PENERAPAN HUFFMAN CODING UNTUK KOMPRESI CITRA PADA JARINGAN WIFI 802.11 B/G SECARA REAL TIME

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

SAFRONI WIBAWANTO
NIM. 0410633066-63

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2009

LEMBAR PERSETUJUAN

PENERAPAN HUFFMAN CODING UNTUK KOMPRESI CITRA PADA JARINGAN WIFI 802.11 B/G SECARA REAL TIME

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

SAFRONI WIBAWANTO
NIM. 0410633066-63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rusmi Ambarwati ST.,MT	**Rudy Yuwono ST., MSc.**
NIP 19720204 200003 2 002	**NIP 19710615 199802 1 003**

LEMBAR PENGESAHAN

PENERAPAN HUFFMAN CODING UNTUK KOMPRESI CITRA PADA JARINGAN WIFI 802.11 B/G SECARA REAL TIME

SKRIPSI

KONSENTRASI TEKNIK TELEKOMUNIKASI

Diajukan untuk memenuhi persyaratan

Memperoleh gelar Sarjana Teknik

Disusun oleh :

SAFRONI WIBAWANTO
NIM. 0410633066-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 29 Desesmber 2009

Dosen Penguji

Ir. Erfan A. Dahlan,.MT
NIP 19530714 198203 1 003

Ali Mustofa, ST.,MT
NIP 19710601 200003 1 001

Ir. Endah Budi P.,MT.
NIP 19621116 198903 2 002

Mengetahui,
Ketua Jurusan Teknik Elektro

Rudy Yuwono ST.,MSc.
NIP 19710615 199802 1 003

PENGANTAR

Assalamu'alaikum Wr.Wb.

Puji dan syukur penulis sampaiakan kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan berkat-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul **"Penerapan *Huffman Coding* untuk Kompresi Citra pada Jaringan Wifi 802.11 b/g secara *Real Time*"** dengan baik. Skripsi ini disusun sebagai salah satu syarat untuk mencapai gelar Sarjana Teknik dari jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.

Penulis menyadari bahwa tanpa bantuan, bimbingan serta dorongan dari semua pihak penyelesaian skripsi ini tidak mungkin bisa terwujud. Pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Ayah, Ibu dan adik-adikku (Luvita Dwi F. dan Chusnul Riska I.) yang telah banyak memberikan kasih sayang, dukungan dan doa yang tak pernah berhenti mengalir hingga terselesainya skripsi ini.
2. Bapak Rudy Yuwono ST., MSc. Selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya dan Bapak M. Aziz Muslim ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
3. Bapak Ali Mustofa ST., MT. selaku KKDK Telekomunikasi Jurusan Teknik Elektro Universitas Brawijaya.
4. Ibu Rusmi Ambarwati ST., MT. dan bapak Rudy Yuwono ST., MSc. selaku dosen pembimbing yang telah banyak memberikan pengarahan dan bimbingan serta atas segala bantuan dan saran yang membangun dalam penyelesaian skripsi ini.
5. Glorya Rakhmawaty yang telah memberikan motivasi dan cintanya yang tanpa terasa telah banyak memberi arti dalam hidupku hingga terselesainya skripsi ini.
6. Ainur Rofiq, Eri Wibowo, Yoga Eka Ari P., Roghib A., Wirawan Hidayatullah, A. Reza Alhadig, M. Wahyuniarto , M. Riza F., Triaji Surya P., Putra Prima, Gamalia Permata D., Choiriyah Indah S., Aditya Junianto, Tulus Kurniawan, Jatayu W. dan teman-teman elektro angkatan 2004 yang tidak bisa disebutkan satu per satu yang telah banyak memberi bantuan, semangat, dukungan dan motivasi.

7. Teman-teman kos Kerto Raharjo Dalam 4 yang telah memberikan semangat dan doanya.
8. Keluarga besar Teknik elektro Universitas Brawijaya terima kasih untuk semuanya.
9. Semua pihak yang telah banyak memberikan bantuan dan dukungan secara langsung maupun tidak langsung atas penyusunan skripsi ini.

Dalam penyusunan skripsi ini, penulis menyadari bahwa skripsi ini belumlah sempurna, karena keterbatasan kemampuan dan kendala-kendala lain yang terjadi selama pengerjaan skripsi ini. Semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Wassalamu'alaikum Wr.Wb

Malang, 22 November 2009

Penulis

ABSTRAK

Safroni Wibawanto, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, November 2009, Penerapan *Huffman Coding* untuk Kompresi Citra pada Jaringan Wifi 802.11 b/g secara *Real Time*, Dosen Pembimbing : Rusmi Ambarwati, ST., MT., dan Rudy Yuwono, ST., MSc.

Kompresi citra yang merupakan salah satu metode untuk memperkecil ukuran data citra guna mempercepat proses transmisi yang salah satunya diterapkan pada jaringan wireless. Perancangan dan penerapan kompresi citra menggunakan Huffman coding pada jaringan wireless menggunakan wifi 802.11 b/g bertujuan untuk menerapkan Huffman coding untuk kompresi citra dan mengetahui pengaruhnya dalam pengurangan bit informasi dalam proses transmisi. Perancangan sistem kompresi ini meliputi perancangan perangkat keras dan perangkat lunak, pada perangkat lunak skripsi ini menggunakan program aplikasi Borland Delphi 7 sebagai program untuk mengambil citra kemudian mengkompresi citra dan mendekompresi file hasil kompresi. Dari hasil perancangan dan penerapan Huffman coding untuk kompresi citra pada jaringan wifi 802.11 b/g diperoleh hasil pengujian bahwa Huffman coding dapat diterapkan dalam program kompresi dan dekompresi pada jaringan wifi 802.11 b/g. Kompresi menggunakan Huffman coding efektif digunakan untuk citra yang memiliki kombinasi warna yang rendah dengan hasil rasio kompresi sampai dengan 82,23%.

Kata kunci: kompresi, Huffman, citra, dekompresi.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jaringan komputer yang terus berkembang dengan pesat telah banyak membantu kebutuhan manusia akan pemenuhan informasi dan komunikasi dalam melakukan segala aktivitas sehari-hari. Kecenderungan teknologi jaringan *wireless* yang menjadi salah satu alternatif yang dipilih untuk pemenuhan informasi dan komunikasi yang memiliki keunggulan dalam hal praktis (mudah dalam mengakses) dan tidak rumit (konfigurasi) serta penggunaanya yang didukung dengan kecepatan akses yang cukup tinggi sampai 384 Mbps (3G).

Untuk mengakses informasi dengan kecepatan dan efisiensi yang tinggi dalam format ukuran data yang cukup besar seperti data multimedia (MPEG, AVI, JPEG, MP3 dan lain-lain) dibutuhkan adanya sebuah kompresi data. Agar kualitas data yang dikompresi tidak mengalami perubahan atau pengurangan data dan sesuai dengan data aslinya, maka kompresi yang digunakan adalah kompresi *lossless* (*lossless compression*). Kompresi *lossless* merupakan kompresi yang tetap mempertahankan kualitas data tanpa ada perubahan sedikitpun. Kompresi data yang digunakan adalah kompresi *lossless* (*lossless compression*) menggunakan *Huffman coding*.

Saat ini sudah ada skripsi yang berjudul "Perancangan Sistem Monitoring Perangkat Node B (BTS 3 G) Menggunakan SMS (*Short Message Service*) dan *Webcam*" yang sudah disusun oleh Choiriyah Indah S. (0410630022-63) dan telah diseminarkan pada tanggal 28 Januari 2009 dengan dosen pembimbing Ir. Endah Budi P.,MT dan Ali Mustofa ST.,MT. Pada Skripsi tersebut sistem monitoring perangkat menggunakan sms dan *webcam, webcam* akan melakukan *capture* secara *real time* dan menampilkan gambar hasil *capture* tersebut pada web secara *real time*.

Perbedaan skripsi tersebut dengan skripsi "Penerapan *Huffman Coding* untuk Kompresi Citra pada Jaringan Wifi b/g secara *Real Time*" terletak pada kompresi citra dengan menggunakan *Huffman coding* untuk pengurangan bit informasi dalam proses transmisi pada jaringan wifi 802.11 b/g. Pada skripsi ini akan dilakukan perancangan suatu sistem dengan menggunakan *webcam.* Dimana citra hasil *capture* akan dikompresi dengan menggunakan *Huffman coding* kemudian ditransmisikan melalui

jaringan wifi 802.11 b/g menggunakan *access point router* untuk bisa diterima pada *client.*

1.2 Rumusan Masalah

Berdasarkan latar belakang yang diuraikan di atas, maka rumusan masalah yang akan dibahas antara lain sebagai berikut:

1. Bagaimana cara mengaplikasikan kompresi citra menggunakan *Huffman coding* pada jaringan Wifi 802.11 b/g secara *real time.*
2. Bagaimana merancang sistem kompresi citra menggunakan *Huffman coding* yang diterapkan pada jaringan Wifi 802.11 b/g secara *real time.*
3. Bagaimana pengaruh *Huffman coding* dalam mengkompresi citra sehingga bit informasinya berkurang.
4. Bagaimana mengembalikan citra yang telah dikompresi kedalam bentuk aslinya.
5. Bagaimana penerapan kompresi citra menggunakan *Huffman coding* pada jaringan wifi 802.11 b/g.

1.3 Batasan Masalah

Mengacu pada permasalahan yang ada maka batasan masalah pada skripsi ini dibatasi pada:

1. Metode kompresi citra yang diterapkan menggunakan *Huffman coding*.
2. Penerapan dilakukan pada jaringan wifi dengan standar 802.11 b/g.
3. Menggunakan OS (*Operating System*) Windows XP.
4. Menggunakan *access point* yang mendukung wifi 802.11 b/g.
5. Menggunakan aplikasi *software* Borland Delphi 7.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah menerapkan *Huffman coding* untuk kompresi citra pada jaringan Wifi 802.11 b/g secara *real time* dan mengetahui pengaruhnya dalam pengurangan bit informasi dalam proses transmisi.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan skripsi ini sebagai berikut:

BAB I : Memuat latar belakang, rumusan masalah, batasan masalah, serta tujuan penulisan skripsi ini.

BAB II : Menjelaskan teori-teori yang mendukung dalam penerapan *Huffman coding* pada jaringan wifi 802.11b/g.

BAB III : Menjelaskan tentang metodologi penelitian yang digunakan untuk mengerjakan skripsi ini. Diataranya studi literatur terhadap dasar teori, pengumpulan data, perencanaan sistem.

BAB IV : Menjelaskan perancangan sistem yang meliputi perancangan perangkat keras, perancangan perangkat lunak, dan prinsip kerja sistem.

BAB V : Menjelaskan tentang penerapan dan pengujian sistem yang telah dibuat.

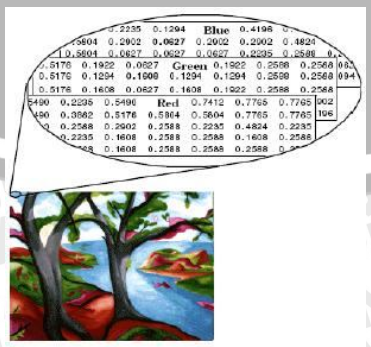
BAB VI : Memberikan kesimpulan dari perancangan sistem yang telah dibuat dan saran-saran yang diperlukan untuk pengembangan aplikasi selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Citra Digital

Citra atau gambar didefinisikan sebagai fungsi intensitas cahaya dua dimensi f(x,y) dimana x dan y menunjukkan koordinat spasial, dan nilai f pada suatu titik (x,y) sebanding dengan kecerahan (*brightness*) yang biasanya dinyatakan dalam tingkatan abu-abu (*grey level*) dari citra di titik tersebut. Citra digital merupakan citra dengan f (x,y) yang nilainya didigitalisasikan (dibuat diskrit) baik dalam koordinat spasial maupun dalam *grey level*-nya. Digitalisasi dari koordinat spasial citra disebut *grey level quantization*. Citra digital dapat dibayangkan sebagai suatu matriks dimana baris dan kolomnya menunjukkan *grey level* di titik tersebut. Elemen-elemen dari citra digital tersebut biasanya disebut dengan piksel yang merupakan singkatan dari *picture element*. Sebenarnya nilai fungsi pada tiap piksel tidak hanya ditentukan oleh parameter x maupun y, tetapi terdiri dari banyak variabel yang lain diantaranya adalah kedalaman (z), warna (λ), dan waktu (t).

Sebagai contoh, citra hitam-putih dengan 256 level artinya mempunyai skala abu-abu dari 0 sampai 255 atau [0, 255]. Yang dalam hal ini nilai intensitas 0 menyatakan hitam, nilai intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabu-abuan antara hitam dan putih. Citra RGB disebut juga citra *truecolor*. Citra RGB merupakan citra digital yang mengandung matriks data berukuran m x n x 3 yang merepresentasikan warna merah, hijau, dan biru untuk setiap pikselnya. Setiap warna dasar diberi rentang nilai. Untuk monitor komputer, nilai rentang paling kecil 0 dan paling besar 255. Warna dari tiap *pixel* ditentukan oleh kombinasi dari intensitas merah, hijau, dan biru.



Gambar 2. 1 Kombinasi intensitas warna merah, hijau dan biru
Sumber: http://www.ittelkom.ac.id

2.2 Kompresi Citra

Kompresi merupakan pengurangan ukuran suatu data (citra, teks ,video dan suara) menjadi ukuran yang lebih kecil dari aslinya. Pada prinsipnya secara umum kompresi citra adalah mengurangi duplikasi data di dalam dalam citra sehingga memori yang dibutuhkan untuk merepresentasikan citra menjadi lebih sedikit daripada representasi citra semula. Kompresi akan sangat menguntungkan manakala terdapat suatu data yang berukuran besar dan data didalamnya mengandung banyak pengulangan karakter. Pada skripsi ini, kompresi data yang yang diambil sebagai studi kasus adalah kompresi citra/gambar. Gambar-gambar yang kita dapatkan di berbagai situs internet pada umumnya merupakan hasil kompresi ke dalam format GIF atau JPEG. File video MPEG adalah hasil kompresi pula. Penyimpanan data berukuran besar pada *server* pun sering dilakukan melalui kompresi.

Adapun teknik kompresi citra yang sering digunakan dan diterapkan dalam hal kompresi adalah:

1. Kompresi *lossy* (*lossy compression*)

   Kompresi *lossy* merupakan kompresi yang dilakukan dengan cara mengeliminasi beberapa data dari suatu berkas. Namun data yang dieliminasikan biasanya adalah data yang kurang diperhatikan atau di luar jangkauan manusia, sehingga pengeliminasian data tersebut kemungkinan besar tidak akan mempengaruhi manusia yang berinteraksi dengan berkas tersebut. Keuntungan dari kompresi ini adalah rasio kompresi (perbandingan antara ukuran berkas yang yang telah dikompresi dengan berkas yang belum dikopresi) cukup tinggi. Contohnya pada pengkompresian berkas audio, kompresi *lossy* akan mengeliminasi data dari berkas audio yang memiliki frekuesi sangat tinggi/rendah yang berada di luar jangkauan manusia. Beberapa jenis data yang biasanya masih dapat mentoleransi kompresi *lossy* adalah gambar, audio dan video.

2. Kompresi *lossless* (*lossles compression*)

   Kompresi *lossless* berbeda dengan kompresi *lossy*, pada kompresi *lossless*, tidak terdapat perubahan data ketika mendekompresi berkas yang telah dikompresi dengan kompresi *lossless* ini. Kompresi ini biasanya diimplementasikan pada kompresi berkas teks, seperti program komputer (berkas zip, rar, gzip dan lain-lain). Sebagai ilustrasi bahwa proses kompresi bisa memperkecil ukuran, 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-masng *byte* merepresentasikan warna merah, hijau, dan biru. Sehingga gambar 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*) gambar. Gambar tersebut tidak bisa disimpan ke dalam disket yang berukuran 1,4 MB. Selain itu, pengiriman gambar berukuran 9 MB memerlukan waktu lebih lama. Untuk koneksi internet *dial-up* (56 kbps), pengiriman gambar berukuran 9 MB memerlukan waktu 21 menit. Untuk itulah diperlukan kompresi gambar sehingga ukuran gambar gambar tersebut menjadi lebih kecil dan waktu pengiriman gambar menjadi lebih cepat.

Implementasi proses kompresi pada data citra akan menghasilkan berbagai macam ekstensi file, diantaranya seperti yang tercantum pada tabel berikut:

Tabel 2.1 Macam-macam format file citra

Ekstensi file	Nama	Keterangan
bmp	Windows Bitmap	Biasanya digunakan oleh aplikasi dan sistem operasi Microsoft Windows.
gif	Graphics Interchange Format	Gif biasanya digunakan di website. Format gif mendukung gambar bergerak/animasi. Namun format gif hanya mendukung 255 warna tiap *frame*. Format gif juga mendukung gambar transparan.
Jpeg. jpg	Joint Photographic Expert Group	JPEG biasanya digunakan untuk foto atau gambar di website. JPEG2000 bisa bervariasi tergantung *setting* kompresi yang digunakan. Kompresi JPEG pada umumnya berbasis DCT (*Discrete Cosine Transform*)
Jpg2. jp2. j2k	Joint Photographic Expert Group 2000	Merupakan pengembangan dari JPEG yang berbasis transformasi wavelet.
Pbm	Portable Bitmap Format	Merupakan format gambar hitam putih yang sederhana. PBM memerlukan 1 bit tiap piksel.
ppm	Portable Pixmap Format	Merupakan format gambar berwarna yang sederhana.
png	Portable Network Graphichs	PNG adalah format gambar dengan kedalam bit berkisar antara 1 sampai dengan 32. PNG didesain untuk menggantikan format GIF untuk diimplementasikan di website.

Sumber: http://en.wikipedia.org/wiki/Graphics_file_format

2.3 Huffman Coding

Algoritma Huffman yang dibuat oleh seorang mahasiswa MIT bernama David Huffman, merupakan salah satu metode paling lama dan paling terkenal dalam kompresi teks. Algoritma Huffman menggunakan prinsip pengkodean yang mirip dengan kode Morse, yaitu tiap karakter (simbol) dikodekan hanya dengan rangkaian beberapa bit, dimana karakter yang sering muncul dikodekan dengan rangkaian bit yang pendek dan karakter yang jarang muncul dikodekan dengan rangkaian bit yang lebih panjang.

Sebagai contoh, dalam kode ASCII *string* 7 huruf "ABACCDA" membutuhkan representasi 7 × 8 bit = 56 bit (7 *byte*), dengan rincian sebagai berikut:

01000001	01000010	01000001	01000011	01000011	01000100	01000001
A	B	A	C	C	D	A

Untuk mengurangi jumlah bit yang dibutuhkan, panjang kode untuk tiap karakter dapat dipersingkat, terutama untuk karakter yang frekuensi kemunculannya besar. Pada *string* di atas, frekuensi kemunculan A = 3, B = 1, C = 2, dan D = 1, sehingga dengan menggunakan algoritma di atas diperoleh kode Huffman seperti pada tabel berikut:

Tabel 2.2 Kode Huffman untuk "ABACCDA"

Simbol	Frekuensi	Peluang	Kode Huffman
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

Sumber: Perencanaan

Dengan menggunakan kode Huffman ini, *string* "ABACCDA" direpresentasikan menjadi rangkaian bit : 0 110 0 10 10 111 0. Jadi, jumlah bit yang dibutuhkan hanya 13 bit. Dari Tabel 2.2 tampak bahwa kode untuk sebuah simbol/karakter tidak boleh menjadi awalan dari kode simbol yang lain guna menghindari keraguan (ambiguitas) dalam proses dekompresi atau *decoding*.

Karena tiap kode Huffman yang dihasilkan unik, maka proses dekompresi dapat dilakukan dengan mudah. Contoh: saat membaca kode bit pertama dalam rangkaian bit "011001010110", yaitu bit "0", dapat langsung disimpulkan bahwa kode bit "0" merupakan pemetaan dari simbol "A". Kemudian baca kode bit selanjutnya, yaitu bit

"1". Tidak ada kode Huffman "1", lalu baca kode bit selanjutnya, sehingga menjadi "11". Tidak ada juga kode Huffman "11", lalu baca lagi kode bit berikutnya, sehingga menjadi "110". Rangkaian kode bit "110" adalah pemetaan dari simbol "B".

Metode Huffman yang diterapkan dalam penelitian ini adalah tipe statik, dimana dilakukan dua kali pembacaan (*two-pass*) terhadap berkas yang akan dikompresi, pertama untuk menghitung frekuensi kemunculan karakter dalam pembentukan pohon Huffman, dan kedua untuk mengkodekan simbol dalam kode Huffman.

2.3.1 Pembentukan Pohon Huffman

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:

a. Baca semua karakter di dalam teks untuk menghitung frekuensi kemunculan setiap karakter. Setiap karakter penyusun teks dinyatakan sebagai pohon bersimpul tunggal. Setiap simpul di-*assign* degan frekuensi kemunculan karakter tersebut.
b. Terapkan strategi algoritma *greedy* sebagai berikut:
   Gabungkan daun buah pohon yang mempunyai frekuensi terkecil pada sebuah akar. Setelah digabungkan akar tersebut akan mempunyai frekuensi yang merupakan jumlah dari frekeunsi dua buah pohon-pohon penyusunnya.
c. Ulangi langkah 2 samapai hanya tersisa satu buah pohon Huffman. Agar pemilihan dua pohon yang digabungkan berlangsung cepat, maka semua yang ada selalu terurut menaik berdasarkan frekuensi.

