BAB 2

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Teknik kompresi digunakan untuk memperkecil ukuran suatu *file*, sehingga saat disimpan memerlukan memory yang lebih kecil dari *file* aslinya, misalnya suatu citra berwarna. Berkembangnya teknik kompresi citra saat ini memberikan suatu hasil yang bervariasi tergantung dari teknik yang digunakan untuk mengkompresi suatu citra tersebut. Salah satu teknik yang digunakan untuk mengkompresi citra yaitu dengan menggunakan jaringan saraf tiruan.

Solaiman B. dan Maillard E. (1995), mengimplementasikan jaringan syaraf tiruan HLVQ yang mengkombinasikan *supervised* dan *unsupervised learning* dalam proses *Learning Vector Quantization*. Kualitas hasil rekonstruksi citra dari teknik ini lebih bagus dari algoritma SOFM.

Jurnal yang berjudul "*Image Compression by Self-Organized Kohonen Map*", Amerijckx C. dkk (1998), menggunakan arsitektur Kohonen tidak hanya pada fitur *Vector Quantization* tetapi juga pada topologi propertinya. Hal ini memungkinkan peningkatan rasio kompresi sampai sekitar 80%. Kualitas dari citra hasil kompresi dengan teknik ini lebih baik daripada JPEG untuk rasio kompresi diatas 30%.

Harandi M. dan Gharavi-Alkhansari M. (2003), mengemukakan algoritma kompresi citra *low bitrate* berbasis Kohonen Self-Organized Maps. Teknik kompresinya berdasarkan pada *Vector Quantization* (VQ) dari koefisien DCT pada blok citra, dimana VQ diimplementasikan oleh jaringan Kohonen. Algoritma ini menunjukkan hasil subjektif yang lebih baik dibandingkan dengan JPEG.

HGANN merupakan penggabungan antara algoritma genetika dan *back-propagation*. Teknik ini diimplementasikan untuk kompresi citra *fractal* oleh Chakrapani Y. dan Soundera Rajan K. (2009) dan dipublikasikan pada jurnalnya yang berjudul "*Implementation of*

Fractal Image Compression Employing Hybrid Genetic-Neural Approach". Hasilnya menunjukkan bahwa teknik ini berhasil diimplementasikan untuk kompresi citra berwarna dan citra *gray-scale*.

2.2. Landasan Teori

2.2.1. Kompresi Citra

Kompresi Citra adalah aplikasi kompresi data yang dilakukan terhadap citra digital dengan tujuan untuk mengurangi redundansi yang terdapat pada citra digital (Salomon, 2004, hal.256). Sehingga *memory* yang dibutuhkan untuk menyimpannya data menjadi lebih kecil maka dapat menghemat kebutuhan media penyimpanan dan mempercepat waktu pengiriman. Teknik kompresi dibedakan menjadi dua, yaitu sebagai berikut :

- Teknik Kompresi *Lossy*

  Ukuran *file* citra menjadi lebih kecil dengan menghilangkan beberapa informasi dalam citra asli. Teknik ini mengubah detail dan warna pada citra digital menjadi lebih sederhana tanpa terlihat adanya perbedaan yang mencolok dalam pandangan manusia sehingga ukurannya menjadi lebih kecil. Biasanya digunakan pada citra foto atau *image* lain yang tidak terlalu memerlukan detail citra, dimana kehilangan bit rate foto tidak berpengaruh pada citra.

  Rasio kompresi yang dihasilkan oleh metode ini lebih tinggi daripada teknik *lossless*.

- Teknik Kompresi *Lossless*

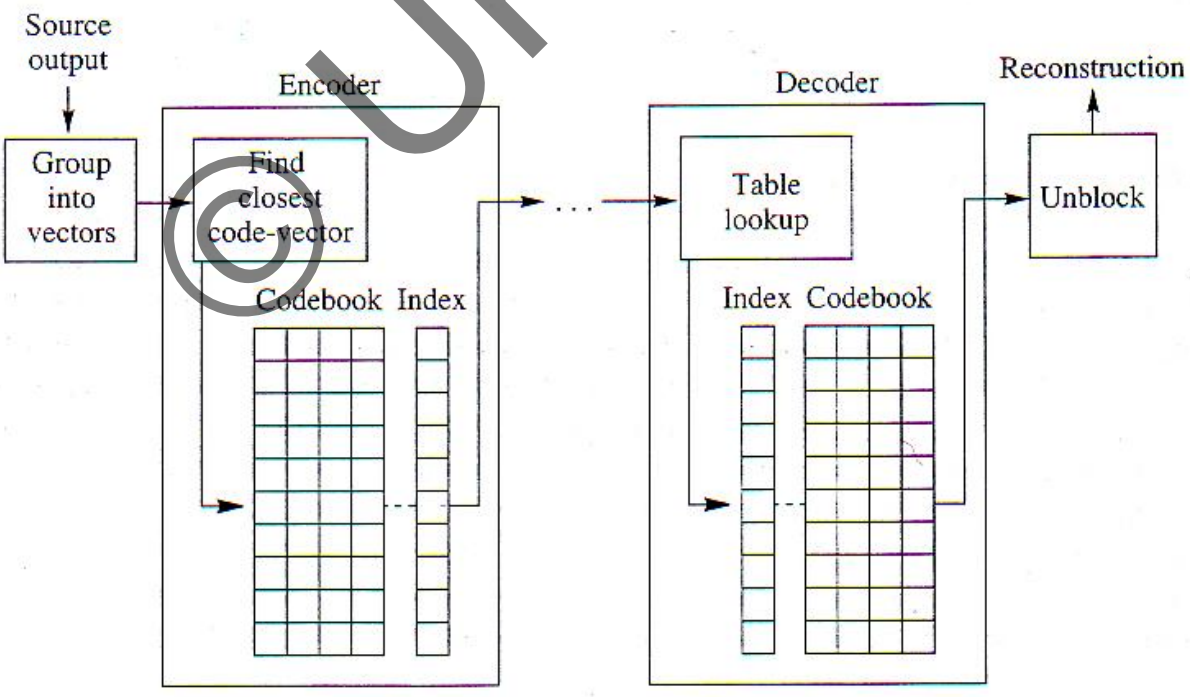
  Teknik kompresi dimana hasil dekompresi sama seperti citra asli tanpa adanya informasi yang hilang. Teknik kompresi ini digunakan pada citra yang mengandung informasi penting seperti citra hasil diagnosa medis.

Perhitungan rasio kompresi dapat dilakukan dengan rumus berikut :

$$\text{Rasio kompresi} = 100\% - \left(\frac{\text{ukuran citra hasil kompresi}}{\text{kuran citra asli}} \text{ x } 100\%\right) \quad \ldots\ldots (2-1)$$

2.2.2. Kuantisasi Vektor

Kuantisasi merupakan salah satu metode kompresi *lossy* dimana terdapat informasi yang hilang tetapi tidak terlihat perbedaan yang mencolok dalam pandangan manusia (Salomon, 2004, hal. 274). Citra dikelompokkan menjadi beberapa blok *pixel* dengan ukuran yang sama (vektor) dan disimpan dalam sebuah tabel warna yang disebut *codebook*. Setiap blok *pixel* pada citra yang akan dikompresi dibandingkan nilainya dengan semua nilai *pixel* yang ada pada *codebook* dan dicari nilai kesamaan yang terdekat. Masing-masing blok *pixel* ini kemudian memiliki informasi *pointer* yang menunjuk ke indeks dari *codebook* dan disimpan sebagai hasil kompresi. Untuk lebih jelasnya dapat dilihat pada Gambar 2.1.



Gambar 2.1. Proses Kuantisasi Vektor (Sayood, 2006, hal. 274)

2.2.3. Pengukuran *Error*

Standar untuk pengukuran *error* ini adalah sebagai berikut :

- *Mean Square Error (MSE)*

  Dihitung untuk menentukan jumlah *error* diantara citra asli dan citra hasil dekompresi, semakin kecil nilai MSE maka *error* yang dihasilkan semakin kecil juga. Rumus matematika untuk menghitung MSE sebagai berikut :

$$MSE = \frac{1}{mn}\sum_{y=0}^{m}\sum_{x=0}^{n}\left(\frac{(IR_{(x,y)} - IR'_{(x,y)})^2 + (IG_{(x,y)} - IG'_{(x,y)})^2 + (IB_{(x,y)} - IB'_{(x,y)})^2}{3}\right) \ldots (2-2)$$

  dengan :

  m,n adalah dimensi citra

  I adalah nilai pixel citra asli

  I' adalah nilai pixel citra hasil rekonstruksi

- *Peak Signal to Noise Ratio (PSNR)*

  Digunakan untuk menghitung *peak error*. Semakin besar nilai PSNR berarti citra hasil dekompresi semakin menyerupai citra asli. Satuan dari PSNR adalah *decibels (dB)*. Rumus matematika untuk menghitung PSNR sebagai berikut :

$$PSNR = 20\ \log_{10}\left(\frac{255}{\sqrt{MSE}}\right) \qquad \ldots\ldots\ldots\ldots (2-3)$$

2.2.4. Algoritma Huffman

Algoritma Huffman dikembangkan oleh David Huffman pada tahun 1952. Algoritma Huffman disebut juga algoritma prefix, dimana setiap kode yang dihasilkan tidak akan menghasilkan kode yang sama dari setiap karakter yang muncul. Algoritma ini didasarkan pada frekuensi kemunculan tiap symbol. Algoritma Huffman akan

menghasilkan *codebook* yang dihasilkan dari pembentukan pohon Huffman. Pembentukan pohon Huffman dilakukkan secara bottom-up.

Secara umum Algoritma Huffman diterangkan sebagai berikut :

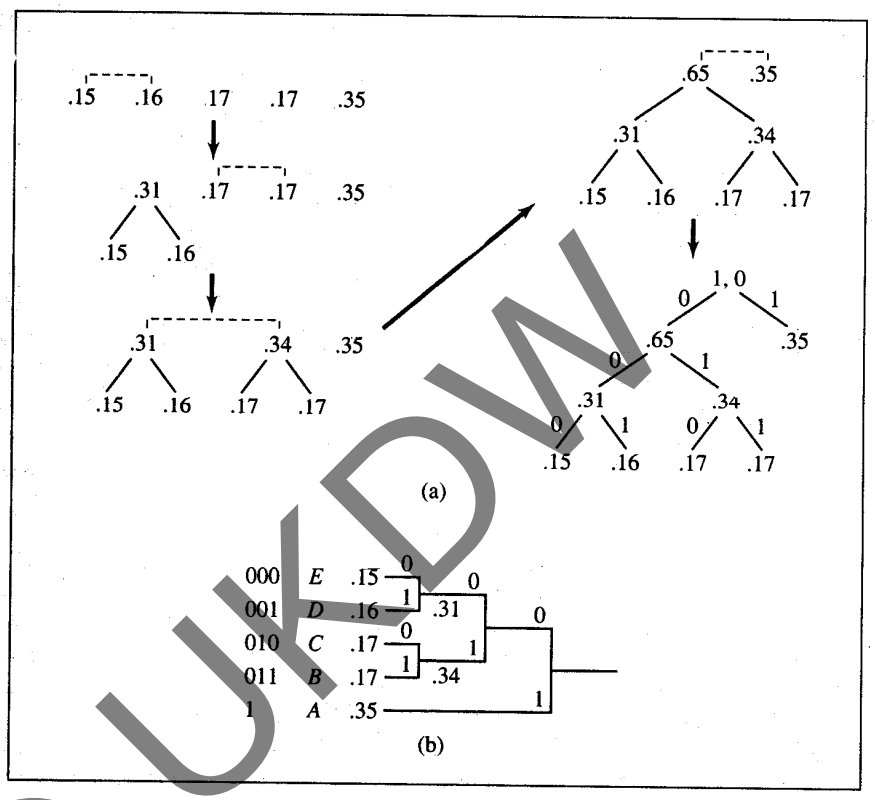
1. Hitung masing-masing kemunculan symbol.
2. Urutkan frekuensi kemunculan symbol dari yang terbesar, masing-masing symbol ini dapat direpresentasikan sebagai sebuah *node*.
3. Gabungkan *node* dengan jumlah frekuensi kemunculan symbol terkecil pertama dan kedua serta tambahkan kode ‘0’ untuk *node* pertama dan kode ‘1’ untuk *node* kedua. Kemudian jumlahkan kedua frekuensi kemunculan simbolnya, maka akan menghasilkan sebuah *node parent*.
4. Masukkan *node parent* kedalam kumpulan *node* dan urutkan berdasarkan jumlah frekuensi kemunculan symbol dari yang terkecil ke yang terbesar.
5. Hapus *node* yang jumlah frekuensi kemunculan symbol terkecil pertama dan kedua dari kumpulan *node*.
6. Ulangi langkah kedua sampai keempat hingga semua symbol habis dibangkitkan.
7. Subtitusi simbol-simbol yang ada dengan kode yang telah dihasilkan.

