchart teknik Kompresi	
2	18 Mei 2011	1. BAB III : Perbaikan Flowchart teknik Kompresi dan tabel Huffman	
3	07 Juni 2011	1. BAB IV : Tambahan citra uji 2. Revisi makalah Seminar Hasil	
4	23 Juni 2011	1. BAB V : Perbaikan tabel Hasil uji dan penambahan rumus 2. ACC makalah seminar hasil	
5	30 Juli 2011	1. ACC Buku Skripsi	

Malang, Agustus 2011

Dosen Pembimbing I

(I Komang Somawirata, ST, MT)
NIP.Y.1030100361

FORM S-4b

INSTITUT TEKNOLOGI NASIONAL
JL. RAYA KARANGLO KM.2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : **BARA FIRMANA BUDIONO**
Nim : **04.12.716**
Masa Bimbingan : **2 APRIL 2011 s/d 2 SEPTEMBER 2011**
Judul Skripsi : **KOMPRESI CITRA MENGGUNAKAN METODE HUFFMAN DENGAN BAHASA PEMROGRAMAN DELPHI 7.0**

NO.	TANGGAL	URAIAN	PARAF PEMBIMBING
1.	23 April 2011	BAB II : Proses konversi citra analog ke digital BAB III : Gambar flowchart Teknik kompresi	
2.	18 Mei 2011	Penambahan Bab Rumusan masalah terbalik BAB III flowchart desain dan tabel Huffman	
3.	7 Juni 2011	Revisi makalah seminar hasil	
4.	8 Juni 2011	BAB IV : Tambahan citra uji	
5.	14 Juni 2011	BAB V : Revisi tabel hasil pengujian	
6.	23 Juni 2011	BAB V : Penambahan rumus dan Perhitungan teknik kompresi	
7.	24 Juni 2011	Revisi makalah seminar hasil	
8.	25 Juni 2011	BAB VI : Revisi kesimpulan	
9.	26 Juni		
10.	05		

Listing Program kompresi dan Dekompresi

```
procedure TFormutama.TampilKode;
var
 i,j:integer;
 tmpstr:string;
begin
 fcompress.memo1.lines.beginupdate;
 for i:=0 to 255 do
  begin
   if huffcodes[i].ada=false then
    continue;
   for j:=0 to huffcodes[i].panjangKode-1 do
    begin
     if huffcodes[i].kode[j] =1 then
      tmpstr:=tmpstr+'1'
     else
      tmpstr:= tmpstr+'0';
    end;
   fcompress.memo1.lines.add('Nilai piksel: '+ inttostr(i))'
'+'Frekuensi:  '+inttostr(charlist[i].frek)+'   '+'Code:
'+tmpstr+'   '+'Panjang Kode :
'+inttostr(huffcodes[i].panjangKode));

   tmpstr:='';
  end;
 fcompress.memo1.lines.endupdate;
end;



function IsGrayscaleBitmap: Boolean;
  function IsPixelGray(AColor: TColor): Boolean;
  const
    HEX_DIGIT = 6;
  var
    colorInHex, parseColor: string;
    i: Integer;
  begin
    Result := True;
    colorInHex := IntToHex(AColor, HEX_DIGIT);
    { Ambil 2 huruf pertama dari colorInHex }
    parseColor := Copy(colorInHex, 1, 2);

    for i := 1 to (Length(colorInHex) div 2)-1 do
    begin
      { Ambil 2 huruf selanjutnya dan bandingkan dengan 2
        huruf pertama }
      Result := Result and (Copy(colorInHex, 2*i+1, 2) =
parseColor);
      { Jika tidak sama maka keluar, Result fungsi bernilai
false }
      if not Result then
        Exit;
    end;
  end;
```

```
var
  row, column: Integer;
  color: TColor;
  bitmap: TBitmap;
begin
  Result := True;
  try
    { Memeriksa sebanyak width * height pixel }
    for row := 0 to Fcitra.image1.Width-1 do
    begin
      for column := 0 to Fcitra.image1.Height-1 do
      begin
        color := Fcitra.image1.Canvas.Pixels[row, column];
        { Memeriksa apakah komponen warna pixel[row, column]
          memiliki nilai yang setara/seimbang/sama }
        Result := Result and IsPixelGray(color);
        a:=true;
        { Jika ada salah satu pixel saja yang bukan  rayscale
maka
          keluar, dan Result dari fungsi adalah false }

      end;
    end;
 finally

  end;
   if not Result then
   begin
     a:=false;
   end;
end;


procedure Inisialisasi;
begin