Sebagai contoh, dalam kode ASCII *string* 7 huruf "ABACCDA" membutuhkan representasi 7 × 8 bit = 56 bit (7 *byte*), dengan rincian sebagai berikut:

A = 01000001
B = 01000010
A = 01000001

C = 01000011

C = 01000011

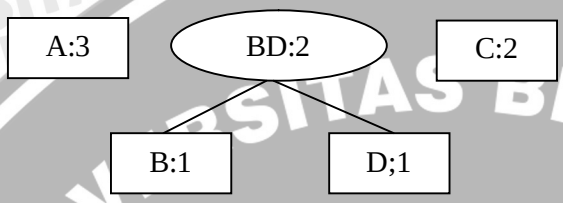
D = 01000100

A = 01000001

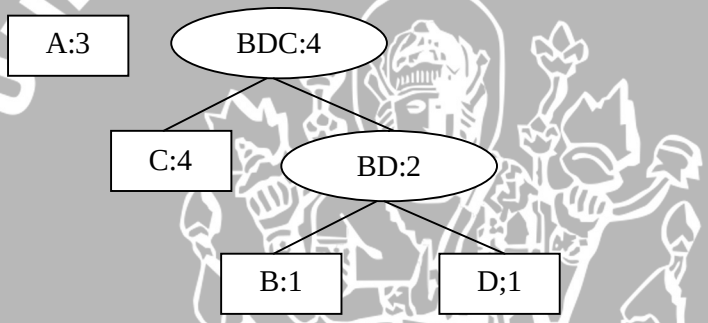
Pada *string* di atas, frekuensi kemunculan A = 3, B = 1, C = 2, dan D = 1,

1.

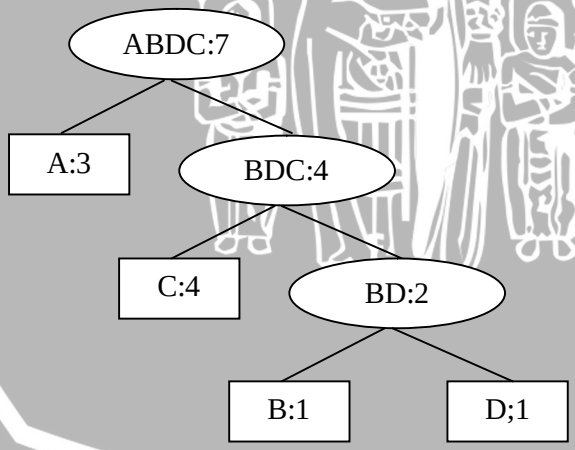
2.



3.



4.



Gambar 2.2 Pohon Huffman untuk karakter “ABACCDA”
Sumber: Perencanaan

2.3.2 Proses Encoding

Encoding adalah cara menyusun *string* biner dari teks yang ada. Proses *encoding* untuk satu karakter dimulai dengan membuat pohon Huffman terlebih dahulu. Setelah

itu, kode untuk satu karakter dibuat dengan menyusun nama *string* biner yang dibaca dari akar sampai ke daun pohon Huffman. Langkah-langkah untuk mengenkoding suatu *string* biner adalah sebagai berikut:

1. Tentukan karakter yang di-*encoding*.
2. Mulai dari akar, baca setiap bit yang ada pada cabang yang bersesuaian sampai ketemu daun dimana karakter itu berada.
3. Ulangi langkah 2 sampai seluruh karakter di-*encoding*.

Sebagai contoh kita dapat melihat tabel dibawah ini, yang merupakan hasil *encoding* untuk pohon Huffman pada gambar 2.2.

Tabel 2.3 Kode Huffman untuk karakter "ABCD"

Karakter	***String* biner Huffman**
A	0
B	110
C	10
D	111

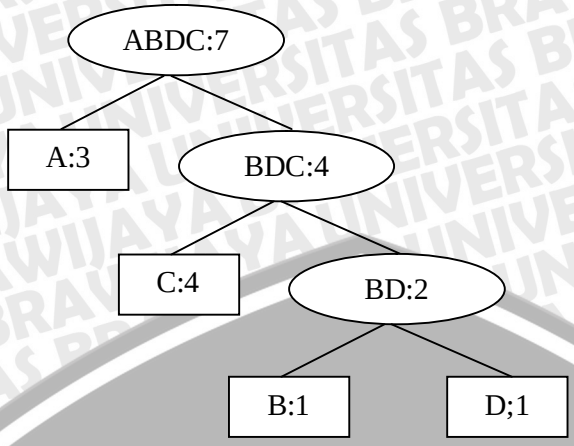
Sumber : Perencanaan

2.3.3 Proses Decoding

Decoding merupakan kebalikan dari *encoding*. *Decoding* berarti menyusun kembali data dari *string* biner menjadi sebuah karakter kembali. *Decoding* dapat dilakukan dengan dua cara, yang pertama dengan menggunakan pohon Huffman dan yang kedua dengan menggunakan tabel kode Huffman. Langkah-langkah men-*decoding* suatu *string* biner dengan menggunakan pohon Huffman adalah sebagai berikut :

1. Baca sebuah bit dari *string* biner.
2. Mulai dari akar.
3. 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 daun.
5. Ulangi dari langkah 1 sampai semua bit di dalam *string* habis.

Sebagai contoh kita akan men-decoding *string* biner yang bernilai "111".



Gambar 2.3 Proses decoding dengan menggunakan pohon Huffman
Sumber: Perencanaan

setelah kita telusuri dari akar, maka kita akan menemukan bahwa *string* yang mempunyai kode Huffman "111" adalah karakter D. Cara yang kedua adalah dengan menggunakan tabel kode Huffman. Sebagai contoh kita akan menggunakan kode Huffman pada Tabel 2.3 untuk merepresentasikan *string* "ABACCDA". Dengan menggunakan Tabel 2.3 *string* tersebut akan direpresentasikan menjadi rangkaian bit : 0 110 0 10 10 111 0. Jadi, jumlah bit yang dibutuhkan hanya 13 bit. Dari Tabel 2.3 tampak bahwa kode untuk sebuah simbol/karakter tidak boleh menjadi awalan dari kode simbol yang lain guna menghindari keraguan (ambiguitas) dalam proses dekompresi atau *decoding*. Karena tiap kode Huffman yang dihasilkan unik, maka proses *decoding* dapat dilakukan dengan mudah. Contoh: saat membaca kode bit pertama dalam rangkaian bit

"011001010110", yaitu bit "0", dapat langsung disimpulkan bahwa kode bit "0" merupakan pemetaan dari simbol "A". Kemudian baca kode bit selanjutnya, yaitu bit "1". Tidak ada kode Huffman "1", lalu baca kode bit selanjutnya, sehingga menjadi "11". Tidak ada juga kode Huffman "11", lalu baca lagi kode bit berikutnya, sehingga menjadi "110". Rangkaian kode bit "110" adalah pemetaan dari simbol "B".

2.4 Kompresi Huffman pada Citra

Metode kompresi Huffman menggunakan prinsip bahwa nilai (derajat keabuan) 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.

Adapun cara yang dilakukan untuk melakukan kompresi dengan menggunakan Huffman coding pada citra 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 piksel di dalam gambar (n)). Setiap frekuensi nilai keabuan dinyatakan sebagai pohon bersimpul tunggal. Setiap di-*assign* dengan frekuensi kemunculan nilai keabuan tersebut.
2. Gabung dua buah pohon yang mempunyai frekuensi kemunculan paling kecil pada sebuah akar. Akar mempunyai frekuensi yang merupakan jumlah dari frekuensi dua buah pohon penyusunnya.
3. Ulangi langkah 2 sampai tersisa hanya satu buah pohon biner.
4. Agar pemilihan dua pohon pohon yang akan digabungkan berlangsung cepat, maka semua pohon yang ada selalu terurut menaik berdasarkan frekuensi. Kemudian beri label setiap sisi pada pohon biner. Sisi kiri dilabeli dengan 0 dan sisi kanan dilabeli dengan 1.
5. Simpul-simpul daun pada pohon biner menyatakan nilai keabuan yang terdapat di dalam citra semula. Untuk mengkodekan setiap piksel di dalam citra, telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar ke daun menyatakan kode Huffman untuk derajat keabuan yang bersesuaian.

Setiap kode Huffman merupakan kode prefix, yang artinya tidak ada kode biner atau nilai keabuan yang merupakan awalan bagi kode biner derajat keabuan yang lain. Dengan cara ini tidak ada ambiguitas pada proses dekompresi citra.

Misalkan terdapat citra yang berukuran 64 x 64 dengan 8 derajat keabuan (k) dan jumlah seluruh piksel (n) = 64 x 64 = 4096.

k	***n_k***	***P (k) = n_k/n***
0	790	0,19
1	1023	0,25
2	850	0,21
3	656	0,16
4	329	0,08
5	245	0,06

Lanjutan tabel derajat keabuan

6	122	0,03
7	81	0,02

Proses pembentukan pohon huffman yang terbetuk dapat dilihat pada gambar 2.4. Setiap simpul di dalam pohon berisi pasangan nilai a:b, yang dalam hal ini a menyatakan nilai keabuan dan menyatakan peluang kemunculan nilai keabuan di dalam citra. Dari pohon Huffman diperoleh kode untuk setiap derajat keabuan sebagai berikut:

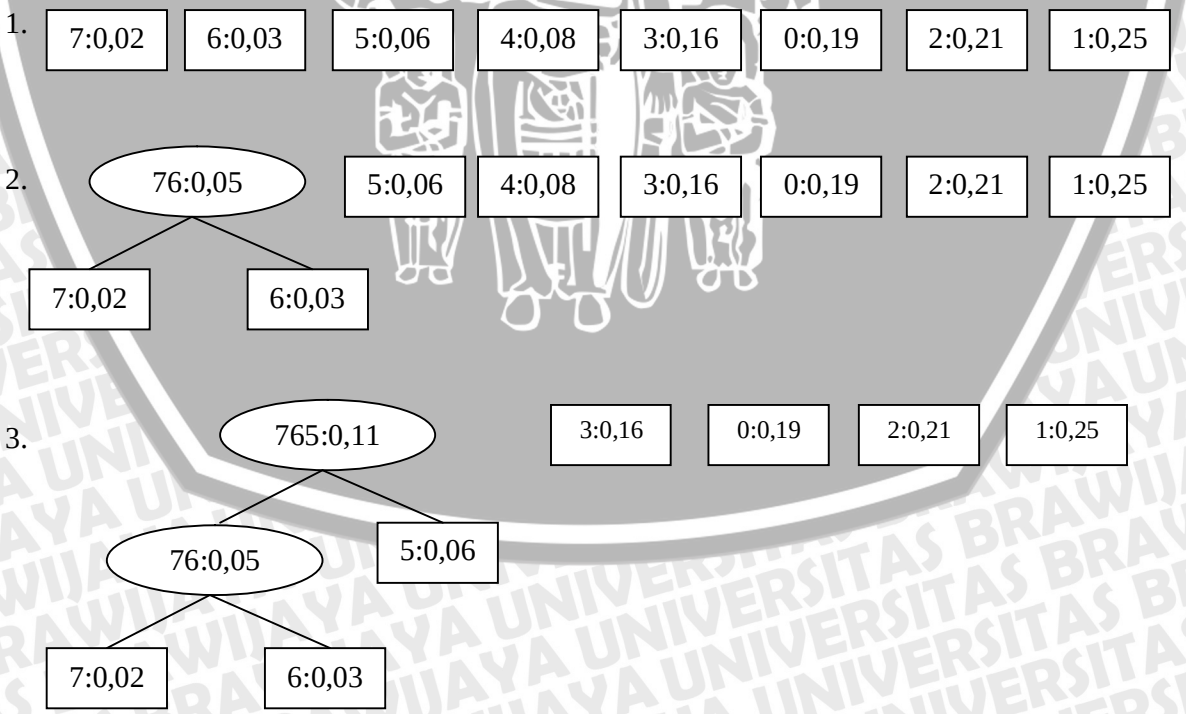
0 = 00 2 = 01 4 = 1110 6 = 111101

1= 10 3 = 110 5 = 11111 7 = 111100

Ukuran citra sebelum kompresi (1 derajat keabuan = 3 bit) adalah 4096 x 3 bit = 12288 bit sedangkan ukuran citra setelah kompresi adalah

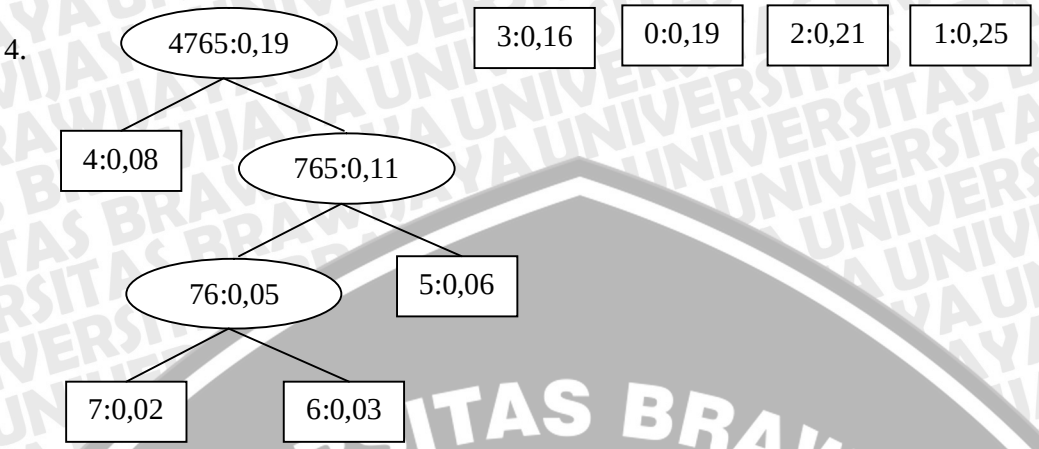
(790 x 2 bit) + (1023 x 2 bit) + (850 x 2 bit) + (656 x 3 bit) + (329 x 4 bit) + (245 x 5 bit) + (122 x 6 bit) + (81 x 6) bit = 11053 bit

Jadi, kebutuhan memori telah dikurang dari 12288 bit menjadi 11053 bit. Jelas ini tidak banyak menghemat, tetapi jika 256 nilai keabuan yang digunakan (dibandingkan dengan 8 derajat keabuan seperti pada contoh) penghematan memori akan lebih besar.

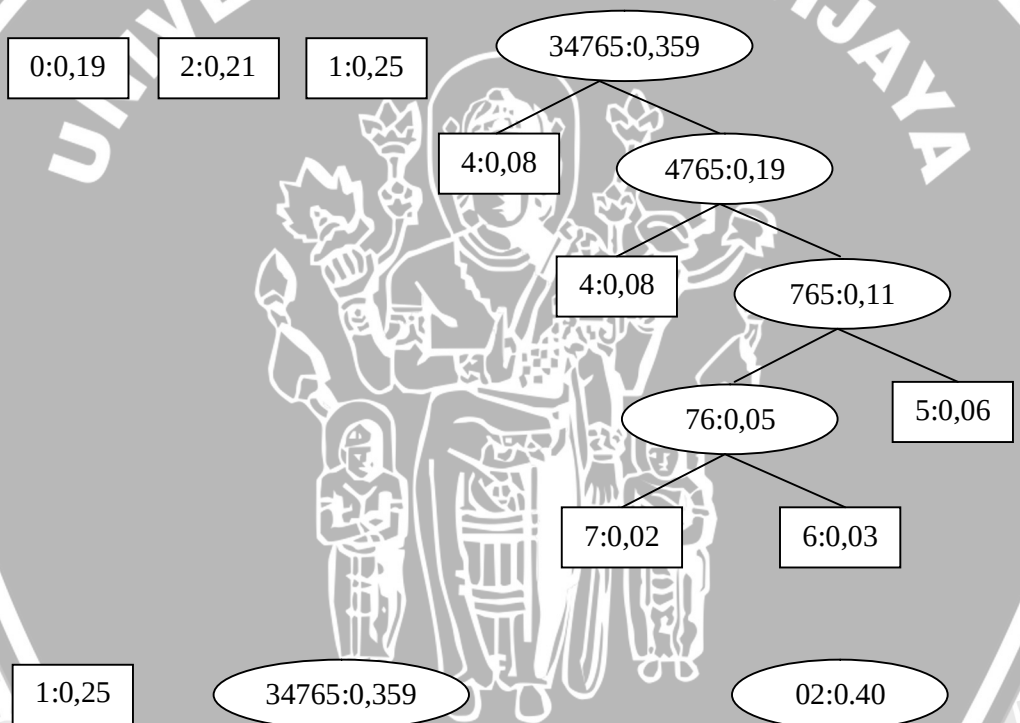


Lanjutan gambar 2.4

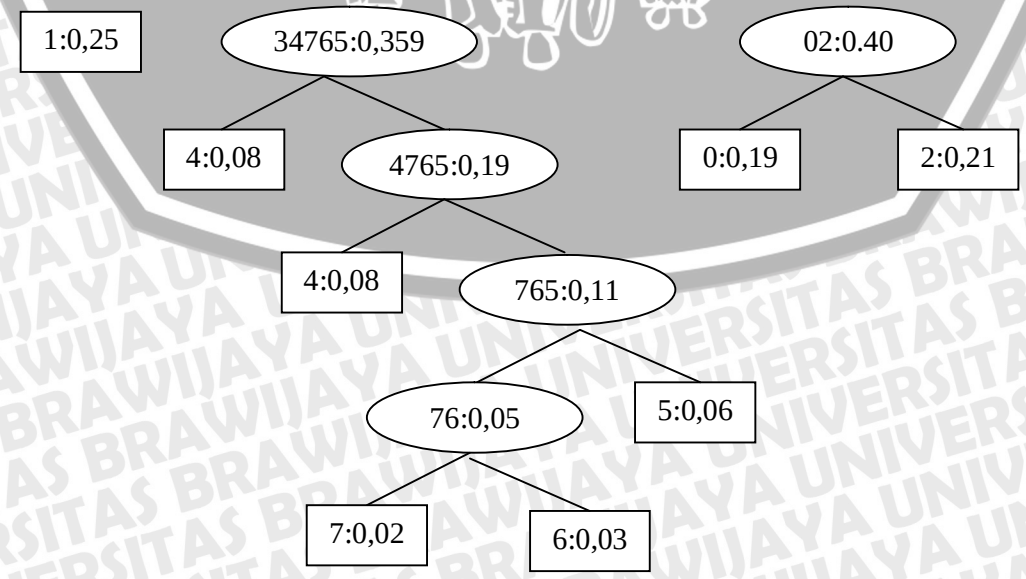
4.



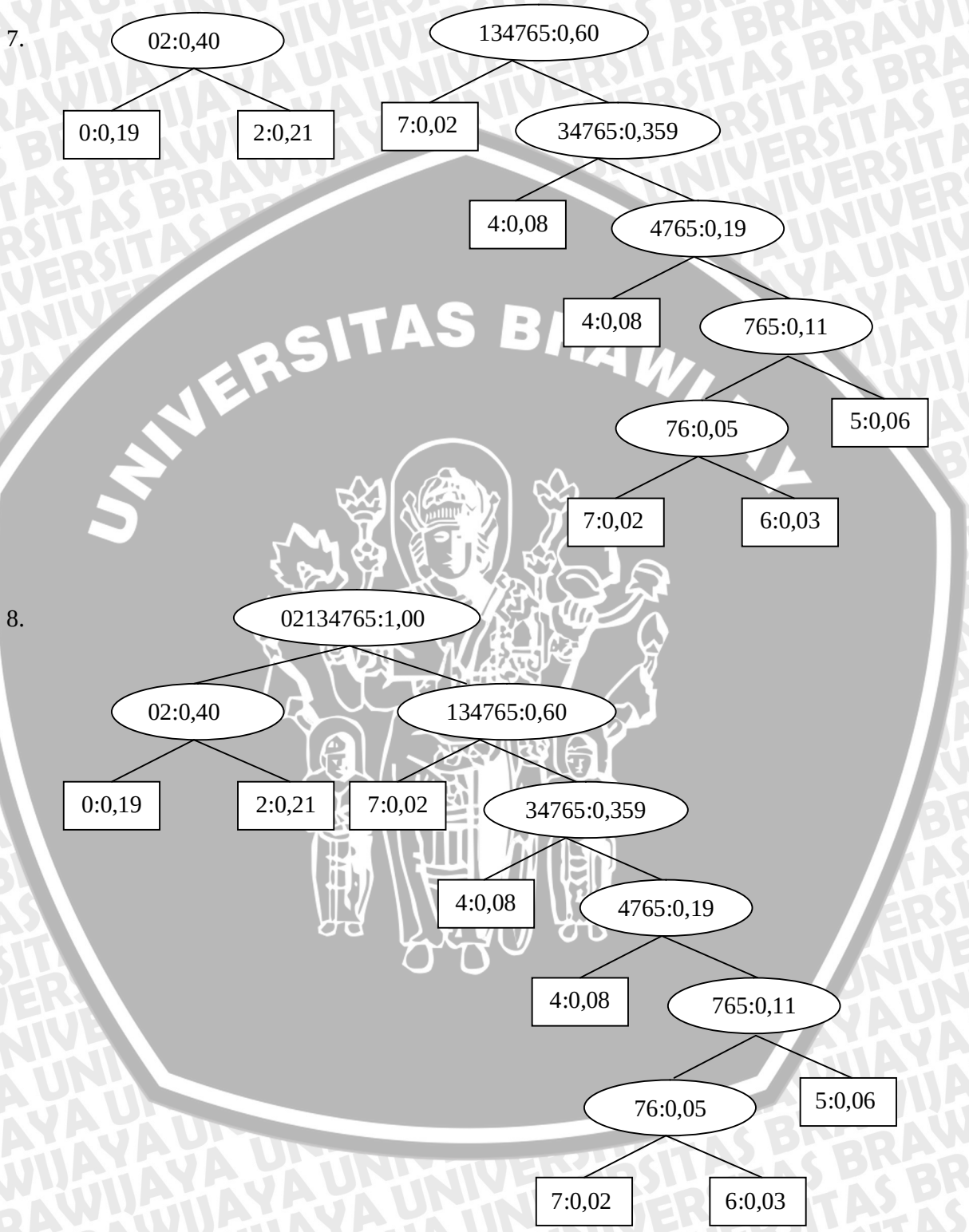
5.