Contoh *Huffman Tree* :

1. Diketahui beberapa symbol dengan peluangnya yakni :

Xi	Pi
A	**0.35**
B	**0.17**
C	**0.17**
D	**0.16**
E	**0.15**

2. Pembentukan *Huffman* tree

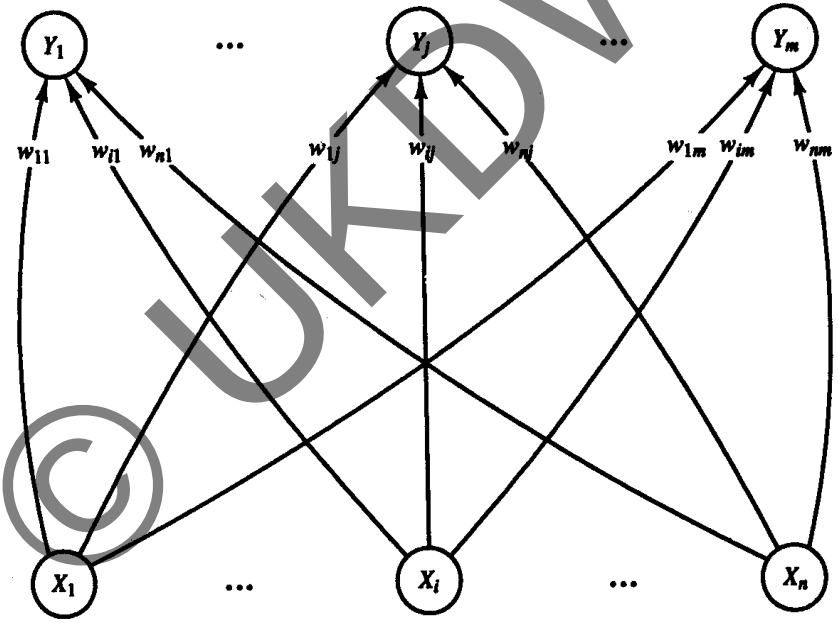


2.2.5. Kohonen Self-Organizing Map

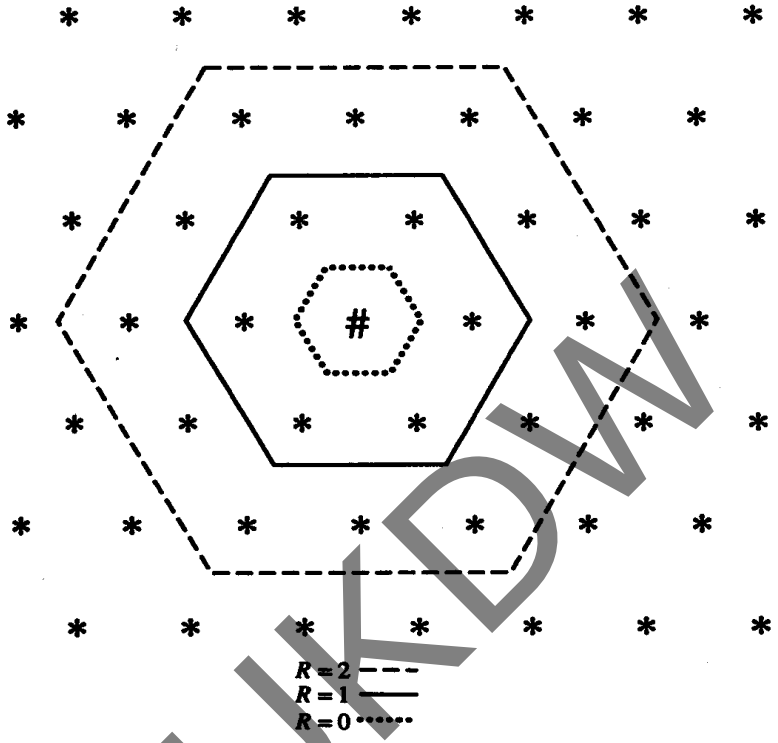
Kohonen Self-Organizing Map diperkenalkan pertama kali pada tahun 1982 oleh Profesor Teuvo Kohonen (Kohonen, 2000, hal. 106). Arsitektur ini banyak dipakai untuk membagi pola masukan kedalam beberapa kelompok. Salah satu hal menarik dari arsitektur ini adalah SOM tidak membutuhkan pengawasan untuk mengklasifikasikan data, karena itu arsitektur ini termasuk dalam *unsupervised learning*. Pada arsitektur ini, lapisan masukan terhubung secara penuh dengan lapisan keluaran dan pada hubungan tersebut terdapat nilai bobot. Selama proses pelatihan, jaringan menerima pola masukan yang berbeda, mencari fitur signifikan pada pola tersebut dan belajar bagaimana

mengklasifikasikan data masukan kedalam kategori yang tepat. Kohonen SOM memakai sistem *competitive learning*, karena itu unit *cluster* yang bobotnya paling mirip dengan pola masukan dipilih sebagai pemenang dan dimodifikasi bobotnya beserta bobot tetangganya. Topologi tetangga yang akan dimodifikasi berbentuk topologi satu dimensi, dua dimensi dan hexagonal. Arsitektur Kohonen Self-Organizing Map dapat dilihat pada Gambar 2.2.

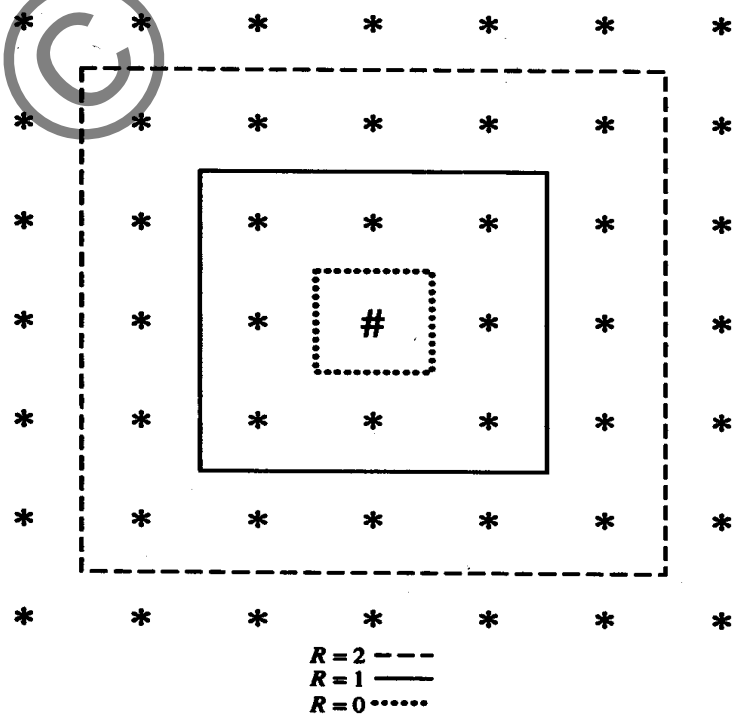
Bentuk topologi tetangga pada arsitektur Kohonen SOM dapat dilihat pada Gambar 2.3, Gambar 2.4 dan Gambar 2.5. Unit pemenang ditandai dengan simbol "#" sedangkan unit lainnya dengan symbol "*".



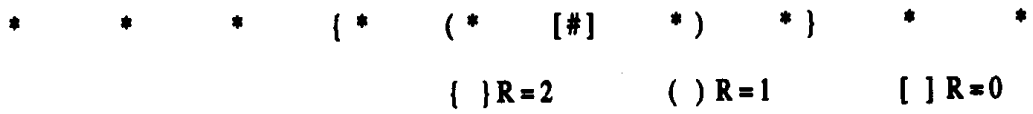
Gambar 2.2. Arsitektur Kohonen Self-Organizing Map (Fausset, 1994, hal. 170)



Gambar 2.3. Topologi Hexagonal (Fausset, 1994, hal. 171)



Gambar 2.4. Topologi Dua Dimensi (Fausset, 1994, hal. 171)



Gambar 2.5. Topologi Satu Dimensi (Fausset, 1994, hal. 170)

Simbol “x” pada Gambar 2.2 merupakan vektor masukan dimana “n” adalah jumlah data yang terdapat dalam vektor tersebut. Setiap nilai pada vektor masukan terhubung dengan unit keluaran yang ditandai dengan simbol “y” dimana “m” adalah jumlah kelompok yang akan dibentuk. Pada hubungan ini terdapat bobot yang ditandai dengan simbol “w”. Vektor bobot ini juga memiliki “n” jumlah data yang sama seperti vektor masukan dan memiliki “m” buah vektor dimana vektor inilah yang kemudian menjadi hasil pengelompokkan dari arsitektur ini.

2.2.5.1. Algoritma Kohonen SOM

Algoritma dari arsitektur ini adalah sebagai berikut :

1. Inisialisasi
   - Bobot dengan nilai acak 0.0 – 1.0
   - *Learning rate* (α) dengan $0 < \alpha < 1$
   - Bentuk dan jari-jari (σ) topologi tetangga
2. Selama kondisi perulangan bernilai benar, lakukan langkah 3-7
3. Untuk setiap vektor masukan x, lakukan langkah 3-5
4. Hitung :

$$D(j) = \sum_i (w_{ji} - x_i)^2 \qquad \ldots\ldots(2-4)$$

dengan:

j adalah indeks keluaran

x_i adalah masing-masing data pada vektor masukan

w adalah bobot penghubung

5. Tentukan indeks j dimana D(j) adalah nilai terkecil dari semua j
6. Modifikasi bobot unit j tersebut dan tetangganya dengan

$$w_{ji}^{baru} = w_{ji}^{lama} + \propto \left(x_i - w_{ji}^{lama}\right)$$

7. Modifikasi *learning rate* dan jari-jari topologi tetangga pada waktu tertentu dengan fungsi :

   - *Learning rate* :

   $$\propto_{baru} = \propto_{awal} x\ exp^{(-t/i)} \qquad \ldots.(2-5)$$

   dengan :

   - t adalah iterasi sekarang
   - i adalah jumlah iterasi

   - Jari-jari :

   $$\sigma_{baru} = \sigma_{awal}\ x\ exp^{(-t/\lambda)} \qquad \ldots.(2-6)$$

   dengan :

   - t adalah iterasi sekarang
   - λ adalah nilai jumlah iterasi dibagi $^{e}\log \sigma_0$

8. Uji kondisi perulangan

Learning rate dan jari-jari tetangga akan dikurangi nilainya selama proses pelatihan. Kondisi perulangan dapat berupa jumlah iterasi yang sudah ditentukan atau bobot hasil pelatihan sudah tidak mengalami perubahan (konvergen).

2.2.5.2. Contoh Pelatihan Kohonen SOM

Diketahui 4 kelompok vector untuk dikelompokkan menjadi 2 kelompok vector, yaitu :

- X(1) = (1,1,0,0) ; X(3) = (1,0,0,0)

- X(2) = (0,0,0,1) ; X(4) = (0,0,1,1)

Dengan nilai jari-jari dan learning rate :

$R=0 ; \propto (t) = 0.6 ; \propto (t+1) = 0.5 * \propto (t)$

dengan : t adalah iterasi

alisasi bobot awal : $\begin{bmatrix} w_{11}=0.2 & w_{12}=0.6 & w_{13}=0.5 & w_{14}=0.9 \\ w_{21}=0.8 & w_{22}=0.4 & w_{23}=0.7 & w_{24}=0.3 \end{bmatrix}$

Jumlah iterasi : 100

✓ Iterasi - 1

- Vektor x(1) = (1,1,0,0)

  $D(1) = (0.2-1)^2 + (0.6-1)^2 + (0.5-0)^2 + (0.9-0)^2 = 1.86$

  $D(2) = (0.8-1)^2 + (0.4-1)^2 + (0.7-0)^2 + (0.3-0)^2 = 0.98$

  D(j) terkecil adalah D(2), maka j = 2

  Modifikasi bobot :

  $w_{21}^{baru} = 0.8 + 0.6(1-0.8) = 0.92 ; \quad w_{22}^{baru} = 0.4 + 0.6(1-0.4) = 0.76$

  $w_{23}^{baru} = 0.7 + 0.6(0-0.7) = 0.28 ; \quad w_{24}^{baru} = 0.3 + 0.6(0-0.3) = 0.98$

  Bobot baru : $\begin{bmatrix} w_{11}=0.2 & w_{12}=0.6 & w_{13}=0.5 & w_{14}=0.9 \\ w_{21}=0.92 & w_{22}=0.76 & w_{23}=0.28 & w_{24}=0.12 \end{bmatrix}$

- Vektor x(2) = (0,0,0,1)

  $D(1) = (0.2-0)^2 + (0.6-0)^2 + (0.5-0)^2 + (0.9-1)^2 = 0.66$

  $D(2) = (0.92-0)^2 + (0.76-0)^2 + (0.28-0)^2 + (0.12-1)^2 = 2.28$

  D(j) terkecil adalah D(1), maka j = 1

  Modifikasi bobot :

  $w_{21}^{baru} = 0.8 + 0.6(1-0.8) = 0.92 ; \quad w_{22}^{baru} = 0.4 + 0.6(1-0.4) = 0.76$

  $w_{23}^{baru} = 0.7 + 0.6(0-0.7) = 0.28 ; \quad w_{24}^{baru} = 0.3 + 0.6(0-0.3) = 0.98$

  Bobot baru : $\begin{bmatrix} w_{11}=0.2 & w_{12}=0.6 & w_{13}=0.5 & w_{14}=0.9 \\ w_{21}=0.92 & w_{22}=0.76 & w_{23}=0.28 & w_{24}=0.12 \end{bmatrix}$

- Vektor x(3) = (1,0,0,0)

  $D(1) = (0.08-1)^2 + (0.24-0)^2 + (0.2-0)^2 + (0.96-0)^2 = 1.87$

  $D(2) = (0.92-1)^2 + (0.76-0)^2 + (0.28-0)^2 + (0.12-0)^2 = 0.68$

  D(j) terkecil adalah D(2), maka j = 2

  Modifikasi bobot :

w_{23}^{baru} = 0.28 + 0.6(0-0.28) = 0.112; w_{24}^{baru} = 0.12 + 0.6(0-0.12) = 0.048

Bobot baru : $\begin{bmatrix} w_{11}=0.08 & w_{12}=0.24 & w_{13}=0.2 & w_{14}=0.96 \\ w_{21}=0.968 & w_{22}=0.304 & w_{23}=0.112 & w_{24}=0.048 \end{bmatrix}$

- Vektor x(4) = (0,0,1,1)

  D(1) = $(0.08-0)^2 + (0.24-0)^2 + (0.2-1)^2 + (0.96-1)^2 = 0.7056$

  D(2) = $(0.968-0)^2 + (0.304-0)^2 + (0.112-1)^2 + (0.048-1)^2 = 2.724$

  D(j) terkecil adalah D(1), maka j = 1

  Modifikasi bobot :

  w_{11}^{baru} = 0.08+0.6(0-0.08)= 0.032 ; w_{12}^{baru} = 0.24+0.6(0-0.24)= 0.096

  w_{13}^{baru} = 0.2+0.6(1-0.2) = 0.68 ; w_{14}^{baru} = 0.96+0.6(1-0.96) = 0.984

  Bobot akhir iterasi - 1:

  $\begin{bmatrix} w_{11}=0.032 & w_{12}=0.096 & w_{13}=0.68 & w_{14}=0.984 \\ w_{21}=0.968 & w_{22}=0.304 & w_{23}=0.112 & w_{24}=0.048 \end{bmatrix}$

Modifikasi *learning rate* : $\propto(2) = 0.5 * 0.6 = 0.3$

✓ Iterasi – 2

Setelah melakukan perhitungan, bobot pada iterasi kedua adalah sebagai berikut:

$\begin{bmatrix} w_{11}=0.0053 & w_{12}=-0.17 & w_{13}=0.7 & w_{14}=1 \\ w_{21}=0.99 & w_{22}=0.3 & w_{23}=0.02 & w_{24}=0.086 \end{bmatrix}$

Perhitungan dilanjutkan hingga iterasi ke-100

✓ Iterasi – 100

Bobot akhir pada iterasi ke-100 adalah :

$\begin{bmatrix} w_{11}=6.7e^{-17} & w_{12}=2e^{-16} & w_{13}=0.51 & w_{14}=1 \\ w_{21}=1 & w_{22}=0.49 & w_{23}=2.3e^{-16} & w_{23}=1e^{-16} \end{bmatrix}$

Nilai bobot ini kemudian dibulatkan.

