KOMPRESI CITRA DIGITAL MENGGUNAKAN *MODIFIED HAAR WAVELET TRANSFORM* DAN RLE PADA CITRA ASTRONOMI

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

oleh :
ADIN EKA FIYANZAR
0410960003 - 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2008

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

KOMPRESI CITRA DIGITAL MENGGUNAKAN *MODIFIED HAAR WAVELET TRANSFORM* DAN RLE PADA CITRA ASTRONOMI

oleh :

ADIN EKA FIYANZAR
0410960003 - 96

Telah dipertahankan di depan Majelis Penguji
pada tanggal 22 Juli 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Muh. Arif Rahman, M.Kom	**Edy Santoso, S.Si, M.Kom**
NIP. 131 971 481	**NIP. 132 304 307**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Drs. Agus Suryanto, M.Sc
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	**Adin Eka Fiyanzar**
NIM	**:**	**0410960003**
Program Studi	**:**	**Ilmu Komputer**
Penulis Skripsi Berjudul	**:**	**Kompresi Citra Digital Menggunakan *Modified Haar Wavelet Transform* dan RLE pada Citra Astronomi**

Dengan ini menyatakan bahwa :

1. **Isi dari Skripsi yang saya buat adalah benar – benar karya sendiri dan tidak menjiplak karya orang lain, selain nama – nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 22 Juli 2008
Yang menyatakan,

(Adin Eka Fiyanzar)
NIM. 0410960003

KOMPRESI CITRA DIGITAL MENGGUNAKAN *MODIFIED HAAR WAVELET TRANSFORM* DAN RLE PADA CITRA ASTRONOMI

ABSTRAK

Citra astronomi memiliki ukuran *file* yang besar, sehingga memerlukan kompresi untuk menghemat ruang penyimpanan dan mempercepat waktu pengiriman citra. Teori *wavelet* dapat digunakan untuk kompresi citra. Dan kompresi RLE dapat digunakan untuk mengkompresi matrik transformasi *wavelet*. Berdasarkan penelitian yang telah dilakukan sebelumnya, maka permasalahan pada penelitian ini adalah bagaimana merancang dan mengimplementasikan *Modified Haar Wavelet Transform* pada citra astronomi yang dilanjutkan dengan kompresi RLE.

Tahapan yang dilakukan untuk melakukan proses kompresi adalah konversi RGBtoYUV, *Haar Wavelet Transform*, *Quantization*, dan RLE *Encoding*. Sedangkan tahap untuk melakukan proses dekompresi adalah RLE *Decoder*, *Invers Haar Wavelet Transform*, dan konversi YUVtoRGB .

Berdasarkan penelitian yang telah dilakukan, didapatkan nilai rata-rata rasio dari ke-20 citra uji menggunakan metode *hard thresholding* sebesar 89,08 %. Dan nilai rata-rata rasio menggunakan *soft thresholding* sebesar 88,96 %. Pada pengujian tingkat kesalahan yang diukur menggunakan nilai MSE, didapatkan nilai rata-rata menggunakan metode *hard thresholding* sebesar 18,38. Dan nilai rata-rata MSE menggunakan *soft thresholding* sebesar 25,95, sehingga dapat diambil kesimpulan bahwa metode *hard thresholding* lebih layak digunakan untuk mengkompresi citra astronomi karena menghasilkan MSE yang lebih kecil.

DIGITAL IMAGE COMPRESSION USING MODIFIED HAAR WAVELET TRANSFORM AND RLE ON ASTRONOMICAL IMAGES

ABSTRACT

Astronomical images has big file size, hence compression is needed to save storage space and to minimize time required for images transmition. Wavelet theory can be used for digital images compression. And RLE compression can be used for compress wavelet matrix. Based on previous research, the purpose of this research are to design and to implement Modified Haar Wavelet Transform at astronomical images which completed by RLE compression.

Step for compression process are conversion from RGB to YUV, Haar Wavelet Transform, quantization, and RLE encoding. Step for decompression are RLE decoder, Invers Haar Wavelet Transform, and conversion from YUV to RGB.

The result of experiment show mean value ratio from 20 test images that use hard thresholding method equal to 89,09%. And mean value ratio that use soft thresholding method equal to 88,96%. Error rate test that measured using MSE value, show mean value of MSE that use hard thresholding method equal to 18,38. And mean value of MSE that use soft thresholding equal to 25,95. Then, it can be concluded that hard thresholding method is more competent for astronomical images compression, because it has smaller MSE.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga Skripsi yang berjudul "Kompresi Citra Digital Menggunakan *Modified Haar Wavelet Transform* dan RLE pada Citra Astronomi" dapat diselesaikan.

Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, Universitas Brawijaya.

Banyak pihak yang berperan atas terselesaikannya penelitian dan penulisan Skripsi ini. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Drs. Muh. Arif Rahman, M.Kom. selaku dosen pembimbing I yang telah memberikan saran dan bimbingan atas penelitian "Kompresi Citra Digital Menggunakan *Modified Haar Wavelet Transform Transform* dan *RLE* pada Citra Astronomi" dan selaku Penasehat Akademik.
2. Edy Santoso, S.Si., M.Kom. selaku dosen pembimbing II atas bimbingan dalam penulisan laporan.
3. Dr. Agus Suryanto, Msc., selaku ketua jurusan Matematika FMIPA UB Malang.
4. Wayan Firdaus Mahmudy, SSi, MT., selaku Ketua Program Studi Ilmu Komputer UB Malang.
5. Segenap bapak dan ibu dosen yang telah mendidik, dan mengamalkan ilmunya kepada penulis.
6. Segenap staf dan karyawan di Jurusan Matematika FMIPA UB.
7. Papa, mama, dan adikku yang telah memberikan semangat dan mendoakan selama belajar di Program Studi Ilmu Komputer FMIPA UB dan dalam mengerjakan skripsi.
8. Dimas, Gheeta, Arya, Andika, serta teman-teman ilkomers '04 yang telah memberikan semangat dan dukungan pada penulis.
9. Semua pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, dan memiliki banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga penulisan skripsi ini bermanfaat bagi pembaca.

Malang, Juli 2008

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR LAMPIRAN

Halaman

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Citra digital telah banyak digunakan dalam berbagai bidang untuk merepresentasikan berbagai tipe informasi yang kompleks. Misalnya, pada bidang astronomi (ilmu perbintangan), kedokteran, SIG dan lain-lain. Citra dalam bidang astronomi biasa disebut dengan citra astronomi.

Citra astronomi sebagian besar berisi ruang hampa. Permasalahannya citra astronomi dapat berukuran sangat besar. Sebagai contohnya, *Space Telescope Science Institute* mempunyai *plate* foto digital yang mencakup seluruh angkasa, menghasilkan 1500 citra yang masing-masing memiliki 14000x14000 (0.4GB) atau 23040x23040 (1.1GB) 16-bit pixel. Beberapa kelompok astronomi kini membuat kamera dengan mosaik CCD yang besar (masing-masing berukuran 2048x2048 atau lebih). Alat ini akan digunakan pada proyek yang menghasilkan data kira-kira melebihi 100 Mbytes tiap 5 menit selama bertahun-tahun. Oleh karena itu, kompresi untuk citra semacam ini diperlukan untuk mengurangi volume data yang disimpan (suatu pertimbangan penting untuk survei angkasa digital dalam skala besar) dan dapat menyingkat waktu yang diperlukan untuk mengirim citra (berguna untuk peninjauan jarak jauh atau akses jarak jauh ke berkas data).(White, 1995)

Kompresi pada citra biasa disebut dengan kompresi citra. Kompresi Citra adalah pengurangan jumlah data yang dibutuhkan untuk merepresentasikan citra digital. Dasar proses pengurangannya yaitu dengan menghilangkan data yang berlebihan (*redundant*)(Gonzalez, 2002).

White (1995) mengatakan bahwa citra astronomi memiliki beberapa ciri yang tidak biasa. Suatu citra astronomi terdiri dari objek yang hampir seluruhnya menyebar. Oleh karena itu, kompresi *lossless* (kompresi yang tidak menghilangkan informasi) tidak bekerja begitu baik. Selain itu, citra ini biasanya ditujukan untuk analisis kuantitatif, jadi metode kompresi yang digunakan harus dapat menjamin untuk tidak menghilangkan informasi yang berguna.

Pada saat ini teori wavelet merupakan metode yang banyak digunakan dalam pengolahan citra, *denoising*, dan kompresi. Salah satu jenis wavelet adalah *Haar Wavelet Transform*. *Haar Wavelet*

mentransformasikan data dari daerah ruang (*spatial domain*) ke daerah frekuensi (*frequency domain*). Cara ini menghasilkan data yang lebih mudah diproses untuk kompresi lebih lanjut (Salomon, 2004).

Ada berbagai sebab yang dikemukakan oleh Ian Kaplan (2004) mengapa metode Haar Wavelet digunakan, diantaranya: memiliki konsep yang sederhana, prosesnya cepat, efisiensi memori, dan dapat dibalikkan dengan tepat tanpa *edge effects* yang merupakan masalah bagi transformasi wavelet yang lainnya.

Tujuan utama *Haar Wavelet Transform* adalah mengecilkan atau mengosongkan koefisien pada matrik *Haar transform* dari daerah matrik asli yang mengandung sedikit variasi. Sehingga terbentuk matrik *Haar transform* yang banyak memiliki koefisien bernilai nol (*sparse matrix*). Matrik yang miliki koefisien bernilai nol ini hanya membutuhkan sedikit memory untuk penyimpanannya. Untuk membentuk matrik yang demikian, maka diterapkan nilai *thresholding* non negatif pada koefisien matrik *Haar transform.* Nilai ini yang akan menentukan koefisien mana yang dinolkan dan mana yang tidak.(Ames, 2002)

Metode *thresholding* ada beberapa macam, diantaranya *hard thresholding*, *soft thresholding*, dan *universal thresholding. Hard thresholding* merupakan *thresholding* yang membuang (menolkan) koefisien-koefisien *wavelet* yang memiliki nilai di bawah *threshold* yang telah ditentukan. *Soft thresholding* merupakan jenis *thresholding* lain yang meredam koefisien-koefisien *wavelet* yang memiliki nilai di bawah *threshold* yang telah ditentukan. (Walidainy, 2004).

Setelah terbentuk *sparse matrix,* maka penyimpanan matrik lebih efektif jika matrik dikompresi terlebih dahulu menggunakan kompresi *lossless.* Salah satu contoh kompresi *lossless* adalah RLE yang memiliki prinsip kerja mengelompokkan koefisien yang bernilai sama. RLE (*Run Length Encoding*) merupakan metode yang telah banyak digunakan baik untuk kompresi teks maupun citra. Konsepnya yang mudah, cocok untuk metode kompresi yang membutuhkan proses cepat (Salomon, 2004).

Penelitian yang pernah dilakukan yang berhubungan dengan *Haar Wavelet* yaitu: Porwik dan Lisowska (2004) dalam penelitiannya yang berjudul *The Haar–Wavelet Transform in Digital Image Processing: Its Status and Achievements* menjelaskan bahwa

pemrosesan sinyal dua dimensi (citra) merupakan area yang efisien bagi pengaplikasian *Haar transforms* berdasarkan pada struktur wavelet yang dimilikinya. Ravinaj dan Sanavullah (2007), dengan judul penelitian *The Modified 2D-Haar Wavelet Transformation in Image Compresion*, melakukan modifikasi metode kompresi *Haar Wavelet* pada tahap kuantisasinya (*thresholding*) yaitu dengan cara menggunakan beberapa nilai *threshold* kompresi yang berbeda pada koefisien *wavelet.*

Dengan adanya kebutuhan kompresi citra astronomi, hasil penelitian yang dilakukan sebelumnya, keuntungan metode *Haar Wavelet Transform* dan *RLE*. Maka timbul inisiatif untuk merancang sebuah model kompresi citra menggunakan *Modified Haar Wavelet Transform* dan *RLE*, serta menerapkannya pada citra astronomi. Sehingga dapat diketahui rasio/nisbah kompresi dan tingkat kesalahan (*error rate*) pada citra astronomi. Dengan demikian, skripsi ini berjudul "**Kompresi Citra Digital Menggunakan *Modified Haar Wavelet Transform* dan RLE pada Citra Astronomi**".

1.2 Perumusan Masalah

Berdasarkan uraian pada latar belakang masalah, maka dalam skripsi ini dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana merancang dan mengimplementasikan model transformasi citra menggunakan *Modified Haar Wavelet Transform* sehingga dapat menghasilkan matrik tranformasi sesuai dengan nilai dan metode *thresholding* yang digunakan.
2. Bagaimana merancang dan mengimplementasikan model kompresi *RLE* sehingga dapat mengkompres matrik *Haar Wavelet Transform*.
3. Bagaimana rasio kompresi citra astronomi menggunakan *Modified Haar Wavelet Transform* dan *RLE*.
4. Bagaimana tingkat kesalahan citra astronomi menggunakan *Modified Haar Wavelet Transform* dan *RLE*.

1.3 Batasan Masalah

Dari permasalahan yang dijelaskan pada subbab 1.2, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Citra yang digunakan dalam penelitian ini adalah citra berwarna (*true color*) 24 bit.
2. Ukuran citra yang digunakan dalam penelitian adalah orthogonal 256 x 256.
3. Format citra yang digunakan adalah bitmap (.bmp).
4. Nilai *Threshold* (*e*) yang akan digunakan pada pengujian kompresi citra bernilai *e*=25, 10, 5 dan 1. Nilai tersebut sesuai dengan nilai yang digunakan pada penelitian Raviraj (2007).
5. Metode *thresholding* yang digunakan adalah *hard thresholding* dan *soft thresholding* (Walidainy, 2004).

1.4 Tujuan Penelitian

Tujuan penelitian yang ingin dicapai adalah:

1. Mengetahui rancangan dan implementasi model transformasi citra menggunakan *Modified Haar Wavelet Transform.*
2. Mengetahui rancangan dan implementasi model kompresi *RLE* untuk mengkompres matrik *Haar Wavelet Transform.*
3. Mengetahui rasio kompresi citra astronomi menggunakan *Modified Haar Wavelet Transform* dan *RLE.*
4. Mengetahui tingkat kesalahan citra astronomi terkompresi menggunakan *Modified Haar Wavelet Transform* dan *RLE.*

1.5 Manfaat Penelitian

Manfaat yang bisa didapatkan melalui penelitian ini adalah:

1. Menyediakan model perangkat lunak (*software*) untuk mengkompres dan mendekompres citra digital menggunakan *Modified Haar Wavelet Transform* dan *RLE*.
2. Menyediakan informasi mengenai rasio pemampatan dan tingkat kesalahan citra astronomi menggunakan *Modified Haar Wavelet Transform* dan *RLE*.
3. Menyediakan referensi mengenai teknik kompresi citra menggunakan *Modified Haar Wavelet Transform* dan *RLE* bagi peneliti teknik kompresi yang lain.