6.



Lanjutan gambar 2.4



Gambar 2.4 Tahapan pembentukan pohon Huffman
Sumber: Perencanaan

Misalkan sebuah citra dinyatakan dalam bentuk matriks berikut:

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 200 & 200 & 200 & 100 \\ 250 & 100 & 200 & 100 & 250 \end{bmatrix}$$

Bila matriks ini mewakili sebuah citra gray-level berukuran 5x3 piksel, maka nilai elemen matriks (piksel) menyatakan tingkat keabuan citra. Tetapi bila matriks ini mewakili sebuah citra berwarna, maka nilai elemen matriks menyatakan warna. Setiap piksel dalam sebuah citra yang dikodekan dalam 8 bit, berarti citra tersebut memiliki 256 tingkat keabuan atau memiliki 256 warna.

Dengan mengambil contoh citra di atas dan dengan menggunakan metode Huffman, dikembangkan sebuah algoritma untuk mengkompres data citra sebagai berikut:

a. Buat data citra yang berupa matriks tersebut menjadi vektor, sehingga didapat vektor [100,100,100,100,100,100,200,200,200,100,250,100,200,100,250]. Besar data citra adalah 15 byte.
b. Baca vektor tersebut dan tentukan nilai warna yang ada serta frekuensi kemunculannya. Hasilnya adalah 100 = 9, 200 = 4 dan 250 = 2.
c. Urutkan warna dari yang frekuensinya terkecil ke frekuensi yang terbesar. 250,200,100.
d. Membuat pohon biner berdasarkan urutan warna.
e. Mengganti data warna dengan kode bit berdasarkan pohon biner:
   100 = “1” 200 = “01” 250 = “00”
f. Mengganti data citra dengan kode bit menjadi: 111111010101100101100.
g. Menyimpan lebar citra, tinggi citra, kode bit untuk warna yang terbesar frekuensi munculnya (00 untuk contoh diatas), data warna yang terdapat di dalam citra dan data citra sudah dikodekan ke dalam file hasil kompresi.

Untuk mengembalikan data citra terkompresi menjadi data citra aslinya, diperlukan suatu algoritma dekompresi yang merupakan kebalikan dari algoritma kompresi. Berikut ini adalah langkah-langkah untuk mengembalikan data citra yang sudah dikodekan menjadi data citra semula adalah sebagai berikut:

a. Baca file hasil kompresi dan data-datanya dimasukkan ke variabel yang sesuai yaitu variabel ukuran citra, variabel kode bit data warna terakhir, variabel warna dan variabel data dan kode.
b. Baca data kode bit per bit dari kiri ke kanan dan dicocokkan dengan data warna yang didapat. Bit hasil kompresi 111111010101100101100. Bit pertama = “1”,

karena nilainya “1” maka bit ini mewakili warna pertama dalam listing variabel warna yaitu warna 100. Kemudian bit berikutnya juga “1” berarti mewakili warna 100, dan seterusnya hingga bit ke 7 bernilai “0”. Karena bit ini bernilai “0” maka perlu dibandingkan dengan nilai kode biner data warna terakhir yaitu “00” dan karena “0” tidak sama dengan “00” maka dilakukan pembacaan berikutnya yaitu bit “1”. Karena bit berikutnya yang terbaca = “1” maka pembacaan kode selanjutnya dihentikan. Dengan demikian kode bit yang terbaca menjadi “01”, kode ini memiliki jumlah bit sebesar 2 bit dan dengan demikian kode ini mewakili warna pada urutan kedua dalam listing warna. Demikian seterusnya dilakukan konversi hingga data terakhir. Dari contoh diatas, hasil rekonstruksi menjadi:

[100 100 100 100 100 100 200 200 200 100 250 100 200 100 250].

c. Rekostruksi citra 2D dengan menggunakan data ukuran citra 5x3 berarti data piksel berbentuk 1D dipenggal menjadi 3 baris dan setiap barisnya berisi 5 piksel. Hasilnya menjadi:

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 200 & 200 & 200 & 100 \\ 250 & 100 & 200 & 100 & 250 \end{bmatrix}$$

2.5 Standard Wifi

Wifi merupakan kependekan dari *wireless fidelity*, memiliki pengertian yaitu sekumpulan standard yang digunakan untuk jaringan lokal nirkabel (*Wireless Local Area Networks* - WLAN) yang didasari pada spesifikasi IEEE 802.11. Wifi dirancang berdasarkan spesifikasi IEEE 802.11, sekarang ini ada empat variasi dari 802.11, yaitu: 802.11a, 802.11b, 802.11g, dan 802.11n. Spesifikasi Wifi yang banyak digunakan di Indonesia adalah 802.11b/g yang beroperasi pada frekuensi 2,4 GHz.

Tabel 2.4 Spesifikasi 802.11

	802.11	802.11a	802.11b	802.11g
Dikeluarkan	Juli 1997	September 1999	September 1999	2002
Bandwidth	83.5 MHz	300 MHz	83.5 MHz	83.5 MHz
Frekuensi	2.4 – 2.4835 GHz DSSS, FHSS	5.15-5.35 GHz OFDM 5.725-5.825 GHz OFDM	2.4-2.4835 GHz DSSS	2.4-2.4835 GHz DSSS, OFDM

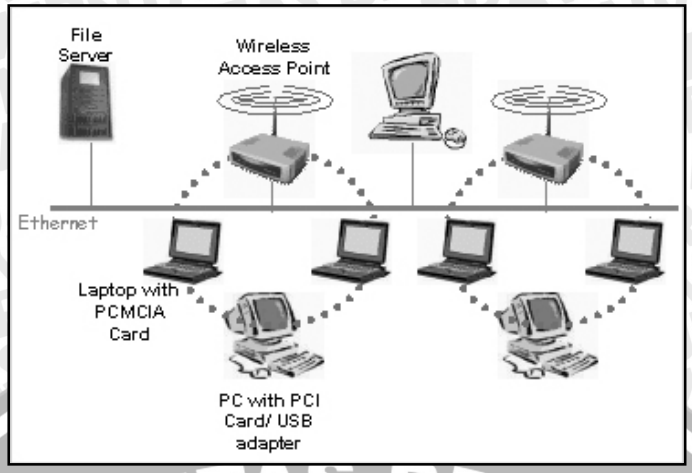
Lanjutan tabel 2.4

Jumlah kanal yang tidak *overlapping*	3 (indoor/outdoor)	4 (indoor/outdoor)	3 (indoor/outdoor)	3 (indoor/outdoor)
Data rate	2 Mbps, 1 Mbps	54, 48, 36, 24, 18, 12, 9, 6 Mbps	11, 5.5 , 2, 1 Mbps	54, 36, 33, 24, 22, 12, 11, 9, 6, 5.5, 2, 1 Mbps
Tipe modulasi	DQPSK (2 Mbps DSSS) DBPSK (1 Mbps DSSS) 4GFSK (2 Mbps FHSS) 2GFSK (1 Mbps FHSS)	BPSK (6,9 Mbps) QPSK (12, 18 Mbps) 16-QAM (24,36 Mbps) 64-QAM (48, 54 Mbps)	DQPSK/CCK (11, 5,5 Mbps) DQPSK (2 Mbps) DBPSK (1 Mbps)	OFDM/CCK (6,9,12,18,24,36, 48,54) OFDM (6,9,12,18,24,36, 48,54 Mbps) DQPSK/CCK (22,33,11,5.5 Mbps) DQPSK (2 Mbps) DBPSK (1 Mbps)

Sumber : http://www.wlana.org

2.5.1 Arsitektur Jaringan Wifi

Setiap perangkat berkomunikasi (mengirim dan menerima data) satu sama lain melalui titik akses, menghubungkan perangkat *wireless* dengan jaringan kabel (LAN). Saat AP menerima data, AP akan mengirimkan kembali data berupa sinyal radio ini ke perangkat *wireless* yang berada dalam area. Struktur kerjanya sama dengan *Base Station* (BS) pada jaringan selular.



Gambar 2.5 Model konfigurasi Infrastruktur
Sumber : http//:Net4India Think Net, Think Net4India.htm

2.6 Access Point

Pada jaringan *wireless, access point* mutlak diperlukan untuk memberikan layanan pada *client wireless. access point* pada dasarnya berfungsi sebagai *bridge* antara jaringan wireless dengan jaringan kabel LAN melalui konektor UTP RJ-45 yang umumnya tersedia dibagian belakang *access point.* Dengan menghubungkan sebuah *access point* dengan jaringan kabel, *wireless client* secara otomatis akan terhubung ke dalam jaringan kabel. Dengan cara ini, *wireless client* bisa tetap berhubungan dengan komputer lain yang masih menggunakan kabel, bisa saling berbagi file, berbagi koneksi internet dan menggunakan *resource* jaringan yang lain.

Gambar 2.6 Access point router WRT54GL
Sumber : www.linksys.com

2.7 Webcam

Webcam (singkatan dari *web camera*) adalah sebutan bagi kamera *real-time* yang gambarnya bisa diakses atau dilihat melalui *World Wide Web*, program *instant messaging*, atau aplikasi *video call*. Istilah "webcam" juga merujuk kepada jenis kamera yang digunakan untuk keperluan ini. Ada berbagai macam merek webcam, diantaranya LogiTech, SunFlowwer, dan sebagainya. Webcam biasanya berresolusi sebesar 352 x 288/640 x 480 piksel. Namun ada yang kualitasnya hingga 1 Megapiksel. Sekarang hampir semua kamera digital dan *handphone* bisa dijadikan sebagai kamera web (webcam).

Gambar 2. 7 Webcam
Sumber: http://geniousgeneration.blogspot.com

Web camera memiliki fitur-fitur dan setting yang bermacam-macam, diantaranya adalah:

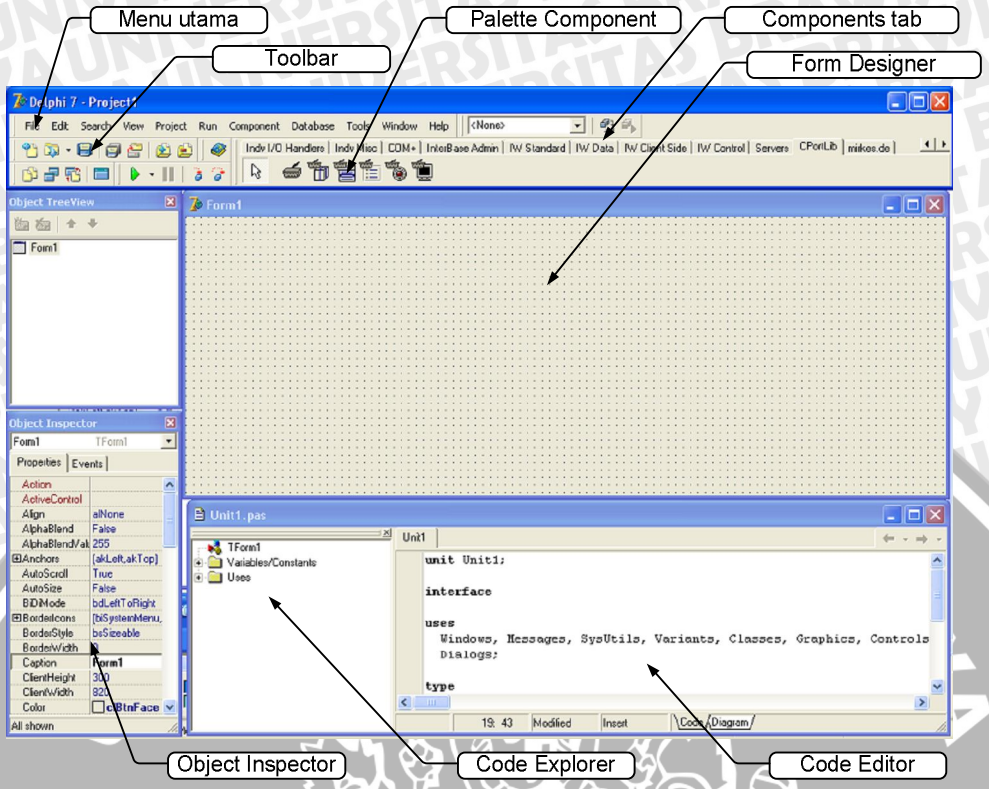
1. *Motion sensing* – web camera akan mengambil gambar ketika kamera mendeteksi gerakan.
2. *Image archiving* – pengguna dapat membuat sebuah *archive* yang menyimpan semua gambar dari web camera atau hanya gambar-gambar tertentu saat interval pre-set.
3. *Video messaging* – beberapa program messaging mendukung fitur ini.
4. *Advanced connections* – menyambungkan perangkat *home theater* ke *web camera* dengan kabel maupun nirkabel.
5. *Automotion* – kamera robotik yang memungkinkan pengambilan gambar secara pan atau tilt dan setting program pengambilan *frame* berdasarkan posisi kamera.
6. *Streaming media* – aplikasi profesional, *setup web camera* dapat menggunakan kompresi MPEG4 untuk mendapatkan *streaming* audio dan video yang sesungguhnya.

7. *Custom coding* – mengimport kode komputer pengguna untuk memberitahu *web camera* apa yang harus dilakukan (misalnya *automatically refresh*).
8. *AutoCam* – memungkinkan pengguna membuat *web page* untuk web cameranya secara gratis di *server* perusahaan pembuat web camera.

2.8 *Software* Borland Delphi

Delphi adalah perangkat lunak untuk menyusun program aplikasi yang berdasarkan pada bahasa pemrograman pascal dan bekerja dalam lingkungan sistem operasi windows. Dengan delphi penyusunan program aplikasi akan lebih mudah karena delphi menggunakan komponen-komponen yang akan menghemat penulisan program. Kelebihan delphi adalah dalam hal kecepatan proses kompilasi program. Delphi menyediakan *Integrated Development Environtment* (IDE) untuk memudahkan perancangan dengan dua cara, tampilan *secara visual (object)* dan penulisan kode. Secara ringkas, *object* adalah suatu komponen yang mempunyai bentuk fisik dan biasanya dapat dilihat (visual). *Object* biasanya dipakai untuk melakukan tugas-tugas tertentu dan mempunyai batasan-batasan tertentu. Sedangkan bahasa pemrograman secara singkat dapat disebut sebagai sekumpulan teks yang mempunyai arti tertentu dan disusun dengan aturan tertentu serta untuk menjalankan tugas tertentu. Delphi menggunakan struktur bahasa pemrograman *Object* Pascal yang sudah sangat dikenal.

Delphi dapat digunakan untuk membuat berbagai jenis program, diantaranya aplikasi *execute (.exe)*, ActiveX, WebService, dan lain-lain. IDE Delphi menyediakan berbagai jendela yang akan sering dilibatkan dalam pengembangan aplikasi, antara lain menu utama, *SpeedBar*, Jendela Form, *Object Inspector*, dan *Component Palette*. Gambar 2.8 menunjukkan tampilan IDE Delphi. Tampilan dari delphi ditunjukkan pada gambar berikut ini.



Gambar 2.8 IDE *Software* Borland Delphi
Sumber: MADCOMS (2003:2)

Menu-menu yang disediakan terebut antara lain: File, Edit, Search, View, Project, Run, Component, Database, Tool, Windows, dan Help. Isi dari setiap menu dan kegunaan dari menu-menu tersebut antara lain:

1. Pada file menu terdapat perintah untuk:

Tabel 2.5 Kegunaan menu File pada Program Borland Delphi 7

No	Menu	Kegunaan
1	New	Untuk membuat objek baru dan objek ini dapat berupa aplikasi, modul, form, file DLL dan lain-lain
2.	New Application	Untuk membuat aplikasi baru yang berisi form kosong
3.	New Form	Untuk membuat form baru
4.	New Data Module	Untuk membuat modul baru
5.	New Unit	Untuk membuka unit baru
6.	Open	Untuk membuka file yang berupa file project, form, unit atau text file
7.	Open Project	Untuk membuka project yang pernah dibuat
8.	Reopen	Untuk membuka project yang pernah dibuat
9.	Save	Untuk menyimpan file yang sedang dibuka
10.	Save As	Untuk menyimpan file dengan nama baru
11.	Save Project As	Untuk menyimpan project
12.	Save All	Untuk menyimpan semua file yang sedang dibuka
13.	Close	Untuk menutup file yang sedang dibuka
14.	Close All	Untuk menutup semua file

Lanjutan tabel 2.5

15.	Print	Untuk mencetak file yang sedang dibuka
16.	Exit	Keluar

Sumber: Suprapto, 2008:2

2. Menu Project berisi:

Tabel 2.6 Kegunaan menu Project pada program Borland Delphi 7

No	Menu	Kegunaan
1.	Add to Project	Memungkinkan untuk menambahkan file ke project
2.	Remove to Project	Untuk menghapus file/unit/form dari project
3.	Add to Repository	Untuk memudahkan menambah project ke object repository
4.	Compile Unit	Untuk mengkompilasi file unit
5.	Make	Untuk mengkompilasi file yang baru/telah diubah
6.	Build All Project	Untuk mengkompilasi semua file yang terkait dengan project

Sumber: Suprapto, 2008:2

3. Menu Run berisi:

Tabel 2.7 Kegunaan menu Run pada program Borland Delphi 7

No	Menu	Kegunaan
1.	Run	Untuk mengkompilasi dan mengeksekusi aplikasi yang dibuat
2.	Parameters	Mendefinisikan parameter pada aplikasi yang dibuat
3.	Step over	Mengeksekusi program secara baris per baris yang ada pada suatu unit
4.	Trace info	Mengeksekusi program secara baris per baris secara langsung yang ada pada suatu unit
5.	Program Pause	Menghentikan sementara eksekusi

Sumber: Suprapto, 2008:2

2.9 *Throughput*

Throughput didefinisikan sebagai jumlah total *byte* yang diterima di sisi penerima dengan baik. Jika *buffer* telah penuh, otomatis penerima tidak dapat menerima paket data yang baru sampai, sampai paket data yang ada di *buffer* tersebut diantarkan ke tujuan selanjutnya.

Setiap stasiun pengirim dan penerima mengetahui besarnya bit data yang dikirim dan ukuran *buffer*-nya masing-masing. Untuk menghitung *throughput end-to-end* pada jaringan ini dapat diperoleh dengan rumus: [Schwartz, 1987:129]

$$\lambda = \frac{1}{t_v} = \frac{(1-p)}{t_1[1+(\alpha-1)p]} \qquad (2.1)$$

dengan :

λ = *throughput* (paket/detik)

t_v = rata-rata waktu transmisi sebuah paket (detik/paket)

t_I = waktu transmisi sebuah paket data/frame (detik)

p = probabilitas frame error pada jaringan TCP/IP

α = konstanta = $1 + (t_{out} / t_I)$

$t_{out} = 2t_{prop} + t_I$

2.10 *Packet Loss*

Adalah jumlah paket yang hilang. Umumnya perangkat *network* memiliki *buffer* untuk menampung data yang diterima. Jika terjadi kongesti yang cukup lama, buffer akan penuh, dan data baru tidak diterima. Paket yang hilang ini harus diretransmisi, yang akan membutuhkan waktu tambahan (Khairunnisa. 2003:68)

$$\text{Packet loss} = \frac{\text{Jumlah byte yang dikirim} - \text{jumlah byte yang diterima}}{\text{Jumlah byte dalam satu paket}} \quad (2.2)$$

2.11 *Delay end to end*

Delay pada sistem ini adalah waktu yang dibutuhkan untuk mengirimkan sebuah paket data. *Delay end to end* merupakan *delay* antara *node* sumber dan *node* tujuan. *Delay end to end* pada sistem ini dihitung dari *server* ke *client*.

$$t_{end\ to\ end} = t_{enc} + t_{trans} + t_p + t_w + t_{dec} \quad (2.3)$$

Dengan :

$t_{end\ to\ end}$ = *delay end to end* (s)

t_{enc} = *delay* enkapsulasi (s)

t_{trans} = *delay* transmisi (s)

t_p = *delay* propagasi (s)

t_w = delay antrian (s)

t_{dec} = *delay* dekapsulasi (s)

2.12 Kecepatan transmisi

Untuk menghitung kecepatan transmisi pada jaringan diperlukan nilai panjang gelombang pada frekuensi *carrier*. Yang didapat dengan menggunakan persamaan (2.4):

$$\lambda = \frac{V}{f_c} \qquad (2.4)$$

V = Kecepatan gelombang wi-fi pada media *wireless* = 3 x 10^8 m/s

f_c = frekuensi *carrier* pada *access point* WRT54GL

Bandwidth pada AP Linksys WRT54GL adalah sebesar 22 Mhz, maka besar bit ratenya adalah :

$$\textit{Bit rate} = \frac{1}{t_b} \text{ (bps)} = \text{f x n (bps)}$$

$$t_b = \frac{1}{\text{f x n}}$$

Dengan :

f = frekuensi yang digunakan berbanding lurus dengan *bandwidth* yang digunakan (Hz)

n = kemungkinan kondisi pada jaringan (kondisi 0 dan 1)

BAB III
METODE PENELITIAN

Skripsi ini membahas mengenai penerapan *Huffman Coding* untuk kompresi citra pada jaringan wifi 802.11 b/g secara *real time*. Metode penelitian yang digunakan dalam skripsi adalah sebagai berikut:

3.1 Studi Literatur

Studi literatur dilakuan untuk mempelajari literatur yang menunjang dalam perencanaan dan pembuatan sistem pada skripsi ini. Sumber dapat berupa buku cetak, buku panduan pemrograman, skripsi, *paper*, tutorial pemrograman, dan sumber bacaan *softcopy* lain yang didapatkan dari internet. Sumber bacaan juga dibutuhkan untuk mendapatkan data sekunder (Data sekunder adalah data yang diperoleh dari berbagai buku, jurnal-jurnal dan internet).