Dari perhitungan ini didapat dua kelompok vector sebagai berikut:

y(1) = (0, 0, 0.5, 1)

y(2) = (1, 0.5 ,0 ,0)

Hasil pengelompokan vector ini nantinya akan disimpan sebagai tabel warna atau *codebook.*

Setelah *codebook* selesai dibuat akan dilakukan proses klasifikasi piksel terhadap citra *inputan* dengan menggunakan rumus *Euclidean Distance* :

$$ed(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \qquad \ldots\ldots(3-3)$$

dengan :

- ed adalah nilai *euclidean distance* antara p dan q
- p adalah nilai warna pada citra masukkan
- q adalah warna pada *codebook*

Klasifikasi piksel ini berguna untuk mencari jarak terkecil citra *inputan* terhadap *codebook*, setelah itu dilanjutkan dengan proses kompresi menggunakan *Huffman Coding.*

BAB 3

PERANCANGAN SISTEM

Perancangan sistem perlu dibuat sebagai titik awal dimana program akan dibuat dan dirancang. Perancangan sistem dapat diartikan sebagai proses untuk mempersiapkan kebutuhan yang diperlukan dalam rangka implementasi dengan tujuan untuk memberikan gambaran mengenai *interface* program, *input*, proses Algoritma Kohonen SOM dan *Huffman Coding*, dan *output* program.

3.1 Spesifikasi *Hardware* dan *Software*

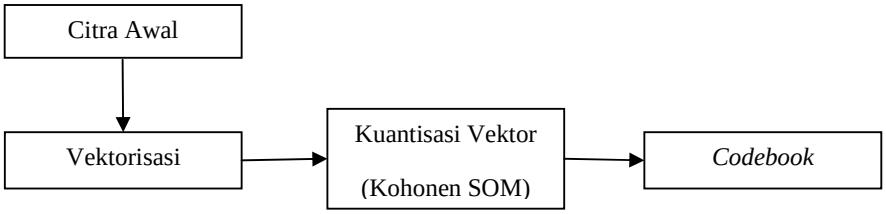
Spesifikasi yang digunakan dalam membangun dan menjalankan sistem ini adalah :

- Hardware, yaitu :
  - *processor* **Intel I5,2.2 GHz**
  - *memory (RAM)* 2Gb
  - *harddisk* 2Gb
- Software, yaitu :
  - Sistem operasi **Windows XP Ultimate**
  - Bahasa pemrograman **Microsoft Visual Basic.net 2008**

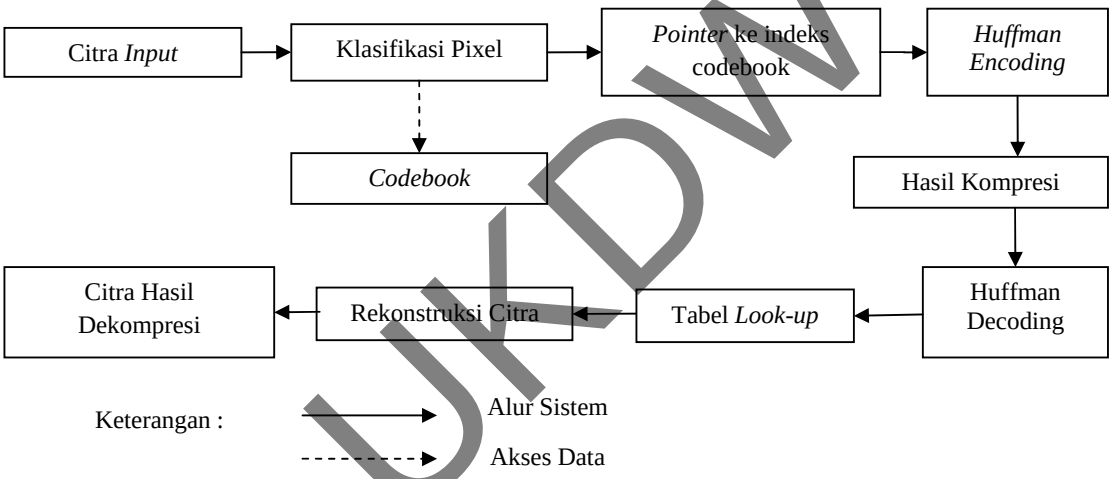
3.2 Rancangan Kerja Sistem

Rancangan kerja sistem diawali dengan proses kuantisasi vektor dengan menggunakan Kohonen SOM, dimana warna tersebut dibagi kedalam kelompok yang lebih kecil dan hasil dari pengelompokan warna itu akan disimpan sebagai *codebook*. Pada citra *input*, tiap piksel warna akan diklasifikasikan dengan setiap warna pada *codebook*. Indeks pada *codebook* yang memiliki nilai terdekat dengan citra *input* tersebut akan disimpan sebagai *file* hasil kompresi. Proses terbentuknya

codebook dapat dilihat pada Gambar 3.1. Gambar 3.2 adalah proses kompresi dan dekompresi.



Gambar 3.1 Proses Pembuatan *Codebook*



Gambar 3.2 Proses Kompresi dan Dekompresi

3.2.1 Algoritma Pembuatan *Codebook*

Proses awal dalam sistem ini dimulai dengan pembuatan *codebook* dengan menggunakan Kohonen SOM. *Codebook* yang terbentuk melalui pelatihan Kohonen SOM ini akan disimpan disuatu tempat dan akan digunakan untuk proses selanjutnya yaitu proses kompresi. Algoritma dalam pembuatan *codebook* dapat dilihat sebagai berikut :

1. Mulai
2. Masukkan citra awal 24-bit
    a. Proses inisialisasi nilai RGB sebagai vektor masukkan untuk arsitektur Kohonen SOM
    b. Menentukan jumlah warna (n) yang akan disimpan pada *codebook*
    c. Menentukan nilai iterasi (i), jari-jari tetangga (σ), dan learning rate (α)
    d. Bentuk topologi tetangga yang digunakan adalah dua dimensi
    e. Inisialisasi vektor bobot awal (w) sebanyak n
3. Proses pelatihan pada Kohonen SOM sebanyak i iterasi
    a. Perhitungan jarak (D)
    b. Menentukan indeks yang memiliki nilai D terkecil sebagai pemenang
    c. Modifikasi bobot pemenang dan tetangganya
4. Modifikasi learning rate dan jari-jari tetangga
    a. Pengurangan learning rate dengan fungsi
    b. Pengurangan jari-jari dengan fungsi
5. Menyimpan hasil akhir bobot pelatihan sebagai daftar warna pada *codebook*
6. Selesai

3.2.2 Algoritma Proses Kompresi

Proses kompresi dilakukan setelah proses pembuatan *codebook* selesai. Proses kompresi ini dilakukan dengan tujuan untuk mengkompres suatu citra menjadi *file* baru dengan ukuran yang lebih kecil. Dimana proses ini diawali dengan pengklasifikasian piksel citra inputan terhadap *codebook* yang sudah terbentuk dengan menghitung jarak terdekat dengan menggunakan rumus *Euclidean Distance*, setelah itu baru diproses dengan

metode *Huffman encoding*. Untuk lebih jelasnya dapat dilihat pada algoritma dibawah ini :

1. Mulai
2. Masukkan citra / gambar yang akan dikompres
3. Pilih jenis *codebook* yang akan digunakan
4. Klasifikasi piksel dari piksel pertama sampai terakir pada citra / gambar
    a. Menghitung jarak terdekat warna piksel dengan masing-masing warna pada *codebook* dengan rumus *Euclidean distance*
    b. Menyimpan indeks *codebook* dengan nilai *Euclidean distance* terkecil
5. Proses Huffman Encoding pada indeks hasil klasifikasi
6. Menyimpan indeks, informasi ukuran citra dan jenis *codebook* yang akan dipakai pada hasil kompresi
7. Menghitung rasio kompresi
8. Selesai

3.2.3 Struktur *File* Kompresi

File yang akan disimpan sebagai hasil kompresi dari sistem memiliki struktur sebagai berikut :

1. File Header

   File header berisi informasi tentang ukuran piksel citra, jumlah warna yang digunakan pada *codebook*, nilai MSE, nama *codebook.*
2. Isi

   Isi dari *file* merupakan indeks yang telah di *encode* menggunakan *Huffman encoding*.

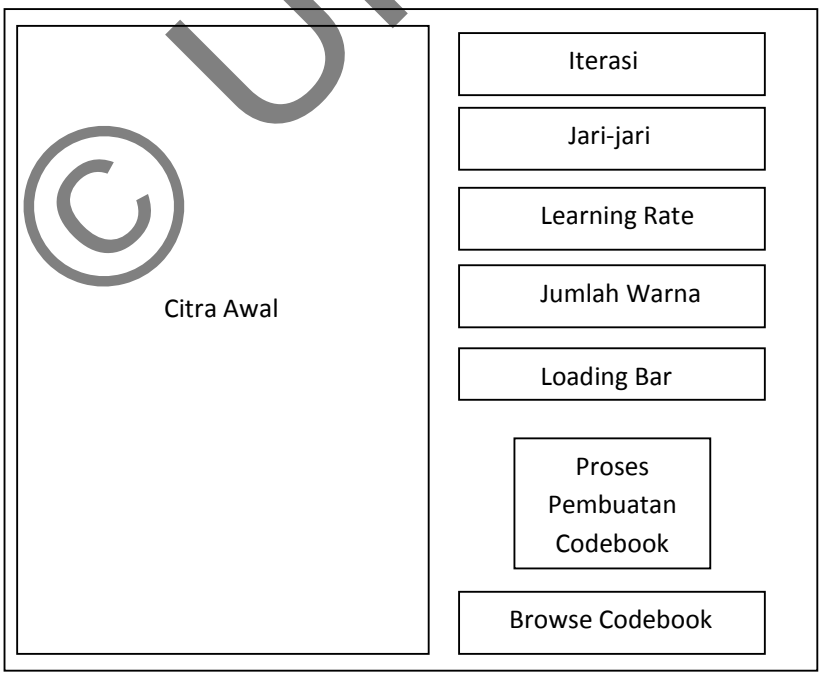
3.2.4 Algoritma Proses Dekompresi

Proses dekompresi adalah proses dimana hasil *file* yang terbentuk dari kompresi direkonstruksi menjadi suatu citra baru dengan ukuran yang lebih kecil. Algoritma proses dekompresi dapat dilihat dibawah ini :

1. Mulai
2. Masukkan *file* kompresi yang akan didekompresi
3. Mengambil nilai indeks, ukuran citra, dan jenis *codebook* yang dipakai
4. Proses *Huffman decoding* pada indeks
5. Menghitung MSE dan PSNR
6. Menampilkan citra hasil rekonstruksi, MSE dan PSNR
7. Selesai

3.3 Rancangan User *Interface*

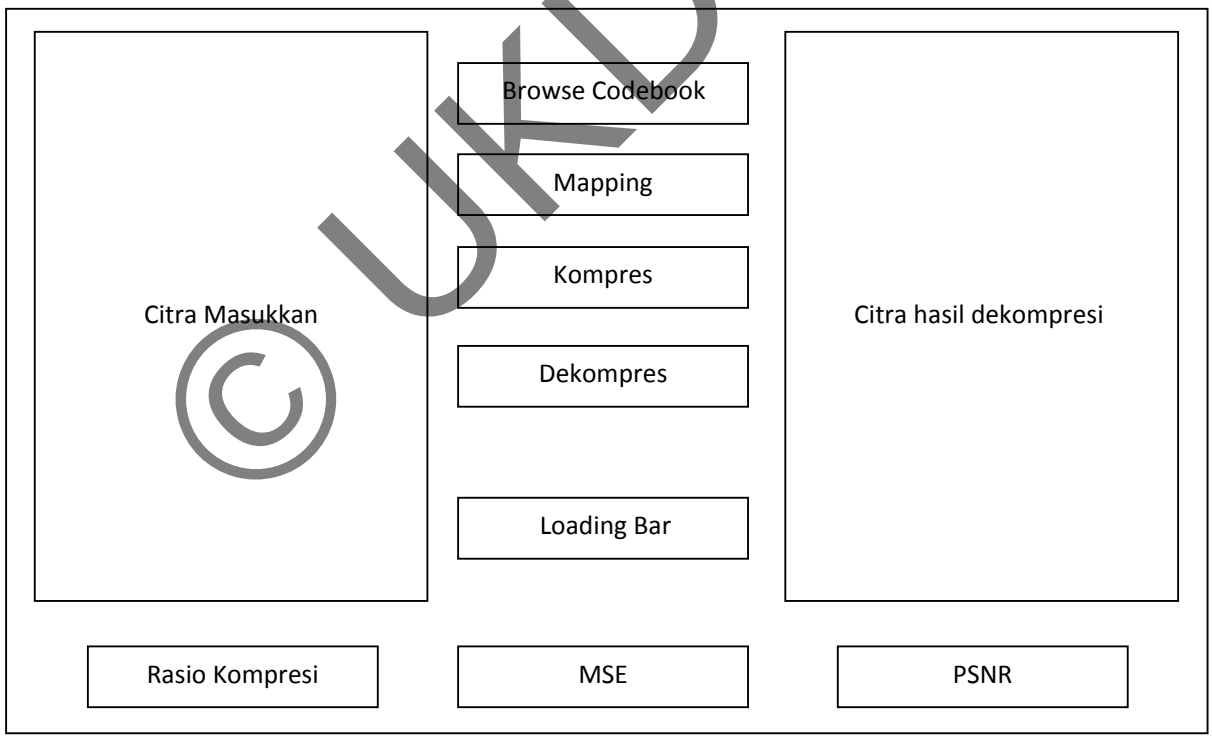
3.3.1 Rancangan Pembuatan *Codebook*



Gambar 3.3 Rancangan Pembuatan Codebook

Pada Gambar 3.3 merupakan rancangan dari proses pembuatan codebook dengan menggunakan Kohonen SOM. Klik pada kolom “citra awal” untuk memilih citra yang akan dibuat *codebook*. Jumlah iterasi, nilai jari-jari, *learning rate*, dan jumlah warna merupakan inputan dari *user*. Jumlah iterasi diberi batasan 1 sampai dengan 500, sedangkan nilai pada *learning rate* diberi batasan 0.0 sampai dengan 1.0. *Loading Bar* menunjukkan proses *running* yang ditampilkan dalam bentuk animasi (gif). Tombol “Proses pembuatan *codebook*” digunakan untuk memulai proses pelatihan Kohonen SOM pada citra awal setelah semua *inputan* dimasukkan oleh *user*. Tombol “Browse *codebook*” digunakan untuk melihat hasil *codebook* yang dihasilkan dari pelatihan Kohonen SOM tersebut.