 zeromemory(@charlist,sizeof(charlist));
 zeromemory(@table,sizeof(table));
 zeromemory(@huffcodes,sizeof(huffcodes));
 zeromemory(@hh,sizeof(hh));
 zeromemory(@ww,sizeof(ww));
 infile:='';
 outfile:='';
 endbits:=0;
 ufilesize:=0;
 cfilesize:=0;
 root:=nil;
end;
procedure Finalize;
begin
 freeandnil(ifile);
 freeandnil(ofile);
end;

function SetInputfile(afilename:string):boolean;
begin
```

```
 result:=fileexists(afilename);
 if result=true then
  begin
   ifile:=Tfilestream.create(afilename,fmOpenRead or
fmShareDenyNone);
   infile:=afilename;
   ufilesize:=GetFileSize(ifile.Handle,nil);
  end
 else
  raise EError.Create('Invalid Filename');
end;

function SetOutputfile(afilename:string):boolean;
begin
 ofile:=Tfilestream.create(afilename,fmCreate or
fmShareDenyNone);
 outfile:=afilename;
 result:=true;
end;

procedure InitList;
var
 i:integer;
begin
 zeromemory(@charlist,sizeof(charlist));
 zeromemory(@huffcodes,sizeof(huffcodes));
 for i:=0 to 255 do
  begin
   charlist[i].pikse [charlist[i].jml kode]:=i;
   inc(charlist[i].jml kode);
  end;
end;


procedure jumlahfrekuensi;
var
i,j,x :integer;
PC:PBytearray;
picture:tpicture;

begin
  if (picture=nil) then
    exit;
   if( fcitra.Image1.Picture.bitmap.PixelFormat= pf8Bit) then
     begin
        for i := 0 to 255 do
        begin
          charlist[i].frek:= 0;
        end;

       x:=0;
       for i := 0 to fcitra.Image1.Picture.Height-1 do
         begin
           PC :- fcitra.Image1.Picture.Bitmap.ScanLine[i ;
           for j := 0 to fcitra.Image1.Picture.Width-1 do
             begin
```

```
                InBuffer[x]:=PC[j];
                Inc(charlist[InBuffer[x]].frek);
                inc(x);
              end;
          end;
        byk:=x;
        jum:=0;
        for i:= 0 to 255 do
        jum  :=jum+ charlist[i].frek;
   end;
end;


procedure pohonHuffman;
var
 i,cnt,tmp:integer;
 pinfo1,pinfo2:Phuffinfo;
begin
 pinfo1:=nil;
 pinfo2:=nil;
 cnt:=255;
 while true do
  begin
   tmp:=maxint;
   for i:=0 to cnt do
    begin
     if (charlist[i].frek<tmp) and (charlist[i].frek > 0)
and(charlist[i].sudah=false) then
      begin
       pinfo1:=@charlist[i];
       tmp:=pinfo1.frek;
      end;
    end;
   if pinfo1=nil then
    break;
   pinfo1.sudah:=true;
   tmp:=maxint;
   for i:=0 to cnt do
    begin
     if (charlist[i].frek<tmp) and (charlist[i].frek > 0) and
     (charlist[i].sudah=false) then
      begin
       pinfo2:=@charlist[i];
       tmp:=pinfo2.frek;
      end;
    end;
   if pinfo2=nil then
    break;
   pinfo2.sudah:=true;
   inc(cnt);
   charlist[cnt].frek:=pinfo1.frek+pinfo2.frek;
   strcat(@charlist[cnt].piksel,@pinfo1.piksel);
   strcat(@charlist[cnt].piksel,@pinfo2.piksel);
   charlist[cnt].jml_kode:=pinfo1.jml_kode+pinfo2.jml_kode;
   charlist[cnt].kiri:=pinfo1;
   charlist[cnt].kanan:=pinfo2;
```

```
    pinfo1:=nil;
    pinfo2:=nil;
   end;
  rootnode:=@charlist[cnt];
 end;


 procedure KodeHuffman;
 var
  i,j:integer;
  tmpnode:phuffinfo;
  flag:integer;
 begin
  for i:=0 to 255 do
   begin
    flag:=-1;
    tmpnode:=rootnode;
    while tmpnode.kiri<>nil do
     begin
      for j:=0 to tmpnode.kiri.jml_kode-1 do
       if tmpnode.kiri.piksel[j]= i then
        begin
         flag:=0;
         tmpnode:=tmpnode.kiri;
         break;
        end;
       if flag=-1 then
        begin
         for j:=0 to tmpnode.kanan.jml_kode-1 do