Adapun literatur yang dipelajari adalah teori yang berhubungan dengan perancangan sistem pada skripsi ini, meliputi teori kompresi citra, aplikasi jaringan wifi 802.11 b/g, aplikasi *real time*, dan perangkat lunak (*software*) yang digunakan dalam perancangan sistem.

3.2 Perancangan Perangkat Keras (*Hardware*)

Perangkat keras (*hardware*) terdiri dari perangkat komputer (PC) sebagai *server*, *webcam* sebagai media untuk melakukan *capture*, dan *access point* sebagai *router* untuk komunikasi *wireless* dari *server* ke *client* menggunakan jaringan wifi 802.11 b/g.

Langkah pertama untuk merancang sistem ini adalah menentukan spesifikasi alat. Penentuan spesifikasi alat yang dilakukan adalah sebagai berikut:

1. Perancangan dan penentuan spesifikasi komputer (PC)/laptop sebagai *server* dan kontrol komunikasi jaringan.
2. Perancangan dan penentuan spesifikasi *webcam* sebagai media untuk melakukan *capture*.
3. Perancangan dan penentuan spesifikasi *access point* sebagai *router* komunikasi *wireless* pada jaringan wifi 802.11 b/g.

Perancangan alat dilakukan dengan mengacu pada spesifikasi yang dibutuhkan sistem. Untuk perancangan alat dilakukan langkah-langkah sebagai berikut:

1. Pembuatan blok diagram sistem.
2. Perancangan perangkat keras dari masing-masing blok yang meliputi perencanaan dan pembuatan sistem.
3. Menggabungkan beberapa blok menjadi keseluruhan sistem yang direncanakan.
4. Perancangan perangkat lunak untuk menangani kebutuhan kontrol komunikasi jaringan sistem yang direncanakan.

3.3 Perancangan Perangkat Lunak (*Software*)

Perancangan perangkat lunak pada sistem ini adalah sebagai berikut:

1. Perancangan perangkat lunak sistem berupa perangkat lunak aplikasi yang dibuat dari *softwar*e Borland Delphi.
2. Perancangan perangkat lunak untuk meng-*capture* citra melalui *webcam*.
3. Perancangan perangkat lunak untuk mengkompresi citra.
4. Perancangan perangkat lunak untuk mendekompresi citra.

3.4 Pengujian

Pengujian ini dilakukan dalam menerapkan *huffman coding* untuk kompresi citra pada jaringan wifi 802.11 b/g secara *real time*. Proses pengujian ini terdiri dari beberapa bagian menurut arsitektur sistem yang dirancang yaitu sebagai berikut:

1. Pengujian koneksi jaringan wireless pada sistem

   Pengujian ini bertujuan untuk mengetahui apakah koneksi antara komputer (PC)/laptop sebagai *server* dapat berkomunikasi baik dengan *client* melalui *router* yang menggunakan *access point*.
2. Pengujian koneksi *webcam* sebagai media *capture*

   Pengujian ini bertujuan untuk mengetahui apakah koneksi *webcam* pada komputer (PC)/laptop sebagai *server* dapat melakukan *capture*.
3. Pengujian perangkat lunak kompresi citra.

   Pengujian ini bertujuan untuk mengetahui apakah perangkat lunak dapat mengkompresi data citra dari hasil *capture webcam* untuk ditransmisikan dan disimpan ke komputer/laptop *client*.
4. Pengujian perangkat lunak dekompresi citra.

   Pengujian ini bertujuan untuk mengetahui apakah perangkat lunak dapat mendekompresi file hasil kompresi tetap seperti file aslinya.
5. Pengujian sistem keseluruhan

Pengujian ini bertujuan untuk mengetahui apakah sistem dapat berfungsi secara keseluruhan.

3.5 Analisis

Pada skripsi ini dilakukan dua macam pengujian dan analisis untuk melihat keberhasilan sistem yang telah dibuat, yaitu:

1. Analisis pada sistem

   Analisis ini dilakukan untuk melihat kesesuaian antara hasil data pengujian dan teori. Analisis menggunakan cara perhitungan matematis dan membandingkan teori tersebut dengan hasil pengujian.

2. Analisis sistem secara keseluruhan

   Analisis ini bertujuan untuk melihat tingkat keberhasilan penerapan *Huffman coding* untuk kompresi citra pada jaringan wifi 802.11 b/g.

3.6 Pengambilan Kesimpulan dan saran

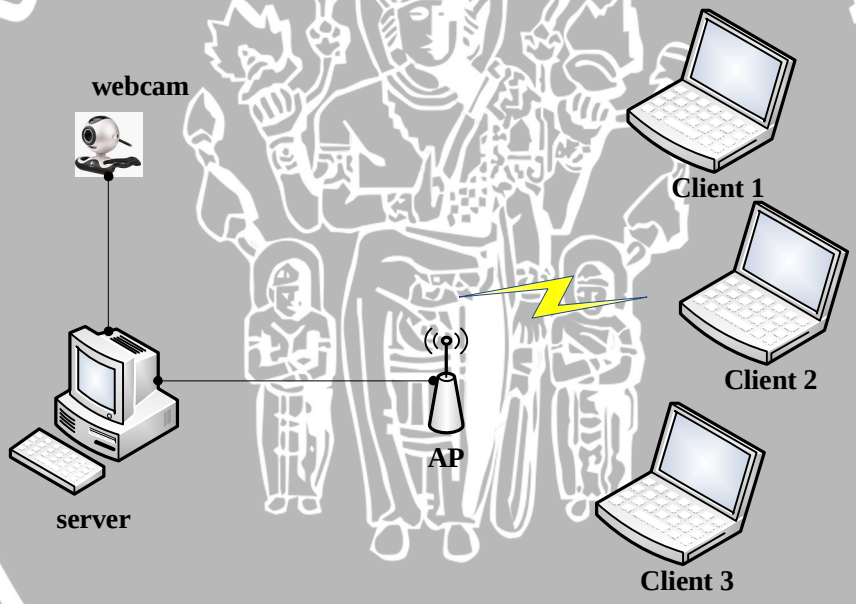
Tahap akhir dalam pembuatan skripsi adalah pengambilan kesimpulan dan saran dari sistem yang telah dibuat. Pengambilan kesimpulan didasarkan pada kesesuaian antara perancangan dengan hasil pengujian sistem. Sedangkan saran bertujuan untuk memperbaiki kekurangan-kekurangan yang terjadi dan kemungkinan-kemungkinan yang dapat dilakukan dalam pengembangan alat dimasa mendatang.

BAB IV
PERANCANGAN

4.1 Umum

Penerapan *Huffman coding* pada jaringan wifi 802.11 b/g yang dimaksudkan disini adalah penerapan *Huffman coding* untuk kompresi citra yang diimplementasikan pada jaringan wifi 802.11 b/g yang dapat diaplikasikan secara *real time*. Bentuk jaringan yang digunakan pada wifi adalah jaringan infrastruktur dengan menggunakan *access point router*.

Pada bab ini dijelaskan mekanisme perancangan jaringan wifi 802.11 b/g menggunakan *Huffman coding*, perancangan penerapan perangkat keras, dan perancangan penerapan perangkat lunak. Sistem perancangan jaringan wifi 802.11 b/g dapat dilihat pada gambar berikut:



Gambar 4. 1 Gambar perancangan sistem
Sumber: perancangan

Cara kerja sistem yaitu:

1. Pada saat PC *server* dalam keadaan “on”, webcam akan melakukan *capture* pada suatu objek.
2. Hasil *capture* akan dikompresi dengan menggunakan *Huffman coding* yang kemudian akan terkirim secara otomatis kepada *client*.

3. Proses kompresi dilakukan dengan menggunakan perangkat lunak aplikasi Borland Delphi 7 yang sudah dirancang pada PC *server*.
4. Citra hasil *capture* akan disimpan dalam PC *server* dan data hasil kompresi akan dikirim ke *client* dalam bentuk format kompresi.
5. Dengan menggunakan jaringan wifi 802.11 b/g pada PC *server* data hasil kompresi akan ditransmisikan melalui jaringan wireless menggunakan *access point router* WRT54GL.
6. Data hasil kompresi akan ditampilkan pada *client* melalui *access point router* dengan menggunakan laptop atau PC komputer yang memiliki *wireless card*.
7. Pada *client* akan melakukan dekompresi citra untuk memperoleh citra semula tanpa mengubah ukuran citra aslinya (*lossless*).

4.2 Perancangan Perangkat Keras (*Hardware*)

Penerapan perangkat keras dan *interface* yang direncanakan pada sistem ini adalah sebagai berikut:

1. Sebuah PC untuk *server* dengan spesifikasi (minimal):
   - *Processor* : Intel Pentium Celeron 1,6 GHz
   - RAM : 512 MB
   - *Operating system* : Microsoft Windows XP
   - NIC (*Network Interface card*) : Intel 1000 Base T Ethernet LAN
2. PC/laptop sebagai *client* dengan spesifikasi (minimal) masing-masing:
   - *Operating system* : Microsoft Windows XP/Vista
   - Processor : Intel (R) Pentium (R) CPU 2.66 GHz
   - RAM : 256 MB
   - PCMCIA card/NIC : Intel (R) PRO/wireless 3945 ABG
3. Sebuah *webcam* Sturdy.
4. Sebuah *access point router* WRT54GL dengan spesifikasi:

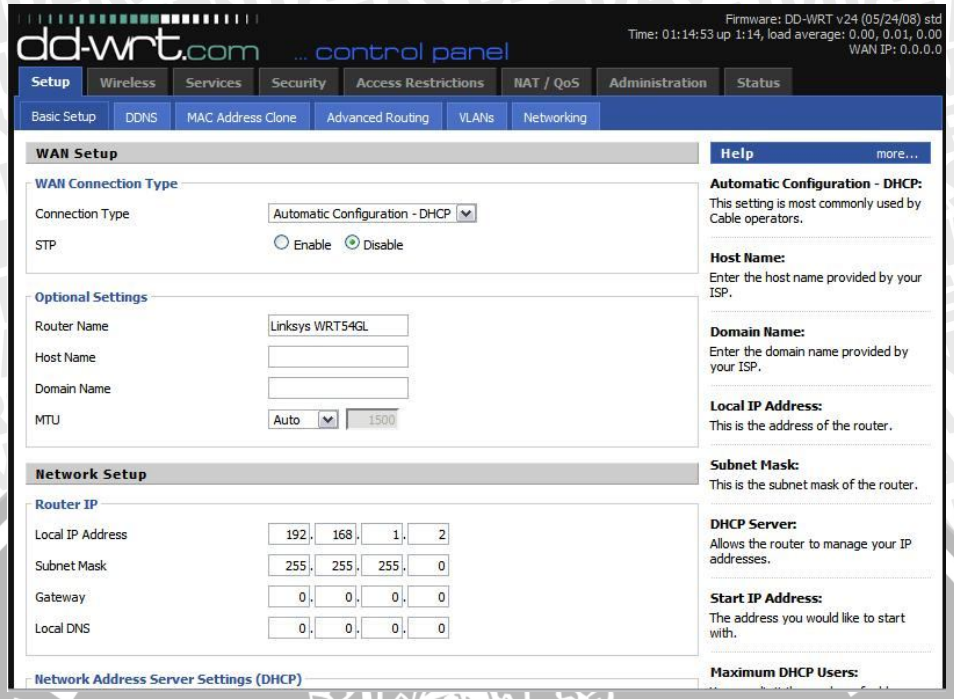
Gambar 4. 2 *Access point router* WRT54GL
Sumber: www.linksys.com

Tabel 4.1Data spesifikasi perangkat *access point router* Linksys WRT54GL

Standards	802.3, 802.3u, 802.11g and 802.11b
*Channel*s 802.11g	11 *Channel*s
Data Rate	Up to 54Mbps
Internet Port	One 10/100 RJ-45 Port for DSL modem
LAN Port	Four 10/100 RJ-45 Switch Ports
Transmit Power	18 dBm
Power External,	12V DC
Wireless Frequency range	2400 - 2483.5 MHz
Media Access Control	CSMA/CA with ACK
Operating Mode	Access Point, Router, Switch
Operating Temp	0ºC to 40ºC (32ºF to 104ºF)

Sumber: Perencanaan

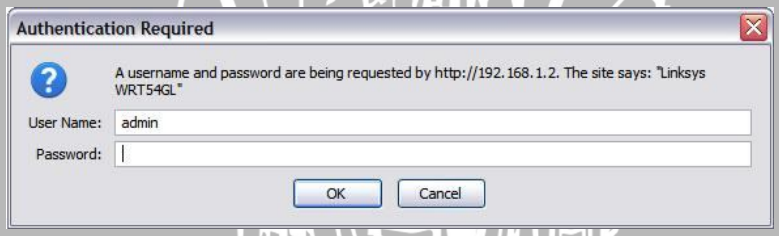
5. Pada skripsi ini dilakukan konfigurasi IP address dari *access point router* Linksys WRT54GL sebagai penyedia media komunikasi antara *server* dengan *client*. Konfigurasi tersebut dijelaskan dalam gambar berikut:



Gambar 4.3 Konfigurasi *IP address access point router*
Sumber : Perencanaan

Penjelasan:

a. Pada browser diketikkan alamat *default* AP *router* http://192.168.1.1
b. Masukan *User Name* : root dan *Password* : admin lalu tekan OK.



Gambar 4.4 Autentikasi *User Name* dan *Password*
Sumber: Perencanaan

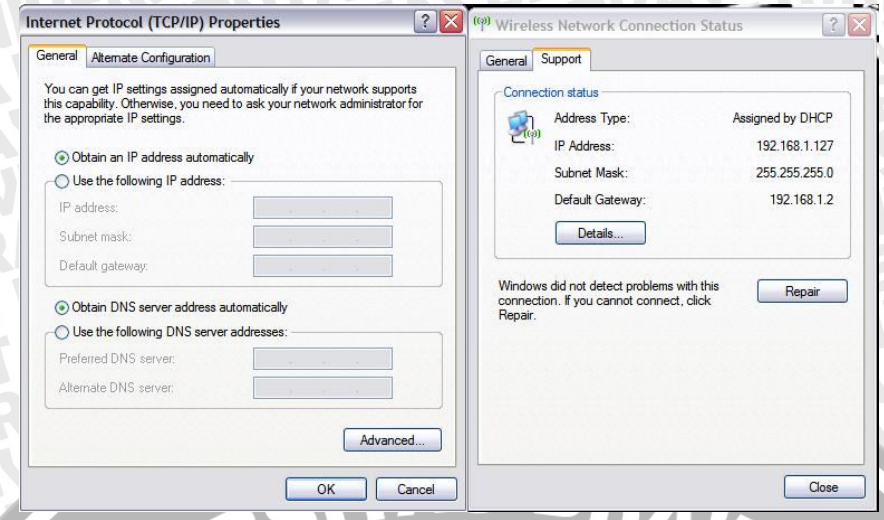
Pada Basic Setup dalam bagian Network Setup diisikan :

Local IP Address : 192.168.1.2

Subnet Mask : 255.255.255.0

c. Setelah itu disimpan dan *refresh* browser dan akses alamat http://192.168.1.2

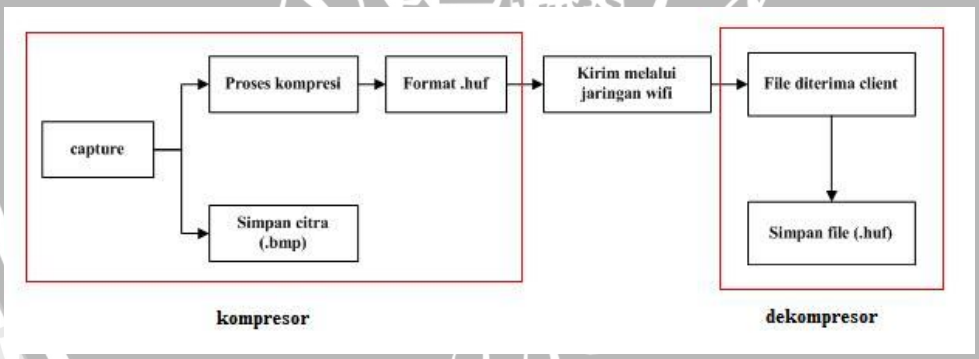
6. Konfigurasi *IP address* dari masing masing *client* dilakukan dengan konfigurasi *IP address* otomatis DHCP (*Dynamic Host Configuration Protocol*). *Setting IP address* dengan DHCP dapat dilihat pada gambar berikut:



Gambar 4.5 Konfigurasi IP *address* otomatis dengan DHCP
Sumber: Perencanaan

4.3 Perancangan Perangkat Lunak (*Software*)

Perancangan perangkat lunak pada sistem ini adalah perancangan *server* sebagai penyedia dan pemroses data, perancangan perangkat lunak *Huffman coding* untuk kompresi citra (kompresor) dengan menggunakan Borland Delphi 7 untuk diterapkan pada *server* dan perancangan perangkat lunak dekompresi citra (dekompresor) yang diterapkan pada *client.*



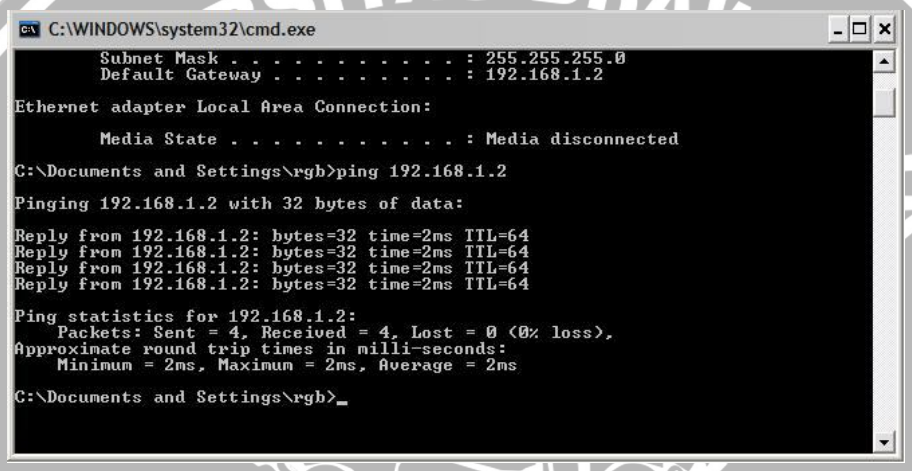
Gambar 4.6 Diagram proses penyimpanan dan pengiriman file
Sumber: Perencanaan

4.3.1 *Server*

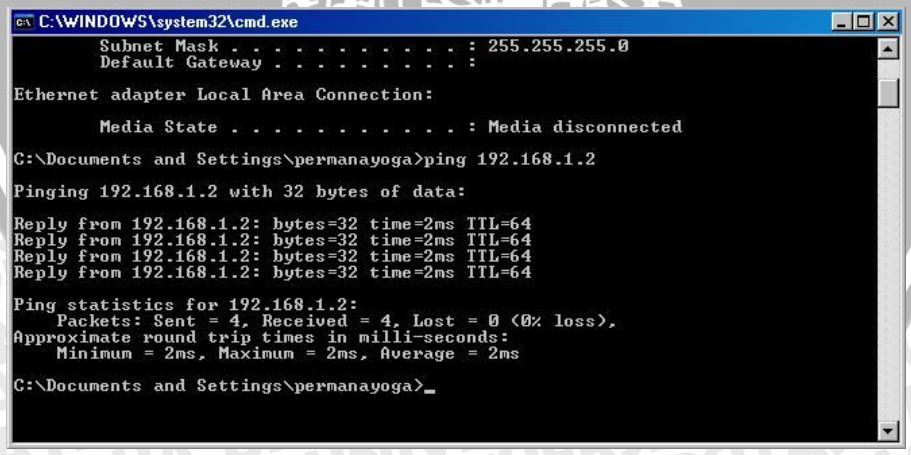
Server merupakan penyedia data dan pusat kontrol terjadinya komunikasi pada sebuah jaringan. Pada sistem ini *server* berfungsi sebagai penyedia, pengontrol dan pemroses data untuk dikirimkan kepada *client.* Data citra akan di-*capture* oleh *server* kemudian dilakukan proses kompresi menggunakan *Huffman coding* dan data hasil kompresi akan dikirimkan kepada *client* melalui jaringan wifi 802.11 b/g.