1.6 Metodologi Penelitian

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan konsep kompresi citra dan *Modified Haar Wavelet Transform* dan *RLE* dari berbagai referensi.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat rancangan model perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu membuat piranti lunak kompresi dan dekompresi citra dengan menggunakan *Modified Haar Wavelet Transform* dan *RLE*.
4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak dengan berbagai macam citra astronomi. Kemudian menganalisa hasil dari implementasi tersebut sesuai dengan tujuan yang dirumuskan sebelumnya, untuk kemudian dievaluasi dan disempurnakan.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

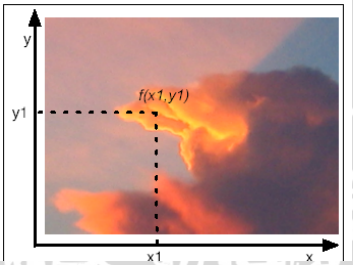
1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan citra astronomi, kompresi citra, *Haar Wavelet Transform* dan *RLE*.
3. BAB III METODOLOGI PENELITIAN
   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam mengkompresi dan mendekompresi citra menggunakan *Modified Haar Wavelet Transform* dan *RLE*.

4. BAB IV HASIL DAN PEMBAHASAN
   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa model sistem perangkat lunak yang dibangun, yaitu mengetahui rasio/nisbah kompresi dan tingkat kesalahan citra terkompresi yang dikompresi menggunakan *Modified Haar Wavelet Transform* dan *RLE*.
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

2.1 Citra Digital

Citra digital dapat didefinisikan sebagai fungsi dua variabel, *f(x,y)*, dimana *x* dan *y* adalah koordinat spasial dan nilai *f(x,y)* adalah intensitas citra pada koordinat tersebut, hal tersebut diilustrasikan pada Gambar 2.1. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*Red, Green, Blue* - RGB). (Balza, 2004)

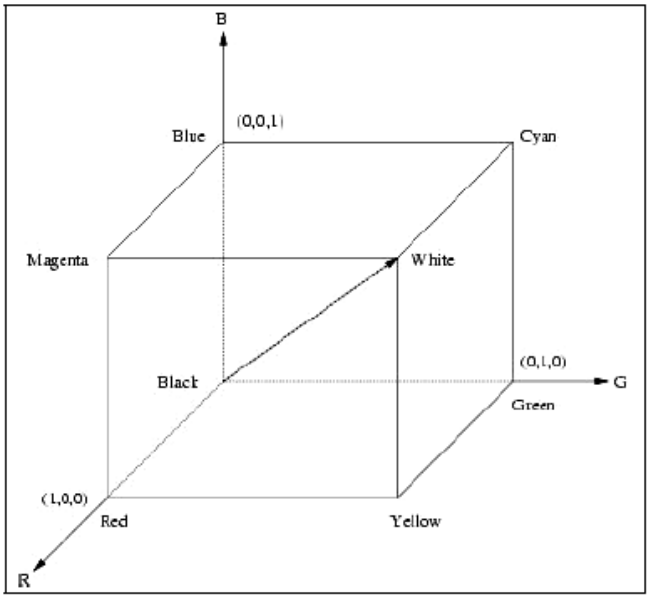


Gambar 2.1. Koordinat Citra Digital
Sumber: Suhendra, 2007

2.1.1 Citra Warna (*true color*)

Pada citra warna, setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari 3 warna dasar, yaitu merah, hijau, dan biru. Format citra ini sering disebut sebagai citra RGB (*red-green-blue*). Setiap warna dasar memiliki nilai maksimum 255 (8 bit), misalnya warna kuning merupakan kombinasi warna merah dan hijau sehingga nilai RGB-nya adalah 255 255 0. Dengan demikian setiap titik pada citra warna membutuhkan data 3 *byte*. (Balza, 2004)

Gambar 2.2 menunjukkan bentuk geometri dari model warna RGB untuk menspesifikasikan warna menggunakan sistem koordinat Cartesian. Spektrum *greyscale* (tingkat keabuan) yaitu warna yang dibentuk dari gabungan tiga warna utama dengan jumlah yang sama, berada pada garis yang menghubungkan titik hitam dan putih. (Suhendra, 2007)



Gambar 2.2. Koordinat warna RGB
Sumber: Suhendra, 2007

2.1.2 Model Warna YUV

YUV, dikenal juga sebagai Y'CbCr dan YpbPr, adalah sebuah *color space* dengan Y merupakan komponen *luminance* (*brightness*) dan U dan V adalah komponen *chrominance* (warna). YUV biasanya digunakan pada aplikasi video. Sinyal YUV diciptakan dari sumber RGB (*red*, *green*, *blue*), dengan diberi bobot tertentu, nilai R, G dan B ditambahkan untuk menghasilkan sinyal Y, yang menyatakan terang-gelap secara keseluruhan, atau *luminance* dari titik tersebut. Sinyal U lalu dibentuk dengan mengurangkan sinyal *blue* dari RGB dengan Y, dan V dengan mengurangkan *red.*

Untuk mengkonversi nilai RGB ke YUV digunakan Persamaan 2.1, 2.2, dan 2.3 (Handoko, 2005):

$$Y = (0.256788 * R + 0.504129 * G + 0.097906 * B) + 16 \quad (2.1)$$
$$U = (-0.148223 * R - 0.290993 * G + 0.439216 * B) + 128 \quad (2.2)$$
$$V = (0.439216 * R - 0.367788 * G - 0.071427 * B) + 128 \quad (2.3)$$

Dan pengubah balik dari YUV ke RGB dilakukan dengan Persamaan 2.4, 2.5, dan 2.6 (Handoko, 2005):

$$R = 1.164383 * Y * 0.9 + 1.596027 * (V - 128) \quad (2.4)$$

$$G = 1.164383 * Y * 0.9 - 0.391762 * (U - 128) - 0.812968 * (V - 128) \quad (2.5)$$

$$B = 1.164383 * Y * 0.9 + 2.017232 * (U - 128) \quad (2.6)$$

Mata manusia paling sensitif terhadap Y (*luminance*) dan paling tidak sensitif terhadap V (*chrominance*). Metode kompresi *lossy* biasanya memanfaatkan hal ini dengan cara membiarkan komponan Y, menghapus beberapa data U, dan menghapus lebih banyak data dari komponen V. (Salomon, 2004)

2.1.3 Citra Astronomi (*Astronomy Images*)

Menurut Hartman (1991) Astronomi/ilmu perbintangan adalah studi dari semua hal dan energi di alam semesta. Ilmu ini menekankan konsentrasi tentang energi dan semual hal dalam mengembangkan materialnya seperti planet, bintang, dan galaksi.

Sedangkan pengertian yang diambil dari wikipedia (2008) adalah ilmu yang melibatkan pengamatan dan penjelasan kejadian yang terjadi di luar Bumi dan atmosfernya. Ilmu ini mempelajari asal-usul, evolusi, sifat fisik dan kimiawi benda-benda yang bisa dilihat di langit dan di luar Bumi, juga proses yang melibatkan mereka.

Menurut Dhani (2002), bagi para astronom yang biasa berkutat dengan pengamatan visual, sebenarnya kegiatan meneropong atau memotret dengan pelat fotografi bukan jamannya lagi. Walaupun desain dasar teleskop hanya mengalami sedikit perubahan dalam ratusan tahun kurun waktu pemanfaatannya di dunia astronomi, namun para astronom masa kini banyak terbantu oleh teknologi berbasis digital, diantaranya adalah CCD (*chart Coupled Device*). Peralatan ini terdiri dari sebuah chip dengan permukaan peka terhadap cahaya. Dengan menempatkan perangkat ini pada teleskop, memungkinkan para astronom untuk menampilkan citra objek yang diamati di layar monitor. Selanjutnya citra tersebut dapat disimpan dalam *hard-disk* untuk kemudian dianalisis lebih lanjut. Gambar 2.3 menunjukkan beberapa contoh citra astronomi.

(a) (b)

Gambar 2.3. (a) Coronet Cluster, (b) M81 spiral galaxy
Sumber: http://photojournal.jpl.nasa.gov/

Salah satu contoh teleskop yang digunakan untuk mengambil citra astronomi adalah teleskop Hubble. Teleskop ini menghasilkan sekitar 10 gigabyte data setiap harinya. (http://www.bbc.co.uk/,2007)

2.2 Kompresi Citra

Kompresi Citra adalah pengurangan jumlah data yang dibutuhkan untuk merepresentasikan citra digital. Dasar proses pengurangannya yaitu dengan menghilangkan data yang berlebihan (*redundant*). Sedangkan Dekompresi citra adalah kebalikan proses kompresi, yaitu merekontruksi citra berdasarkan data kompresi. (Gonzalez, 2002)

Ada dua tipe utama kompresi citra, yaitu :

1. **Kompresi tipe *Lossy***

   Kompresi dimana terdapat data yang hilang selama proses kompresi. Akibatnya kualitas citra yang dihasilkan jauh lebih rendah dari pada kualitas citra asli. Contoh: *Color reduction, Transform coding*.

2. **Kompresi tipe *Lossless***

   Kompresi yang tidak menghilangkan informasi setelah proses kompresi terjadi, akibatnya kualitas citra hasil kompresi tidak menurun. Contoh: *RLE*, *Entropy Encoding* (Huffman, Aritmatik).(Gonzales, 2002)

2.3 *Wavelet Transform*

2.3.1 Tranformasi Citra

Transformasi citra, sesuai namanya, merupakan proses perubahan bentuk citra untuk mendapatkan suatu informasi tertentu. Transformasi bisa dibagi menjadi dua, yaitu Transformasi piksel/transformasi geometris dan transformasi ruang/domain/*space*. Transformasi piksel masih bermain di ruang/ domain yang sama (domain spasial), hanya posisi piksel yang kadang diubah, misalkan rotasi, translasi, *scaling*, *invers*, *shear*, dan lain-lain. Transformasi jenis ini relatif mudah diimplementasikan dan banyak aplikasi yang dapat melakukannya (Paint, ACDSee, dan lain-lain). Transformasi ruang merupakan proses perubahan citra dari suatu ruang/domain ke ruang/domain lainnya, sebagai contoh dari ruang spasial ke ruang frekuensi. (Suhendra, 2004)

2.3.2 Penelitian Pendahuluan

Pada tahun 2007, Ravinaj dan Sanavullah melakukan penelitian dengan judul *The Modified 2D-Haar Wavelet Transformation in Image Compresion.* Pada penelitian ini dilakukan modifikasi *Haar wavelet transform* yaitu dengan cara menerapkan beberapa nilai threshold yang berbeda pada proses kuantitasinya. Algoritma yang diusulkan pada penelitian ini antara lain:

1. Membaca citra dari *User.*
2. Mengaplikasikan 2D DWT menggunakan *haar wavelet transform.*
3. Menetapkan beberapa nilai theshold. Misalnya 25,10,5,1.
4. Menampilkan citra hasil.
5. Menghitung nilai MSE, MAE, dan PNSR.

Penelitian ini dilakukan hanya pada citra *grayscale* dengan berbagai ukuran. Selain melakukan kompresi, pada penelitian ini juga dilakukan penghilangan *noise* menggunakan *haar wavelet transform.*

2.3.3 *Haar Wavelet Transform* (HWT)

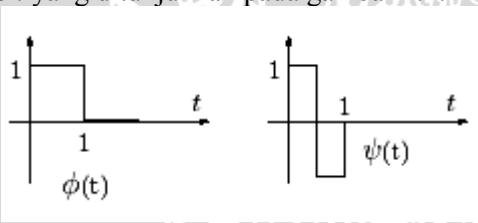
Transformasi *wavelet* dalam konteks pemrosesan sinyal adalah suatu metode yang digunakan untuk mendekomposisi sinyal

masukan ke dalam bentuk suatu gelombang yang disebut *wavelet* induk, dan menganalisis sinyal dengan memberi perlakuan terhadap koefisien *wavelet*.(Walidainy, 2004)

Haar wavelet transform adalah salah satu jenis transformasi *wavelet* yang paling lama dikenal. Fungsi Haar sudah digunakan sejak 1910 oleh matematikawan Hungaria bernama Alfred Haar. *Haar Transform* merupakan transformasi simetris dengan ukuran data $N=2^n$. (Porwik,2004)

Wavelet merupakan sebuah basis. Basis *wavelet* berasal dari sebuah fungsi penskalaan (*scaling function*). *Scaling function* memiliki sifat dapat disusun dari sejumlah salinan dirinya yang telah didilasikan, ditranslasikan, dan diskalakan. Fungsi ini diturunkan dari persamaan dilasi (*dilation equation*), yang dianggap sebagai dasar dari teori *wavelet*. Dari persamaan *scaling function* dapat dibentuk persamaan *wavelet* yang pertama (atau disebut juga *mother wavelet*). Dari mother *wavelet* dapat dibentuk *wavelet-wavelet* berikutnya (ψ^1, ψ^2 dan seterusnya) dengan cara mendilasikan (memampatkan atau meregangkan) dan menggeser *mother wavelet*. (Murni, 2003)

Haar transform menggunakan *Haar scaling function* dan *Haar wavelet function* yang ditunjukkan pada gambar 2.4.



Gambar 2.4. *Haar scaling function* dan *Haar wavelet function*
Sumber: Salomon, 2004

a. Haar scaling function

Fungsi dilasi Haar (*scaling*) didefinisikan dari rumus (Porwik, 2004):

$$\Phi(t) = \begin{cases} 1, & \text{for } t \in [0,1), \\ 0, & \text{for } t \notin [0,1). \end{cases} \tag{2.7}$$

Untuk sembarang fungsi Haar, maka akan menghasilkan fungsi:

$$\Phi_i^j(t) = \sqrt{2^j}\,\Phi(2^j t - i) \tag{2.8}$$

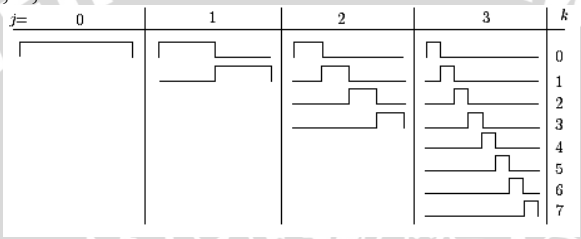
Untuk i = 0,1,...,2^j-1
j = 0,1,…, $\log_2$N-1

Konstanta $\sqrt{2^j}$ dipilih agar hasil skalar. $\left\langle \Phi_i^j, \Phi_i^j \right\rangle = 1, \Phi_i^j \in L^2(R)$.

Indek j menyatakan dilasi dan indek i menyatakan translasi. Sehingga didapatkan rentang linier ruang vektor V^j :

$$V^j = span\left\{\Phi_i^j\right\}_{i=0,\ldots,2^j-1} \qquad (2.9)$$

Pada gambar 2.5 diberikan contoh fungsi $\Phi(2^j t-i)$ dengan j= 0,1,2,3 dan i= 0,1,...,7.