3.3.2 Rancangan Pembuatan Proses Kompresi dan Dekompresi



Gambar 3.4 Rancangan Proses Kompresi dan Dekompresi

Pada Gambar 3.4 merupakan rancangan proses kompresi dan dekompresi. Klik pada kolom “Citra Masukkan” untuk memilih citra yang akan dikompresi. Tombol “Browse Codebook” digunakan untuk memilih jenis *codebook* yang akan digunakan untuk proses kompresi pada citra. Tombol “Mapping” digunakan untuk memulai proses klasifikasi warna antara citra masukkan dengan *codebook* yang dipilih untuk proses kompresi. Tombol “Kompres” digunakan untuk memulai proses kompresi pada citra setelah melewati proses *mapping*. Tombol “Dekompres” digunakan untuk melakukan proses rekonstruksi citra dari *file* hasil kompresi. *Loading Bar* menunjukkan proses *running* yang ditampilkan dalam bentuk animasi (gif). Nilai rasio kompresi akan ditampilkan ketika proses kompresi selesai. Nilai MSE dan PSNR akan ditampilkan setelah proses dekompresi selesai.

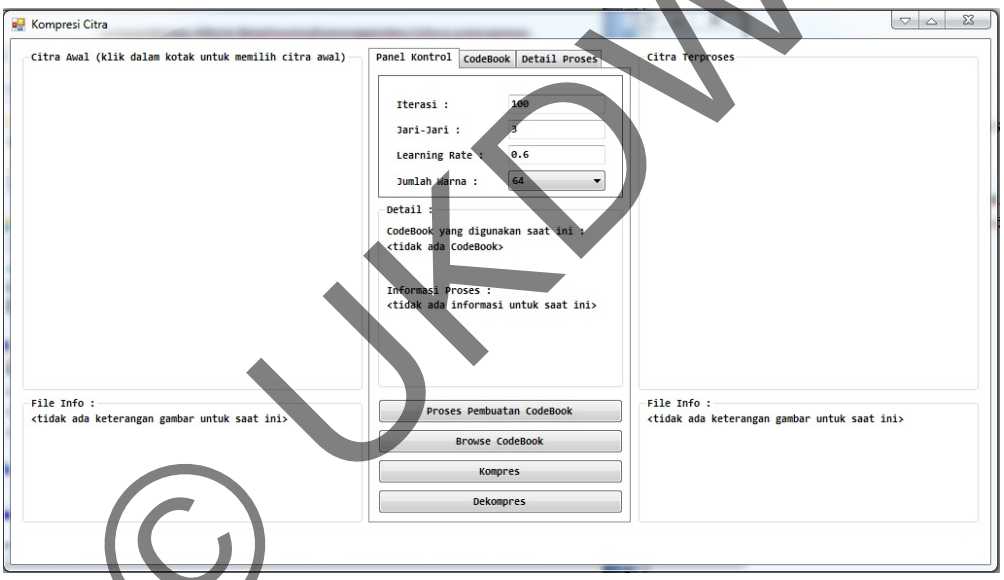
BAB 4

IMPLEMENTASI DAN ANALISIS SISTEM

Pada bab ini akan dijelaskan mengenai implementasi dan analisis dari algoritma yang dibahas pada Bab 3 dalam sebuah program dengan menggunakan Visual Basic 2008.

4.1 Implementasi Sistem

4.1.1 Form Utama

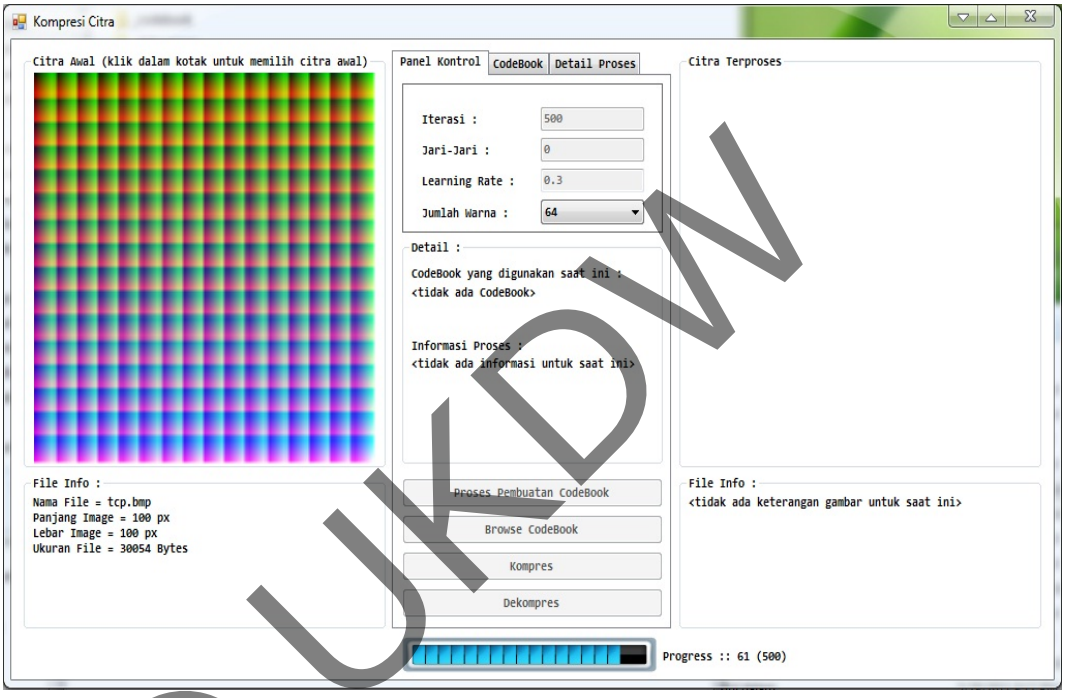


Gambar 4.1 Tampilan Form Utama

Gambar 4.1 adalah tampilan utama saat program dijalankan. Pada program ini berintikan tiga proses utama, yaitu proses pembuatan *codebook*, proses kompresi, dan proses dekompresi. Pada *form* utama terbagi menjadi tiga kolom inti, kolom pertama menjelaskan tentang citra inputan bitmap dan file info tentang citra tersebut. Kolom kedua berisikan tentang seluruh *inputan* dalam tiga proses sistem (pembuatan *codebook*, proses kompresi, proses dekompresi) beserta

informasinya, sedangkan kolom tiga menjelaskan tentang citra hasil dekompresi beserta informasi tentang citra tersebut.

4.1.2 Proses Pembuatan *Codebook*

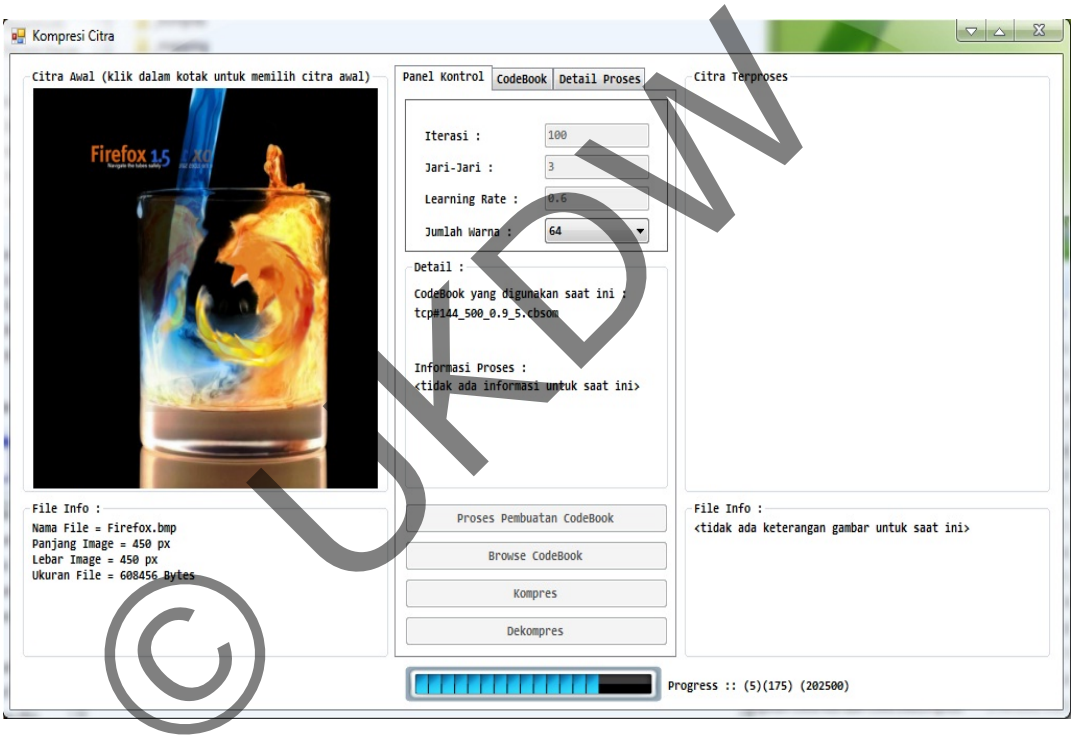


Gambar 4.2 Tampilan Proses Pembuatan *Codebook*

Pada Gambar 4.2, langkah pertama pada pembuatan *codebook* adalah memasukkan citra awal dengan mengklik pada *box* gambar dibawah tulisan "Citra Awal (klik dalam kotak untuk memilih citra awal)". Setelah citra selesai dipilih, akan tampil pada *box* tersebut dan akan muncul informasi pada kolom "File Info" yang berisikan tentang nama citra, panjang citra, lebar citra, dan ukuran citra. Langkah selanjutnya memberikan *inputan* nilai pada *tab* "Panel Kontrol" dengan memasukkan nilai Iterasi, Jari-Jari, *Learning Rate*, dan Jumlah warna. Batas nilai Iterasi adalah 1 - 1000, sedangkan batas nilai *Learning Rate* 0.1 - 1.0. Jumlah

warna tinggal memilih pada data yang sudah disediakan pada *box input* "Jumlah Warna", sedangkan besar jari-jari maximal tergantung dari besar jumlah warna yang dipilih. Setelah semua *input* dimasukkan, langkah terakhir adalah menekan tombol "Proses Pembuatan *Codebook*" untuk memulai proses pelatihan Kohonen SOM untuk menghasilkan *codebook* yang akan digunakan untuk proses kompresi.

4.1.3 Proses Kompresi

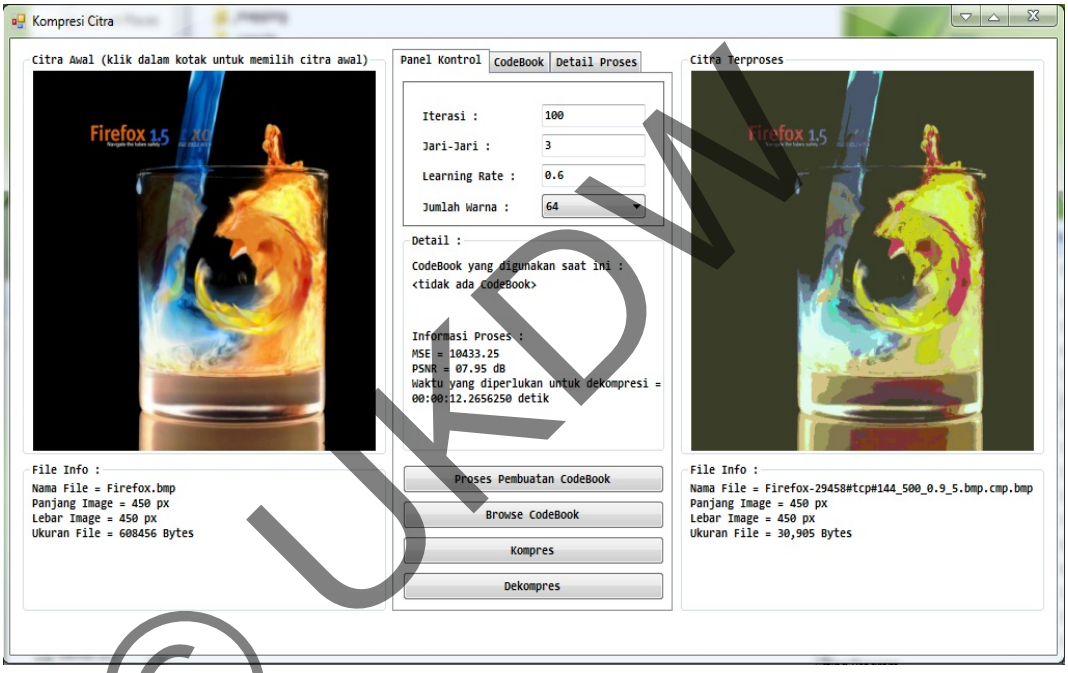


Gambar 4.3 Tampilan Proses Kompresi

Pada Gambar 4.3, langkah pertama pada proses kompresi adalah memilih citra *input* pada *box* gambar "Citra Awal". Selanjutnya memilih *file codebook* yang akan digunakan untuk mengkompres citra yang sudah dipilih dengan menekan tombol "Browse *Codebook*". Pada kolom "Detail" akan ada informasi tentang *file codebook* yang dipilih. Sedangkan pada *tab "Codebook"* yang terletak diatas, kita dapat melihat nilai *codebook* nya.