Untuk mentransmisikan data dari *server* ke *client*, perlu adannya konfigurasi jaringan antara *server* dengan *client*. Konfigurasi yang yang dapat dilakukan untuk komunikasi antara *server* dengan *client* adalah sebagai berikut:

1. Lakukan setting IP *address* untuk *server* dan *client*. Untuk sistem ini IP *address* akan diberikan secara otomatis dengan DHCP oleh AP *router*.
2. Mengecek perangkat keras yang digunakan apakah bisa melakukan koneksi *client – server* seperti yang diharapkan.
3. Untuk mengecek koneksi *client–server* dapat melakukan *ping* pada jaringan sistem melalui perintah *command prompt*.



Gambar 4.7 Koneksi antara *server* dengan *router*
Sumber: Perencanaan



Gambar 4.8 Koneksi antara *client* dengan *router*
Sumber: Perencanaan

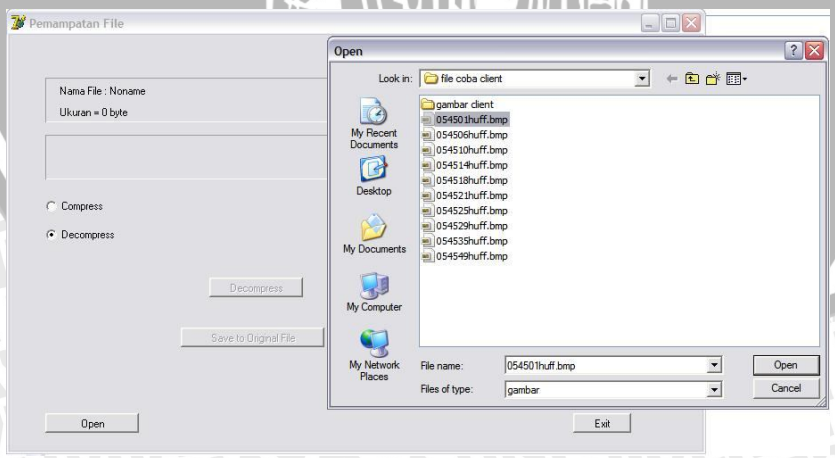
4. Setelah proses pengecekan koneksi selesai sistem siap digunakan.

4.3.2 Perangkat Lunak Kompresi (Kompresor)

Setelah setting konfigurasi antara *server* dan *client* berhasil, langkah yang dilakukan berikutnya adalah membuat perangkat lunak yang bisa mentransmisikan data antara *server* dengan *client* menggunakan perangkat lunak Borland Delphi 7. Berikut ini adalah gambar hasil perancangan perangkat lunak *Huffman coding* untuk kompresi citra.

Gambar 4. 9 Tampilan kompresor untuk meng-*capture*
Sumber : Perencanaan

Setelah data di-*capture* dari *server*, data hasil kompresi dengan metode Huffman akan langsung ditransmisikan ke *client* berupa file kompresi dengan ekstensi *.huf. Untuk memperoleh citra yang terkompresi seperti citra asli digunakan dekompresor untuk mendekompresi file terkompresi (.huf).



Gambar 4.10 Tampilan dekompresor
Sumber: Perencanaan

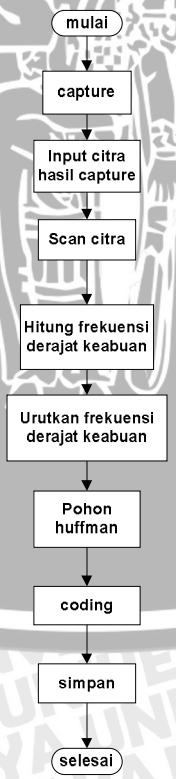
4.3.3 Tampilan Aplikasi Real Time

Untuk tampilan aplikasi *real time* pada sisi *client* adalah berbasis desktop dimana pada sisi *client* dibuat folder sharing untuk menyimpan file hasil transmisi. Pada file hasil transmisi sudah memiliki identitas sendiri berdasarkan waktu pengambilan citra citra dengan format waktu jam, menit dan detik (hh:mm:ss).

4.4 Diagram Alir Sistem

4.4.1 Diagram Alir Proses Kompresi Huffman

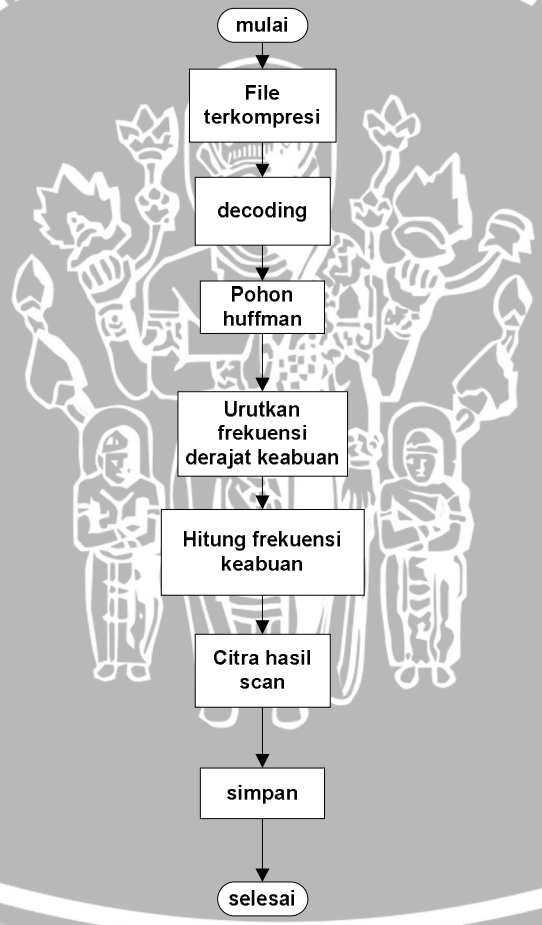
1. Program meminta inputan file citra hasil *capture*.
2. Men-scan file citra dan menentukan derajat keabuan setiap piksel citra.
3. Menghitung frekuensi dari setiap jenis derajat keabuan (nilai warna).
4. Mengurutkan frekuensi kemunculan derajat keabuan dari yang terkecil sampai yang terbesar. Jika frekuensinya sama diurutkan berdasarkan kode warna.
5. Membuat pohon Huffman sesuai urutan frekuensi nilai warna.
6. Menentukan kode Huffman untuk setiap cabang pohon.
7. Mengganti nilai derajat keabuan file citra asli dengan kode Huffman.
8. Menyimpan hasil kompresi ke dalam memori buffer.



Gambar 4.11 Diagram alir proses kompresi Huffman
Sumber: Perencanaan

4.4.2 Diagram Alir Proses Dekompresi Huffman

1. Program meminta inputan file hasil kompresi.
2. Mengganti nilai derajat keabuan file kompresi dengan decoding Huffman.
3. Menentukan kode Huffman untuk setiap cabang pohon.
4. Membuat pohon Huffman sesuai urutan frekuensi nilai warna.
5. Mengurutkan frekuensi kemunculan derajat keabuan dari yang terkecil sampai yang terbesar. Jika frekeuensinya sama diurutkan berdasarkan kode warna.
6. Menghitung frekuensi dari setiap jenis derajat keabuan setiap piksel citra.
7. Menentukan derajat keabuan setiap piksel citra.
8. Menyimpan file hasil dekompresi.



Gambar 4. 12 Diagram alir proses dekompresi Huffman
Sumber: Perencanaan

BAB V
HASIL PENGUJIAN DAN ANALISIS SISTEM

5.1 Umum

Bab ini membahas hasil pengujian *Huffman Coding* yang telah diterapkan pada jaringan wifi 802.11 b/g untuk kompresi citra. Tujuan pengujian ini adalah untuk mengetahui performansi jaringan wifi 802.11 b/g dalam penerapan kompresi citra menggunakan *Huffman coding*. Setelah pengujian dilakukan, sistem akan dianalisis untuk mengetahui performansi berdasarkan parameter-parameter yang telah ditentukan. Hal tersebut dilakukan untuk mendukung pengambilan kesimpulan dan saran.

5.2 Pengujian

Proses pengujian ini terdiri dari beberapa bagian menurut perancangan sistem yang digunakan. Pengujian dilakukan dengan menggunakan jaringan wifi 802.11 b/g yang dilakukan pada sistem yang dirancang. Pengujian ini meliputi:

1. Pengujian koneksi antara *server - acces point router – client.*
2. Pengujian koneksi *webcam* sebagai media *capture.*
3. Pengujian perangkat lunak kompresi citra.
4. Pengujian perangkat lunak dekompresi citra.
5. Pengujian sistem secara keseluruhan.

5.2.1 Pengujian Koneksi *server-access point router-client*

5.2.1.1 Tujuan Pengujian

Pengujijan ini dilakukan dengan tujuan untuk mengetahui apakah koneksi *server-access point router-client* dapat dilakukan dengan baik. Pengujian ini dilakukan dengan cara membuat koneksi antara *server* dan *access point* terlebih dahulu lalu koneksi antara *client* dan *server* melalui *access point router.*

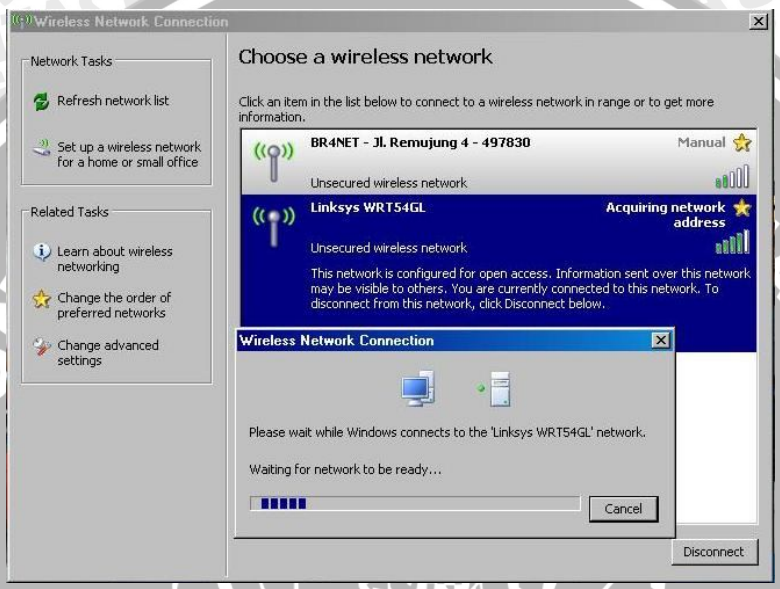
5.2.1.2 Peralatan Pengujian

1. Laptop yang digunakan untuk *server*
2. Access point router Linksys WRT54GL
3. Laptop untuk *client*

5.2.1.3 Prosedur Pengujian Koneksi Server-Access Point Router-Client

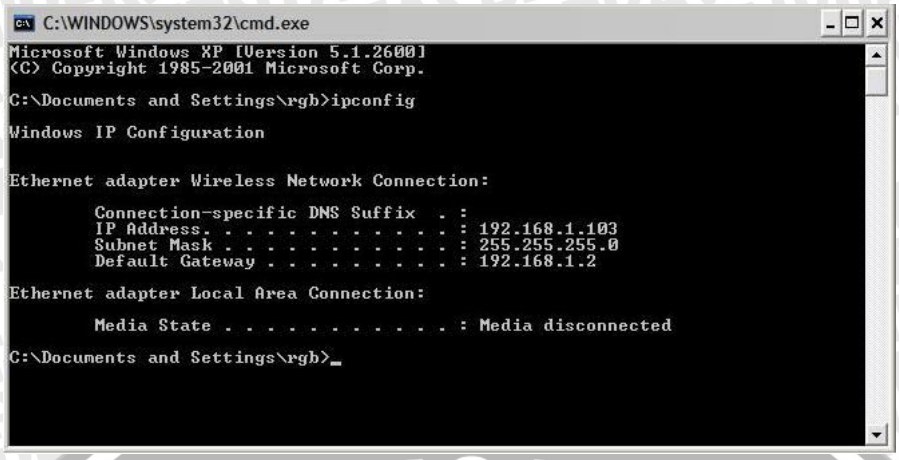
1. Menghubungkan *server* dengan *accces point router.*
2. Melakukan setting AP *router* melalui laptop yang digunakan sebagai *server.*
3. Mengaktifkan wifi 802.11 b/g pada komputer *client* dengan cara *enable wireless*

4. *card* yang ada pada laptop *client*.
5. Mengoneksikan server dengan AP dengan cara mebuka *icon wireless network connection* dan klik *connect* untuk menghubungkan antara AP dengan *server*.
6. Mengoneksikan *client* dengan AP dengan cara membuka *icon wireless network connection* dan klik *connect* untuk menghubungkan antara AP dengan *client*. Koneksi *client* dengan AP dapat dilihat pada gambar berikut:



Gambar 5.1 Koneksi antara *client* dengan AP
Sumber: Hasil pengujian

7. Setelah koneksi antara AP dengan *server* dan AP dengan *client* berhasil maka *server* dan *client* akan memperoleh *IP address* dari AP *router* Linksys WRT54GL. Untuk *server* dan *client* akan mendapatkan *ip address* secara otomatis dari AP *router*.
8. Untuk mengetahui koneksi yang sudah berhasil, dapat diketahui dengan cara membuka *command prompt* kemudian ketik *ipconfig*. Koneksi dapat dilihat pada gambar berikut:



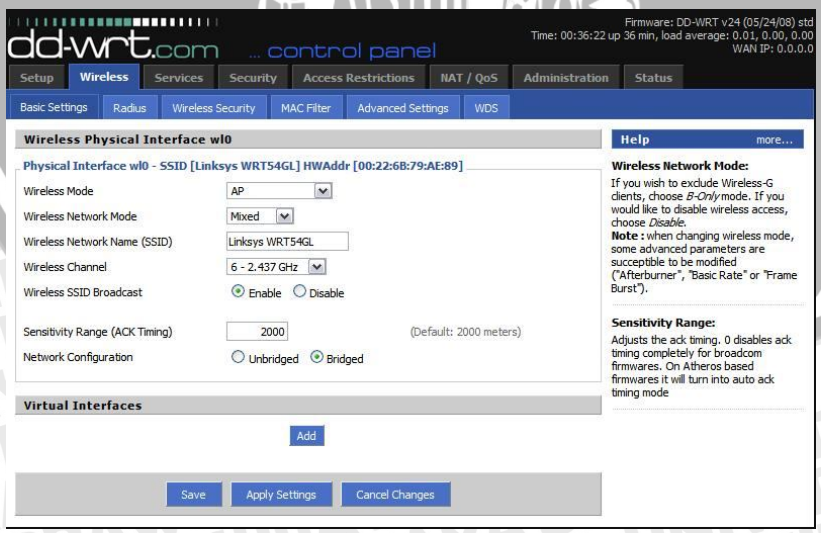
Gambar 5.2 Tampilan koneksi pada *server*
Sumber: Hasil pengujian

9. Setelah masuk *command prompt*, untuk mengetahui terjadinya koneksi dengan AP dapat dilakukan dengan cara mengetik ping 192.168.1.2. IP 192.168.1.2 adalah IP *default* dari AP *router*.

5.2.1.4 Data Hasil Pengujian

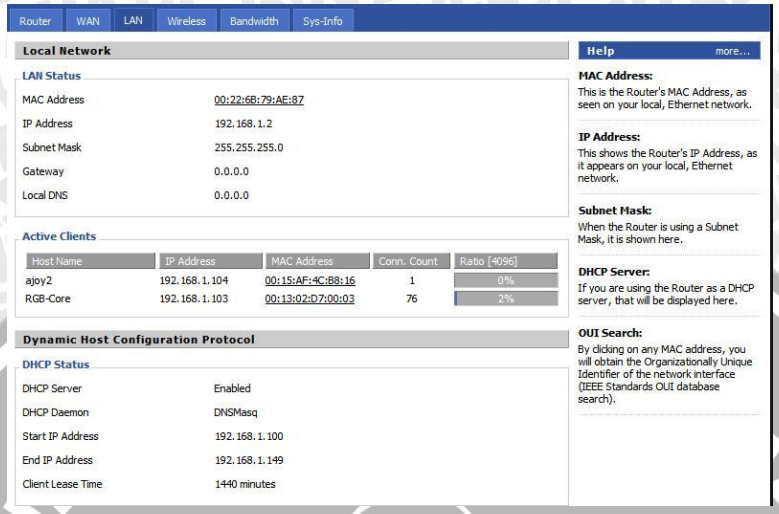
Dari pengujian yang dilakukan didapatkan data sebagai berikut:

1. IP yang diperoleh oleh *server* dari AP menggunakan DHCP adalah 192.168.1.103.
2. IP yang diperoleh oleh *client* dari AP menggunakan DHCP adalah 192.168.1.104.
3. IP untuk AP adalah 192.168.1.2 yang merupakan *default* dari AP *router*. Koneksi AP dapat dilihat pada gambar berikut:



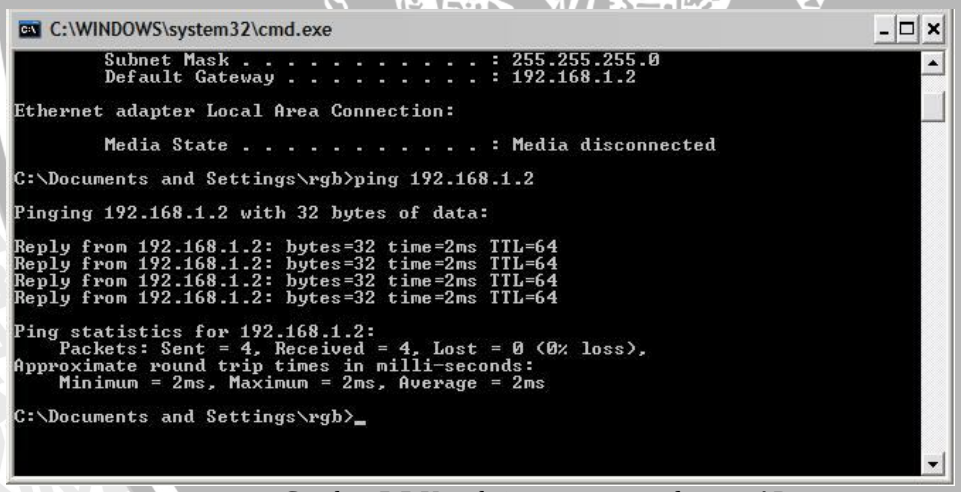
Gambar 5.3 Setting *default* AP router
Sumber: Hasil pengujian

4. Koneksi antara *server-access point router-client* berjalan baik. Koneksi ini dapat dilihat pada gambar berikut:

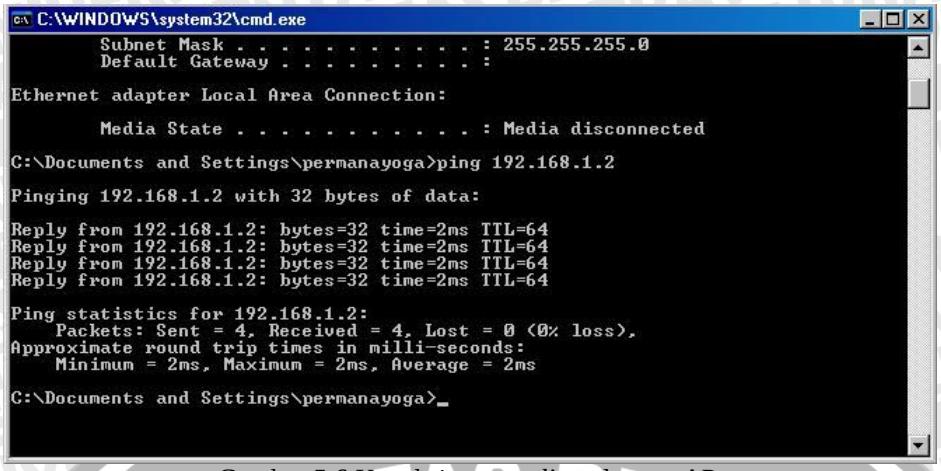


Gambar 5.4 Koneksi antara *server* dan *client* dengan AP
Sumber: Hasil pengujian

5. Koneksi antara *server-access point router-client* dapat dilihat juga melalui *command prompt*. Koneksi dapat dilihat pada gambar berikut:



Gambar 5.5 Koneksi antara *server* dengan AP
Sumber: Hasil pengujian



Gambar 5.6 Koneksi antara *client* dengan AP
Sumber: Hasil pengujian

5.2.1.5 Kesimpulan Hasil Pengujian

Dari hasil pengujian koneksi *server* dan *client* terlihat bahwa *client* berhasil melakukan koneksi melalui AP *router*. *Client* memperoleh alamat IP secara otomatis melalui DHCP (*Dynamic Host Configuration Protocol*). Dari pengujian didapatkan bahwa koneksi antara *server-access point router-client* dapat terhubung dengan baik.