Gambar 2.5. Fungsi $\Phi(2^j t-i)$
Sumber: Salomon, 2004

b. Haar wavelet function

Fungsi *wavelet* Haar didefinisikan dari rumus (Porwik, 2004):

$$\Psi(t) = \begin{cases} 1, & \text{for } t \in [0, \frac{1}{2}), \\ -1, & \text{for } t \in [\frac{1}{2}, 1), \\ 0, & \text{otherwise.} \end{cases} \qquad (2.10)$$

Untuk sembarang fungsi Haar, maka akan menghasilkan fungsi:

$$\Psi_i^j(t) = \sqrt{2^j}\,\Psi(2^j t - i) \qquad (2.11)$$

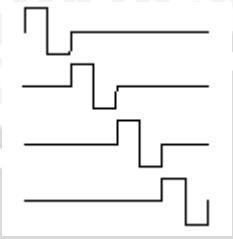
Untuk i = 0,1,...,2^j-1
j = 0,1,…, $\log_2$N-1

Konstanta $\sqrt{2^j}$ dipilih agar hasil skalar $\left\langle \Psi_i^j, \Psi_i^j \right\rangle = 1, \Psi_i^j \in L^2(R)$.

Indek j menyatakan dilasi dan indek i menyatakan translasi. Sehingga didapatkan rentang linier ruang vektor W^j :

$$W^j = span\left\{\Psi_i^j\right\}_{i=0,\ldots,2^j-1} \qquad (2.12)$$

Sebagai contoh pada gambar 2.6 ditunjukkan 4 *Haar wavelet* $\Psi(2^2 t-i)$ dengan i= 0,1,2,3.

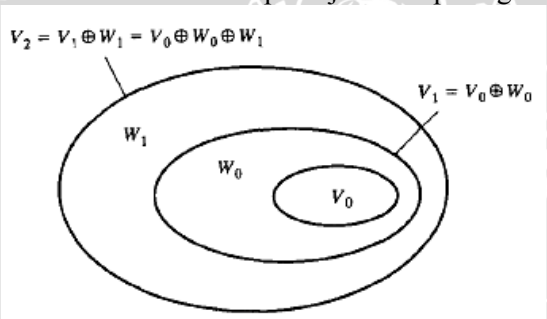


Gambar 2.6. Fungsi $\Psi(2^2t-i)$
Sumber: Salomon, 2004

Dalam multiresolusi analisis, basis *Haar* memiliki sifat yang sangat penting (Sandberg, 2000):

$$V^{j+1}=V^j \oplus W^j \qquad (2.13)$$

Misalnya $V^3=V^2 \oplus W^2$. Hal ini dapat dijelaskan pada gambar 2.7:



Gambar 2.7. Hubungan antara ruang fungsi *scaling* dan *wavelet*
Sumber: Gonzales, 2002

Haar wavelet bekerja pada data dengan metode transformasi vektor yang disebut *averaging* dan *differencing*. Hasil dari operasi *averaging* disebut *approximation coefficients*. Hasil dari operasi *differencing* disebut *detail coefficients*. (Ames, 2002)

Berdasarkan persamaan 2.8 dan 2.11 mengenai fungsi Haar dan fakta bahwa V^j dan W^j merentang seperti pada persamaan 2.9 dan 2.12 . Maka fungsi Φ dan fungsi Ψ dapat dinyatakan sebagai kombinasi linier fungsi basis dari ruang V dan W berturut-turut. Dalam *Haar wavelet*, fungsi Φ dan Ψ dapat ditulis (Porwik, 2004):

$$\Phi(t) = h(0)\sqrt{2}\,\Phi(2t) + h(1)\sqrt{2}\,\Phi(2t-1) \qquad (2.14)$$
$$\Psi(t) = g(0)\sqrt{2}\,\Phi(2t) + g(1)\sqrt{2}\,\Phi(2t-1) \qquad (2.15)$$

Dimana {h(0),h(1)} dan {g(0),g(1)} mendefinisikan tapis(*filter*) *low-pass* dan *high-pass* berturut-turut. Dalam hal ini, h(k), k=0,1 koefisiennya diketahui (Porwik, 2004):

$$h(0) = 1/\sqrt{2}\ ,h(1) = 1/\sqrt{2}\ ,g(0) = 1/\sqrt{2}\ ,g(1) = -1/\sqrt{2} \quad (2.16)$$

jika C menyatakan matrik citra, maka:

$$A(i) = 1/\sqrt{2}\ C(2i) + 1/\sqrt{2}\ C(2i+1) \quad (2.17)$$

$$D(i) = 1/\sqrt{2}\ C(2i) - 1/\sqrt{2}\ C(2i+1) \quad (2.18)$$

Dimana C(i) = vektor berukuran N, berisi baris atau kolom matrik C, $i \in \{0,1,...,N/2 -1\}$, A(i) = vektor berukuran N/2, berisi *approximation coefficients*, D(i) = vektor berukuran N/2, berisi *detail coefficients.* (Porwik, 2004)

Dari persamaan 2.17 dan 2.18, maka persamaan untuk rekonstruksi citra (*Invers Haar Wavelet Transform*) dinyatakan sebagai berikut (Porwik, 2004):

$$C(2i) = 1/\sqrt{2}\ A(i) + 1/\sqrt{2}\ D(i) \quad (2.19)$$

$$C(2i+1) = 1/\sqrt{2}\ A(i) - 1/\sqrt{2}\ D(i) \quad (2.20)$$

Hal tersebut juga dapat dijelaskan melalui perkalian matrik. Berdasarkan persamaan 2.13, maka dapat dibentuk matrik *wavelet* sebagai berikut. Untuk langkah ke 1, persamaan $V^n = V^{n-1} \oplus W^{n-1}$. matrik *wavelet* M_1 berbentuk:

$$M_1 = [\, \Phi^{n-2}_{j=0,...,2^{n-2}-1} \subset V^{n-1}, \Psi^{n-1}_{j=0,...,2^{n-1}-1} \subset W^{n-1} \,]^T$$

Untuk langkah ke 2, persamaan $V^{n-1} = V^{n-2} \oplus W^{n-2} \oplus W^{n-1}$, matrik *wavelet* M_2 berbentuk:

$$M_2=[\, \Phi^{n-2}_{j=0,...,2^{n-2}-1} \subset V^{n-2}, \Psi^{n-2}_{j=0,...,2^{n-2}-1} \subset W^{n-2}, \Psi^{n-1}_{j=0,...,2^{n-1}-1} \subset W^{n-1} \,]^T$$

Untuk langkah ke n, persamaan $V^1 = V^0 \oplus W^0 \oplus W^1 \oplus W^2 \oplus ... \oplus W^{n-1}$, matrik *wavelet* M_n berbentuk (Porwik, 2004):

$$M_n= [\, \Phi^0_0 \subset V^0, \Psi^0_0 \subset W^0, \Psi^1_{j=0,1} \subset W^1 ... \Psi^{n-1}_{j=0,...2^{n-1}-1} \subset W^{n-1} \,]^T \quad (2.21)$$

Jika M adalah matrik *wavelet* dan I adalah matrik citra asli, maka matrik transformasi T didapatkan dengan persamaan (Ames, 2002):

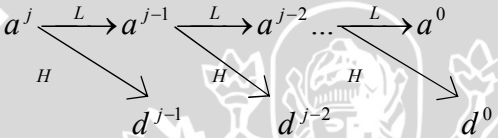
$$T = M^T . I . M \qquad (2.22)$$

Sedangkan untuk penyusunan/rekonstruksi citranya menggunakan persamaan(Ames, 2002):

$$I = (M^{-1})^T . I . M^{-1} \qquad (2.23)$$

2.3.3.1 Dekomposisi Koefisien *Haar Wavelet*

Stephane Mallat memperkenalkan cara mudah menghitung koefisien *Haar Wavelet* yang dikenal dengan algoritma piramida Mallat. Algoritma tersebut ditunjukkan dengan gambar 2.8. (Murni, 2003)



Gambar 2.8. Piramida Mallat untuk dekomposisi
Sumber : Murni, 2003

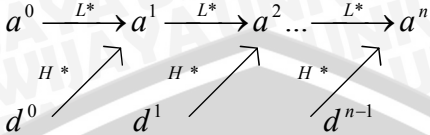
Dengan a^j adalah vektor awal dengan ukuran 2^j, dan koefisien *Haar Wavelet* dapat diperoleh dari aproksimasi a^0 detail-detail d^0, d^1 dan seterusnya. Dari Gambar 2.8 terlihat bahwa vektor awal dilewatkan filter *lowpass* (L) dan *highpass* (H) untuk mendapatkan nilai aproksimasi dan detail pada level selanjutnya. Matriks L dan H masing-masing adalah matriks *lowpass* (*averaging*) dan *highpass* (*differencing*) yang koefisiennya didefinisikan pada persamaan 2.16. Matrik LH ditunjukkan pada gambar 2.9.

$$L = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \qquad H = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Gambar 2.9. Matriks L dan H
Sumber : Murni, 2003

2.3.3.2 Rekonstruksi Koefisien *Haar Wavelet*

Proses kebalikan dari dekomposisi ini disebut proses rekonstruksi, dan algoritma piramida Mallat untuk rekonstruksi ditunjukan pada gambar 2.10.



Gambar 2.10. Piramida Mallat untuk rekonstruksi
Sumber : Murni, 2003

Matriks pengali L* dan H* memiliki hubungan dengan matriks L dan H pada proses dekomposisi yaitu LL* = HH* = I (L merupakan invers dari L*). Isi dari L* dan H* untuk basis Haar ditunjukkan pada gambar 2.11. (Murni, 2003)

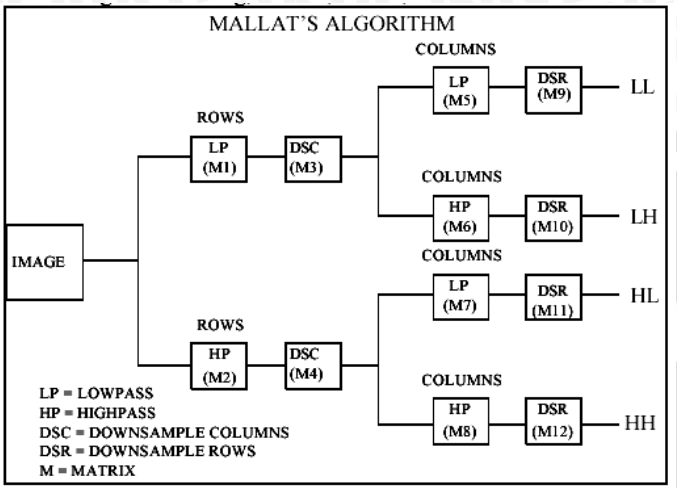
$$L^* = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix} \qquad H^* = \begin{bmatrix} 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \\ 0 & -1/\sqrt{2} \end{bmatrix}$$

Gambar 2.11. Matriks L* dan H*
Sumber : Murni, 2003

2.3.3.3 Transformasi *Haar Wavelet* Pada Citra

Suatu citra merupakan matriks dua dimensi. Sehingga dapat dilakukan transformasi terhadap baris-baris pada citra, dan dilanjutkan dengan transformasi terhadap kolom-kolom pada citra, seperti pada Gambar 2.12.

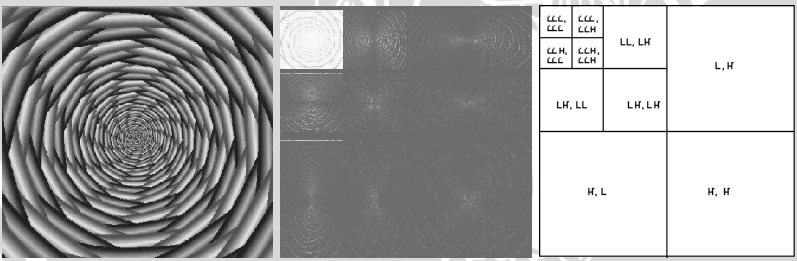
Sinyal pertama-tama dilewatkan pada rangkain filter *high-pass* dan *low-pass*, kemudian setengah dari masing-masing keluaran diambil sebagai *sample* melalui operasi *sub-sampling*. Proses ini disebut sebagai proses dekomposisi satu tingkat. Keluaran dari filter *low-pass* digunakan sebagai masukkan di proses dekomposisi tingkat berikutnya. Proses ini diulang sampai tingkat proses dekomposisi yang diinginkan. Gabungan dari keluaran-keluaran filter *high-pass* dan satu keluaran filter *low-pass* yang terakhir, disebut sebagai koefisien *wavelet*, yang berisi informasi sinyal hasil transformasi (Walidainy, 2004). Gambar 2.13 adalah contoh dekomposisi pada citra.



LL	LH
HL	HH

LL: hasil lowpass terhadap baris dan kolom
LH: hasil lowpass terhadap baris diteruskan dengan Highpass terhadap kolom
HL: hasil highpass terhadap baris diteruskan dengan lowpass terhadap kolom
HH: hasil highpass terhadap baris dan kolom

Gambar 2.12. Algoritma Mallat untuk dekomposisi citra
Sumber : Saha, 2004



(a) (b) (c)

Gambar 2.13 (a) citra asli, (b)dekomposisi level 3, (c)spektrum dan penataan dekomposisi
Sumber : Sanberg, 2000

2.3.3.4 Kuantisasi (*Quantization*)

Secara sederhana, kuantisasi merupakan proses pengurangan jumlah bit yang diperlukan untuk menyimpan koefisien transformasi dengan cara mengurangi ketelitian/ketepatan nilai keofisien tersebut. Kuantisasi dapat dilakukan pada tiap-tiap koefisien, yang disebut *Skalar Quantization* (SQ). Kuantisasi juga dapat dilakukan pada sekelompok koefisien bersama-sama yang dikenal dengan *Vector Quantization* (VQ). (Saha, 2004)

Pada kompresi *wavelet*, kuantisasi dilakukan dengan cara menetapkan nilai *threshold* non-negatif yang dikenal dengan ε. Kemudian setiap nilai koefisien matrik *Haar wavelet transform* yang bernilai kurang dari ε akan diubah menjadi nol. Pengubahan nilai tersebut bertujuan agar terbentuk *sparse matrix*. Jika nilei ε= 0 maka tidak ada nilai yang diubah dan tidak ada informasi yang hilang. Kompresi *lossy* terjadi jika nilai ε lebih dari nol. Karena beberapa nilai elemen diubah menjadi nol, beberapa data asli akan hilang. Dalam kasus ini kompresi *lossless* dapat membalik perhitungan dan mendapatkan citra asli. Dengan kompresi *lossy* hanya bisa didapatkan pendekatan dari citra asli. (Ames, 2002)

Dua hal yang pokok harus diketahui/dipahami adalah bagaimana memilih ambang (*threshold*) dan bagaimana melakukan *thresholding*. Banyak metode diperkenalkan untuk menset batas ambang. Cara yang paling banyak memakan waktu adalah menset batas ambang dengan dasar kasus-per-kasus. Dua aturan umumnya digunakan (seperti yang telah dijelaskan di atas) untuk *thresholding* koefisien *wavelet* adalah *thresholding* lunak dan keras. Bila ε menyatakan ambang/*threshold* maka sinyal *threshold* keras adalah:

$$T_{hard}(x) = \begin{cases} x & |x| > \varepsilon \\ 0 & |x| \leq \varepsilon \end{cases} \qquad (2.24)$$

Pada *thresholding* jenis ini seluruh koefisien-koefisien *wavelet* dengan nilai di bawah *threshold* yang ditentukan dibuang (dinolkan). Hal ini dihubungkan dengan pemodelan sederhana, secara murni koefisien-koefisien *wavelet* di bawah *threshold* digolongkan sebagai derau, sedangkan koefisien-koefisien di atas *threshold* digolongkan sebagai sinyal.