Tombol “Kompres” digunakan untuk memulai proses kompresi setelah semua *inputan* dimasukkan. Setelah proses kompresi selesai kita dapat melihat pada “Informasi Proses”, dimana terdapat informasi tentang nama *file* terkompres, ukuran *file* terkompres, rasio kompresi, dan waktu yang dibutuhkan selama proses kompresi.

4.1.4 Proses Dekompresi



Gambar 4.4 Tampilan Proses Dekompresi

Pada Gambar 4.4, proses dekompresi ini hanya mengambil dari hasil *file* kompresi pada proses sebelumnya kemudian di dekompres menjadi *file* baru dengan ukuran yang lebih kecil. Langkah yang dilakukan hanya menekan tombol “Dekompres”, dimana kita akan memilih *file* yang telah dikompres sebelumnya. Setelah proses dekompres selesai maka akan muncul pada *box* gambar di “Citra Terproses” dan pada “Informasi Proses” kita bisa melihat informasi dari citra hasil dekompresi tentang

nilai MSE, PSNR, dan waktu dekompresi. Pada "*File* Info" dibawah *box* gambar akan mencetak kembali Nama *File*, Panjang gambar, Lebar gambar, dan ukuran *file*.

4.2 Pengujian Citra Pada Sistem

Pengujian dilakukan untuk meneliti :

- Pengaruh dari jumlah warna, nilai jari-jari, dan *learning rate* terhadap citra hasil dekompresi.
- Pengaruh besarnya jumlah warna, jari-jari, dan *learning rate* pada lamanya proses pembuatan *codebook* dan proses kompresi.

Pengujian sistem dilakukan dengan spesifikasi *hardware* sebagai berikut :

- Processor : Intel(R) Core(TM) i5 CPU M430 @2.27GHz
- Memory : 2GB
- Chip Type : Intel(R) HD Graphics (Corei5)

4.2.1 Pengujian Lama Waktu Proses Pembuatan *Codebook*

Tabel 4.1 Pembuatan *codebook*

Iterasi	Learning Rate	Jumlah Warna	Jari-jari	Waktu (h:m:s)
500	0.3	64	0	00:06:19.43
500	0.9	64	0	00:06:11.96
500	0.3	64	0	00:06:58.48
500	0.9	64	0	00:06:52.21
500	0.3	144	2	00:12:41.98
500	0.9	144	2	00:12:26.96
500	0.3	144	5	00:15:51.18
500	0.9	144	5	00:15:43.98

Dilihat dari Tabel 4.1 dapat dilihat bahwa semakin besar jumlah warna untuk pembuatan *codebook* maka proses yang dibutuhkan juga akan semakin lama. Begitu juga dengan jari-jari yang lebih besar dari 0 maka prosesnya juga akan semakin meningkat. Tetapi proses dengan jumlah warna dan jari-jari yang sama dengan *learning rate* yang lebih besar maka proses pembuatan bisa menurun. Pengujian *codebook* dilakukan dengan menggunakan citra truecolorpalette.bmp 24-bit dengan ukuran 100x100 piksel. Semua proses pembuatan *codebook* menggunakan iterasi 500.

4.2.2 Pengujian Kompresi dan Hasil Citra

Pengujian ini dilakukan dengan *codebook* yang mempunyai perbedaan pada jumlah warna, nilai *learning rate*, dan jari-jari.

(a) Pengujian citra Koala.bmp dan Tulip.bmp dengan *codebook* yang menggunakan jari-jari = 0 dan jumlah warna yang berbeda.

Tabel 4.2 Hasil Uji CobaKoala.bmp Dengan Jari-jari = 0

No	Jari-jari	Jumlah Warna	Learning Rate	Waktu (h:m:s)	Rasio Kompresi (%)	MSE	PSNR (db)
1	0	64	0.3	0:0:46.41	86.46	17579.77	05.68
2	0	64	0.9	0:0:41.01	88.43	29269.53	03.47
3	0	144	0.3	0:0:50.23	83.61	17200.74	05.78
4	0	144	0.9	0:0:26.51	88.63	24717.12	04.20

Tabel 4.3 Hasil Uji Coba Tulip.bmp Dengan Jari-jari = 0

No	Jari-jari	Jumlah Warna	Learning Rate	Waktu (h:m:s)	Rasio Kompresi (%)	MSE	PSNR (db)
1	0	64	0.3	0:0:34.71	91.59	23687.88	04.39
2	0	64	0.9	0:0:33.73	91.86	33007.71	02.94
3	0	144	0.3	0:0:49.75	90.49	22572.59	04.60
4	0	144	0.9	0:7:4.54	93.22	29519.62	03.43

Pengujian citra dengan *codebook* yang menggunakan jari-jari = 0 pada citra Koala.bmp dan Tulip.bmp dapat dilihat pada Tabel 4.2 dan Tabel 4.3. Hasil uji coba menunjukkan pemakaian jumlah warna dan *learning rate* yang lebih besar akan meningkatkan waktu proses dan rasio kompresi. Jumlah warna yang lebih banyak akan menghasilkan citra dekompresi yang lebih bagus jg, tetapi dengan nilai *learning rate* yang lebih besar akan menyebabkan hasil dekompresi menurun. Dapat dilihat pada peningkatan nilai MSE dan penurunan pada nilai PSNR.

Hasil citra asli dan dekompresi pada uji cobaTabel 4.2 dan 4.3 dapat dilihat pada Lampiran A-1 dan A-2.

(b) Pengujian citra Firefox.bmp dan Tulip.bmp dengan *codebook* yang menggunakan jari-jari lebih besar dari 0 (jari-jari > 0) dan jumlah warna yang berbeda.

Tabel 4.4 Uji Coba Firefox.bmp Dengan Jari-jari > 0

No	Jari-jari	Jumlah Warna	Learning Rate	Waktu (h:m:s)	Rasio Kompresi (%)	MSE	PSNR (db)
1	2	64	0.3	0:3:41.81	95.47	10885.41	07.76
2	2	64	0.9	0:3:17.70	96.26	18872.98	05.37
3	5	144	0.3	0:3:44.59	94.57	10034.72	08.11
4	5	144	0.9	0:34:53.29	94.87	10433.25	07.95

Tabel 4.5 Uji Coba Tulip.bmp Dengan Jari-jari > 0

No	Jari-jari	Jumlah Warna	Learning Rate	Waktu (h:m:s)	Rasio Kompresi (%)	MSE	PSNR (db)
1	2	64	0.3	0:0:48.35	89.99	24189.71	04.29
2	2	64	0.9	0:0:55.15	91.46	31018.68	03.21
3	5	144	0.3	0:1:2	87.44	22871	04.54
4	5	144	0.9	0:1:14.06	89.88	22623.45	04.59

Pengujian citra dengan *codebook* yang menggunakan jari-jari lebih besar dari nol (jari-jari > 0) dan jumlah warna yang lebih besar pada citra Firefox.bmp danTulip.bmp. Hasil dari uji coba dapat dilihat pada Tabel 4.4 dan Tabel 4.5. Pengujian menunjukkan adanya peningkatan proses kompresi saat nilai jari-jari lebih besar dari 0 dibandingkan dengan saat proses dengan jari-jari = 0. Nilai *learning rate* yang lebih besar akan memberikan rasio kompresi yang lebih besar. Jumlah warna dan jari-jari yang besar akan memberikan hasil dekompresi yang lebih bagus, tetapi semakin tinggi nilai *learning rate* nya maka hasil citra pun juga akan menurun.

Hasil uji coba Tabel 4.4 dan Tabel 4.5 dapat dilihat pada Lampiran A-3 dan A-4.

(c) Membandingkan kompleksitas citra Desert.bmp, Dbz.bmp dengan *codebook* yang menggunakan jumlah warna 144, jari-jari = 0, dan learning rate 0.3

Tabel 4.6 Uji Coba Kompleksitas Warna Yang Berbeda

Nama file	**Ukuran File (bytes)**	**Ukuran Pixel**	**Waktu (h:m:s)**	**Rasio kompresi**	**MSE**	**PSNR**
Desert.bmp	270.054	300x300	0:1:47.93	93.01	20284.72	05.06
Dbz.bmp	270.054	300x300	0:1:46.76	86.26	17527.71	05.69

Pengujian pada Tabel 4.6 dapat dilihat bahwa dengan *codebook* yang sama dan ukuran *file* yang sama menghasilkan rasio kompresi yang berbeda. Ini karena perbedaan kompleksitas warna pada citra, semakin kompleks warna pada suatu citra maka akan menghasilkan rasio yang lebih kecil. Hasil dekompresi akan semakin bagus pada citra dengan warna yg lebih kompleks, dapat dilihat pada nilai MSE dan PSNR nya. Pada citra Dbz.bmp nilai PSNR nya lebih tinggi dibandingkan dengan citra Desert.bmp. Tetapi hal itu juga dipengaruhi oleh nilai dalam pembuatan *codebooknya* juga.

Hasil uji coba pada Tabel 4.6 dapat dilihat pada Lampiran A-5.

4.3 Hasil Analisis

Hasil analisis yang didapat dari pengujian citra pada Tabel-tabel diatas menunjukkan bahwa jumlah warna dan jari-jari yang lebih besar

akan menyebabkan waktu dalam proses pembuatan *codebook* meningkat. Sebaliknya jika nilai *learning rate* semakin besar maka proses waktunya akan menurun.

Jumlah warna yang lebih besar akan menghasilkan ukuran kompresi yang lebih besar dan akan menghasilkan citra hasil dekompresi yang semakin bagus. Nilai *learning rate* yang semakin besar akan membuat rasio kompresi naik dan cenderung akan membuat hasil citra dekompresi kurang bagus. Sedangkan nilai *learning rate* yang semakin kecil akan menghasilkan rasio kompresi menurun dan cenderung menghasilkan citra dekompresi yang lebih bagus. Hal ini dapat dilihat pada penurunan nilai MSE dan penaikan nilai PSNR. Semakin besar nilai PSNR maka citra dekompresi yang dihasilkan akan semakin bagus. Waktu proses juga akan meningkat jika jumlah warna yang digunakan pada proses kompresi semakin besar.

Nilai jari-jari juga berpengaruh pada proses kompresi dan citra hasil dekompresi. Penggunaan jari-jari yang lebih besar dari 0 (jari-jari > 0) akan membuat rasio kompresi relatif menurun dan menyebabkan citra hasil dekompresi lebih bagus. Penggunaan jari-jari = 0 dengan *learning rate* rendah akan cenderung menghasilkan rasio yang lebih kecil dan hasil dekompresi yang bagus dibandingkan dengan pengunaan jari-jari = 0 dengan *learning rate* yang besar akan menghasilkan rasio kompresi yang tinggi dan menyebabkan hasil citra dekompresi menurun. Semakin besar jumlah warna pada jari-jari = 0 dan nilai *learning rate* yang kecil akan menghasilkan citra dekompresi yang jauh lebih bagus. Hal ini berbeda jika jari- jari yang digunakan lebih besar dari 0, maka semakin besar jumlah warna dan *learning rate* yang dipakai dalam proses kompresi akan menghasilkan rasio kompresi yang lebih kecil dan citra dekompresi semakin bagus.

LAMPIRAN

Form Utama

```
Imports System.Threading
Public Class formUtama

    'deklarasi struct
    Structure structWarna
        Dim R, G, B As Integer
    End Structure

    Structure structBobot
        Dim wR, wG, wB As Double
    End Structure

    Structure StructColorRGB
        Dim cR, cG, cB As Int32
    End Structure

    'inisialisasi variabel global
    Dim warna(,) As structWarna
    Dim bobot(,) As structBobot
    Dim citraInput(,) As StructColorRGB
    Dim tempLoadCodebook() As StructColorRGB
    Const key As String = "cb_agung"
    Dim tempM1, tempM2, tempM3, imageSizeCB1, imageSizeCB2, tempMSE As Double
    Dim imageFileNameCB1, imageFileNameCB1wExt, imageFileNameCB2,
imageFileNameCB2wExt As String
    Dim idx() As Integer
    Dim img As Bitmap
    Dim temp_clr As Color
    Dim start_time, end_time As DateTime
    Dim elapsed_time As TimeSpan

    Sub setLoad()
        cbJmlhWarna.SelectedIndex = 0
        txtIterasi.Text = "100"
        txtJariJari.Text = Math.Sqrt(Val(cbJmlhWarna.Text)) / 2 - 1
        txtLearningRate.Text = 0.6

        pbLoader.Visible = False
        lblProgress.Visible = False
    End Sub

    Private Sub setButton(ByVal setBoolean As Boolean)
        btnProses.Enabled = setBoolean
        btnCodeBook.Enabled = setBoolean
        btnKompres.Enabled = setBoolean
        btnDekompres.Enabled = setBoolean

        txtIterasi.Enabled = setBoolean
        txtJariJari.Enabled = setBoolean
        txtLearningRate.Enabled = setBoolean
    End Sub

    Private Sub formUtama_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

```
        setLoad()
    End Sub

    Private Sub pbVector_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles pbVector.Click
        ofdImage.Filter = "Bitmap Files (*.bmp)|*.bmp"
        ofdImage.FileName = ""
        ofdImage.InitialDirectory = My.Application.Info.DirectoryPath &
"\_sample"

        If ofdImage.ShowDialog = Windows.Forms.DialogResult.OK Then
            img = Image.FromFile(ofdImage.FileName)
            If img.PixelFormat = Imaging.PixelFormat.Format16bppGrayScale Or
img.PixelFormat = Imaging.PixelFormat.Format1bppIndexed Or img.PixelFormat =
Imaging.PixelFormat.Format4bppIndexed Or img.PixelFormat =
Imaging.PixelFormat.Format8bppIndexed Then
                MsgBox("Citra bukan merupakan citra berwarna",
MsgBoxStyle.Information, "Peringatan")
                Exit Sub
            End If
            pbVector.Image = img
            imageFileNameCB1 =
System.IO.Path.GetFileNameWithoutExtension(ofdImage.FileName)
            imageFileNameCB1wExt =
System.IO.Path.GetFileName(ofdImage.FileName)
            imageSizeCB1 =
My.Computer.FileSystem.GetFileInfo(ofdImage.FileName).Length

            lblInfo1.Text = "Nama File = " & imageFileNameCB1wExt & vbNewLine
& _
                            "Panjang Image = " & img.Width & " px" &
vbNewLine & _
                            "Lebar Image = " & img.Height & " px" & vbNewLine
& _
                            "Ukuran File = " & imageSizeCB1.ToString("##.##")
& " Bytes"
        End If
    End Sub