          if tmpnode.kanan.piksel[j]= i then
           begin
            flag:=1;
            tmpnode:=tmpnode.kanan;
            break;
           end;
        end;
      if flag=-1 then
       break;
      huffcodes[i].ada:=true;
      huffcodes[i].kode[huffcodes[i].panjangKode]:=flag;
      huffcodes[i].panjangKode:=huffcodes[i].panjangKode+1;
      huffcodes[i].char:=i;
      flag:=-1;
     end;
   end;
 end;

 procedure TabelHuffman;
 var
  i,j,k:integer;
  tmpcode:byte;
  bit:integer;
 begin
 k:=0;
  bit:=0;
  tmpcode:=0;
```

```
  for i:=0 to 255 do
   begin
    if huffcodes[i].ada=false then
     continue;
    table[k]:=i;
    tab_e[k+1]: huffcodes[i].panjangKode;
    k:=k+2;
    for j:=0 to huffcodes[i].panjangKode-1 do
     begin
      tmpcode:=tmpcode shl 1 or huffcodes[i].kode[j];
      bit:=bit+1;
      if bit=8 then
       begin
        table[k]:=tmpcode;
        k:=k+1;
        bit:=0;
        tmpcode:=0;
       end;
     end;
    if bit>0 then
     begin
      tmpcode:=tmpcode shl (8-bit);
      table[k]:=tmpcode;
      k:=k+1;
      bit:=0;
      tmpcode:=0;
     end;
   end;
  Jumtabel:=k;
end;

procedure SimpanFile;
var
x,y, i,j,k:integer;
 tmpcode:byte;
 bit:integer;
 bufcount:integer;
 PC:PByteArray;
begin
k:=0;
 tmpcode:=0;
 bit:=0;
 h:= fcitra.Image1.Picture.Height;
 w:=fcitra.Image1.Picture.Width;
 ofile.seek(1,sofrombeginning);
 ofile.write(Jumtabel,sizeof(Jumtabel));
 ofile.write(table,Jumtabel);
 ofile.write(h,sizeof(h));
 ofile.write(hh,h);
 ofile.write(w,sizeof(w));
 ofile.write(ww,w);
 ifile.seek(0,soFrombeginning);
 for i:=0 to byk do
   begin
    for j:=0 to huffcodes[inbuffer[i]].panjangKode-1 do
     begin
```

```
        tmpcode:=(tmpcode shl 1) or
        huffcodes[inbuffer[i]].kode[j];
        inc(bit);
        if bit=8 then
          begin
            outbuffer[k]:=tmpcode;
            inc(k);
            bit:=0;
            tmpcode:=0;
          end;
         if k=byk then
          begin
            ofile.write(outbuffer,byk);
            k:=0;
          end;
      end;
    end;
  if bit>0 then
   begin
    tmpcode:=tmpcode shl (8-bit);
    outbuffer[k]:=tmpcode;
    k:=k+1;
   end;
  if k>0 then
   ofile.write(outbuffer,k);
   ofile.seek(0,sofrombeginning);
   ofile.write(bit,1);
end;


procedure Compress(var usize,csize:integer);
begin
 InitTis;
 jumlahfrekuensi;
 pohonHuffman;
 kodeHuffman;
 TabelHuffman;
 SimpanFile;
 usize:=ufilesize;
 csize:=HitungKompres;
end;


function HitungKompres:integer;
var
 i:integer;
begin
 result:=0;
 for i:=0 to 255 do
  begin
   if huffcodes[i].ada then
    result:=result+huffcodes[i].panjangKode*charlist[i].frek;
  end;
cfilesize:=result div 8;
 if result mod 8>0 then
```

```
  cfilesize:=cfilesize-1;

cfilesize:=cfilesize+h+w+sizeof(h)+sizeof(w)+Jumtabel+sizeof(Jum
Label)+1;
  result:=cfilesize;
//cfilesize:=result
end;


procedure TformUtama.Open1Click(Sender: TObject);
var
fc: string;
begin

  if (OpenPictureDialog1.Execute) then
        SetInputFile(OpenPictureDialog1.FileName)
         else
        exit;
       if (fcitra =nil) then
        Application.CreateForm(Tfcitra, fcitra);
        Fcitra.image1.picture.LoadFromFile(OpenPictureDialog1.Fi
        leName);
        Fcitra.ClientHeight:=Fcitra.Image1.Picture.Height;
        Fcitra.Client_Width:=Fcitra.Image1.Picture.Width+200;
        Fcitra.Label1.Left:=  Fcitra.Image1.Width+20;