5.2.2 Pengujian Koneksi Webcam

5.2.2.1 Tujuan Pengujian

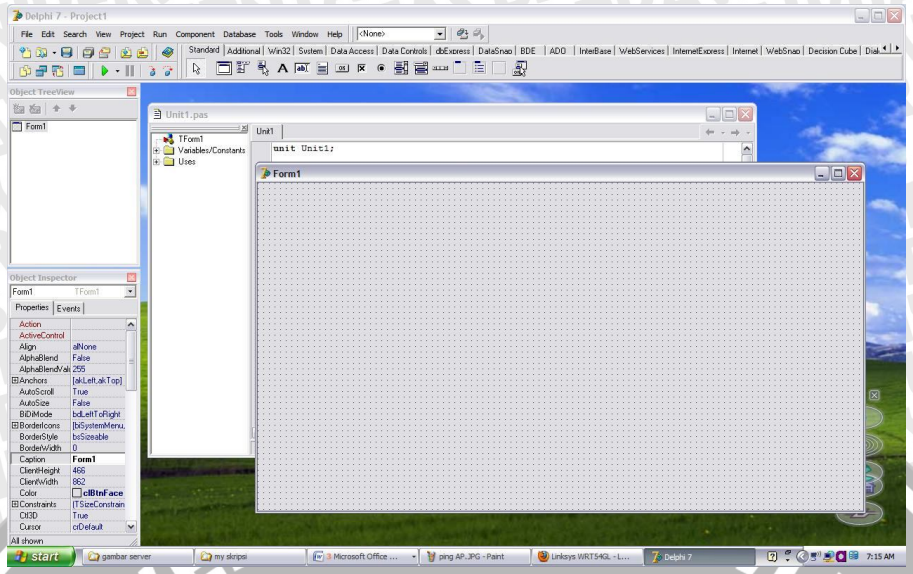
Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah koneksi webcam sebagai media *capture* dapat dilakukan dengan baik. Pengujian ini dilakukan dengan cara membuat koneksi antara *webcam* dengan laptop (*server*) yang telah terintegrasi dengan perangkat lunak aplikasi *capture* dengan menggunakan perangkat lunak Borland Delphi.

5.2.2.2 Peralatan Pengujian

1. Laptop yang digunakan sebagai *server.*
2. Webcam Sturdy.
3. Perangkat lunak Borland Delphi.

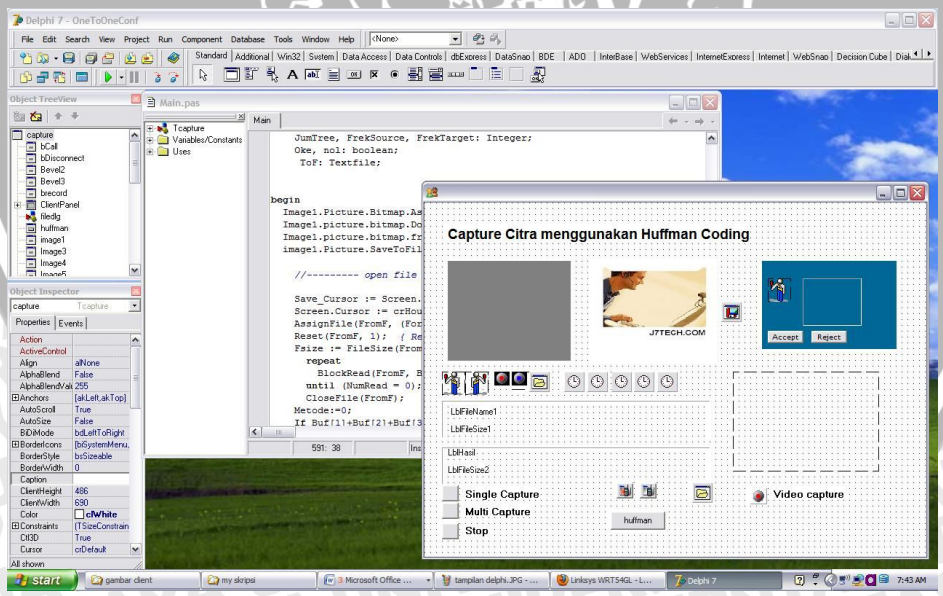
5.2.2.3 Prosedur Pengujian Koneksi Webcam

1. Menghubungkan laptop (*server*) dengan *webcam.*
2. Melakukan instalasi perangkat lunak webcam untuk bisa terkoneksi dengan *server*.
3. Setelah proses instalasi selesai kemudian buka perangkat lunak aplikasi Borland Delphi yang sudah terintegrasi dengan *webcam* untuk aplikasi *capture*. Menjalankan perangkat lunak aplikasi Borland Delphi dapat dilihat pada gambar berikut:



Gambar 5.7 Tampilan perangkat lunak Borland Delphi
Sumber: Hasil pengujian

4. Kemudian buka file program Delphi untuk aplikasi *capture* yang sudah terintegrasi dengan *webcam.*
5. Jalankan program aplikasi *capture* dengan menekan tombol "Run" pada tampilan Delphi atau "F9" pada keyboard laptop. Tampilan program aplikasi Delphi yang sudah terintegrasi dengan *webcam* dapat dilihat pada gambar berikut:



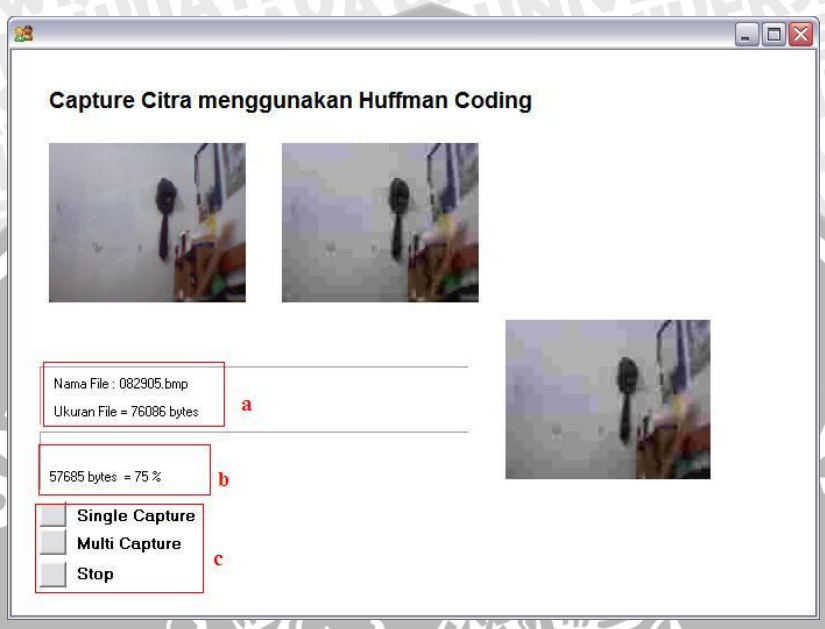
Gambar 5.8 Tampilan program aplikasi Delphi
Sumber: Hasil pengujian

6. Setelah program aplikasi dijalankan maka akan muncul tampilan terintegrasi antara *webcam* dengan *server*.

5.2.1.4 Data Hasil Pengujian

Dari pengujian yang dilakukan didapatkan data sebagai berikut:

1. Koneksi antara *webcam* dengan *server* berjalan dengan baik. Koneksi antara *webcam* dengan *server* dapat dilihat pada gambar berikut:



Gambar 5.9 Tampilan terintegrasi antara *webcam* dengan *server*
Sumber: Hasil pengujian

Keterangan:

a. Nama file dan ukuran file, pada kolom tersebut menunjukkan nama file (citra) yang di-*capture* beserta ukuran filenya dalam satuan byte.
b. Nilai rasio kompresi.
c. Tombol navigasi.

2. Berdasarkan gambar 5.9 aplikasi *capture webcam* sudah dapat berjalan tanpa perangkat lunak Delphi karena format aplikasi adalah .exe.

5.2.2.5 Kesimpulan Hasil Pengujian

Dari hasil pengujian koneksi webcam terlihat bahwa webcam dapat melakukan koneksi dengan laptop (*server*) menggunakan perangkat lunak aplikasi Borland Delphi. Dari pengujian didapatkan bahwa koneksi antara *webcam* dengan *server* dapat terhubung dengan baik.

5.2.3 Pengujian Perangkat Lunak Kompresi

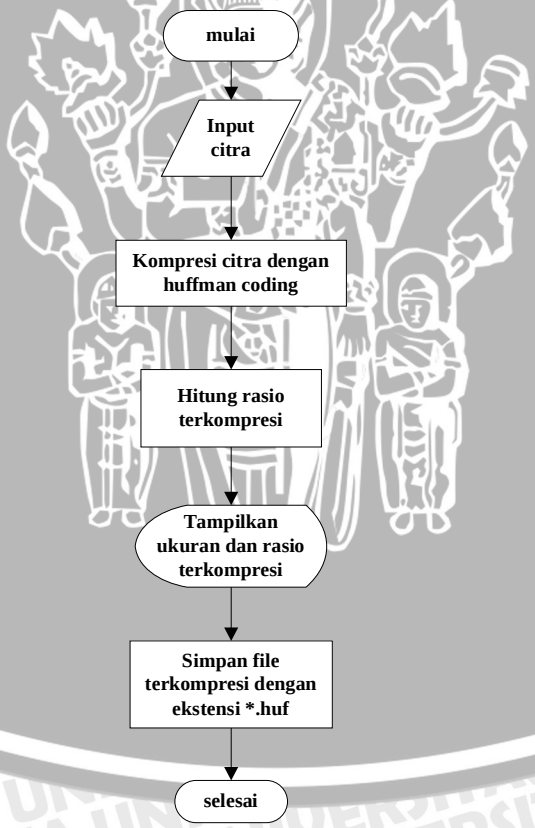
5.2.3.1 Tujuan Pengujian

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah perangkat lunak yang dibuat dapat mengkompresi data citra dari hasil *capture webcam* untuk disimpan ke komputer/laptop melalui jaringan wifi 802.11 b/g. Pengujian ini dilakukan dengan cara mengintegrasikan *webcam*, *server*, AP *router* dan perangkat lunak aplikasi.

5.2.3.2 Peralatan Pengujian

1. Laptop yang digunakan sebagai *server.*
2. Laptop yang digunakan sebagai *client.*
3. AP router Linksys WRT54GL.
4. Webcam Sturdy.
5. Perangkat lunak Borland Delphi.

5.2.3.3 Prosedur Pengujian Perangkat Lunak Kompresi

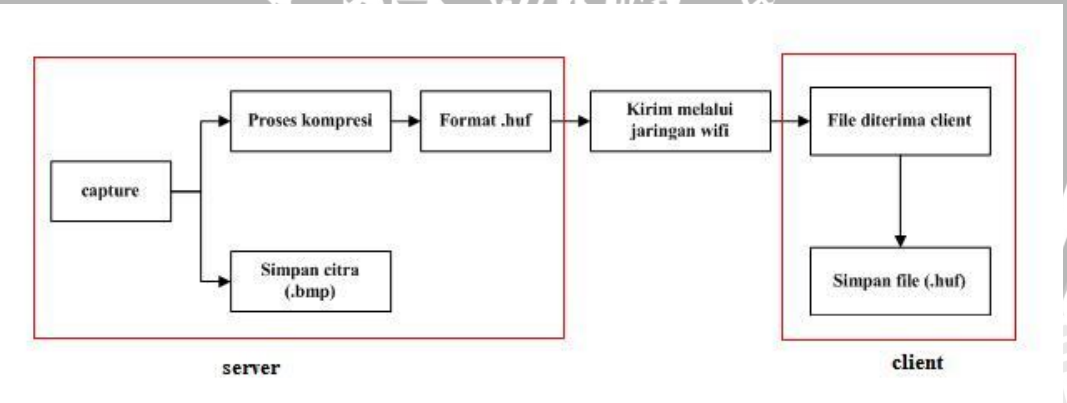


Gambar 5.10 Diagram alir proses kompresi
Sumber: Perencanaan

1. Jalankan program aplikasi perangkat lunak Delphi pada sisi *server*.
2. Pada tampilan aplikasi *capture*, lakukan proses pengambilan citra dengan cara klik

tombol navigasi pada tampilan yang terdiri dari tombol “Single Capture”, “Multi Capture” dan “Stop”.

3. Untuk melakukan proses pengambilan citra dengan satu kali *capture* tekan tombol “Single Capture”.
4. Untuk melakukan proses pengambilan citra secara kontinyu/berulang kali tekan tombol “Multi Capture”.
5. Untuk menghentikan proses pengambilan citra secara kontinyu tekan tombol “Stop”.
6. Setelah proses pengambilan citra dilakukan, program aplikasi Delphi akan melakukan proses kompresi secara otomatis.
7. Setelah proses kompresi dilakukan, pada tampilan aplikasi akan muncul tampilan nama file beserta format filenya, ukuran file dan rasio kompresi dari citra yang di-*capture*.
8. File terkompresi akan langsung dikirim ke sisi *client* dengan format file berekstensi Huffman (.huf) sedangkan pada sisi *server* juga menyimpan file citra hasil *capture* (file asli) sebelum dikompresi dengan format file berekstensi bitmap (.bmp). Proses penyimpanan dan pengiriman file dapat dijelaskan melalui gambar berikut:

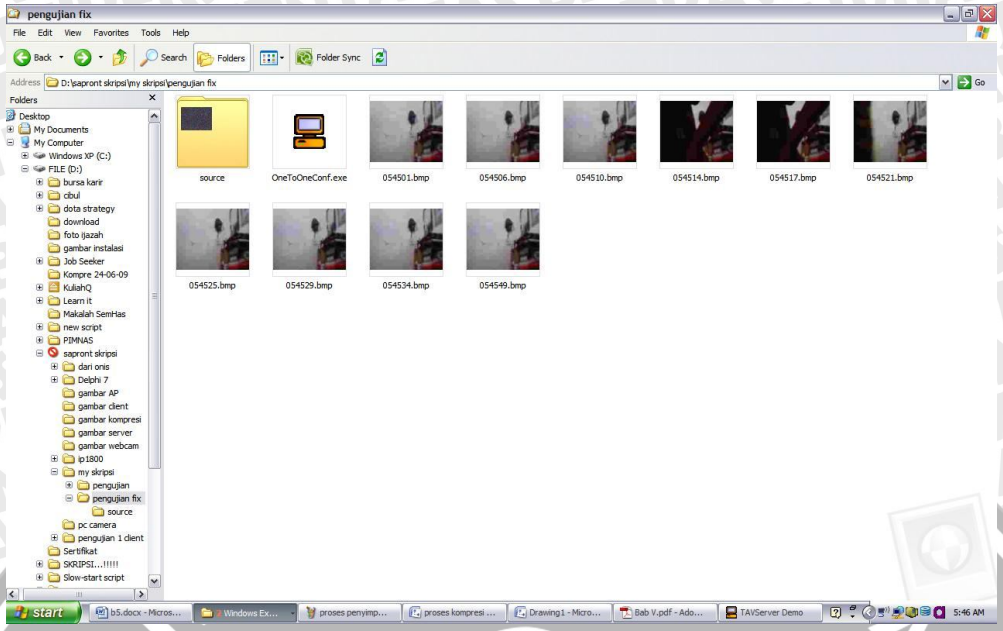


Gambar 5.11 Diagram proses penyimpanan dan pengiriman file
Sumber: Perencanaan

5.2.3.4 Data Hasil Pengujian dan Analisis

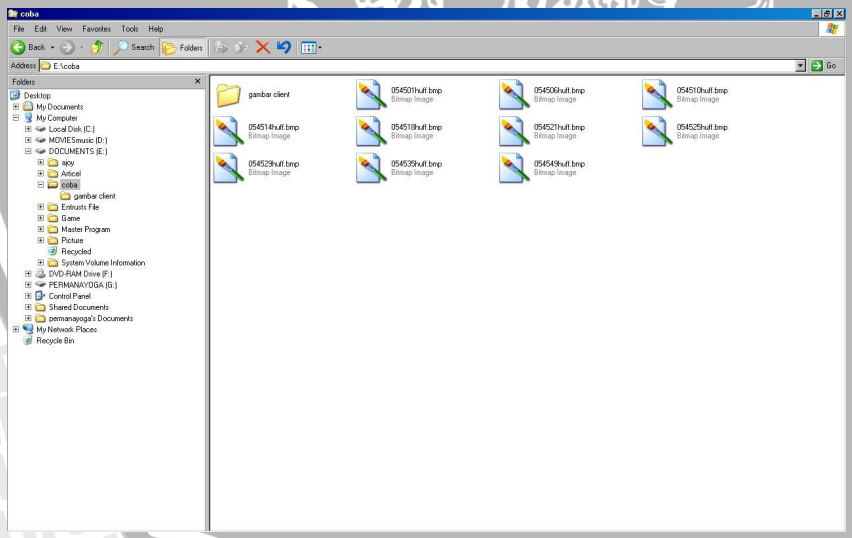
Dari pengujian yang dilakukan didapatkan data sebagai berikut:

1. Pada tampilan aplikasi *capture* terdapat dua mode untuk melakukan pengambilan citra yaitu dengan “Single Capture” dan “Multi Capture”.
2. Untuk file citra hasil *capture* (tanpa kompresi) akan disimpan secara otomatis pada laptop *server*. File citra hasil *capture* dapat dilihat pada gambar berikut:



Gambar 5.12 File citra yang tersimpan otomatis pada *server*
Sumber: Hasil pengujian

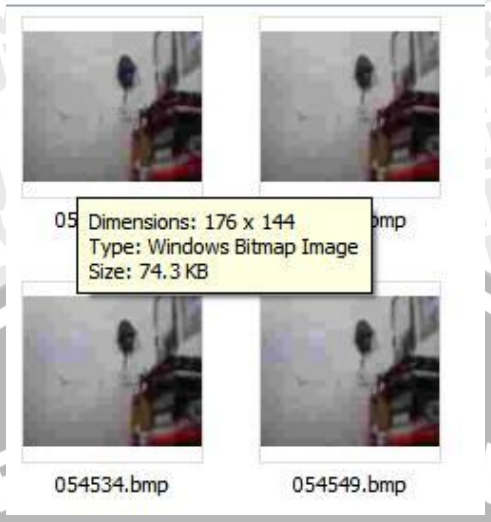
3. Untuk file citra hasil kompresi akan dikirim dan tersimpan secara otomatis pada laptop *client.* File citra hasil kompresi dapat dilihat pada gambar berikut:



Gambar 5.13 File hasil kompresi yang tersimpan pada *client*
Sumber: Hasil pengujian

4. Nama file citra dibuat otomatis berdasarkan waktu pengambilan citra. Seperti contoh pada gambar 5.13, citra dengan nama “054534.bmp” dan “054549” merupakan citra yang diambil pada jam 05:45:34 dan jam 05:45:49 dengan format file bitmap (.bmp). Untuk ukuran file cita hasil *capture* (tanpa kompresi) adalah

74.3 KB. Citra hasil *capture* dapat dilihat pada gambar berikut:



Gambar 5.14 Citra hasil *capture* tanpa kompresi
Sumber: Hasil pengujian

5. Berikut ini data ukuran file hasil *capture* dan hasil kompresi untuk 10 file yang tersimpan pada *server* dan *client*.

Tabel 5.1 File hasil *capture*

Data ke	Nama file	Ukuran (KB)	Format
1	054501	74,3	bmp
2	054506	74,3	bmp
3	054510	74,3	bmp
4	054514	74,3	bmp
5	054517	74,3	bmp
6	054521	74,3	bmp
7	054525	74,3	bmp
8	054529	74,3	bmp
9	054534	74,3	bmp
10	054549	74,3	bmp

Sumber: Hasil pengujian

Tabel 5.2 File hasil kompresi

Data ke	Nama file	Ukuran (KB)	Format
1	054501huff	54,4	.huf
2	054506huff	54,5	.huf
3	054510huff	56,8	.huf
4	054514huff	40,3	.huf
5	054518huff	42,4	.huf
6	054521huff	58,5	.huf
7	054525huff	55,3	.huf
8	054529huff	56,4	.huf
9	054535huff	56,4	.huf
10	054549huff	56,2	.huf

Sumber: Hasil pengujian

Berdasarkan tabel 5.1 dan 5.2 dapat diperoleh nilai rasio kompresi citra hasil *capture*. Dengan menggunakan rumus berikut:

$$rasio\ kompresi = \left(1 - \frac{ukuran\ citra\ hasil\ kompresi}{ukuran\ citra\ asli}\right) 100\%$$

Diperoleh hasil rasio kompresi untuk 10 data citra kompresi hasil *capture.*

$$D1 = \left(1 - \frac{54,4}{74,3}\right) 100\% = 26,78\%$$

$$D2 = \left(1 - \frac{54,5}{74,3}\right) 100\% = 26,64\%$$

$$D3 = \left(1 - \frac{54,8}{74,3}\right) 100\% = 26,24\%$$

$$D4 = \left(1 - \frac{40,3}{74,3}\right) 100\% = 45,76\%$$

$$D5 = \left(1 - \frac{42,4}{74,3}\right) 100\% = 42,93\%$$

$$D6 = \left(1 - \frac{58,5}{74,3}\right) 100\% = 21,27\%$$

$$D7 = \left(1 - \frac{55,3}{74,3}\right) 100\% = 25,57\%$$

$$D8 = \left(1 - \frac{56,4}{74,3}\right) 100\% = 24,09\%$$

$$D9 = \left(1 - \frac{56,4}{74,3}\right) 100\% = 24,09\%$$

$$D10 = \left(1 - \frac{56,2}{74,3}\right) 100\% = 24,36\%$$

Dari perhitungan diatas berikut ini adalah tabel rasio kompresi antara ukuran file sebelum dikompresi dengan file setelah kompesi.