Sedangkan sinyal *threshold* lunak adalah:

$$T_{soft}(x) = \begin{cases} sign(x)(|x| - thresh) & |x| \geq \varepsilon \\ 0 & |x| < \varepsilon \end{cases} \quad (2.25)$$

Pada *thresholding* jenis ini seluruh koefisien-koefisien *wavelet* dengan nilai di bawah *threshold* yang ditentukan dibuang. Selain itu seluruh koefisien-koefisien *wavelet* di atas *threshold* yang ditentukan akan dikurangi dengan nilai *threshold*. Dengan demikian metode ini meredam/ mengurangi jangkauan *koefisien wavelet* dan meratakan sinyal. Dengan kata lain energy sinyal dimodifikasi.

Prosedur keras menciptakan ketidakkontinuan pada x= ± ε; prosedur lunak tidak menciptakan kondisi seperti itu Dalam metode ini, kuncinya adalah pilihan numeris dari *threshold* ε. Pemilihan ini bersifat kritis, jika *threshold* terlalu kecil atau terlalu besar. Dengan demikian dapat terjadi distorsi yang cukup besar. (Walidainy, 2004)

2.4 Run Length Encoding (RLE)

RLE adalah teknik yang digunakan untuk mengurangi ukuran karakter string yang berulang. Jika jenis data *d* muncul sebanyak *n* kali, maka kejadian *n* kali diganti dengan sepasang data tunggal *nd*. Kejadian deret *n* pada jenis data *d* disebut *run length* of *n*. RLE dapat digunakan pada kompresi text maupun citra.

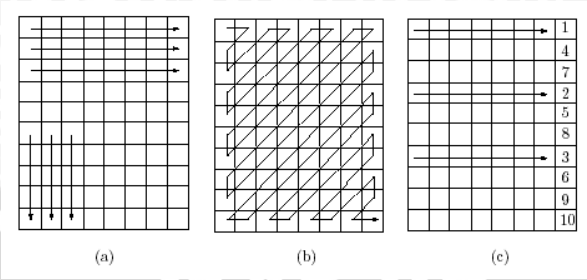
Contoh: Citra *grayscale* bitmap yang dimulai dengan

12, 12, 12, 12, 12, 12, 12, 12, 12, 35, 76, 112, 67, 87, 87, 87, 5, 5, 5, 5, 5, 5, 1, . . .

Kemudian dikompresi menjadi

9 ,12,35,76,112,67, 3 ,87, 6 ,5,1,. . . , dimana angka yang digarisbawahi menunjukkan jumlah.

Output RLE kadang-kadang lebih besar dari citra aslinya untuk citra yang kompleks. Misalnya pada gambar yang memiliki banyak garis vertikal. Ketika di-*scan* secara horizontal, maka akan dihasilkan run yang sangat pendek, hal ini mengakibatkan kompresi yang buruk atau hasilnya lebih besar. Pada prakteknya, kompresor citra RLE dapat men-*scan* citra bitmap secara baris, kolom, maupun zigzag gambar 2.14. (Salomon, 2004)



Gambar 2.14. Berbagai macam cara RLE men-*scan* citra
Sumber : Salomon, 2004

2.5 Rasio Kompresi (Nisbah Pemampatan)

Rasio kompresi merupakan perbandingan ukuran file citra sebelum dikompresi dan ukuran file hasil dekompres. Persentase kompresi dapat dinyatakan dengan Persamaan 2.26 (Khalid, 2000)

$$R = (1 - (K / A)) * 100 \ \% \qquad (2.26)$$

Dimana A = Ukuran *file* citra asli,
K = Ukuran *file* citra terkompresi,
R = Persentase kompresi citra.

2.6 Mean Square Error (MSE)

Dalam statistika, MSE merupakan salah satu cara untuk mengukur jumlah perbedaan antara nilai perkiraan dengan nilai yang sebenarnya. MSE mengukur rata-rata wilayah kesalahan (*error*). MSE (*Mean Square Error*) merupakan sigma dari jumlah kesalahan (*error*) antara citra hasil kompresi dan citra asli. Perhitungan nilai MSE dari citra digital berukuran N x M, dilakukan sesuai dengan rumus:

$$\text{MSE}=\frac{1}{MN}\sum_{Y=1}^{M}\sum_{X=1}^{N}\left[I(x,y)-I'(x,y)\right]^2 \qquad (2.27)$$

Dengan:I(x,y) adalah nilai pixel di citra asli,
I'(x,y) adalah nilai pixel pada citra hasil kompresi,
M,N adalah dimensi citra.
Nilai MSE yang rendah akan lebih baik. (Linda S., 2005)

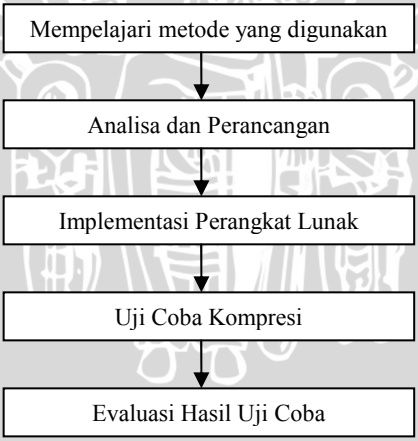
UNIVERSITAS BRAWIJAYA

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan, dan langkah-langkah yang dilakukan dalam mengkompresi dan mendekompress citra menggunakan *Modified Haar Wavelet Transform* dan RLE. Langkah-langkah yang dilakukan dalam penelitian ini meliputi :

1. Mempelajari metode yang digunakan dari berbagai sumber seperti yang telah dijelaskan pada bab 2.
2. Menganalisa dan merancang perangkat lunak dengan metode yang akan digunakan.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Uji coba kompresi dan dekompresi citra astronomi menggunakan perangkat lunak yang telah dibuat.
5. Evaluasi hasil uji coba.

Langkah-langkah penelitian ini dapat digambarkan seperti pada Gambar 3.1.



Gambar 3.1. Langkah-Langkah Penelitian

3.1 Analisis Perangkat Lunak

3.1.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak pengkompres citra ini dikembangkan untuk melakukan kompresi dan dekompresi pada citra astronomi menggunakan *Modified Haar Wavelet Transform* dan RLE. *Input*-an yang dibutuhkan pada proses kompresi berupa citra astronomi digital. Citra ini akan melewati serangkaian proses kompresi hingga dihasilkan *file* kompresi. Sedangkan *input*-an yang dibutuhkan pada proses dekompresi berupa *file* kompresi. *File* ini akan melewati serangkaian proses dekompresi hingga dihasilkan citra rekonstruksi.

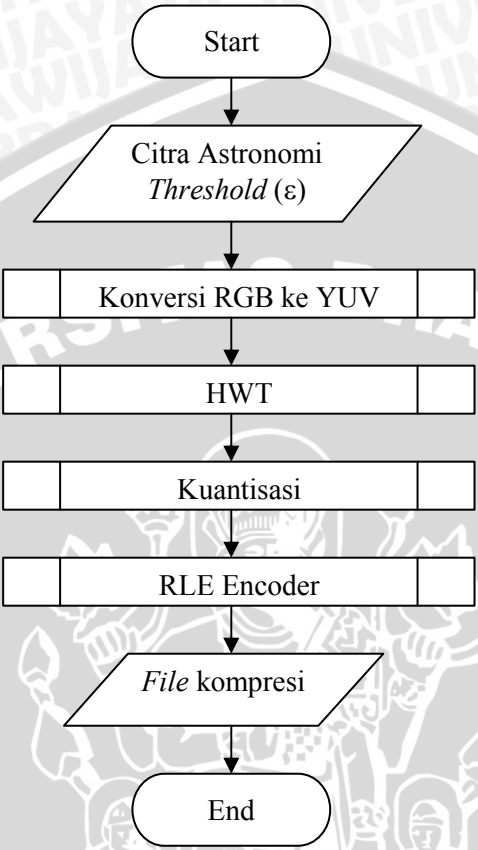
Metode yang digunakan adalah *wavelet transform* yang proses kuantisasinya telah dimodifikasi dan RLE. Adapun proses-proses yang dilakukan untuk mengkompresi adalah sebagai berikut:

1. *User* memasukkan citra warna 24 bit, metode *threshold* dan 4 nilai *threshold*.
2. Dilakukan konversi model warna citra dari RGB ke YUV.
3. Masing-masing komponen warna YUV ditransformasikan menggunakan *HWT*.
4. Masing-masing komponen warna dikuantisasi menggunakan beberapa nilai *threshold* secara acak .
5. Masing-masing komponen warna dikompres menggunakan RLE yang akan menghasilkan *file* kompresi.

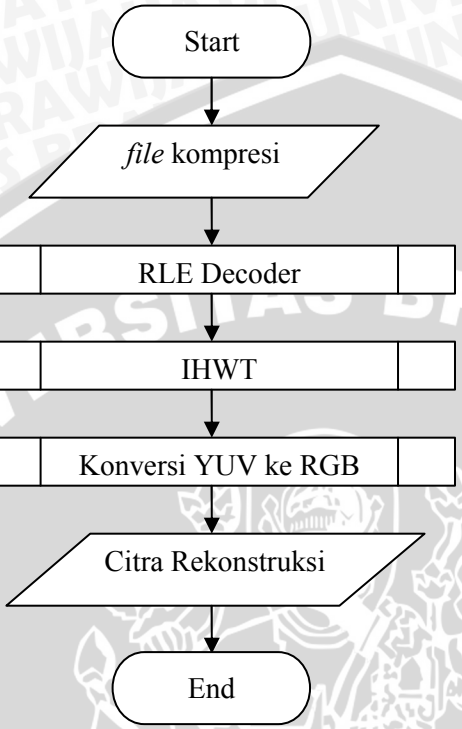
Diagram langkah-langkah proses kompresi citra ditunjukkan pada Gambar 3.2. Sedangkan proses-proses yang dilakukan untuk mendekompres adalah sebagai berikut:

1. *User* memasukkan *file* kompresi.
2. Dilakukan proses *invers* RLE untuk masing-masing komponen warna.
3. Masing-masing komponen warna ditransformasikan menggunakan *IHWT*.
4. Masing-masing komponen warna YUV dikonversikan ke model warna RGB yang kemudian ditampilkan sebagai citra rekonstruksi.

Diagram langkah-langkah proses dekompres citra ditunjukkan pada Gambar 3.3.



Gambar 3.2. *Flowchart* proses kompresi



Gambar 3.3. *Flowchart* proses dekompres

3.1.2 Batasan Perangkat Lunak

Batasan perangkat lunak ini adalah :

1. Citra yang digunakan sebagai inputan hanya citra dengan ekstensi bitmap (.bmp).
2. Citra yang digunakan adalah citra berwarna 24 bit.
3. Dimensi citra yang digunakan memiliki ukuran genap.

3.2 Perancangan Perangkat Lunak

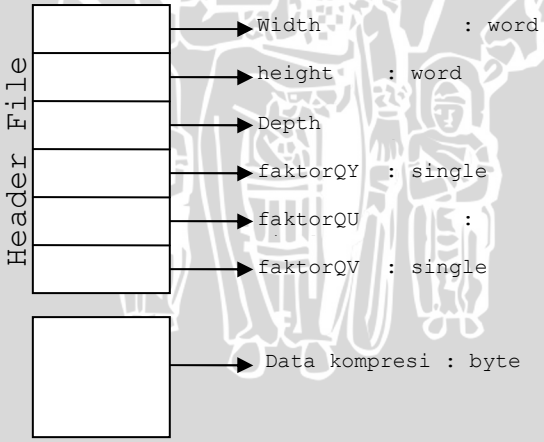
3.2.1 Perancangan Struktur *File* Kompresi

File (berkas) kompresi terdiri dari dua bagian yaitu *Header file* dan data kompresi. *Header file* berisi informasi mengenai data kompresi. Informasi tersebut berguna untuk proses dekompresi. Sehingga citra rekonstruksi sesuai dengan citra aslinya. Bagian *File* dijelaskan pada Tabel 3.1.

Tabel 3.1. Struktur *File* kompresi

Bagian	Keterangan
width	Berisi informasi mengenai lebar citra
height	Berisi informasi mengenai tinggi citra
Depth	Berisi informasi mengenai kedalaman transformasi
factorQY	Berisi informasi mengenai faktor kuantisasi komponen Y yang berguna untuk mengkonversi nilai *float* menjadi *byte* dan sebaliknya.
factorQU	Berisi informasi mengenai faktor kuantisasi komponen U yang berguna untuk mengkonversi nilai *float* menjadi *byte* dan sebaliknya.
factorQV	Berisi informasi mengenai faktor kuantisasi komponen V yang berguna untuk mengkonversi nilai *float* menjadi *byte* dan sebaliknya.
Data Kompresi	Berisi data kompresi ketiga matrik komponen warna YUV.

Struktur *file* kompresi dapat ditunjukkan seperti pada Gambar 3.4. *File* kompresi akan disimpan dengan ekstensi .HWT dan hanya bisa dibuka/didekompresi menggunakan perangkat lunak ini.



Gambar 3.4. Struktur *file* kompresi

3.2.2 Perancangan Proses Kompresi

3.2.2.1 Konversi Model Warna RGB ke YUV

Pada tahap ini, model warna citra RGB (*Red-Green-Blue*) diubah menjadi model warna YUV. Pengubahan ini bertujuan untuk memanfaatkan kemampuan mata manusia yang tidak peka terhadap perubahan *chrominance*. Sehingga kompresi *lossy* dapat dilakukan dengan mengurangi komponen dari YUV. Pengubahan ini menggunakan Persamaan 2.1, 2.2, dan 2.3.

Setelah dikonversikan, masing-masing komponen YUV akan ditranformasikan secara terpisah. Ada 3 *layer* transformasi, yaitu tranformasi komponen Y, U, dan V. Meskipun ditranformasikan secara terpisah, tetapi proses yang akan diterapkan pada ketiga komponen tersebut sama seperti dijelaskan pada Gambar 3.2.