        Fcitra.Label3.Left:=  Fcitra.Image1.Width+20;
         Fcitra.Label1.Caption:=inttostr(ufilesize)+'  '+'BYTE';
        case
          (Fcitra.Image1.Picture.Bitmap.PixelFormat) of
         pf1bit :fc :='biner';
         pf8bit :fc :='keabuan';
         pf24bit :fc :='true colour';
        end;
       StatusBar1.SimpleText := OpenPictureDialog1.FileName
        + '('+ IntToStr(Fcitra.Image1.Picture.Width) +  'x'
       IntToStr(Fcitra.Image1.Picture.Height) + ',  '+fc+')';
end;

procedure TformUtama.Compress1Click(Sender: TObject);
var
 c1,c2:Integer;
begin
  if (Fcitra=nil) then
  begin
  ShowMessage('Citra belum ada!!!!!!!!!');
  exit;
  end
  else
  IsGrayscaleBitmap;
 If (Fcitra.Image1.Picture.Bitmap.PixelFormat <> pf8bit) or
(a=false) then
 begin
  if (MessageDlg('Bukan Citra Grey scale 8 bit ! Konversi ke
Gray scale?',mtWarning,[mbYes,mbNo],0) = mrYes) then
  formabu.Show;
 end
```

```
 else
 begin
  if savedialog1.execute=true then
  SetOutputfile(savedialog1.filename)
 else
 exit;

 fcompress.memo1.clear;
 Compress(c1,c2);
 TampilKode;

 fcompress.Label2.Caption:='Ukuran file asli =  '
+inttostr(c1)+'  '+ 'BYTE';
 fcompress.Label3.Caption:='Ukuran file setelah kompresi =  '
+inttostr(c2)+'  '+ 'BYTE';
 fcompress.Label4.Caption:='Rasio kompresi =  ' +floattostr((1-
c2/c1)*100);
 fcompress.memo1.lines.add('total frekuensi: '+ inttostr(jum));
 fcompress.ShowModal;
 Finalize;

end;

end;

procedure AmbilTabel;
var
 s,j,k,l:integer;
 index:integer;
 length,cnt:integer;
 tmpcode:byte;
begin
 ifile.seek(0,soFromBeginning);
 ifile.read(endbits,1);
 if endbits=0 then
  endbits:= 8;
 ifile.Read(table,sizeof(table));
 Jumtabel:=pinteger(@table[0])^;
 k:=sizeof(Jumtabel);
 while k<Jumtabel+sizeof(Jumtabel) do
  begin
   j:=0;
   index:=table[k];
   k:=k+1;
   length:=table[k];
   k:=k+1;
   huffcodes[index].Ada:=true;
   huffcodes[index].char:=index;
   huffcodes[index].panjangKode:=length;
   while j<=length-1 do
    begin
     tmpcode:=table[k];
     k:=k+1;
     if (length-j)>8 then
      cnt:=8
     else
```

```
        cnt:=length-j;
       for l:=1 to cnt do
        begin
         huffcodes[index].kode[l+j-1]:=(tmpcode shr (8-l)) and 1;
        end;
       j:=j+cnt;
      end;
    end;
end;

procedure BalikPohon;
var
 rinfo:phuffinfo;
 i,j,k:integer;
begin
 k:=0;
 zeromemory(@charlist,sizeof(charlist));
 for i:=0 to 255 do
  begin
   rinfo:=@charlist[511];
   for j:=0 to huffcodes[i].panjangKode-1 do
    begin
     charlist[k].huff:=huffcodes[i].kode[j];
     if huffcodes[i].kode[j]=0 then
      begin
       if rinfo.kiri=nil then
        begin
         rinfo.kiri:=@charlist[k];
         k:=k+1;
        end;
       rinfo:=rinfo.kiri;
      end
     else
      begin
       if rinfo.kanan=nil then
        begin
         rinfo.kanan:=@charlist[k];
         k:=k+1;
        end;
       rinfo:=rinfo.kanan;
      end;
     if j=huffcodes[i].panjangKode-1 then
      rinfo.char:=huffcodes[i].char;
    end;
  end;
end;

procedure AmbilFile;
var
n,m, tmp,bufcount,z,y,i,j,l:integer;
 tmpbit,tmpbyte:byte;
 tmpnode:phuffinfo;
 pc:PByteArray;
begin
 i:=0;j:=1;l:=0;
 tmpnode:=@charlist[511];
```

```
 ifile.seek(1+Jumtabel+sizeof(Jumtabel),soFromBeginning);
 ifile.Read(hh,sizeof(hh));
 m:=pinteger(@hh[0])^;

ifile.seek(1+Jumtabel+sizeof(Jumtabel)+m+sizeof(m),soFromBeginni
ng);
 ifile.Read(ww,sizeof(ww));
 n:=pinteger(@ww[0])^;