Tabel 5.3 Nilai hasil rasio kompresi

Data ke	Ukuran file		Rasio kompresi (%)
	Sebelum kompresi	Setelah kompresi	
1	74,3	54,4	26,78
2	74,3	54,5	26,64
3	74,3	56,8	26,24
4	74,3	40,3	45,76
5	74,3	42,4	42,93
6	74,3	58,5	21,27
7	74,3	55,3	25,57
8	74,3	56,4	24,09

Lanjutan tabel 5.3

9	74,3	56,4	24,09
10	74,3	56,2	24,36

Sumber: Hasil pengujian

Rasio kompresi rata-rata untuk 10 file yang terkompresi adalah sebagai berikut:

$$= \frac{26,78 + 26,64 + 26,24 + 45,76 + 42,93 + 21,27 + 25,57 + 24,09 + 24,09 + 24,36}{10}$$

$$= \frac{287,73}{10}$$

$$= 28,73\%$$

Jadi rata-rata rasio kompresi untuk 10 file citra adalah 28,73%

5.2.3.5 Kesimpulan Hasil Pengujian

Dari hasil pengujian perangkat lunak kompresi terlihat bahwa perangkat lunak kompresi (kompresor) dapat berjalan dengan cukup baik. Dari pengujian didapatkan bahwa kompresor mampu mengkompresi file citra dengan rasio kompresi rata-rata 28,73%.

5.2.4 Pengujian Perangkat Lunak Dekompresi

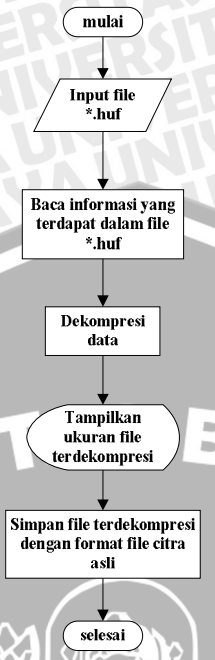
5.2.4.1 Tujuan Pengujian

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah perangkat lunak yang dibuat dapat mendekompresi citra terkompresi tanpa mengubah citra asli yang ditransmisikan melalui jaringan wifi 802.11 b/g. Pengujian ini dilakukan dengan cara mengintegrasikan perangkat lunak dekompresor dengan laptop *client*.

5.2.4.2 Peralatan Pengujian

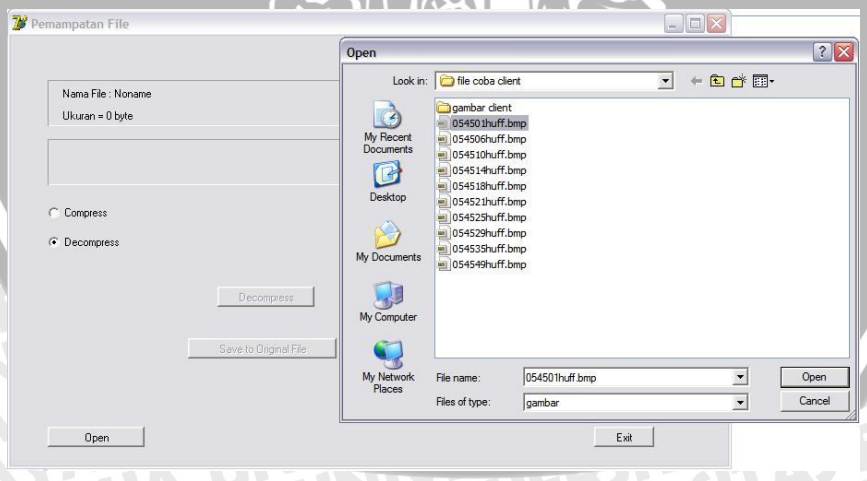
1. Laptop yang digunakan sebagai *client*.
2. Aplikasi perangkat lunak Delphi (dekompresor).

5.2.4.3 Prosedur Pengujian Perangkat Lunak Dekompresi



Gambar 5.15 Diagram alir proses dekompresi
Sumber: Perencanaan

1. Jalankan program aplikasi dekompresor yang sudah dirancang pada sisi *client*.
2. Setelah program aplikasi dekompresor dijalankan, masukkan data terkompresi (.huf) yang sudah tersimpan pada *client* untuk di dekompresi. Program dekomproser dapat dilihat pada gambar berikut:



Gambar 5.16 Tampilan dekompresor
Sumber: Hasil pengujian

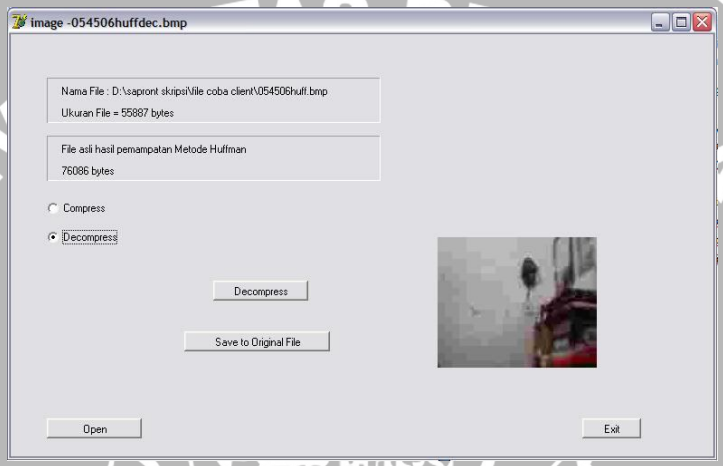
3. Setelah file terkompresi (.huf) dijadikan input pada dekompresor, secara otomatis dekompresor akan memproses dekompresi citra.

4. File terkompresi (.huf) yang sudah didekompresi akan ditampilkan pada dekompresor beserta ukuran file yang telah terdekompresi.
5. Kemudian file terdekompresi akan disimpan pada *client* sesuai dengan citra asli tanpa mengubah citra sedikitpun (*lossless*).

5.2.4.4 Data Hasil Pengujian dan Analisis

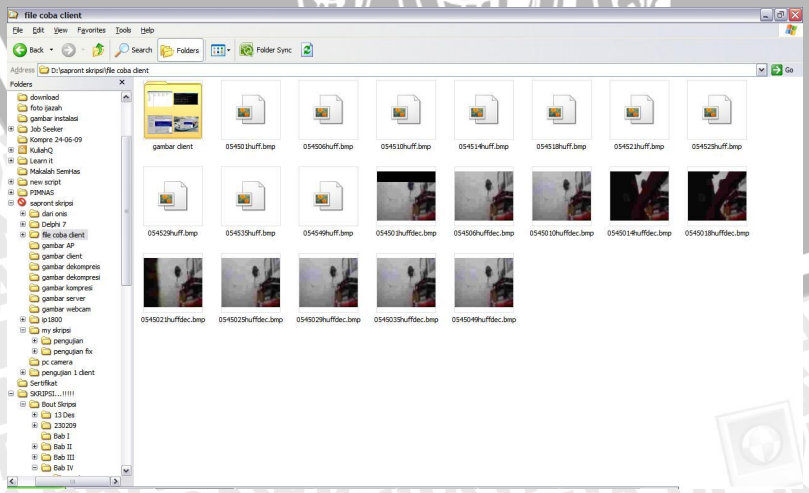
Dari pengujian yang dilakukan didapatkan data sebagai berikut:

1. Citra terkompesi dapat didekompresi seperti citra asli dan ditampilkan pada dekompresor. Citra terdekompresi dapat dilihat pada gambar berikut:



Gambar 5.17 Citra terdekompresi pada dekompresor
Sumber: Hasil pengujian

2. Ukuran file citra terdekompresi sama persis dengan ukuran file citra hasil *capture* yaitu sebesar 74,3 KB. Citra hasil dekompresi dengan ukuran yang sama dapat dilihat pada gambar berikut:



Gambar 5.18 Citra hasil dekompresi
Sumber: Hasil pengujian

Gambar 5.19 Citra hasil dekompresi dengan ukuran yang sama
Sumber: Hasil pengujian

5.2.4.5 Kesimpulan Hasil Pengujian

Dari hasil pengujian perangkat lunak dekompresi terlihat bahwa perangkat lunak dekompresi (dekompresor) dapat berjalan dengan cukup baik. Dari pengujian didapatkan bahwa dekompresor mampu mengembalikan citra terkompresi seperti citra asli (citra hasil *capture*) dengan ukuran file 74,3 KB.

5.2.5 Pengujian Sistem secara Keseluruhan

5.2.5.1 Tujuan Pengujian

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah Huffman coding dapat diterapkan pada jaringan wifi 802.11 b/g untuk aplikasi kompresi citra. Pengujian ini dilakukan dengan cara mengintegrasikan semua peralatan pengujian meliputi: *server, client, router, webcam* dan perangkat lunak aplikasi.

5.2.5.2 Peralatan Pengujian

1. Laptop yang digunakan sebagai *server.*
2. Laptop yang digunakan sebagai *client.*
3. AP *router* Linksys WRT54GL.
4. Webcam Surdy.
5. Perangkat Lunak Borland Delphi.

5.2.5.3 Prosedur Pengujian

1. Mengintergrasikan semua peralatan pengujian meliputi: *server, client, router, webcam* dan perangkat lunak aplikasi (kompresor dan dekompresor).

2. Melakukan pengujian secara keseluruhan dengan cara berurutan mulai koneksi antara *server – access point router – client*, koneksi *webcam* dan perangkat lunak aplikasi (kompresor dan dekompresor).
3. Menganalisis hasil pengujian kompresi citra dari *webcam* dengan membandingkan hasil pengujian kompresi citra untuk berbagai macam ukuran.
4. Menganalisis *throughput* dan *delay end to end* data pengujian pada jaringan wifi 802.11 b/g dengan membadingkan antara file yang terkompresi dengan file tanpa kompresi.

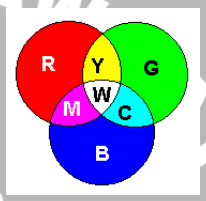
5.2.5.3 Data Hasil Pengujian dan Analisis

Dari Pengujian yang dilakukan didapatkan data sebagai berikut:

Tabel 5.4 Gambar bitmap dengan berbagai macam ukuran

No	Nama gambar	Ukuran (KB)	Gambar
1	Borobudur	177	
2	Butterfly	173	
3	Cat_fish	98,4	

Lanjutan tabel 5.4

4	Cat_mirror	98,7	
5	Colors	93	
6	Flower	81	
7	Kuda	144	

Lanjutan tabel 5.4

8	Nemo	191	
9	Nemo2	172	

Sumber: Perencanaan

1. Tabel 5.4 merupakan citra yang digunakan untuk membandingkan hasil pengujian kompresi citra untuk berbagai macam ukuran dan kombinasi warna.

Tabel 5.5 Hasil kompresi Huffman

No	Nama gambar	Ukuran sebelum dikompresi (KB)	Ukuran setelah dikompresi (KB)	Rasio kompresi (%)
1	Borobudur	177	142	16,45
2	Butterfly	173	164	5,2
3	Cat_fish	98,4	97,2	1,22
4	Cat_mirror	98,7	94,5	4,26
5	Colors	93	15,6	83,23
6	Flower	81	80,9	0,12
7	Kuda	144	144	0

Lanjutan tabel 5.5

8	Nemo	191	190	0,52
9	Nemo2	172	169	1,74

Sumber: Hasil pengujian

2. Dari data pengujian diperoleh bahwa tidak semua citra bisa dikompresi dengan Huffman coding, walaupun ukuran filenya tidak jauh berbeda. Hal ini disebabkan oleh intensitas warna dari citra yang dikandung. Kompresi menggunakan Huffman coding tidak efektif untuk mengkompresi citra dengan kombinasi warna yang tinggi.
3. IP dari 3 *client* yang diperoleh dari AP menggunkan DHCP adalah sebagai berikut:

   IP client 1 : 192.168.1.104

   IP client 2 : 192.168.1.108

   IP client 3 : 192.168.1.103
4. *Delay end to end* yang terjadi dihitung dengan cara RTT (*Run Trip Time*) dibagi dua. Berikut ini adalah tabel perhitungan *delay end to end* untuk semua *client* yang menggunakan kompresi dengan tanpa kompresi:

Tabel 5.6 Hasil pengujian *delay end to end* tanpa kompresi

	IP address	**RTT**	**Delay end to end**
Client 1	192.168.1.104	0,003471 s	0,001736 s
Client 2	192.168.1.108	0,003249 s	0,001625 s
Client 3	192.168.1.103	0,003023 s	0,001512 s

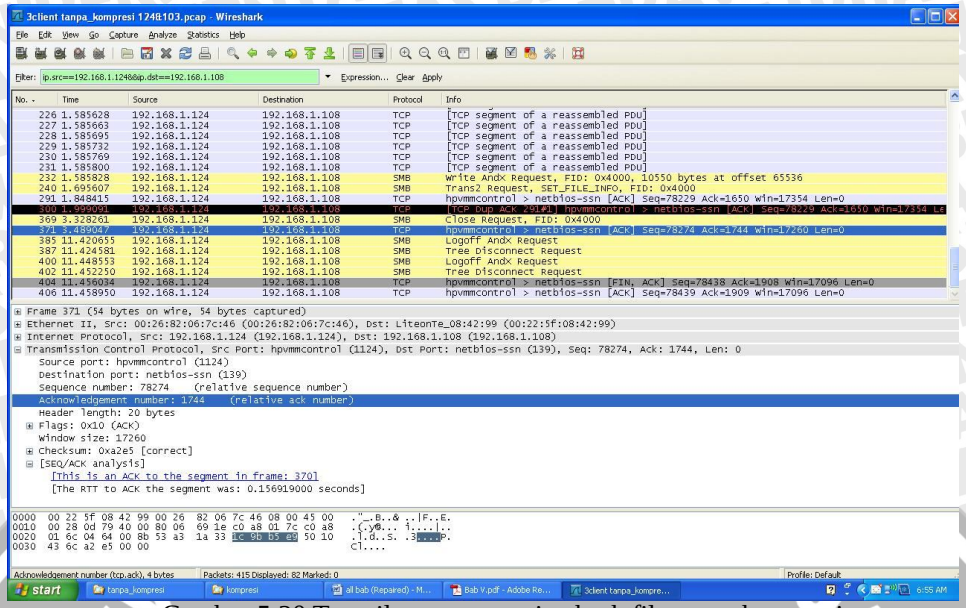
Sumber: Hasil pengujian

Tabel 5.7 Hasil pengujian *delay end to end* dengan kompresi

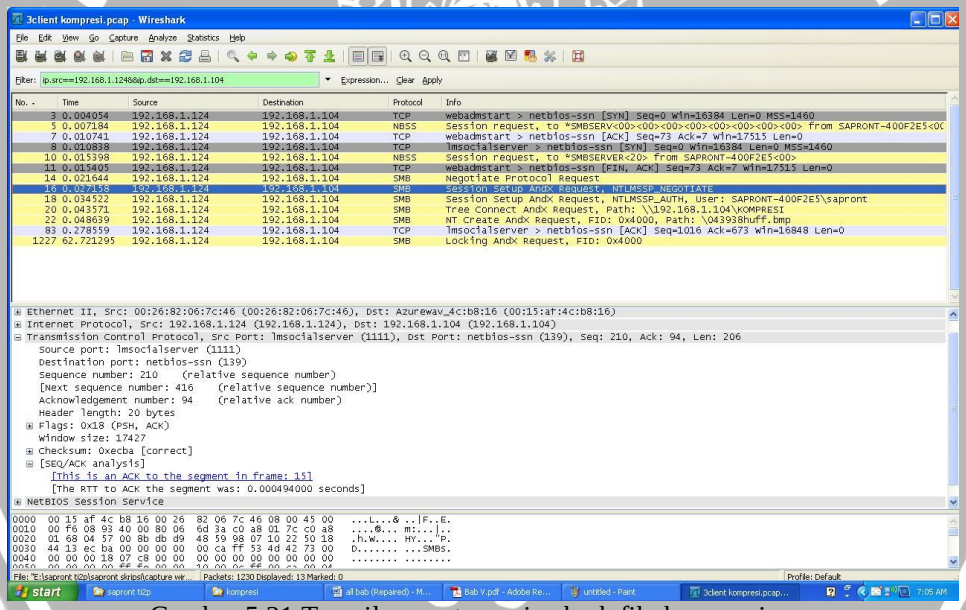
	IP address	**RTT**	**Delay end to end**
Client 1	192.168.1.104	0,001789 s	0,000895 s
Client 2	192.168.1.108	0,002083 s	0,001042 s
Client 3	192.168.1.103	0,00195 s	0,000975 s

Sumber: Hasil pengujian

5. Data hasil *capture* menggunakan *wireshark* dapat dilihat pada gambar berikut:



Gambar 5.20 Tampilan *capture* wireshark file tanpa kompresi
Sumber: Hasil pengujian



Gambar 5.21 Tampilan *capture* wireshark file kompresi
Sumber: Hasil pengujian

5. Salah satu parameter untuk menentukan performansi jaringan adalah *throughput*. *Throughput* menunjukkan jumlah data yang diterima dengan baik pada penerima setela melewati media transmisi. Berikut ini adalah analisis *throughput* untuk transmisi file menggunakan kompresi dengan tidak menggunakan kompresi.

Tabel 5.8 Besar *throughput* pada jaringan menggunakan kompresi

	Jumlah paket	Waktu pengiriman (s)	Throughput (paket/s)
Client 1	10	0,001736 s	5762,028
Client 2	10	0,001625 s	6155,74
Client 3	10	0,001512 s	6615,944

Sumber: hasil pengujian

Tabel 5.9 Besar *throughput* pada jaringan tanpa menggunakan kompresi

	Jumlah paket	Waktu pengiriman (s)	Throughput (paket/s)
Client 1	10	0,000895 s	11179,43
Client 2	10	0,001042 s	9601,536
Client 3	10	0,000975 s	10256,41

Sumber: Hasil pengujian

5.2.5.4 Kesimpulan Hasil Pengujian

Dari data hasil pengujian dapat disimpulkan bahwa kompresi menggunakan Huffman *coding* tidak efektif pada citra yang memiliki kombinasi warna yang tinggi. Sementara pada citra dengan kombinasi warna yang rendah kompresi menggunakan Huffman coding sangat efektif. Berdasarkan pengamatan transmisi data pada jaringan wifi 802.11 b/g semakin kecil nilai RTT semakin kecil pula *delay end to end* yang ditransmisikan. Pada pengujian jaringan wifi 802.11 b/g nilai *throughput* untuk transmisi data menggunakan kompresi lebih besar daripada transmisi data tanpa menggunakan kompresi.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian penerapan Huffman coding untuk kompresi citra pada jaringan wifi 802.11 b/g, dapat disimpulkan bahwa:

1. Interkoneksi antara *server*-AP *router-client* berjalan dengan baik dengan terkoneksinya semua *client* pada jaringan wifi 802.11 b/g. Semua *client* dapat menerima file yang dikirim oleh *server*.
2. Huffman *coding* dapat diterapkan dalam program kompresi dan dekompresi citra pada jaringan wifi 802.11 b/g.
3. Kompresi menggunakan Huffman *coding* efektif untuk citra yang memiliki kombinasi warna yang rendah. Dan sangat tidak efektif untuk citra dengan kombinasi warna yang tinggi.
4. Rasio kompresi yang diperoleh dalam pengurangan bit informasi dalam proses transmisi rata-rata adalah sebesar 28,73% untuk 10 file citra.
5. Dari pengujian performansi jaringan yang telah dilakukan didapatkan hasil sebgai berikut:

	Menggunakan kompresi	**Tanpa menggunakan kompresi**
Jumlah client	3	3
Delay end to end	0,002911 s	0,004872 s
Throughput	31037,38 paket/s	18533,71 paket/s
Kapasitas ukuran file	42,4 KB	74,3 KB
Kualitas	sama	sama

6.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem, antara lain:

1. *Webcam* yang digunakan memiliki resolusi yang lebih besar dan lebih baik agar gambar dapat terlihat dengan baik.
2. Program kompresi dan dekompresi dapat dikembangkan lebih lanjut agar proses kompresi dan dekompresi dapat dilakukan dengan lebih cepat dan dapat mengkompresi citra yang mempunyai resolusi tinggi.
3. Membuat aplikasi *real time* yang lebih baik dan lebih *user interface* pada sisi *server* dan *client*.

DAFTAR PUSTAKA

Munir, Rinaldi. Pengolahan Citra Digital dengan Pendekatan Algoritmik. Bandung: Informatika, 2004.

Fadlisya, Taufiq, Zulfikar & Fauzan . Pengolahan Citra Menggunakan Delphi, Graha Ilmu.Yogyakarta, 2008.

Ahmad Usman, Pengolahan Citra Digital & Teknik Pemrogramanya. Graha Ilmu, Yogyakarta, 2005.

Sigit R,Basuki A, Ramadijanti N, Pramadihanto D, Step by Step Pengolahan Citra Digital. Andi, Yogyakarta 2005

MADCOMS. *Seri Panduan Pemrograman: Pemrograman Borland Delphi 7*. Yogyakarta : CV. ANDI OFFSET, 2003.

Miano, John. Compressed Image File Formats. New York: ACM Pers, 1999.

Sayood, Khalid. Introduction to Data Compression. San Fansisco: Morgan Kaufmann Publisher, 2009.

Sayood, Khalid. 2003. Lossless Compress Handbook. San Diego: Academic Press.