    Private Sub btnProses_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnProses.Click
        If txtJariJari.Text = "" Or txtIterasi.Text = "" Then
            MessageBox.Show("Cek Kembali Nilai Jari-Jari dan atau Iterasi")
        Else
            If lblInfo1.Text = "<tidak ada keterangan gambar untuk saat ini>"
Then
                MessageBox.Show("Pilih Gambar Terlebih Dahulu")
            Else
                Try
                    setButton(False)

                    Dim iPercentage As String = "Progress :: 0 (" &
txtIterasi.Text & ")"
                    lblProgress.Text = iPercentage
                    lblProgress.Visible = True
```

```
                    pbLoader.Visible = True

                    start_time = Now

                    'mendapatkan semua warna untuk setiap pixel dari gambar
                    ReDim warna(img.Width - 1, img.Height - 1)
                    For i As Integer = 0 To img.Height - 1
                        For j As Integer = 0 To img.Width - 1
                            warna(i, j).R = img.GetPixel(i, j).R
                            warna(i, j).G = img.GetPixel(i, j).G
                            warna(i, j).B = img.GetPixel(i, j).B
                            Application.DoEvents()
                        Next
                        Application.DoEvents()
                    Next

                    'inisialisasi random bobot awal
                    Dim jumlahWarna As Integer = Val(cbJmlhWarna.Text)
                    Dim akarJumlahWarna As Integer =
Math.Sqrt(Val(jumlahWarna))
                    ReDim bobot(akarJumlahWarna - 1, akarJumlahWarna - 1)
                    Dim rand As New Random
                    For i As Integer = 0 To akarJumlahWarna - 1
                        For j As Integer = 0 To akarJumlahWarna - 1
                            bobot(i, j).wR = rand.NextDouble()
                            bobot(i, j).wG = rand.NextDouble()
                            bobot(i, j).wB = rand.NextDouble()
                            Application.DoEvents()
                        Next
                        Application.DoEvents()
                    Next

                    'proses kohonen-som
                    Dim iDistance As Double = 0 'jarak
                    Dim iMinimumSekarang As Double = 0 'jarak pembanding
                    Dim tR, tG, tB As Integer 'R, G, B sementara
                    Dim tRNew, tGNew, tBNew As Integer 'R, G, B baru
                    Dim iIndexX, iIndexY As Integer 'index bobot yang
mempunyai jarak terkecil
                    Dim learningRate As Double = Val(txtLearningRate.Text)
'learning rate
                    Dim learningRateAwal As Double = learningRate 'learning
rate awal
                    Dim iterasi As Integer = Val(txtIterasi.Text) 'jumlah
iterasi
                    Dim jariJari As Integer = Val(txtJariJari.Text) 'jari-
jari
                    Dim jariJariAwal As Integer = jariJari 'variabel tampung
untuk jariJari awal
                    Dim lambda As Double = iterasi / Math.Log(jariJariAwal)
'belom tau gunanya, kayanya ikut rumus

                    For h As Integer = 0 To iterasi - 1
                        lblProgress.Text = "Progress :: " & h + 1 & " (" &
txtIterasi.Text & ")"
                        For i As Integer = 0 To img.Height - 1
                            For j As Integer = 0 To img.Width - 1
```

```
                                tR = warna(i, j).R
                                tG = warna(i, j).G
                                tB = warna(i, j).B
                                'proses pencarian bobot pemenang
                                iDistance = 0
                                iMinimumSekarang = 10000000
                                For k As Integer = 0 To akarJumlahWarna - 1
                                    For l As Integer = 0 To akarJumlahWarna -
1
                                        iDistance = Math.Pow((bobot(k, l).wR
- tR), 2) + _
                                                    Math.Pow((bobot(k, l).wG
- tG), 2) + _
                                                    Math.Pow((bobot(k, l).wB
- tB), 2)
                                        If (iDistance < iMinimumSekarang)
Then
                                            iMinimumSekarang = iDistance
                                            iIndexX = k
                                            iIndexY = l
                                        End If
                                        Application.DoEvents()
                                    Next
                                    Application.DoEvents()
                                Next
                                tRNew = bobot(iIndexX, iIndexY).wR
                                tGNew = bobot(iIndexX, iIndexY).wG
                                tBNew = bobot(iIndexX, iIndexY).wB
                                bobot(iIndexX, iIndexY).wR = tRNew +
learningRate * (tR - tRNew)
                                bobot(iIndexX, iIndexY).wG = tGNew +
learningRate * (tG - tGNew)
                                bobot(iIndexX, iIndexY).wB = tBNew +
learningRate * (tB - tBNew)
                                'proses update bobot tetangga
                                If CInt(jariJari) > 0 Then
                                    Dim horMinus, horPlus, verMinus, verPlus,
diaMinus, diaPlus As Integer
                                    '             vertikal
                                    '                ^
                                    '| x-i,y-i | x,y-i | x+i,y-i |
                                    '|  x-i,y  |  x,y  |  x+i,y  | -->
horizontal
                                    '| x-i,y+i | x,y+i | x+i,y+i |
                                    For iX As Integer = 1 To jariJari
                                        Dim xx As Integer = 0
                                        Dim yy As Integer = 0
                                        horMinus = iIndexX - iX
                                        horPlus = iIndexX + iX
                                        verMinus = iIndexY - iX
                                        verPlus = iIndexY + iX
                                        For horVerX As Integer = 1 To 4
                                            If horVerX = 1 Then
                                                'tetangga(kiri)
=============
```

```
                                        xx = horMinus
                                        yy = iIndexY
                                    ElseIf horVerX = 2 Then
                                        'tetangga(kanan)
=============
                                        xx = horPlus
                                        yy = iIndexY
                                    ElseIf horVerX = 3 Then
                                        'tetangga(atas)
=============
                                        xx = iIndexX
                                        yy = verMinus
                                    ElseIf horVerX = 4 Then
                                        'tetangga(bawah)
=============
                                        xx = iIndexX
                                        yy = verPlus
                                    End If
                                    If (xx >= 0 And xx <=
akarJumlahWarna - 1) And (yy >= 0 And yy <= akarJumlahWarna - 1) Then
                                        tRNew = bobot(xx, yy).wR
                                        tGNew = bobot(xx, yy).wG
                                        tBNew = bobot(xx, yy).wB

                                        bobot(xx, yy).wR = tRNew +
learningRate * (tR - tRNew)
                                        bobot(xx, yy).wG = tGNew +
learningRate * (tG - tGNew)
                                        bobot(xx, yy).wB = tBNew +
learningRate * (tB - tBNew)
                                    End If
                                    Application.DoEvents()
                                Next

                                For iY As Integer = 1 To jariJari
                                    diaMinus = iIndexY - iY
                                    diaPlus = iIndexY + iY
                                    Dim xxx As Integer = 0
                                    Dim yyy As Integer = 0
                                    For horVerY As Integer = 1 To 4
                                        If horVerY = 1 Then
                                            'tetangga(kiri-atas)
=============
                                            xxx = horMinus
                                            yyy = diaMinus
                                        ElseIf horVerY = 2 Then
                                            'tetangga(kiri-bawah)
=============
                                            xxx = horMinus
                                            yyy = diaPlus
                                        ElseIf horVerY = 3 Then
                                            'tetangga(kanan-atas)
=============
                                            xxx = horPlus
                                            yyy = diaMinus
                                        ElseIf horVerY = 4 Then
```

```
                                                    'tetangga(kanan-bawah)
=============
                                                    xxx = horPlus
                                                    yyy = diaPlus
                                                End If
                                                If (xxx >= 0 And xxx <=
akarJumlahWarna - 1) And (yyy >= 0 And yyy <= akarJumlahWarna - 1) Then
                                                    tRNew = bobot(xxx,
yyy).wR
                                                    tGNew = bobot(xxx,
yyy).wG
                                                    tBNew = bobot(xxx,
yyy).wB

                                                    bobot(xxx, yyy).wR =
tRNew + learningRate * (tR - tRNew)
                                                    bobot(xxx, yyy).wG =
tGNew + learningRate * (tG - tGNew)
                                                    bobot(xxx, yyy).wB =
tBNew + learningRate * (tB - tBNew)
                                                End If
                                                Application.DoEvents()
                                            Next
                                            Application.DoEvents()
                                        Next
                                        Application.DoEvents()
                                    Next
                                End If
                                Application.DoEvents()
                            Next
                            Application.DoEvents()
                        Next

                        'update learning rate
                        learningRate = learningRateAwal * Math.Exp(-h /
iterasi)
                        If jariJari <> 0 Then
                            'update jari-jari
                            jariJari = jariJariAwal * Math.Exp(-h / lambda)
                        End If
                        Application.DoEvents()
                    Next

                    pbLoader.Visible = False
                    lblProgress.Visible = False


                    'simpan codebook pada text file (.cbsom)
                    Dim infoFileKompres As String = cbJmlhWarna.Text & "_" &
txtIterasi.Text & "_" & txtLearningRate.Text & "_" & txtJariJari.Text
                    Dim fsData As New
IO.FileStream(My.Application.Info.DirectoryPath & "\_codebook\" &
imageFileNameCB1 & "#" & infoFileKompres & ".cbsom", IO.FileMode.Create)
                    Dim writer As New IO.BinaryWriter(fsData)
                    writer.Write(key)
                    writer.Write(CInt(cbJmlhWarna.Text))
                    For Each elementData As structBobot In bobot
```

```
                    writer.Write(CInt(elementData.wR))
                    writer.Write(CInt(elementData.wG))
                    writer.Write(CInt(elementData.wB))
                Next
                writer.Flush()
                writer.Close()
                fsData.Close()

                end_time = Now
                elapsed_time = end_time.Subtract(start_time)

                'buka folder _codebook
                Dim Message As String = "Proses pembuatan codebook sudah
selesai" & vbNewLine & _
                                        "Waktu yang dibutuhkan untuk
pembuatan codebook = " & elapsed_time.ToString & " detik" & vbNewLine & _
                                        vbNewLine & _
                                        "File codebook(" &
infoFileKompres & ")" & vbNewLine & _
                                        "sudah tersimpan pada folder
[_codebook]" & vbNewLine & _
                                        "buka folder [_codebook] untuk
melihat file?"
                Dim Caption As String = "CODEBOOK"
                Dim Buttons As MessageBoxButtons =
MessageBoxButtons.YesNo
                Dim Result As DialogResult
                Result = MessageBox.Show(Message, Caption, Buttons)
                If Result = Windows.Forms.DialogResult.Yes Then
                    Process.Start(My.Application.Info.DirectoryPath &
"\_codebook")
                End If
            Catch ex As Exception
                pbLoader.Visible = False
                lblProgress.Visible = False
                MessageBox.Show("Ada kesalahan proses")
            End Try

            setButton(True)
        End If
    End If
End Sub

Private Sub btnCodeBook_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCodeBook.Click
    'buka folder _codebook
    On Error GoTo wrongs
    ofdImage.Filter = "Codebook SOM Files (*.cbsom)|*.cbsom"
    ofdImage.FileName = ""
    ofdImage.InitialDirectory = My.Application.Info.DirectoryPath &
"\_codebook"
    If ofdImage.ShowDialog = Windows.Forms.DialogResult.OK Then
        Dim fsData As New IO.FileStream(ofdImage.FileName,
IO.FileMode.Open, IO.FileAccess.Read)
        Dim reader As New IO.BinaryReader(fsData)

        Dim temp As String
```

```
            temp = reader.ReadString
            If temp <> key Then
                reader.Close()
                fsData.Close()
                GoTo wrongs
            Else
                lblInfo2.Text = System.IO.Path.GetFileName(ofdImage.FileName)
                Dim lokasiCodebook As String = ofdImage.FileName
                Dim tempTotalCb As Integer = reader.ReadInt32 - 1

                'tampilkan pada datagridview
                dgvCodeBook.Columns.Clear()
                dgvCodeBook.Rows.Clear()
                dgvCodeBook.Columns.Add("colNo", "NO")
                dgvCodeBook.Columns.Add("colCBSOM", "CBSOM")
                dgvCodeBook.Columns(0).Width = 30
                dgvCodeBook.Columns(1).Width = 60
                Dim counter = 0
                For x As Int32 = 0 To tempTotalCb
                    dgvCodeBook.Rows.Add()
                    dgvCodeBook(0, counter).Value = counter + 1
                    dgvCodeBook(1, counter).Value = CInt(reader.ReadInt32) &
"," & CInt(reader.ReadInt32) & "," & CInt(reader.ReadInt32)
                    counter += 1
                    Application.DoEvents()
                Next
                reader.Close()
                fsData.Close()

                tabControl.SelectTab(tabCodeBook)
            End If
        End If
        Exit Sub
wrongs: MsgBox("File Tidak Bisa Dibaca", MsgBoxStyle.Exclamation,
"Perhatian")
    End Sub

    Private Sub btnKompres_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnKompres.Click
        If dgvCodeBook.RowCount = 0 Then
            MessageBox.Show("Pilih CodeBook Terlebih Dahulu")
        Else
            If lblInfo1.Text = "<tidak ada keterangan gambar untuk saat ini>"
Then
                MessageBox.Show("Pilih Gambar Terlebih Dahulu")
            Else
                setButton(False)
                Try
                    lblProgress.Visible = True
                    pbLoader.Visible = True
                    start_time = Now

                    'inisialisasi variabel
                    Dim rr, gg, bb, rrNew, ggNew, bbNew, iIndex As Integer
                    Dim tempCB(2) As String
                    Dim tempDistance, minDistance As Double
```

```
                    Dim new_bmp As New Bitmap(img.Width, img.Height,
Imaging.PixelFormat.Format24bppRgb)

                    'inisialisasi ukuran datagridview
                    dgvHasilMapping.Columns.Add("colIndex", "Index")
                    dgvHasilMapping.Columns.Add("colKSOM", "KSOM")
                    dgvHasilMapping.Columns(0).Width = 30
                    dgvHasilMapping.Columns(1).Width = 100

                    'proses mapping
                    For i As Integer = 0 To img.Width - 1
                        For j As Integer = 0 To img.Height - 1
                            tempDistance = 0
                            minDistance = 100000000
                            rrNew = 0
                            ggNew = 0
                            bbNew = 0
                            iIndex = 0