Algoritma untuk mengkonversi RGB ke YUV ditunjukkan pada Gambar 3.5.

```
For y←0 to img.hight-1
       For x ←0 to img.width-1

  Clr ← Img.Canvas.Pixels[0,0]
  R ← GetRValue (Clr)
  G ← GetGValue (Clr)
  B ← GetBValue (Clr)

  Y ←(0.256788 * R + 0.504129 * G + 0.097906 * B)+ 16
  U ← (-0.148223 * R - 0.290993 * G + 0.439216 * B)+ 128
  V ← (0.439216 * R - 0.367788 * G - 0.071427 * B)+ 128
       endfor
endfor
```

Gambar 3.5. Algoritma konversi RGB ke YUV

Berdasarkan Gambar 3.5, langkah-langkah yang dilakukan pada tahap ini adalah

1. Memasukkan inputan berupa citra.
2. Melakukan *looping* (perulangan) sepanjang lebar dan tinggi citra.
3. Mengambil nilai RGB citra pada pixel tertentu.
4. Mengkonversi nilai RGB menjadi YUV.

3.2.2.2 *Haar Wavelet Transform* (HWT)

Pada tahap ini, sistem akan mentranformasikan citra ke daerah frekuensi dengan inputan matrik (array 2 dimensi) komponen YUV. Matrik ini akan ditransformasikan menggunakan filter *lowpass* dan *highpass*. Proses transformasi akan dilakukan hingga kedalaman level ke n, dimana $N= 2^n$ dengan N adalah ukuran citra. Pada tiap level akan dilakukan tranformasi baris terlebih dahulu melalui filter *lowpass* dan *highpass*. Kemudian dilakukan tranformasi terhadap kolom, seperti yang ditunjukkan pada Gambar 2.12. Pada akhir transformasi tiap level akan dilakukan proses kuantisasi.

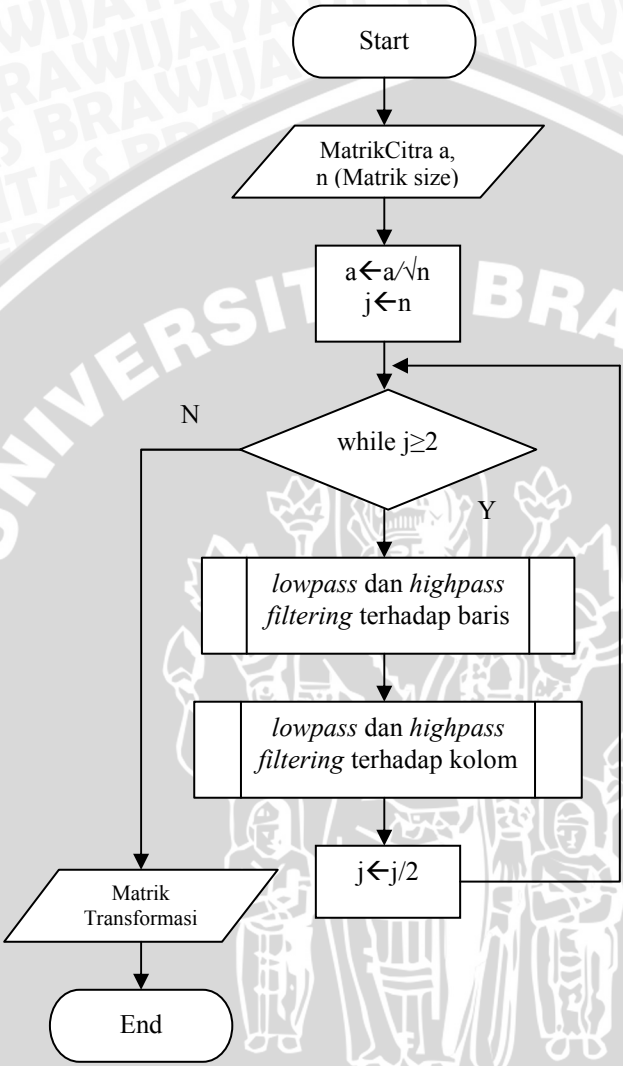
Langkah-langkah yang dilakukan pada tahap ini adalah:

1. Memasukkan matrik komponen YUV
2. Membagi matrik dengan akar ukuran array
3. Mendefinisikan ukuran array
4. Melakukan *conditional while* ukuran array ≥ 2
5. Melakukan *lowpass* dan *highpass filtering* terhadap baris
6. Melakukan *lowpass* dan *highpass filtering* terhadap kolom
7. Membagi dua ukuran array untuk tranformasi pada level selanjutnya (kembali ke point 4).

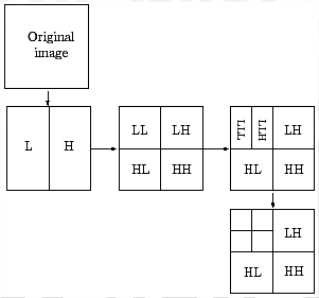
Lowpass dan *highpass filtering* merupakan tahap melewatkan koefisien baris atau kolom melalui Persamaan *filtering* 2.17 dan 2.18. *Flowchart* proses *Haar Wavelet Transform* ditunjukkan pada Gambar 3.6.

Setelah proses tranformasi selesai maka akan terbentuk matrik transformasi domain frekuensi dengan energi yang terkumpul di pojok kiri atas (contoh pada gambar 3.7 (a)).

Struktur wavelet dan *pseudocode* proses HWT ditunjukkan pada Gambar 3.7.



Gambar 3.6. *Flowchart* proses HWT



(a)

```
procedure NStdCalc(a:array of real real, n:int)
          // n is the array size (a power of 2)
      a←a/√n // divide entire array
      j←n
     while j≥ 2 do
       for r←1 to j do HWTstep(row r of a, j)
       endfor
                            for c←1 to j do
                            HWTstep(col c of a, j)
                            endfor
     j←j/2
    endwhile
end
```

```
procedure HWTstep(a:array of real real, j:int)
     for i←1 to j/2 do
       b[i]←(a[2i-1]+a[2i])/√2  //lowpass
       b[j/2+i]←(a[2i-1]-a[2i])/√2 //highpass
     endfor
     a←b // move entire array
end
```

(b)

Gambar 3.7. (a) Struktur dan
(b) *pseudocode Haar wavelet transform*

Berdasarkan pada Gambar 3.7 (b), *procedure* HWTstep merupakan tahap *lowpass* dan *highpass filtering.*

3.2.2.3 Kuantisasi (*Quantization*)

Pada tahap ini akan dilakukan pengurangan informasi berdasarkan nilai *threshold* (ε) yang diinputkan oleh *user*. Sistem akan melakukan proses kuantisasi dengan cara men-*scan* bagian *detail* matrik tranformasi (matrik komponen warna yang telah ditransformasikan) per baris dari kiri ke kanan sambil membandingkan nilai absolut matrik pada koordinat tersebut dengan nilai *threshold*. Jika nilai matrik lebih kecil dari *threshold*, maka nilai matrik tersebut diubah menjadi nol. Jika nilai matrik lebih besar dari nilai *threshold*, maka nilai matrik dibiarkan. Dengan adanya pengubahan nilai koefisien matrik menjadi nol, maka akan terbentuk *sparse matrik* (matrik renggang) yang siap untuk dikompres pada proses selanjutnya.

Pada penelitian ini akan digunakan metode *hard thresholding (*keras*)* dan *soft thresholding* (lunak). Sehingga dapat dibandingkan hasilnya. Persamaan yang digunakan adalah Persamaan 2.24 dan 2.25.

Pada penelitian ini, kuantisasi dilakukan dengan cara menerapkan beberapa nilai *threshold* yang berbeda pada koefisien *wavelet* seperti yang dilakukan pada penelitian Raviraj (2007). Beberapa nilai *threshold* yang digunakan tersebut akan dipilih secara *random* untuk mengkuantisasi suatu koefisien *wavelet*.

Langkah-langkah yang dilakukan pada tahap ini adalah:

1. Memasukkan inputan berupa matrik transformasi.
2. Melakukan *looping* sepanjang dimensi matrik.
3. Memilih ε dari beberapa nilai inputan menggunakan fungsi random.
4. Memilih metode *thresholding* yang akan digunakan (lunak atau keras).
5. Membandingkan nilai absolut koefisien matrik dengan ε.
6. Merubah koefisien matrik menjadi 0 jika nilai absolutnya ≤ ε.

Algoritma kuantisasi untuk salah satu komponen warna ditunjukkan pada Gambar 3.8.

```
For y←0 to hight-1
       For x ←0 to width-1
       If daerah detail then begin
       Pilih random nilai threshold(25/10/5/1)
           If hard threshold then begin
                          If |MatrikTransform[x,y]| ≤
threshold then
                          Begin
                          MatrikTransform[x,y]← 0
                          endif
           If soft threshold then begin
                          If |MatrikTransform[x,y]| <
threshold then
                          Begin
                          MatrikTransform[x,y]← 0
                          endif
                          else
  MatrikTransform[x,y]←| MatrikTransform[x,y]| -
threshold
                          endif
```

Gambar 3.8. Algoritma Kuantisasi

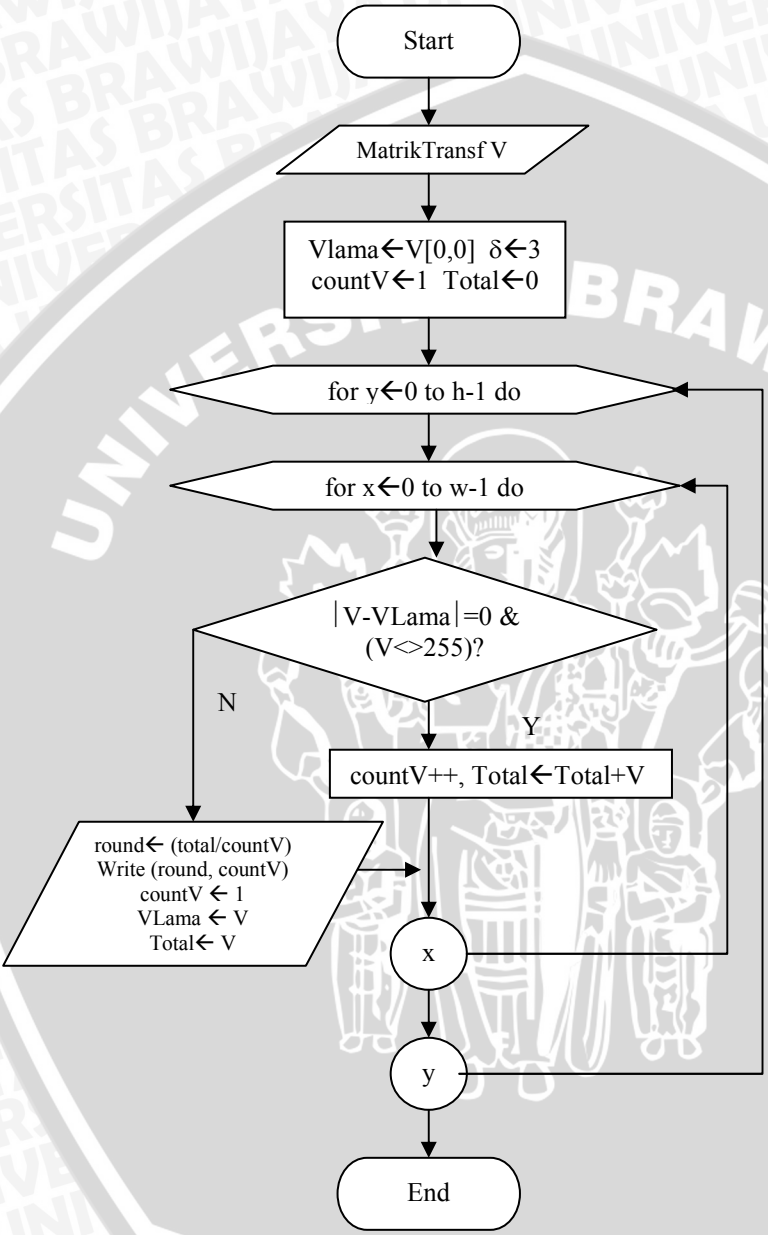
3.2.2.4 RLE Encoder

Pada tahap ini, matrik tranformasi yang telah dikuantisasi akan dikompresi menggunakan metode RLE. Sebelum menuliskan data kompresi, sistem akan menuliskan informasi mengenai ukuran matriks terlebih dahulu pada awal *file* kompresi. Penulisan tersebut bertujuan agar sistem dapat menyusun kembali matrik dengan ukuran yang sama pada saat didekompres nanti. Sistem akan men-*scan* matrik tranformasi per baris dari kiri ke kanan untuk kemudian dikodekan menggunakan metode RLE. Setelah semua koefisien matrik tranformasi dikodekan, maka terbentuk *file* kompresi hasil proses pengkodean.

Langkah-langkah yang dilakukan pada tahap ini adalah:

1. Memasukkan inputan berupa matrik terkuantisasi.
2. Mendefinisikan nilai lama, dan jumlah nilai.
3. Melakukan *looping* sepanjang dimensi matrik.
4. Membandingkan nilai sekarang dengan nilai lama, jika sama maka jumlah nilai bertambah. Jika tidak sama maka sistem akan menuliskan nilai dan jumlah nilai pada *file* kompresi (kembali ke point 3).

Flowchart proses RLE Encoder ditunjukkan pada Gambar 3.9.



Gambar 3.9. *Flowchart* proses RLE Encoder

Algoritma RLE *encoder* untuk salah satu komponen warna ditunjukkan pada Gambar 3.10.

```
Total←0 {inisialisasi total}
for y←0 to h-1 do
begin
  for x←0 to w-1 do
  if (x>0)or(y>0) then
  begin

     if (abs(V-VLama)=0 and (V<>255))
     then
               begin
                     Inc(countV)
               Total←Total+V
      endif
      else
      begin

rounded←round(total/countV)
                     Write (fl,
rounded)
                  Write (fl, countV)
                        countV ← 1
                        VLama ← V
                        Total← V
      endif
```

Gambar 3.10. Algoritma RLE Encoder

3.2.3 Perancangan Proses Dekompresi

3.2.3.1 RLE Decoder

Pada tahap ini sistem akan melakukan rekonstruksi/ penyusunan ulang matrik 2 dimensi dari *file* kompresi. Pertama sistem akan membaca berkas kompresi untuk mengetahui informasi ukuran matrik dan kedalaman transformasi. Dengan diketahuinya informasi ini, sistem akan membentuk matrik kosong 2 dimensi dengan ukuran sesuai informasi. Setelah terbentuk matrik kosong 2 dimensi, sistem akan membaca data kompresi untuk merekonstruksi nilai matrik tersebut. Pengisian nilai matrik dilakukan per baris dari kiri ke kanan.