Yuwono, Rudy. ”Studi Penerapan Teknik Companding (*Compressing-Expanding*) Text File dengan Menggunakan Algoritma Huffman pada Jaringan Sistem ATM (*Asynchronous Transfer Mode*) ”. Skripsi Tidak Diterbitkan. Malang: Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, 1997.

Asfahani, Roghib. ”Penerapan Algoritma Slow-Start pada jaringan wifi 802.11 b/g ”. Skripsi Tidak Diterbitkan. Malang: Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, 2009.

S., Indah Choiriyah. ” Perancangan Sistem Monitoring Perangkat Node B (BTS 3G) menggunakan SMS (*Short Message Service*) dan Webcam”. Skripsi Tidak Diterbitkan. Malang: Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, 2009.

Suprapto. *Short Course ITCC Pemrograman Delphi*. Malang : ITCC FT-Unibraw.

Madenda, Sarifuddin. 2008. Kompresi Citra Berwarna Menggunakan Metode Pohon Biner Huffman. http://www.batan.go.id/ppin/lokakarya/LKSTN_10/Sarifuddin1-.pdf (Diakses tanggal 20 Agustus 2009.

LAMPIRAN

LISTING PROGRAM DELPHI

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, jpeg, ExtCtrls,ScktComp, Buttons, StdCtrls,
  ImgList, ComCtrls, ToolWin, Menus, AVServer, AVClient,mmsystem, Gauges,
   CPort, ExtDlgs;

type
  Tcapture = class(TForm)
    server: TAVServer;
    bCall: TSpeedButton;
    bDisconnect: TSpeedButton;
    ClientPanel: TPanel;
    client: TAVClient;
    RequestPanel: TPanel;
    pictureid: TImage;
    usr: TLabel;
    baccept: TButton;
    breject: TButton;
    Bevel1: TBevel;
    remoteaddress: TLabel;
    manicon: TImage;
    Timer1: TTimer;
    Image3: TImage;
    Image4: TImage;
    brecord: TSpeedButton;
    Image5: TImage;
    Image6: TImage;
    filedlg: TOpenDialog;
    Timer3: TTimer;
    Timer2: TTimer;
    Timer4: TTimer;
    Label6: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    TimerCapture: TTimer;
    multicapture: TSpeedButton;
    SpeedStop: TSpeedButton;
    Stop: TLabel;
    huffman: TButton;
    Label1: TLabel;
    OpenDialog1: TOpenDialog;
    image1: TImage;
    SpeedButton1: TSpeedButton;
    SavePictureDialogbmp: TSavePictureDialog;
```

```
Bevel2: TBevel;
LblFileSize1: TLabel;
LblFileName1: TLabel;
LblHasil: TLabel;
LblFileSize2: TLabel;
Bevel3: TBevel;
procedure FormCreate(Sender: TObject);
procedure bCallClick(Sender: TObject);
procedure bDisconnectClick(Sender: TObject);
procedure clientConnectionRefused(Sender: TObject; from_ip: String);
procedure serverConnectionRequest(Sender: TObject; conn_string,
  Remote_Address: String; SocketHandle: Integer);
procedure serverUserLeave(Sender: TObject; user: String);
procedure bacceptClick(Sender: TObject);
procedure brejectClick(Sender: TObject);
procedure serverPictureReceived(Sender: TObject; user: String;
  picture: TJPEGImage);
procedure Timer1Timer(Sender: TObject);
procedure brecordClick(Sender: TObject);
procedure clientConnectionAccepted(Sender: TObject; from_ip: String);
procedure serverClientDisConnect(Sender: TObject;
  Socket: TCustomWinSocket);
procedure Timer2Timer(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Timer4Timer(Sender: TObject);
procedure singlecaptureClick(Sender: TObject);
procedure TimerCaptureTimer(Sender: TObject);
procedure multicaptureClick(Sender: TObject);
procedure SpeedStopClick(Sender: TObject);
procedure huffmanClick(Sender: TObject);
function BinToDec (Biner : string) : LongInt;
function DecToBin (Desimal : LongInt) : String;


private
   { Private declarations }
FSize, FCSize : LongInt;
FCHSize : LongInt;
FCSaveSize : LongInt;
Metode : String;
Buf : array [1..5000000] of Char;
BufC : array [1..5000000] of Char;
FScanlineSize : Integer;
FPSNR : Double;
FOrigSize : Integer;
FPixelCount : Integer;

{ Private declarations }
```

```
  public
    { Public declarations }
    Origin, MovePt: TPoint;
    Buffer:TBitmap;
    ActiveFileName,CurrentFile:string;
    procedure PlaySound(SND : PansiChar);
  end;

Const CRLF = #13+#10;
      CTRZ = #26;

var
  fname : string;
  capture: Tcapture;
  kondisi:Boolean;
  icon1 : boolean;
  jawab, jawab1, jawabMK, data, data_sms,sms_del,DataSerial,sms_dikirim:
string;
  pducmgc: string;
  status:Boolean;
  modestatus,lev,nilai,g,h,i,ii,j,k,l,m,n,w:integer;

implementation

{$R *.dfm}
{$R one.RES}


  function Tcapture.BinToDec(Biner:string):LongInt;
  var
  PanjangKar, b : Integer;
  Desimal : LongInt;
  SatuBit : String;
begin
    Desimal:=0;
    PanjangKar := Length(biner);
    For b:= 1 to PanjangKar do
begin
    SatuBit := copy(Biner,PanjangKar-b+1,1);
    Desimal := Desimal+Trunc((StrToInt(SatuBit))*exp((b-1)*ln(2)));
end;
  BinToDec := Desimal;
end;

function tcapture.DecToBin (Desimal : LongInt) : String;
var
  SisaBagi : Integer;
  DesBin : LongInt;
```

```
  Biner : String;
begin
  Biner :='';
  DesBin := Desimal;
  repeat
  SisaBagi := DesBin mod 2;
  DesBin := DesBin div 2;
  if SisaBagi = 0 then Biner := '0'+ Biner else
  biner :='1'+ biner;
  until DesBin=0;
  DecToBin :=Biner;
  end;


procedure Tcapture.PlaySound(SND : PansiChar);
var
  hFind, hRes: THandle;
  Song : PChar;
begin
 hFind:=FindResource(HInstance, SND, 'WAVE');
 if hFind<>0 then begin
   hRes:=LoadResource(HInstance, hFind);
   if hRes<>0 then begin
     Song:=LockResource(hRes);
     if Assigned(Song) then
       SndPlaySound(Song, SND_ASYNC or snd_Memory);
     UnlockResource(hRes);
   end;
   FreeResource(hFind);
 end;
end;

procedure Tcapture.FormCreate(Sender: TObject);
var host : string;
begin

status:=true;
Application.ProcessMessages;

//client.UserID := txtuser.Text;
ClientPanel.DoubleBuffered := true; (*Required to avaid flickering
                                when diplaying incoming frames*)
//host := server.mGetLocalHostName;
//self.Caption := '1 to 1 Conferencing  - ' + host;
server.DriverIndex := 0;
server.DriverOpen := true;
server.VideoPreview := true;
server.PreviewScaleToWindow := true;
```

```
server.VideoStreamQuality := 10;
server.Listen := true;

Application.ProcessMessages;
bCall.Click;

kondisi:=true;
end;


procedure Tcapture.bCallClick(Sender: TObject);
begin
  client.isActive := true;
end;

procedure Tcapture.bDisconnectClick(Sender: TObject);
begin
  client.isActive := false;
end;

procedure Tcapture.clientConnectionRefused(Sender: TObject;
  from_ip: String);
begin
  Application.MessageBox('Your connection request has been rejected ...', 'Sorry
..', MB_OK+MB_ICONASTERISK+MB_DEFBUTTON1+MB_APPLMODAL);
end;

procedure Tcapture.serverConnectionRequest(Sender: TObject; conn_string,
  Remote_Address: String; SocketHandle: Integer);
begin
 timer1.Enabled := true;
 usr.Caption := conn_string;
 remoteaddress.Caption := Remote_address;
 RequestPanel.Visible := true;
 Application.ProcessMessages;
 baccept.Click;
end;

procedure Tcapture.serverUserLeave(Sender: TObject; user: String);
begin
client.isActive := false;
end;

procedure Tcapture.bacceptClick(Sender: TObject);
begin
timer1.Enabled := false;
RequestPanel.Visible := false;
pictureid.Picture.Bitmap := nil;
```

```
server.mAcceptConnection(usr.Caption,0);

  if client.GetRemoteIP <> remoteaddress.Caption then
  begin
    client.Address := remoteaddress.Caption;
    client.isActive := true;
  end;
end;

procedure Tcapture.brejectClick(Sender: TObject);
begin
 timer1.Enabled := false;
 RequestPanel.Visible := false;
 pictureid.Picture.Bitmap := nil;
 server.mRefuseConnection(usr.Caption,0)
end;

procedure Tcapture.serverPictureReceived(Sender: TObject; user: String;
  picture: TJPEGImage);
begin
pictureid.Picture.Assign(picture);
end;

procedure Tcapture.Timer1Timer(Sender: TObject);
begin
icon1 := not Icon1;
  case Icon1 of
    True: manicon.Picture.bitmap  := Image3.Picture.Bitmap;
    False: manicon.Picture.bitmap := Image4.Picture.Bitmap;
  end;
PlaySound('RING');
end;


procedure Tcapture.clientConnectionAccepted(Sender: TObject;
  from_ip: String);
begin
PlaySound('CONNECTED');
end;

procedure Tcapture.serverClientDisConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
timer1.Enabled := false;
 RequestPanel.Visible := false;
 pictureid.Picture.Bitmap := nil;
end;
```

```
procedure Tcapture.Timer2Timer(Sender: TObject);
begin
status:=true;
Timer2.Enabled:=true;
end;

procedure Tcapture.Button5Click(Sender: TObject);
begin
 Sleep(500);
end;

procedure Tcapture.Timer4Timer(Sender: TObject);
begin
kondisi:=true;
brecord.Click;
Timer4.Enabled:=false;
end;

 procedure Tcapture.brecordClick(Sender: TObject);
begin
if client.recording_video = false then
begin
brecord.Glyph := image5.Picture.Bitmap;
client.StartRecordingFrames(0,4,500);
brecord.Hint := 'Recording incoming video frames to' + client.VideoFileName;
end
else
begin
brecord.Glyph := image6.Picture.Bitmap;
client.Stoprecordingframes;
brecord.Hint := 'Click here to record the incoming video frames to an AVI file';
end;
end;

procedure Tcapture.singlecaptureClick(Sender: TObject);
begin
 PlaySound('SNAP');
 modestatus:=0;
 huffman.Click;
end;

procedure Tcapture.multicaptureClick(Sender: TObject);
begin
TimerCapture.enabled := true ;
end;

procedure Tcapture.TimerCaptureTimer(Sender: TObject);
```

```
var fname : string;
    countCapture, count : integer;
begin
  modestatus:=1;
  TimerCapture.Enabled := False;
  if count = 1 then
    TimerCapture.Enabled := False
    else begin
    PlaySound('SNAP');
    huffman.Click;
    end;
end;

procedure Tcapture.SpeedStopClick(Sender: TObject);
begin
 TimerCapture.Enabled := False;
end;

procedure Tcapture.huffmanClick(Sender: TObject);
  var
  FromF: file;
  NumRead: Integer;
  Save_Cursor:TCursor;
  Metode: Integer;

   i,ii,iii: LongInt;
    u,uu, uuu, NoAlih, JumPindah: Integer;
    KarakterLD0,HighestLevel :integer;
    SigmaAscii: Array[0..255] of LongInt;
    KarakterHuf: array[0..255] of Integer;
    JumlahKarakterHuf: array[0..255] of Integer;
    KodeHuf: array[0..255] of string;
    KodeTem: string;
    JumlahTem, JumlahKiri: Integer;
    Inbuf: Integer;
    AngkaTerkecil: LongInt;
    LevelTree: array[0..1000] of integer;
    FrekTree: array[0..1000] of integer;
    KarTree: array[0..1000] of string;
    JumTree, FrekSource, FrekTarget: Integer;
    Oke, nol: boolean;
     ToF: Textfile;


begin
  Image1.Picture.Bitmap.Assign(client.Picture.Graphic);
  Image1.picture.bitmap.Dormant;
  Image1.picture.bitmap.freeimage;
```

```
image1.Picture.SaveToFile(FormatDateTime('hhmmss',Now) + '.bmp');

 //--------- open file hasil capture

 Save_Cursor := Screen.Cursor;
 Screen.Cursor := crHourglass;    { Show hourglass cursor }
 AssignFile(FromF, (FormatDateTime('hhmmss',Now) + '.bmp'));
 Reset(FromF, 1);    { Record size = 1 }
 Fsize := FileSize(FromF);
  repeat
    BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
  until (NumRead = 0);
  CloseFile(FromF);
 Metode:=0;
 If Buf[1]+Buf[2]+Buf[3]='HUF' then Metode:=3;
 LblFileName1.Caption := 'Nama File : '+(FormatDateTime('hhmmss',Now) +
'.bmp');
 LblFileSize1.Caption := 'Ukuran File = '+IntToStr(FSize)+' bytes';
 Screen.Cursor := Save_Cursor;
 LblHasil.caption:='';
 LblFileSize2.caption:='';

Save_Cursor := Screen.Cursor;
 Screen.Cursor := crHourglass;    { Show hourglass cursor }
 for uu:=0 to 255 do
 Begin
  SigmaAscii[uu]:=0;
  JumlahKarakterHuf[uu]:=0;
  KarakterHuf[uu]:=0;
 end;
 KarakterLD0:=0;

 for i:=0 to Fsize-1 do   //Jumlah tiap karakter
 begin
  Inbuf:=integer(Buf[1+i]);
  SigmaAscii[Inbuf]:=SigmaAscii[Inbuf]+1;
 end;
 for u:=0 to 255 do     //Pengurutan
 begin
  AngkaTerkecil:=1000000;
  for uu:=0 to 255 do
  begin
   if (SigmaAscii[uu]>0) and (SigmaAscii[uu]<AngkaTerkecil) then
   begin
    AngkaTerkecil:=SigmaAscii[uu];
    KarakterHuf[u]:=uu;
    JumlahKarakterHuf[u]:=AngkaTerkecil;
   end;
```

```
    end;
    if JumlahKarakterHuf[u]>0 then
      begin
        KarakterLD0:=KarakterLD0+1;
        SigmaAscii[KarakterHuf[u]]:=0;
      end;
  end;

//------------------- Huffman Tree
  For u:=0 to KarakterLD0-1 do
  Begin
   LevelTree[u]:=1;
   KarTree[u]:=chr(KarakterHuf[u]);
   FrekTree[u]:=JumlahKarakterHuf[u];
  end;

  JumTree:=KarakterLD0;
  For u:=1 to KarakterLD0-1 do          //n-1 kali
  Begin
   oke:=false;
   uuu:=0;
   repeat    //cari target
     inc(uuu);
     if LevelTree[uuu]=1 then
     begin
       for uu:=JumTree-1 downto 0 do
       begin
         LevelTree[uu+1]:=LevelTree[uu];
         KarTree[uu+1]:=KarTree[uu];
         FrekTree[uu+1]:=FrekTree[uu];
       end;
       LevelTree[0]:=1;
       KarTree[0]:=KarTree[1]+KarTree[uuu+1];
       FrekTree[0]:=FrekTree[1]+FrekTree[uuu+1];
       inc(JumTree);
       for uu:=1 to uuu do          //cabang pohon source
       begin
         inc(LevelTree[uu]);
       end;
       uu:=1;
       repeat                   //target & cabang pohon target
         inc(LevelTree[uu+uuu]);
         inc(uu);
       until (LevelTree[uu+uuu]=1) or (uu+uuu>JumTree);
       oke:=true;
     end;
   until oke=true;
```

```
      FrekSource:=FrekTree[0];   //urut
      NoAlih:=0;
      For uuu:=1 to Jumtree-1 do
      begin
        FrekTarget:=FrekTree[uuu];
        if (FrekSource>FrekTarget) and (LevelTree[uuu]=1) then NoAlih:=uuu;
      end;
      If NoAlih>0 then
      begin
        uuu:=0;
        JumPindah:=0;
        repeat
          LevelTree[JumTree+uuu]:=LevelTree[uuu];
          KarTree[JumTree+uuu]:=KarTree[uuu];
          FrekTree[JumTree+uuu]:=FrekTree[uuu];
          Inc(JumPindah);
          inc(uuu);
        until levelTree[uuu]=1;
        repeat
          LevelTree[uuu-JumPindah]:=LevelTree[uuu];
          KarTree[uuu-JumPindah]:=KarTree[uuu];
          FrekTree[uuu-JumPindah]:=FrekTree[uuu];
          inc(uuu);
        until (LevelTree[uuu]=1) and (FrekTree[uuu]>=FrekSource);
        For uu:=1 to JumPindah do
        begin
          LevelTree[uuu-JumPindah+uu-1]:=LevelTree[JumTree+uu-1];
          KarTree[uuu-JumPindah+uu-1]:=KarTree[JumTree+uu-1];
          FrekTree[uuu-JumPindah+uu-1]:=FrekTree[JumTree+uu-1];
        end;
      end;
    end;

//---------------- Huffman Code
    HighestLevel:=1;
    For uu:=0 to JumTree-1 do
    Begin
      If LevelTree[uu]>HighestLevel then HighestLevel:= LevelTree[uu];
    end;
    For uu:=2 to HighestLevel do  //mulai level 2
    Begin
      Nol:=true;
      For ii:=0 to JumTree-1 do
      Begin
        if LevelTree[ii]=uu then
        begin
          for uuu:=1 to length(KarTree[ii]) do
          begin
```

```
          KodeTem:=copy(KarTree[ii],uuu,1);
          NoAlih:=0;
          for FrekTarget:=0 to KarakterLD0-1 do
          begin
            if KodeTem=chr(KarakterHuf[FrekTarget]) then NoAlih:=FrekTarget;
          end;
          case nol of
            true: KodeHuf[NoAlih]:=KodeHuf[NoAlih]+'0';
            false: KodeHuf[NoAlih]:=KodeHuf[NoAlih]+'1';
          end;
        end;
        nol:=not(nol);
      end;
    end;
  end;


//-------- Hasil Pemampatan
  BufC[1]:='H';
  BufC[2]:='U';
  BufC[3]:='F';
  KodeTem:=InttoHex(FSize,6);
  BufC[4]:=chr(StrtoIntDef('$'+copy(KodeTem,1,2),255));  //jumlah byte
  BufC[5]:=chr(StrtoIntDef('$'+copy(KodeTem,3,2),255));
  BufC[6]:=chr(StrtoIntDef('$'+copy(KodeTem,5,2),255));
  BufC[7]:=chr(KarakterLD0-1);
  FCHSize:=7;
  For uu:=1 to KarakterLD0 do  // karakter, kode, panjang bit
  begin
    BufC[(uu-1)*4+8]:=chr(KarakterHuf[uu-1]);
    BufC[(uu-
1)*4+9]:=chr(StrtoIntDef('$'+copy(IntToHex(BinToDec(KodeHuf[uu-
1]),4),1,2),225));
    BufC[(uu-
1)*4+10]:=chr(StrtoIntDef('$'+copy(IntToHex(BinToDec(KodeHuf[uu-
1]),4),3,2),225));
    BufC[(uu-1)*4+11]:=chr(Length(KodeHuf[uu-1]));
    FCHSize:=FCHSize+4;
  end;
  KodeTem:='';
  For ii:=1 to FSize do   // isi file
  begin
    uu:=-1;
    repeat
      inc(uu);
    until chr(KarakterHuf[uu])=Buf[ii];
    KodeTem:=KodeTem+KodeHuf[uu];
    if length(KodeTem)>=8 then
```

```
  begin
    repeat
      Inc(FCHSize);
      BufC[FCHSize]:=chr(BinToDec(copy(KodeTem,1,8)));
      If Length(KodeTem)>8 then
KodeTem:=copy(KodeTem,9,Length(KodeTem)-8) else KodeTem:='';
    until Length(KodeTem)<8;
   end;
  end;
  If Length(KodeTem)>0 then  //sisa
  begin
   KodeTem:=KodeTem+StringofChar('0',8-Length(KodeTem));
   Inc(FCHSize);
   BufC[FCHSize]:=chr(BinToDec(KodeTem));
  end;
  LblFileSize2.caption:=inttostr(FCHSize)+' bytes  = '+inttostr(FCHSize*100 div
FSize)+' %';
   Screen.Cursor := Save_Cursor;
   //SButSave.caption:='Save with Huffman Algorithm';
   FCSaveSize:=FCHSize;
   //SButSave.enabled:=true;


     AssignFile(ToF, ('\\192.168.1.4\coba\'+FormatDateTime('hhmmss',Now) +
'huff.bmp'));
      Rewrite(ToF);      { Record size = 1 }
     for ii:=1 to FCSaveSize do

     AssignFile(ToF, ('\\192.168.1.1\coba\'+FormatDateTime('hhmmss',Now) +
'huff.bmp'));
      Rewrite(ToF);      { Record size = 1 }
     for ii:=1 to FCSaveSize do
      begin
       Write(ToF, bufC[ii]);
      end;
      CloseFile(ToF);


  //---------Deteksi status mode : mode Singglecapture atau Multicapture-----
   if modestatus=1 then
  begin
   TimerCapture.Enabled := True;
   end;
end;

end.
```