ifile.seek(1+JumLabel+sizeof(Jumtabel)+m+sizeof(m)+n+sizeof(n),s
eFromBeginning);
 bufcount:=ifile.read(inbuffer,1000000);
 tmpbyte:=inbuffer[0];
 i:=i+1;
 repeat
  while true do
   begin
    while (tmpnode.kiri<>nil) do
     begin
      if j>8 then
       begin

        tmpbyte:=inbuffer[i];
        i:=i+1;
        if i bufcount then
         begin
          bufcount:=ifile.read(inbuffer,1000000);
          i:=0;
         end;
        j:=1;
       end;
      tmpbit:=(tmpbyte shr (8-j)) and 1;
      if tmpnode.kiri.huff=tmpbit then
       begin
        tmpnode:=tmpnode.kiri;
        j:=j+1;
       end
      else
       begin
        tmpnode:=tmpnode.kanan;
        j:=j+1;
       end;
     end;
     OutBuffer[l]:=tmpnode.char;
    l:=l+1;
    tmpnode:=@charlist[511];
    if (bufcount=0) and (i=0) and (j>endbits) then
     break;
   end;
until bufcount=0;
 if l>0 then
 begin
    FORM1.Image1.Picture:=N_L;
    z:=0;
    form1.image1.ClientHeight:=m;
    form1.image1.ClientWidth:=n;
```

```
      for i:= 0 to m-1  do
            for j:=0 to n-1    do
              begin
               tmp:=OutBuffer[z];
              form1.image1.Canvas.Pixels[j,i]:=RGB(tmp,tmp,tmp);
                 z:=z+1;
              end;

          form1.image1.Picture.Bitmap.PixelFormat:=pf24bit;
        end;

         form2.Image1.Height:=form1.Image1.Height;
         form2.Image1.width:=form1.Image1.width;
         form2.Image1.Picture:=form1.Image1.picture;
          FORM2.ClientHeight:=form1.Image1.Height;
          FORM2.ClientWidth:=form1.Image1.WIDTH;
          form2.ShowModal;
         Finalize;
end;

procedure Decompress;
begin
 AmbilLabel;
 BalikPohon;
 AmbilFile;
end;


procedure TformUtama.BtClick(Sender: TObject);
var
 FromF: file;
 fsize:longint;
begin
 Inisialisasi;

 If opendialog1.execute=true then
  SetInputfile(opendialog1.FileName)
 else
  exit;
    AssignFile(FromF, OpenDialog1.FileName);
    Reset(FromF, 1);  { Record size = 1 }
    Fsize := FileSize(FromF);
  if (MessageDlg('Dekompres file '+ '('+ opendialog1.FileName
       + ' ) dengan ukuran '+ ' '+ IntToStr(Fsize) +'
'+'byte?',mtWarning,[mbYes,mbNo],0) = mrYes) then
 Decompress;
  Finalize;
end;

procedure TformUtama.Histogram1Click(Sender: TObject);
begin
       if (FormHist = nil) then
       Application.CreateForm(TFormHist, FormHist);
       FormHist.Top := Fcitra.Top;
       FormHist.Left :=Fcitra.Left+Fcitra.Width;
       FormHist.picture :=Fcitra.Image1.Picture;
```

```
        FormHist.MenghitungHistogram(0);
        FormHist.Repaint;
end;

procedure TformUtama.Save1Click(Sender: TObject);
begin
    if savedialog1.execute =true then
    Fcompress.memo1.Lines.SaveToFile(savedialog1.filename);
end;

procedure TformUtama.CITRAAWAL1Click(Sender: TObject);
var
x,y : integer;
Pimg : Pbytearray;

begin
  fpawal.ST.ColCount:= fcitra.Image1.Picture.Width;
   fpawal.St.RowCount:= fcitra.image1.Picture.Height;
   if( fcitra.Image1.Picture.bitmap.PixelFormat= pf8Bit) then
  begin
   for y := 0 to fcitra.image1.Picture.Height-1 do
    begin
      Pimg := fcitra.image1.Picture.Bitmap.ScanLine[y];
      for x := 0 to fcitra.image1.Picture.Width-1 do
      begin
         fpawal.ST.Cells[x,y] := IntToStr(Pimg[x]);
      end;
    end;
   end;

 FPawal.Show;

 FPawal.label1.Caption:='jumlah baris=
'+inttostr(FPawal.St.RowCount)+'';
 FPawal.Label2.Caption:='jumlah kolom=
'+inttostr(FPawal.St.colCount)+'';

end;
```