Langkah-langkah yang dilakukan pada tahap ini adalah:

1. Membaca ukuran matrik dari *file* kompresi.
2. Membentuk matrik kosong.
3. Membaca nilai matrik dan jumlahnya dari *file* kompresi.
4. Mengisikan nilai tersebut pada matrik sesuai dengan jumlah dan posisinya.

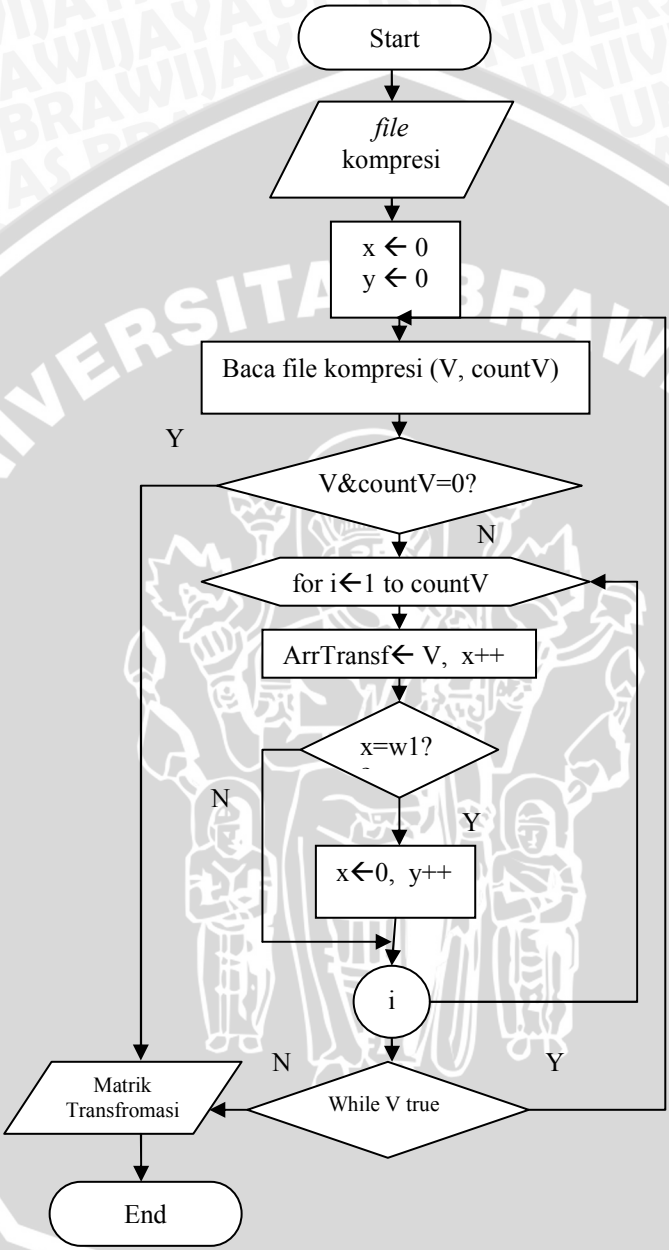
Algoritma RLE *decoder* untuk salah satu komponen warna dapat ditunjukkan pada Gambar 3.11.

```
 x ← 0
  y ← 0
  while true do
  begin
       Read (fl, V)    //membaca nilai V dari file
       Read (fl, countV)//membaca jumlah V

              if (V=0) and (countV=0) then break
              for i←1 to countV do
               begin
                     ArrTransf[x,y]← V  //mengisi
nilai V
                              Inc (x)
                              if x=w1 then
                     begin
                      x ← 0
                        Inc (y)
                              endif
               endfor
  endwhile
```

Gambar 3.11. Algoritma RLE Decoder

Flowchart proses RLE *decoder* ditunjukkan pada Gambar 3.12.



Gambar 3.12. *Flowchart* proses *RLE Decoder*

3.2.3.2 Invers Haar Wavelet Transform (IHWT)

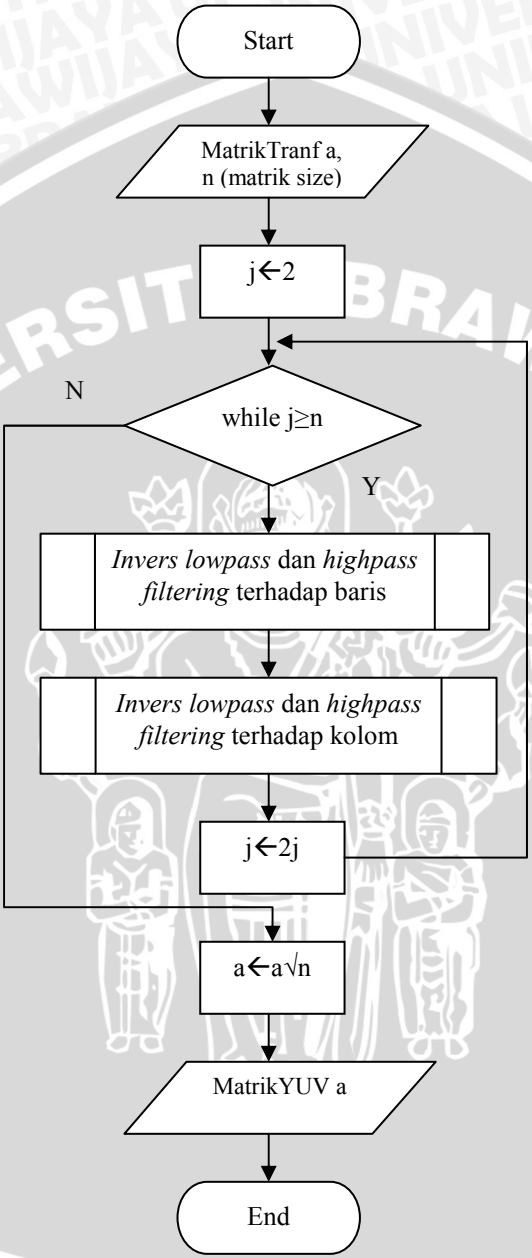
Pada proses ini dilakukan tranformasi matrik dari daerah frekuensi ke daerah ruang, sehingga diperoleh matrik komponen warna YUV. Proses ini merupakan kebalikan (*invers*) dari proses HWT. Proses ini dilakukan dengan cara melewatkan koefisien matrik tranformasi melalui *invers filter* seperti pada Persamaan 2.19 dan 2.20. Transformasi *invers* ini dilakukan hingga level ke n, dimana $N= 2^n$ dengan N adalah ukuran matrik transformasi.

Langkah-langkah yang dilakukan pada tahap ini adalah:

1. Memasukkan matrik tranformasi.
2. Mendefinisikan ukuran array pada level terdalam.
3. Melakukan *conditional while* ukuran array ≤ N
4. Melakukan *invers lowpass* dan *highpass filtering* terhadap baris
5. Melakukan *invers lowpass* dan *highpass filtering* terhadap kolom
6. Mengalikan dua ukuran array untuk tranformasi pada level selanjutnya (kembali ke point 3).

Invers lowpass dan *highpass filtering* merupakan tahap melewatkan koefisien baris atau kolom melalui Persamaan *filtering* 2.19 dan 2.20.

Flowchart proses RLE decoder ditunjukkan pada Gambar 3.13.



Gambar 3.13. *Flowchart* IHWT

Algoritma proses IHWT dapat ditunjukkan dalam bentuk *pseudocode* pada Gambar 3.14.

```
procedure NStdReconst(a:array of real, n:int)
     j←2 //size array pada level terdalam
    while j≤n do
                              for c←j to 1 do //
loop backwards
                       IHWTstep(col c of a, j)
                              endfor
          for r←1 to j do
             IHWTstep(row r of a, j)
           endfor
     j←2j
     endwhile
a←a√n // multiply entire array
```

```
procedure IHWTstep(a:array of real real, j:int)
     for i←1 to j/2 do
       b[2i-1]←(a[i]+a[j/2+i])/√2
       b[2i]←(a[i]-a[j/2+i])/√2
     endfor
a←b          // move entire array
end
```

Gambar 3.14. Algoritma IHWT

Berdasarkan pada Gambar 3.14. *procedure* IHWTstep merupakan tahap *invers lowpass* dan *highpass filtering.*

3.2.3.3 Konversi Model warna YUV ke RGB

Setelah melalui proses IHWT, maka ketiga matrik komponen YUV akan diubah kembali menjadi model warna RGB untuk ditampilkan menjadi citra rekonstruksi. Pengubahan ini menggunakan Persamaan 2.4, 2.5, dan 2.6. Gambar 3.15 adalah algoritma untuk mengubah model warna YUV menjadi RGB.

```
For y← 0 to matriktransform.hight
   For x ← 0 to matriktransform.width

r ← 1.164383 * Y * 0.9+ 1.596027 * (V – 128)
g ← 1.164383 * Y * 0.9 - 0.391762 * (U – 128) –
     0.812968  * (V – 128)
b ← 1.164383 * Y * 0.9 + 2.017232 * (U – 128)

Img.Canvas.Pixels[x,y] ← RGB(r,g,b)

      endfor
endfor
```

Gambar 3.15. Algoritma konversi YUV ke RGB.

Berdasarkan Gambar 3.15, langkah-langkah yang dilakukan pada tahap ini adalah

1. Memasukkan inputan berupa matrik komponen warna YUV.
2. Melakukan *looping* (perulangan) sepanjang lebar dan tinggi matrik.
3. Mengambil nilai YUV pada koordinat tertentu.
4. Mengkonversi nilai YUV menjadi RGB.
5. Memasukkan nilai RGB pada citra rekonstruksi.

3.3 Perancangan Pengujian

Pengujian yang akan dilakukan pada perangkat lunak ini adalah pengujian rasio berkas kompresi dibandingkan berkas citra asli dan tingkat kesalahan pada citra hasil penyusunan kembali (rekonstruksi) dibandingkan citra astronomi asli. Pengujian rasio dan tingkat kesalahan kompresi akan dilakukan dengan beberapa nilai *threshold* (ε) yang berbeda, yaitu pada *e*=25, 10, 5, dan 1. Selain itu juga digunakan metode *thresholding* yang berbeda.

Oleh karena nilai *threshold* yang digunakan adalah *random* dari beberapa nilai *threshold* yang ditentukan. Maka pada tiap-tiap citra uji dilakukan 3 (tiga) kali percobaan kompresi untuk diambil nilai rata-ratanya, baik pada pengujian rasio maupun pada pengujian tingkat kesalahan.

3.3.1 Citra Uji

Citra yang diuji adalah citra astronomi yang memiliki karakteristik sebagai berikut:

1. 20 citra dengan ekstensi bitmap (.bmp).
2. Citra warna 24 bit.
3. Citra berukuran orthogonal 256x 256.
4. Citra astronomi yang diujikan didapatkan dari *http://photojournal.jpl.nasa.gov/* dan *http://www.astronomy-images.com/*

3.3.2 Lingkungan Pengujian

Pengujian ini membutuhkan bantuan dari perangkat lunak yang lain. Perangkat lunak tersebut adalah ACDSee 7.0 yang digunakan untuk menyiapkan citra uji (untuk mengubah citra menjadi warna 24 bit dan merubah ukuran citra menjadi orthogonal NxN) dan MS Excel yang digunakan untuk mengolah data.

3.3.3 Pengujian Rasio Kompresi (Nisbah pemampatan)

Pengujian terhadap rasio merupakan pengujian terhadap perbandingan ukuran berkas kompresi dengan ukuran berkas citra asli. Pengujian ini menggunakan Persamaan 2.26.

Inputan dari pengujian ini berupa ukuran berkas kompresi dan berkas citra asli. Output dari pengujian ini berupa persentase rasio pemampatan. Makin besar pemampatannya, maka makin besar persentasenya dan sebaliknya. Tabel yang digunakan untuk metode *hard thresholding* ditunjukkan seperti pada Tabel 3.2.

Tabel 3.2. Tabel rasio kompresi (*Hard thresholding*)

No.	Nama citra	Rasio Kompresi				
		Kompresi			Rata-rata	Persentase (%)
		1	2	3		

Tabel yang digunakan untuk metode *soft thresholding* ditunjukkan seperti pada Tabel 3.3.

Tabel 3.3. Tabel rasio kompresi (*Soft thresholding*)

No.	Nama citra	Rasio Kompresi				
		Kompresi			Rata-rata	Persentase (%)
		1	2	3		

3.3.4 Pengujian Tingkat Kesalahan (*error rate*)

Pengujian terhadap tingkat kesalahan merupakan pengujian terhadap ketepatan metode dekompresi dalam menyusun kembali citra asli. Pengujian ini menggunakan Persamaan MSE 2.27.

Inputan dari pengujian ini berupa nilai pixel dari citra asli dan citra hasil penyusunan kembali. Output dari pengujian ini berupa nilai kesalahan. Makin besar kesalahannya, maka makin besar nilai MSE yang dihasilkan dan sebaliknya. Tabel yang digunakan untuk metode *hard thresholding* ditunjukkan pada Tabel 3.4.

Tabel 3.4. Tabel tingkat kesalahan (*Hard thresholding*)

No.	Nama citra	Tingkat kesalahan (MSE)			
		Kompresi			Rata-rata
		1	2	3	

Tabel yang digunakan untuk metode *soft thresholding* ditunjukkan seperti pada Tabel 3.5.

Tabel 3.5. Tabel tingkat kesalahan (*Soft thresholding*)

No.	Nama citra	Tingkat kesalahan (MSE)			
		Kompresi			Rata-rata
		1	2	3	

3.4 Contoh Perhitungan

3.4.1 Proses Dekomposisi Citra

Dimisalkan citra berukuran 8x8 dengan nilai pada Gambar 3.16.

47	73	47	30	45	37	27	8
67	69	46	42	63	26	24	14
72	40	59	36	62	11	15	57
67	35	72	52	51	35	30	83
39	37	65	61	41	20	24	70
51	70	54	68	37	36	111	88
64	50	44	50	44	48	119	148
86	46	50	37	34	86	99	145

Gambar 3.16. citra 8x8

Untuk memudahkan pemahaman dan perhitungan, maka digunakan metode pendekatan perkalian matrik. Langkah-langkah dekomposisi wavelet Haar terhadap potongan citra tersebut adalah:

1. Menentukan filter dekomposisi LH, ditunjukkan pada Gambar 3.17.

$1/\sqrt{2}$	$1/\sqrt{2}$	0	0	0	0	0	0
$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0	0	0	0	0
0	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0	0	0
0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0	0	0
0	0	0	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0
0	0	0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0
0	0	0	0	0	0	$1/\sqrt{2}$	$1/\sqrt{2}$
0	0	0	0	0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$

Gambar 3.17. filter dekomposisi LH

2. Mengalikan tiap kolom dengan matriks dekomposisi LH. Contoh untuk kolom pertama ditunjukkan pada Gambar 3.18.

$1/\sqrt{2}$	$1/\sqrt{2}$	0	0	0	0	0	0		47		81
$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0	0	0	0	0		67		-14
0	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0	0	0		72		98
0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0	0	0	*	67	=	3.5
0	0	0	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0		39		64
0	0	0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0		51		-8.5
0	0	0	0	0	0	$1/\sqrt{2}$	$1/\sqrt{2}$		64		106
0	0	0	0	0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$		86		-15

Gambar 3.18. Perkalian kolom 1 dengan filter LH

Hasil dekomposisi perkolom ini akan menghasilkan matriks D2 ditunjukkan pada Gambar 3.19.