                            lblProgress.Text = "Progress :: (" & i & ")(" & j
& ") (" & img.Width * img.Height & ")"

                            For k As Integer = 0 To dgvCodeBook.RowCount - 1
                                tempCB = Split(dgvCodeBook(1, k).Value, ",")
                                rr = img.GetPixel(i, j).R
                                gg = img.GetPixel(i, j).G
                                bb = img.GetPixel(i, j).B
                                tempDistance = Math.Pow((Val(rr) -
Val(tempCB(0))), 2) + _
                                               Math.Pow((Val(gg) -
Val(tempCB(1))), 2) + _
                                               Math.Pow((Val(bb) -
Val(tempCB(2))), 2)
                                tempDistance = Math.Sqrt(tempDistance)
                                If tempDistance < minDistance Then
                                    minDistance = tempDistance
                                    rrNew = tempCB(0)
                                    ggNew = tempCB(1)
                                    bbNew = tempCB(2)
                                    iIndex = k
                                End If
                                Application.DoEvents()
                            Next
                            dgvHasilMapping.Rows.Add(iIndex, rrNew & "," &
ggNew & "," & bbNew)
                            new_bmp.SetPixel(i, j, Color.FromArgb(rrNew,
ggNew, bbNew))
                            Application.DoEvents()
                        Next
                        Application.DoEvents()
                    Next
                    'pbReal.Image = new_bmp

                    imageFileNameCB1 = imageFileNameCB1 & "#" &
lblInfo2.Text.Replace(".cbsom", "")
                    Dim newImage As String =
My.Application.Info.DirectoryPath & "\_mapping\" & imageFileNameCB1 & ".bmp"
```

```
                    new_bmp.Save(newImage)

                    'proses kompresi huffman
                    Dim L As Long = FileSystem.FileLen(newImage)

                    'buat sebuah array dan load file ke dalamnya
                    Dim B(0 To L - 1) As Byte
                    FileSystem.FileOpen(1, newImage, OpenMode.Binary,
OpenAccess.Read)
                    FileSystem.FileGet(1, B)
                    FileSystem.FileClose(1)

                    'proses kompresi
                    lblProgress.Visible = True
                    Application.DoEvents()
                    Dim CMP() As Byte = HuffmanCoding.CompressByteArray(B)
                    Application.DoEvents()
                    lblProgress.Visible = False
                    'akhir of proses kompresi

                    'simpan file hasil kompresi (berupa byte array) dengan
extension .cmp
                    FileSystem.FileOpen(1, newImage & ".cmp",
OpenMode.Binary, OpenAccess.Write)
                    FileSystem.FilePut(1, CMP)
                    FileSystem.FileClose(1)

                    'pindahkan hasil (.cmp) ke folder _kompres
                    Dim dFrom As String = My.Application.Info.DirectoryPath &
"\_mapping\" & imageFileNameCB1 & ".bmp.cmp"
                    Dim dTo As String = My.Application.Info.DirectoryPath &
"\_kompres\" & imageFileNameCB1 & ".bmp.cmp"
                    If System.IO.File.Exists(dFrom) = True Then
                        If System.IO.File.Exists(dTo) = True Then
                            System.IO.File.Delete(dTo)
                        End If
                        System.IO.File.Move(dFrom, dTo)
                    End If

                    end_time = Now
                    elapsed_time = end_time.Subtract(start_time)

                    Dim fileCompressedName As String =
System.IO.Path.GetFileName(dTo)
                    Dim fileSize As String =
My.Computer.FileSystem.GetFileInfo(dTo).Length
                    lblInfo2B.Text = "File Terkompres : " & vbNewLine &
fileCompressedName & vbNewLine & _
                                     "Ukuran File Terkompres : " & fileSize &
" Bytes" & vbNewLine & _
                                     "Rasio : " & FormatNumber(100 -
(Val(fileSize) / Val(imageSizeCB1.ToString("##.##"))) * 100, 2) & " %" &
vbNewLine & _
                                     "Waktu yang diperlukan untuk kompresi =
" & vbNewLine & elapsed_time.ToString & " detik"

                    lblProgress.Visible = False
```

```
                    pbLoader.Visible = False
                    MessageBox.Show("PROSES KOMPRES SELESAI")

                    'tampilkan hasil mapping
                    tabControl.SelectTab(tabDetail)

                    'buka folder _kompress
                    Dim Message As String = "File " & imageFileNameCB1 &
".bmp.cmp" & vbNewLine & _
                                            "sudah tersimpan pada folder
[_kompress]" & vbNewLine & _
                                            "buka folder [_kompress] untuk
melihat file?"
                    Dim Caption As String = "HASIL KOMPRESI"
                    Dim Buttons As MessageBoxButtons =
MessageBoxButtons.YesNo
                    Dim Result As DialogResult
                    Result = MessageBox.Show(Message, Caption, Buttons)
                    If Result = Windows.Forms.DialogResult.Yes Then
                        Process.Start(My.Application.Info.DirectoryPath &
"\_kompres")
                    End If

                Catch ex As Exception
                    lblProgress.Visible = False
                    pbLoader.Visible = False
                    MessageBox.Show("Ada kesalahan proses")
                End Try
                setButton(True)
            End If
        End If
    End Sub

    Private Sub btnDekompres_Click_1(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnDekompres.Click
        setButton(False)
        'buka file dialog untuk memilih file
        ofdImage.Filter = "Huffman Compression Files (*.cmp)|*.cmp"
        ofdImage.FileName = ""
        ofdImage.InitialDirectory = My.Application.Info.DirectoryPath &
"\_kompres"
        If ofdImage.ShowDialog = Windows.Forms.DialogResult.OK Then

            'dapatkan panjang isi file
            Dim L As Long = FileSystem.FileLen(ofdImage.FileName)

            'buat sebuah array dan load file ke dalamnya
            Dim B(0 To L - 1) As Byte
            FileSystem.FileOpen(1, ofdImage.FileName, OpenMode.Binary,
OpenAccess.Read)
            FileSystem.FileGet(1, B)
            FileSystem.FileClose(1)

            'proses kompresi
            start_time = Now
            lblProgress.Visible = True
            Application.DoEvents()
```

```
            Dim CMP() As Byte = HuffmanCoding.DecompressByteArray(B)
            Application.DoEvents()
            lblProgress.Visible = False
            'akhir of proses kompresi

            'simpan file hasil dekompresi dengan extension .bmp
            FileSystem.FileOpen(1, ofdImage.FileName & ".bmp",
OpenMode.Binary, OpenAccess.Write)
            FileSystem.FilePut(1, CMP)
            FileSystem.FileClose(1)

            'pindahkan hasil (.bmp) ke folder _dekompres
            Dim dFrom As String = ofdImage.FileName & ".bmp"
            Dim dTo As String = My.Application.Info.DirectoryPath &
"\_dekompres\" & ofdImage.SafeFileName & ".bmp"
            If System.IO.File.Exists(dFrom) = True Then
                If System.IO.File.Exists(dTo) = True Then
                    System.IO.File.Delete(dTo)
                End If
                System.IO.File.Move(dFrom, dTo)
            End If

            'tampilkan hasil dekompress
            Dim newImage As Bitmap = Image.FromFile(dTo)
            pbReal.Image = newImage
            imageFileNameCB2 =
System.IO.Path.GetFileNameWithoutExtension(dTo)
            imageFileNameCB2wExt = System.IO.Path.GetFileName(dTo)
            imageSizeCB2 = My.Computer.FileSystem.GetFileInfo(dTo).Length

            lblInfo3.Text = "Nama File = " & imageFileNameCB2wExt & vbNewLine
& _
                            "Panjang Image = " & newImage.Width & " px" &
vbNewLine & _
                            "Lebar Image = " & newImage.Height & " px" &
vbNewLine & _
                            "Ukuran File = " & imageSizeCB2.ToString("##,##")
& " Bytes"


            MessageBox.Show("PROSES DEKOMPRES SELESAI")
            Process.Start(My.Application.Info.DirectoryPath & "\_dekompres")


            ReDim citraInput(newImage.Width, newImage.Height)
            Dim jumlahpixelkompres As Int32 = newImage.Width *
newImage.Height
            Dim n_progress As Int32 = jumlahpixelkompres * 0.01
            ReDim idx(jumlahpixelkompres - 1)
            Dim counter, distance, min As Integer
            counter = 0

            Dim arrayTemp(2) As String
            arrayTemp = imageFileNameCB2wExt.Split("#")
            Dim arrayTempDetail(3) As String
            arrayTempDetail = arrayTemp(2).Split("_")
            Dim tempTotalCb As Integer = arrayTempDetail(0)
```

```
            ReDim tempLoadCodebook(tempTotalCb)

            Dim MSE As Double = 0

            For x As Int32 = 0 To newImage.Width - 1
                For y As Int32 = 0 To newImage.Height - 1
                    distance = 0
                    min = 1000000
                    temp_clr = newImage.GetPixel(x, y)
                    citraInput(x, y).cR = temp_clr.R
                    citraInput(x, y).cG = temp_clr.G
                    citraInput(x, y).cB = temp_clr.B

                    For ed As Int32 = 0 To tempTotalCb
                        distance = Math.Abs(citraInput(x, y).cR -
tempLoadCodebook(ed).cR) + Math.Abs(citraInput(x, y).cG -
tempLoadCodebook(ed).cG) + Math.Abs(citraInput(x, y).cB -
tempLoadCodebook(ed).cB)
                        distance = Math.Sqrt(distance)
                        If distance < min Then
                            min = distance
                            idx(counter) = ed
                        End If
                    Next
                    tempM1 = citraInput(x, y).cR -
tempLoadCodebook(idx(counter)).cR
                    tempM2 = citraInput(x, y).cG -
tempLoadCodebook(idx(counter)).cG
                    tempM3 = citraInput(x, y).cB -
tempLoadCodebook(idx(counter)).cB
                    tempMSE = (tempM1 * tempM1) + (tempM2 * tempM2) + (tempM3
* tempM3)
                    tempMSE = tempMSE / 3
                    MSE += tempMSE
                    counter = counter + 1
                Next
            Next

            MSE = MSE / jumlahpixelkompres
            Dim strMSE As String = MSE.ToString("00.00")
            Dim PSNR As Double
            PSNR = 20 * (Math.Log10(255 / Math.Sqrt(MSE)))
            Dim strPSNR As String = PSNR.ToString("00.00") & " dB"

            end_time = Now
            elapsed_time = end_time.Subtract(start_time)

            lblInfo2B.Text = "MSE = " & strMSE & vbNewLine & _
                             "PSNR = " & strPSNR & vbNewLine & _
                             "Waktu yang diperlukan untuk dekompresi = " &
vbNewLine & elapsed_time.ToString & " detik"
        End If
        setButton(True)
    End Sub
```

```
    Private Sub cbJmlhWarna_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cbJmlhWarna.SelectedIndexChanged
        txtJariJari.Text = Math.Sqrt(Val(cbJmlhWarna.Text)) / 2 - 1
    End Sub

    Private Sub txtJariJari_MouseHover(ByVal sender As Object, ByVal e As
System.EventArgs) Handles txtJariJari.MouseHover
        toolTip.UseFading = True
        toolTip.UseAnimation = True
        toolTip.Show("Nilai Jari-Jari = " & 0 & "-" &
Math.Sqrt(Val(cbJmlhWarna.Text)) / 2 - 1, txtJariJari)
    End Sub

    Private Sub txtJariJari_TextChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles txtJariJari.TextChanged
        If Val(txtJariJari.Text) > Val(Math.Sqrt(Val(cbJmlhWarna.Text)) / 2 -
1) Then
            txtJariJari.Text = Math.Sqrt(Val(cbJmlhWarna.Text)) / 2 - 1
        End If
    End Sub

    Private Sub txtIterasi_MouseHover(ByVal sender As Object, ByVal e As
System.EventArgs) Handles txtIterasi.MouseHover
        toolTip.UseFading = True
        toolTip.UseAnimation = True
        toolTip.Show("Nilai Iterasi = 1-1000", txtIterasi)
    End Sub

    Private Sub txtLearningRate_MouseHover(ByVal sender As Object, ByVal e As
System.EventArgs) Handles txtLearningRate.MouseHover
        toolTip.UseFading = True
        toolTip.UseAnimation = True
        toolTip.Show("Nilai Learning Rate = 0.1-1.0", txtLearningRate)
    End Sub
End Class
```

```
Module HuffmanCoding

    ' first define the BitsArray, which is used to perform bitstoring and
reading.
    Private Class bitsarray
        Public BytesUsed As Integer
        Public BytesArray() As Byte
        Public ByteBuffer As Integer
        Public ByteBufferUsed As Integer

        ' the constructer
        Public Sub New()
            BytesUsed = 0
            ByteBufferUsed = 0
            ReDim BytesArray(0 To 1023)
        End Sub
```

```
        ' SET BYTE ARRAY
        Public Sub SetArray(ByVal A() As Byte)
            BytesArray = A
            BytesUsed = 0
            ByteBufferUsed = -1
        End Sub

        ' add single bit to the buffer
        Public Sub AddBit(ByVal BitValue As Integer)
            ByteBufferUsed = ByteBufferUsed + 1
            BitValue = BitValue Mod 2
' extract the first bit
            ByteBuffer = ByteBuffer Or (BitValue * Math.Pow(2, ByteBufferUsed
- 1)) ' add the bit to the buffer
            If ByteBufferUsed = 8 Then
                BytesArray(BytesUsed) = ByteBuffer
                BytesUsed = BytesUsed + 1
                ByteBuffer = 0
                ByteBufferUsed = 0
                If BytesUsed = BytesArray.Length Then
                    ReDim Preserve BytesArray(0 To BytesUsed + 1023)
                End If
            End If
        End Sub


        ' read single bit from the buffer
        Public Function ReadBit() As Integer
            If ByteBufferUsed = -1 Then
                ByteBuffer = BytesArray(0)
                BytesUsed = 0
                ByteBufferUsed = 0
            End If
            If ByteBufferUsed = 8 Then
                BytesUsed = BytesUsed + 1
                ByteBuffer = BytesArray(BytesUsed)
                ByteBufferUsed = 0
            End If
            Dim Bit As Integer
            Bit = ByteBuffer Mod 2
            ByteBuffer = ByteBuffer \ 2
            ByteBufferUsed = ByteBufferUsed + 1
            Return Bit
        End Function