81	93	66	60	76	44.5	36	16
-14	2	0.5	-6	-9	5.5	1.5	-3
98	37.5	65.5	44	56.5	23	22.5	70
3.5	2.5	-6.5	-8	5.5	-12	-7.5	-13
64	53.5	59.5	64.5	39	28	67.5	79
-8.5	-16.5	5.5	-3.5	2	-8	-43.5	-9
106	48	47	43.5	39	67	109	146.5
-15	2	-3	6.5	5	-19	10	1.5

Gambar 3.19. Matrik D2

Keterangan:

- warna biru adalah hasil aproksimasi
- warna merah adalah hasil detail

3. Mengatur hasil pada point 3 agar bagian aproksimasi berkumpul di bagian atas dan bagian detail mengumpul di bagian bawah. Sehingga terbentuk matrik D3 yang ditunjukkan pada Gambar 3.20.

81	93	66	60	76	44.5	36	16
98	37.5	65.5	44	56.5	23	22.5	70
64	53.5	59.5	64.5	39	28	67.5	79
106	48	47	43.5	39	67	109	146.5
-14	2	0.5	-6	-9	5.5	1.5	-3
3.5	2.5	-6.5	-8	5.5	-12	-7.5	-13
-8.5	-16.5	5.5	-3.5	2	-8	-43.5	-9
-15	2	-3	6.5	5	-19	10	1.5

Gambar 3.20. Matrik D3

4. Mengalikan tiap baris matrik D3 dengan matriks dekomposisi LH. Contoh untuk baris pertama ditunjukkan pada Gambar 3.21.

$1/\sqrt{2}$	$1/\sqrt{2}$	0	0	0	0	0	0		81	123
$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0	0	0	0	0		93	-8.5
0	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0	0	0		65.7	89
0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0	0	0	*	60	= 4
0	0	0	0	$1/\sqrt{2}$	$1/\sqrt{2}$	0	0		76	84
0	0	0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$	0	0		44.5	-22
0	0	0	0	0	0	$1/\sqrt{2}$	$1/\sqrt{2}$		36	37
0	0	0	0	0	0	$1/\sqrt{2}$	$-1/\sqrt{2}$		16	14

Gambar 3.21. Perkalian baris 1 dengan filter LH

Hasil dekomposisi perbaris ini akan menghasilkan matriks D4 yang ditunjukkan pada Gambar 3.22.

123	-8.5	89	4	84	-22	37	14
53.5	16	54.75	10.7	39.75	16.7	46.25	-23.7
49.25	-4.2	62	-2.5	33.5	5.5	73.25	-5.7
61.6	13.5	45.25	1.75	53	-14	127.75	-18.7
-4	-6	-2.75	3.2	-1.75	-7.2	-0.75	2.25
2.5	0	-7.25	0.75	-3.25	8.7	10.25	2.75
-11.25	5.25	1	4.5	-3	5	-26.25	-17.2
-4.5	-6.5	1.75	-4.75	-7	12	5.75	4.25

Gambar 3.22. Matrik D4

5. Mengatur hasil pada point 4 agar bagian aproksimasi berkumpul di bagian kiri dan bagian detail mengumpul di bagian kanan. Sehingga terbentuk matrik D5 yang ditunjukkan pada Gambar 3.23.

123	89	84	37	-8.5	4	-22	14
53.6	54.8	39.8	46.25	16	10.8	16.8	-23.7
49.25	62	33.5	73.25	-4.25	-2.5	5.5	-5.75
61.6	45.25	53	127.75	13.5	1.7	-14	-18.7
-4	-2.75	-1.75	-0.8	-6	3.25	-7.2	2.2
2.5	-7.25	-3.25	10.2	0	0.75	8.7	2.75
-11.25	1	-3	-26.25	5.25	4.5	5	-17.2
-4.5	1.75	-7	5.75	-6.5	-4.75	12	4.25

Gambar 3.23. Matrik D5

6. Telah dilakukan dekomposisi wavelet Haar 1 level terhadap citra. Matriks yang dihasilkan pada point 6 ditunjukkan pada Gambar 3.24.

123	89	84	37	-8.5	4	-22	14
53.5	54.75	39.75	46.25	16	10.75	16.75	-23.75
49.25	62	33.5	73.25	-4.25	-2.5	5.5	-5.75
61.5	45.25	53	127.75	13.5	1.75	-14	-18.75
-4	-2.75	-1.75	-0.75	-6	3.25	-7.25	2.25
2.5	-7.25	-3.25	10.25	0	0.75	8.75	2.75
-11.25	1	-3	-26.25	5.25	4.5	5	-17.25
-4.5	1.75	-7	5.75	-6.5	-4.75	12	4.25

Gambar 3.24. Matrik Dekomposisi 1 level

Keterangan:

- warna biru adalah bagian aproksimasi
- warna merah adalah bagian detail horizontal
- warna coklat adalah bagian detail vertikal
- warna hijau adalah bagian detail diagonal

Merujuk pada Gambar 3.17. matriks LH sebenarnya tidak perlu dibuat secara eksplisit, cukup dilakukan beberapa operasi perkalian saja. Langkah selanjutnya adalah melakukan transformasi level 2 dengan inputan bagian aproksimasi dari level 1. Tranformasi dikakukan hingga level ke n, dimana $N=2^n$, dengan N adalah ukuran citra. Oleh karena ukuran citra=8, maka iterasi dilakukan hingga

level 3 ($8=2^3$). Hasil akhir transformasi citra ini ditunjukkan pada Gambar 3.25.

235	1	12	8	-7	5.25	11.25	7.25
6	8	2	28	16	10.75	16.75	-23.75
1	-7	0	5	-4.25	-2.5	5.5	-5.75
0	-23	-7	9	13.5	1.75	-14	-18.75
-4	-2.75	-1.75	-0.75	-6	3.25	-7.25	2.25
2.5	-7.25	-3.25	10.25	0	0.75	8.75	2.75
-11.25	1	-3	-26.25	5.25	4.5	5	-17.25
-4.5	1.75	-7	5.75	-6.5	-4.75	12	4.25

Gambar 3.25. Matrik Dekomposisi 3 level

3.4.2 Proses Rekonstruksi Citra

Misalkan matrik yang akan akan direkonstruksi adalah matrik D5. Berikut adalah langkah-langkah yang perlu dilakukan untuk merekonstruksi citra dekomposisi kembali menjadi citra semula:

1. Menentukan filter rekonstruksi L* dan H*, ditunjukkan pada Gambar 3.26.

L*=

$1/\sqrt{2}$	0	0	0
$1/\sqrt{2}$	0	0	0
0	$1/\sqrt{2}$	0	0
0	$1/\sqrt{2}$	0	0
0	0	$1/\sqrt{2}$	0
0	0	$1/\sqrt{2}$	0
0	0	0	$1/\sqrt{2}$
0	0	0	$1/\sqrt{2}$

H* =

$1/\sqrt{2}$	0	0	0
$-1/\sqrt{2}$	0	0	0
0	$1/\sqrt{2}$	0	0
0	$-1/\sqrt{2}$	0	0
0	0	$1/\sqrt{2}$	0
0	0	$-1/\sqrt{2}$	0
0	0	0	$1/\sqrt{2}$
0	0	0	$-1/\sqrt{2}$

Gambar 3.26. filter rekonstruksi L*H*

2. Untuk setiap baris hasil dekomposisi (matriks D5), matriks L* dan H* dikalikan dengan bagian aproksimasi dari matriks D5 seperti ditunjukkan pada Gambar 3.27.

$1/\sqrt{2}$	0	0	0
$1/\sqrt{2}$	0	0	0
0	$1/\sqrt{2}$	0	0
0	$1/\sqrt{2}$	0	0
0	0	$1/\sqrt{2}$	0
0	0	$1/\sqrt{2}$	0
0	0	0	$1/\sqrt{2}$
0	0	0	$1/\sqrt{2}$

*

123
89
84
37

+

$1/\sqrt{2}$	0	0	0
$-1/\sqrt{2}$	0	0	0
0	$1/\sqrt{2}$	0	0
0	$-1/\sqrt{2}$	0	0
0	0	$1/\sqrt{2}$	0
0	0	$-1/\sqrt{2}$	0
0	0	0	$1/\sqrt{2}$
0	0	0	$-1/\sqrt{2}$

*

-8.5
4
-22
14

=

87
87
63
63
59
59
26
26

+

-6
6
3
-3
16
-16
10
-10

=

81
93
66
60
75
43
36
16

Gambar 3.27. Perkalian baris 1 dengan filter L*H*

Kemudian hasilnya kembali diletakkan perbaris di matriks hasil, yang nantinya akan sama persis dengan matriks D3

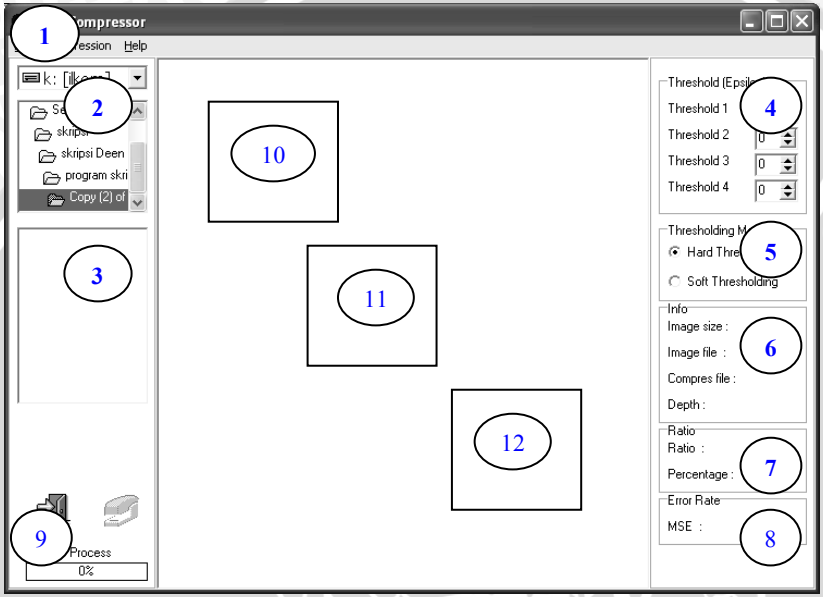
3. Dengan cara yang sama, dilakukan perkalian pada tahap 2 terhadap setiap kolom dari matriks yang diperoleh pada langkah sebelumnya.

4. Langkah ini akan menghasilkan kembali citra awal. Proses rekonstruksi selesai.

Merujuk pada Gambar 3.26, matriks L* dan H* sebenarnya tidak perlu dibuat secara eksplisit, cukup dilakukan beberapa operasi perkalian saja.

3.5 Perancangan Antarmuka

Perancangan antarmuka *software* yang akan dibuat, ditunjukkan pada Gambar 3.28



Gambar 3.28. Perancangan Antarmuka

Keterangan:

1. Menu *file*, *compression*, dan *help*.
2. *File directory*.
3. *File listbox*.
4. Input nilai *threshold*.
5. Metode *thresholding*.
6. Laporan kompresi, berupa nilai *file* citra, *file* kompresi dalam satuan *byte*, kedalaman transformasi, dan dimensi citra.
7. Nilai rasio hasil kompresi
8. Nilai tingkat kesalahan (*error rate*) kompresi.
9. *Progress bar*.
10. *Form* untuk menampilkan citra yang akan dikompresi.
11. *Form* untuk menampilkan citra hasil transformasi.
12. *Form* untuk menampilkan citra hasil kompresi.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan perangkat lunak kompresi citra digital ini adalah:

1. Prosesor Intel P3 667 Mhz
2. Memori 256 MB
3. Harddisk dengan kapasitas 80 GB
4. Monitor 15”
5. Keyboard
6. Mouse

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan perangkat lunak kompresi citra digital ini adalah :

1. Sistem operasi yang digunakan adalah Microsoft Windows XP Professional.
2. *Software* yang digunakan dalam mengembangkan sistem adalah Borland Delphi 6.
3. *Software* yang digunakan untuk menyiapkan citra uji adalah ACDSee 7.0.
4. *Software* yang digunakan untuk mengolah data adalah MS Excel 2003.

4.2 Implementasi Perangkat Lunak

Berdasarkan analisa dan perancangan proses yang terdapat pada subbab 3.2, maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

4.2.1 Struktur Data

Struktur data yang digunakan pada perangkat lunak ini direpresentasikan pada Gambar 4.1.

```
type
    ArSingle = array of array of single;
    ArTemp   = array of single;

    mYL,mU,mV : ArSingle;
    Temp,Temp2: ArTemp;
    b         : ArTemp;

     HWT_Header = packed record
                   width        : word;
                   height       : word;
                   depth        : byte;
                   factorQY     : single;
                   factorQU     : single;
                   factorQV     : single;
                   end;
```

Gambar 4.1. Struktur data

Keterangan struktur data pada Gambar 4.1 ditunjukkan pada Tabel 4.1.

Tabel 4.1 Keterangan Struktur Data

Struktur Data	Keterangan
`mYL,mU,mV:ArSingle;`	Matrik yang digunakan untuk menampung nilai komponen warna Y, U, dan V.
`Temp,Temp2: ArTemp;` `b : ArTemp;`	Array temporary yang digunakan pada saat transformasi HWT dan IHWT.

`HWT_Header = packed record`
`width : word;`
`height : word;`
`depth : byte;`
`factorQY : single;`
`factorQU : single;`
`factorQV : single;`
`end;`
	Record yang digunakan sebagai *Header file* kompresi. *Header* ini berisi beberapa informasi mengenai *file* kompresi yang berguna pada proses kompresi dan dekompresi.

4.2.2 Implementasi Kompresi

Langkah pertama yang dilakukan *User* untuk melakukan proses kompresi adalah memasukkan citra inputan, menentukan nilai *threshold*, menentukan metode *thresholding,* dan menentukan tempat *file* kompresi diletakkan. Setelah itu akan dilakukan proses seperti yang telah dijelaskan pada subbab 3.2.2.