        Public Function ReadBits(ByVal Bits As Integer) As Integer
            Dim V As Integer
            Dim K As Integer
            V = 0
            For K = 0 To Bits - 1
                V = V + (Math.Pow(2, K) * ReadBit())
            Next
            Return V
        End Function
```

```
        ' add a number of bits to the array
        Public Sub AddBits(ByVal Value As Integer, ByVal Bits As Integer)
            Dim I As Integer
            For I = 1 To Bits
                AddBit(Value Mod 2)
                Value = Value \ 2
            Next
        End Sub

        ' save the rest of the byte
        Public Sub FlushBuffer()
            Do While ByteBufferUsed <> 0
                AddBit(0)
            Loop
        End Sub

        ' get the bits array
        Public Function GetBitsArray() As Byte()
            Dim B() As Byte
            Dim I As Integer
            ReDim B(0 To BytesUsed - 1)
            For I = 0 To BytesUsed - 1
                B(I) = BytesArray(I)
            Next
            Return B
        End Function

    End Class

    ' next define the tree nodes data struction
    Private Class BinaryTreeNode
        Public Probability As Long
        Public RightNode As BinaryTreeNode
        Public LeftNode As BinaryTreeNode
        Public Symbol As Integer
    End Class

    ' find the probabilities for a set of symbols
    Private Function FindProbabilitiesForSymbols(ByVal Data() As Byte) As
Long()
        Dim B(0 To 255) As Long
        Dim I As Integer

        ' set probabilities to zero
        For I = 0 To 255
            B(I) = 0
        Next

        ' compute the probabilities
        For I = 0 To Data.Length - 1
            B(Data(I)) += 1
        Next

        Return B
    End Function
```

```
    Private Function GetSymbolCodes(ByVal N As BinaryTreeNode, ByVal Prefix
As String) As Collection

        If N.LeftNode Is Nothing And N.RightNode Is Nothing Then
            Dim C As New Collection
            C.Add(N.Symbol)
            C.Add(Prefix)
            Return C
        End If

        Dim C1 As Collection
        Dim C2 As Collection
        Dim C3 As New Collection

        If Not N.LeftNode Is Nothing Then
            C1 = GetSymbolCodes(N.LeftNode, Prefix & "0")
        Else
            C1 = New Collection
        End If

        If Not N.RightNode Is Nothing Then
            C2 = GetSymbolCodes(N.RightNode, Prefix & "1")
        Else
            C2 = New Collection
        End If

        Dim I As Integer
        For I = 1 To C1.Count
            C3.Add(C1(I))
        Next
        For I = 1 To C2.Count
            C3.Add(C2(I))
        Next
        Return C3
    End Function


    Private Sub StoreTree(ByVal Node As BinaryTreeNode, ByRef BA As
bitsarray)
        If Node Is Nothing Then
            Exit Sub
        End If

        If Node.Symbol <> -1 Then
            BA.AddBits(3, 2)
            BA.AddBits(Node.Symbol, 8)
            Exit Sub
        End If

        If Not Node.LeftNode Is Nothing Then
            BA.AddBits(0, 2)
            StoreTree(Node.LeftNode, BA)
        Else
            BA.AddBits(2, 2)
        End If

        If Not Node.RightNode Is Nothing Then
```

```
            BA.AddBits(1, 2)
            StoreTree(Node.RightNode, BA)
        Else
            BA.AddBits(2, 2)
        End If

    End Sub

    ' load tree from bits array
    Private Sub LoadTree(ByRef Node As BinaryTreeNode, ByVal BA As bitsarray)
        ' get two bits for the command
        Dim CMD As Integer
        Dim K1 As Integer
        Dim K2 As Integer
        CMD = BA.ReadBits(2)

        If CMD = 3 Then
            Node.Symbol = BA.ReadBits(8)
            Return
        End If

        If CMD = 0 Then
            Node.LeftNode = New BinaryTreeNode
            LoadTree(Node.LeftNode, BA)
        Else
            K1 = 0
        End If

        CMD = BA.ReadBits(2)
        If CMD = 1 Then
            Node.RightNode = New BinaryTreeNode
            LoadTree(Node.RightNode, BA)
        Else
            K2 = 0
        End If
    End Sub


    ' compress the data
    Public Function CompressByteArray(ByVal Data() As Byte) As Byte()
        ' first get the probabilities
        Dim P() As Long = FindProbabilitiesForSymbols(Data)

        ' next start creating the tree
        Dim NodeList(0 To 255) As BinaryTreeNode
        Dim NodeListUsed As Integer = 0
        Dim Node As BinaryTreeNode
        Dim I As Integer
        For I = 0 To 255
            If P(I) > 0 Then
                ' create a node
                Node = New BinaryTreeNode
                Node.Symbol = I                 ' the symbol the node is
representing
                Node.Probability = P(I)         ' copy the probability
                Node.RightNode = Nothing        ' the default is nothing, but
it is okay to assign a value
```

```
            Node.LeftNode = Nothing         ' same here
            NodeList(NodeListUsed) = Node   ' add the new node
            NodeListUsed += 1               ' update the number of nodes.
        End If
    Next

    ' next build the tree structure from the leaves
    Do While NodeListUsed > 1

        ' find the two nodes with the minimum probabilities
        Dim MinLoc1 As Integer
        Dim MinLoc2 As Integer
        Dim N1 As BinaryTreeNode
        Dim N2 As BinaryTreeNode

        ' find location of first min probability
        MinLoc1 = 0
        For I = 1 To NodeListUsed - 1
            If NodeList(I).Probability < NodeList(MinLoc1).Probability
Then
                MinLoc1 = I
            End If
        Next

        ' remove the item
        N1 = NodeList(MinLoc1)
        For I = MinLoc1 To NodeListUsed - 2
            NodeList(I) = NodeList(I + 1)
        Next
        NodeListUsed -= 1

        ' find the location of the second min probability
        MinLoc2 = 0
        For I = 1 To NodeListUsed - 1
            If NodeList(I).Probability < NodeList(MinLoc2).Probability
Then
                MinLoc2 = I
            End If
        Next

        ' remove the item
        N2 = NodeList(MinLoc2)
        For I = MinLoc2 To NodeListUsed - 2
            NodeList(I) = NodeList(I + 1)
        Next
        NodeListUsed -= 1


        ' set parent node
        Node = New BinaryTreeNode
        Node.Probability = N1.Probability + N2.Probability
        Node.LeftNode = N1
        Node.RightNode = N2
        Node.Symbol = -1

        ' add this node to the liost
        NodeList(NodeListUsed) = Node
```

```
            NodeListUsed += 1
        Loop

        ' now we got the tree, create the look up table for the symbols with
code
        Dim SymbolCodes As Collection = GetSymbolCodes(NodeList(0), "")
        Dim SYM(0 To 255) As String
        Dim I2 As Integer
        For I = 0 To 255
            For I2 = 1 To SymbolCodes.Count Step 2
                If SymbolCodes(I2) = I Then
                    SYM(I) = SymbolCodes(I2 + 1)
                End If
            Next
        Next

        ' define the bit array
        Dim BA As New bitsarray

        ' save tree structure
        Node = NodeList(0)
        I = 0
        BA.AddBits(Data.Length, 32)             ' length of data
        BA.AddBits(SymbolCodes.Count \ 2, 8)    ' number of symbols
        StoreTree(Node, BA)                     ' store the tree

        ' now store the symbols
        For I = 0 To Data.Length - 1
            For I2 = 0 To SYM(Data(I)).Length - 1
                BA.AddBit(SYM(Data(I)).Substring(I2, 1))
            Next
        Next

        BA.FlushBuffer()
        Return BA.GetBitsArray
    End Function

    Public Function DecompressByteArray(ByVal Data() As Byte) As Byte()
        Dim BA As New bitsarray
        BA.SetArray(Data)
        Dim Length As Long = BA.ReadBits(32)    ' get the length of data
        Dim NoOfSymbols = BA.ReadBits(8)        ' get howmany symbols in the
tree
        Dim Node As New BinaryTreeNode
        Node.Symbol = -1
        LoadTree(Node, BA)                      ' generate the tree

        ' now we got the tree, create the look up table for the symbols with
code
        Dim SymbolCodes As Collection = GetSymbolCodes(Node, "")
        Dim SYM(0 To 255) As String
        Dim I2 As Integer
        For I = 0 To 255
            For I2 = 1 To SymbolCodes.Count Step 2
                If SymbolCodes(I2) = I Then
                    SYM(I) = SymbolCodes(I2 + 1)
                End If
```

```
            Next
        Next

        ' the last part is to extract the data from the stream
        Dim Result(0 To Length - 1) As Byte
        Dim Bt As String = ""
        Dim RC As Long = 0
        Dim FLG As Boolean
        Do While RC < Length
            FLG = False
            ' check if the bit is in the table
            For i = 0 To 255
                If SYM(i) = Bt And (Not SYM(i) Is Nothing) Then
                    Result(RC) = i
                    RC = RC + 1
                    Bt = ""
                    FLG = True
                    Exit For
                End If
            Next

            If Not FLG Then
                ' add a bit
                Bt = Bt & BA.ReadBit
            End If
        Loop
        Return Result
    End Function


End Module
```

Universitas Kristen Duta Wacana
Fakultas Teknik Program Studi Teknik Informatika

KARTU KONSULTASI TUGAS AKHIR

NIM : 22 05 384

Nama : Agung Prasetyo

Judul : KOMPRESI CITRA BERWARNA DENGAN METODE KOHONEN SELF-ORGANIZED MAP

Dosen Pembimbing : Bp. Sri Suwarno

No. : 1 Tanggal : 22 Des '10	Paraf
Catatan Perkembangan : Bab I & II Ok	

No. : 2 Tanggal : 22 Mei	Paraf
Catatan Perkembangan : Bab III Implementasi interface tambah Tabel	

Catatan : -Mahasiswa wajib membawa Kartu Konsultasi pada saat pertemuan pembimbingan dengan Dosen Pembimbing
- Jumlah minimal pertemuan yang diakui program Studi adalah 6x (enam kali)

No. : 3	Paraf
Tanggal : 30 Mei 2011	

Catatan Perkembangan :

Bab IV revisi.

No. : 4	Paraf
Tanggal : 30 Mei	

Catatan Perkembangan :

Bab I OK

No. :	Paraf
Tanggal :	

Catatan Perkembangan :

No. :	Paraf
Tanggal :	

Catatan Perkembangan :

Universitas Kristen Duta Wacana
Fakultas Teknik Program Studi Teknik Informatika

KARTU KONSULTASI TUGAS AKHIR

NIM : 22 05 3811

Nama : Agung Prasetyo

Judul : KOMPRESI CITRA BERWARNA DENGAN METODE KOHONEN SELF-ORGANIZED MAP

Dosen Pembimbing II : Ibu Widi Hapsari

No. : 1	**Paraf**
Tanggal : 22 Desember '10	
Catatan Perkembangan : jabarkan tentang kompresi citra, dikaitkan dg SOM.	

No. : 2	**Paraf**
Tanggal : 20 Mei '11	
Catatan Perkembangan : Bab I & II Ok	

Catatan : - Mahasiswa wajib membawa Kartu Konsultasi pada saat pertemuan pembimbingan dengan Dosen Pembimbing.
- Jumlah minimal pertemuan yang diakui Program Studi adalah 6x (enam kali)

No. : 3	Paraf
Tanggal : 23 Mei '11	

Catatan Perkembangan :

Bab III oh

No. : 4	Paraf
Tanggal : 27 Mei '11	

Catatan Perkembangan :

Demo Program + revisi tampilan

No. : 5	Paraf
Tanggal : 1 Juni '11	

Catatan Perkembangan :

Bab IV & V oh

No. :	Paraf
Tanggal :	

Catatan Perkembangan :

Universitas Kristen Duta Wacana
Fakultas Teknologi Informasi
Program Studi Teknik Informatika

FORMULIR CATATAN REVISI TUGAS AKHIR

(Diisi oleh Ketua Team Penguji)

Pada hari ini : Kamis Tanggal : 9 Juni 2011

Setelah dilakukan Ujian Tugas Akhir maka dengan ini Ketua Team Penguji Tugas Akhir menyatakan bahwa mahasiswa tersebut dibawah ini :

NAMA MAHASISWA : **AGUNG PRASETYO**
Nomor Induk Mahasiswa : 22 05 3811
Judul Tugas Akhir : Kompresi Citra Berwarna dengan Kohonen Self-Organized Map dan Metode Huffman.

Dosen Pembimbing I : Ir. Sri Suwarno, M.Eng.
Dosen Pembimbing II : Dra. Widi Hapsari, M.T.

Menyatakan beberapa perubahan tugas akhir yang harus dilakukan oleh mahasiswa tersebut diatas :

No.	CATATAN PERBAIKAN
1.	Abstrak, kesimpulan
2.	Gb. hal 27. dilengkapi gambar
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

Perubahan diatas harus sudah diselesaikan paling lambat tanggal : 9 Juli 2011

Yogyakarta, 9 Juni 2011
Ketua Team Penguji,

Dra. Widi Hapsari, MT

Catatan
- 1 (satu) lembar untuk mahasiswa
- 1 (satu) lembar untuk arsip

Universitas Kristen Duta Wacana
Fakultas Teknik Program Studi Teknik Informatika
Jl. Dr. Wahidin Sudirohusodo 5 – 25 Yogyakarta 55224
Telp : (0274) 563929 Faks. : (0274) 51323

FORMULIR PERBAIKAN (REVISI) TUGAS AKHIR

Dengan ini kami menyatakan bahwa mahasiswa yang melakukan Tugas Akhir dibawah ini :

Nama Mahasiswa	: Agung Prasetyo
Nim	: 22053811
Judul Tugas Akhir	: Kompresi Citra Berwarna Dengan Kohonen Self-Organized Map dan Metode Huffman
Tgl. Pendadaran	: 9 Juni 2011
Tgl. Revisi	: 16 Juni 2011

Telah melakukan perbaikan Tugas Akhir dengan lengkap.
Demikian pernyataan kami agar dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 22 Juni 11

Dosen Pembimbing I	Dosen Pembimbing II
(Ir. Sri Suwarno, M.Eng.)	(Dra. Widi Hapsari, M.T.)