4.2.2.1 Konversi Model Warna RGB ke YUV

Tahap awal dari proses kompresi ini adalah melakukan konversi model warna RGB citra masukan ke model warna YUV. Fungsi untuk melakukan konversi warna ditunjukkan pada Gambar 4.2.

```
function RGBtoYUV (var Img:TImage; warna:string;
                   wi,hi:integer):ArSingle;
var r,g,b :byte;     mYUV  :ArSingle;
    x,y   :integer;  Clr   :TColor;
begin
    SetLength(mYUV,wi,hi); //deklarasi dimensi matrik mYUV
    For y:=0 to hi-1 do
            For x:=0 to wi-1 do
            begin
    Clr := Img.Canvas.Pixels[x,y];
    r := GetRValue (Clr);
    g := GetGValue (Clr);
    b := GetBValue (Clr);

    if warna='Y' then  //konversi Komponen Y
    mYUV[x,y]:=((0.256788 * r) + (0.504129 * g) +
               (0.097906 *  b))+ 16
    else
    if warna='U' then  //konversi Komponen U
    mYUV[x,y]:=((-0.148223 * r) - (0.290993 * g) +
               (0.439216 * b))+ 128
    else
    if warna='V' then  //konversi Komponen V
    mYUV[x,y]:= ((0.439216 * r) - (0.367788 * g) -
               (0.071427 * b))+ 128;

     Result:=mYUV;
end;
```

Gambar 4.2. *SourceCode* fungsi RGBtoYUV

Fungsi RGBtoYUV akan menghasilkan matrik 2D bertipe *single* dengan dimensi sesuai citra masukan. Matrik yang dihasilkan dapat berupa matrik komponen Y, U, atau V.

4.2.2.2 HWT (*Haar Wavelet Transform*)

Setelah proses konversi selesai dilakukan, langkah selanjutnya adalah melakukan transformasi *haar wavelet* pada masing-masing komponen warna YUV. Prosedur untuk melakukan *HWT* ditunjukkan pada Gambar 4.3.

```
procedure NStdCalc(var a:ArSingle; Header:HWT_Header;
                   detw,deth:integer);
var
     j,k,r,c,x,y,n,o :integer;
     Temp,Temp2      : ArTemp;
Begin
     n:=Header.width;
     o:=Header.hight;

     SetLength(Temp,o); //deklarasi panjang array temp
     SetLength(Temp2,n);
     For y:=0 to o-1 do
        For x:=0 to n-1 do
        begin
        a[x,y]:=a[x,y]/sqrt(n); // divide entire array
        end;
     j:=n;
     k:=o;
     while (j>=2) and (k>=2) do
     begin
        //transformasi terhadap baris
        for r:=0 to j-1 do
        begin
        //mengkopi baris r dari array a ke array Temporary
                for c:=0 to k-1 do
                begin
                Temp[c]:=a[r,c];
                end;
          Temp:=HWTstep(Temp, k); //proses HWT thd baris
        //mengkopi array Temporary ke baris r dari array a
                for c:=0 to k-1 do
                begin
                a[r,c]:=Temp[c];
                end;
        end;
       //transformasi terhadap kolom
        for  c:=0 to k-1 do
        begin
       //mengkopi kolom c dari array a ke array Temporary
                for r:=0 to j-1 do
                begin
                Temp2[r]:=a[r,c];
                end;
```

```
        Temp2:=HWTstep(Temp2, j); //proses HWT thd kolom
       //mengkopi array Temporary ke kolom c dari array a
                for r:=0 to j-1 do
                begin
                a[r,c]:=Temp2[r];
                end;
        end;

     j:=round(j/2);//ukuran array pada level transformasi
     k:=round(k/2);//selanjutnya
     inc(m);       //level transformasi +1

  //jika dimensi pada transf selanjutnya ganjil maka keluar
    if (j mod 2=1) or (k mod 2=1) then
    break;
    end;
    detw:=j;
    deth:=k;
  //tampilkan depth level
    FormUtama.Label13.Caption:=inttostr(m);
end;
```

```
function HWTstep(ATemp:ArTemp; j:integer):ArTemp;
var i:integer;
    b:ArTemp;
begin

    SetLength(b,j); //deklarasi panjang array b
    //proses filtering
    for i:=0 to round(j/2)-1 do
     begin
     b[i]:=(ATemp[2*i]+ATemp[2*i+1])/sqrt(2);  //low pass
     b[round(j/2)+i]:=(ATemp[2*i]-ATemp[2*i+1])/sqrt(2);
                                                //high pass
     end;
     ATemp:=b; // move entire array
     Result:=ATemp;

end;
```

Gambar 4.3. *SourceCode* HWT

Prosedur HWT menerima masukan komponen warna YUV dalam matrik *a* dan akan menghasilkan matrik komponen warna YUV yang telah ditransformasi. Tingkat kedalaman transformasi citra bervariasi, bergantung pada dimensi citra masukkan. Misalnya citra orthogonal berdimensi 128x128 akan mengalami transformasi hingga level 7 ($128=2^7$). Sedangkan citra non orthogonal MxN akan

mengalami transformasi hingga dimensi citra berukuran ganjil. Misalnya citra berdimensi 8x6 hanya mengalami sekali transformasi, karena setelah ditransformasi dimensi citra menjadi 4x3. Contoh citra hasil transformasi ini ditunjukkan pada Gambar 4.4.

(a) Citra asli (b) Citra transformasi

Gambar 4.4. Citra hasil transformasi

Selain itu, prosedur HWT juga akan menghasilkan data integer *detw* dan *deth*. Data ini akan digunakan pada proses kuantisasi. Dengan data ini, sistem akan dapat membedakan daerah aproksimasi dan daerah detail pada matrik transformasi. Hal ini dikarenakan proses kuantisasi hanya dilakukan pada daerah detail saja.

4.2.2.3 Kuantisasi (Quantization)

Setelah proses konversi selesai dilakukan, langkah selanjutnya adalah melakukan kuantisasi pada masing-masing matrik komponen warna YUV. Selain matrik komponen warna YUV, proses ini juga membutuhkan inputan berupa data *detw*, *deth*, nilai dan metode *thresholding* yang digunakan. Nilai *detw* dan *deth* didapatkan dari proses HWT. Prosedur untuk melakukan kuantisasi ditunjukkan pada Gambar 4.5.

```
procedure quantization(var a:ArSingle;
                       wi,hi,th1,th2,th3,th4,det,deth:int);
var y,x,thres:integer;
begin

For y:=0 to hi-1 do
   For x:=0 to wi-1 do
   begin
        //mengecek apakah daerah detail?
       If (x>=detw) and (y>=deth) then
        begin
```

```
            //Pilih random nilai threshold(25/10/5/1)
             thres:=threshold(th1,th2,th3,th4);
            If FormUtama.RadioGroup1.ItemIndex=0 then
             begin
                //hard thresholding
                 If abs(a[x,y])<=thres then
                         Begin
                         a[x,y]:=0;
                         end
                         end
            else
            If FormUtama.RadioGroup1.ItemIndex=1 then
             begin
                 //soft thresholding
                 If abs(a[x,y]) < thres then
                         Begin
                         a[x,y]:= 0;
                         end
                 else
                          begin
                         a[x,y]:=abs(a[x,y])-thres;
                         end
           end;
   end;
end;
```

Gambar 4.5. *SourceCode* Kuantisasi

Prosedur ini menghasilkan output berupa matrik komponen warna YUV yang sudah dikuantisasi. Dalam proses ini sangat memungkinkan banyak koefisien matrik yang dinolkan. Sehingga matrik hasil kuantisasi banyak memiliki koefisien bernilai nol (*sparse matrik*).

Matrik komponen warna YUV bertipe data *float*, sehingga penyimpananya membutuhkan ruang yang cukup banyak. Oleh karena itu, diperlukan adanya konversi *range* data *float* ke *range* data *byte* untuk menghemat ruang penyimpanan. Konversi dilakukan dengan cara mengalikan koefisien matrik dengan faktor kuantisasi. Faktor kuantisasi didapatkan dari perhitungan nilai maksimal tipe data *byte* (255) dibagi nilai maksimal koefisien matrik yang akan dikonversi. Nilai faktor kuantisasi tersebut akan disimpan pada *header file* untuk proses dekompresi. Kode untuk melakukan konversi ini ditunjukkan pada Gambar 4.6.

```
//menghitung faktor kuantisasi
factorQ:=((255)/abs(max)) ;

For y:=0 to hi-1 do
   For x:=0 to wi-1 do
   begin
        a[x,y]:=(a[x,y]*factorQ );
   end;
end;
```

Gambar 4.6. *SourceCode* konversi data float ke range byte

4.2.2.4 RLE Encoder

Tahap akhir dari proses kompresi adalah menuliskan data hasil transformasi ke dalam file kompresi. Sebelum dituliskan dalam file kompresi, tiap-tiap matrik komponen YUV akan dikompresi terlebih dahulu menggunakan metode *RLE*. Selain itu, pada tahap ini juga akan dilakukan penulisan *Header file* yang berisi informasi mengenai dimensi matrik, kedalaman transformasi, dan faktor kuantisasi tiap komponen warna. Potongan kode untuk menyimpan header file ditunjukkan pada Gambar 4.7.

```
   //menulis header ke file kompresi
   Assignfile(fcompr,Filename);
   filemode:=2;
   Rewrite(fcompr,1);
   blockwrite(fcompr,Header,sizeof(Header));
   CloseFile (fcompr);
   //akhir simpan header
```

Gambar 4.7. *SourceCode* menyimpan *Header File*

Pada Gambar 4.7, *fcompr* adalah file kompresi tanpa type. *Header* merupakan record HWT_Header.

Setelah menuliskan *Header file*, langkah selanjutnya adalah melakukan RLE *encoding* terhadap masing-masing matrik komponen warna YUV. Proses ini dilanjutkan dengan menuliskan data hasil proses *encoding* pada *file* kompresi. Potongan prosedur untuk melakukan RLE *Encoder* pada komponen warna V ditunjukkan pada Gambar 4.8.

```
AssignFile (fl, Filename); //membuka file
ReWrite (fl);

 // Proses encoding komponen V
  Write (fl, Nol);
  Write (fl, Nol);
  total:= VLama;//inisialisasi total
  for y:=0 to hi-1 do
  begin
    for x:=0 to wi-1 do
    if (x>0)or(y>0) then
    begin

       if (Abs(maV[x,y]-VLama)=delta) and (cV<255)   then
       begin
         Inc(cV);
         total:=total+maV[x,y];
       end
       else
         begin
         rounded:=round(total/cV);
         Write (fl, rounded); //menuliskan nilai V
         Write (fl, cV);      //menuliskan jumlah V
         cV := 1;
         VLama := maV[x,y];
        total := maV[x,y];
      end;
    end;
  end;
rounded:=round(total/cV);
Write (fl, rounded); //menuliskan nilai V
Write (fl, cV);      //menuliskan jumlah V
CloseFile (fl);      // menutup file
```

Gambar 4.8. *SourceCode* RLE Encoder

Proses RLE encoder akan menghasilkan data matrik transformasi yang sudah dikelompokkan. Data tersebut berupa sepasang nilai koefisien matrik dan jumlahnya. Pada Gambar 4.5 data tersebut adalah *rounded* dan *cV*. Data inilah yang dituliskan dalam file kompresi.

4.2.3 Implementasi Dekompresi

Langkah pertama yang dilakukan *User* untuk melakukan proses dekompresi adalah memasukkan *file* kompresi. Setelah itu akan dilakukan proses seperti yang telah dijelaskan pada subbab 3.2.3.

4.2.3.1 RLE Decoder

Tahap awal dari proses dekompresi adalah membaca *Header file* kompresi. Kode yang digunakan untuk membaca *header file* ditunjukkan pada Gambar 4.9.

```
//membaca header file kompresi
Assignfile(fcompr,Filename);
filemode:=0;
Reset(fcompr,1);
blockread(fcompr,header,sizeof(header));
wi:=header.width;
hi:=header.height;
d :=header.depth;
CloseFile(fcompr);
//akhir membaca header
```

Gambar 4.9. *SourceCode* membaca *Header File*

Pada Gambar 4.9, *fcompr* adalah file kompresi tanpa tipe. *Header* merupakan *record* HWT_Header. Melalui *header* ini, didapatkan informasi mengenai dimensi matrik. Sehingga dapat dibuat matrik kosong yang digunakan untuk menampung data hasil proses dekompresi.

Setelah membaca *header file*, langkah selanjutnya adalah membaca data *file* kompresi. Kemudian data tersebut akan dimasukkan dalam matrik kosong yang telah disiapkan. Potongan prosedur untuk melakukan RLE *decoder* pada salah satu komponen warna (V) ditunjukkan pada Gambar 4.10.

```
AssignFile (fl, Filename);    //membuka file
  Reset (fl);                 // mode membaca file
  seek(fl, 18);

  //proses decoding matrik V
  SetLength(maV,wi,hi);
  x := 0;
  y := 0;
  while true do
  begin
    Read (fl, V);  //membaca nilai komponen V
    Read (fl,CoV); //membaca frekuensi komponen V
    cV:=round(CoV);
    if (V=0)and(CoV=0) then break;
```

```
    for i:=1 to cV do
    begin
         //memasukkan nilai V pada matrik V
      maV[x,y]:=V;
      Inc (x);
      if x=wi then
      begin
        x := 0;
        Inc (y);
      end;
    end;
end;
CloseFile (fl);
```

Gambar 4.10. *SourceCode* RLE Decoder

Prosedur ini akan menghasilkan matrik komponen warna YUV yaitu *maYL, maU*, dan *maV* yang bertipe data *single*. Tetapi nilai matrik-matrik tersebut masih berkisar antara 0 sampai 255, sehingga perlu dikonversi kembali menggunakan faktor kuantisasi untuk mendapatkan nilai yang sebenarnya. Proses konversi ini akan dilakukan pada saat akan melakukan proses IHWT.

4.2.3.2 IHWT (*Invers Haar Wavelet Transform*)

Setelah membaca data matrik komponen warna YUV, langkah selanjutnya adalah melakukan *invers* transformasi pada tiap-tiap komponen warna. Namun sebelum itu perlu dilakukan pencarian nilai awal *width* dan *height* matrik yang akan ditransformasi balik. Pencarian nilai ini membutuhkan informasi kedalaman transformasi (*depth*) yang didapatkan dari *header file*. Proses untuk mencari nilai awal ditunjukkan pada Gambar 4.11.

```
N    :=Header.width;
O    :=Header.height;
Depth:=Header.depth;

j:=n;
     k:=o;
     for l:=0 to depth-1 do
     begin
     if (l<>0) then begin
         j:=round(j/2);
         k:=round(k/2);
         end;
     end;
```

Gambar 4.11. *SourceCode* pencarian dimensi awal matrik

Langkah selanjutnya mengkonversi nilai pada tiap-tiap matrik komponen warna dari *range byte* ke *single*. Konversi nilai dilakukan dengan cara membagi nilai koefisien matrik dengan faktor kuantisasi tiap komponen warna. Faktor kuantisasi ini didapatkan dari *header file* yang telah dibaca sebelumnya. Kode untuk melakukan konversi nilai salah satu komponen warna ditunjukkan pada Gambar 4.12.

```
factorQ:=Header.factorQV; //deklarasi factor kuantisasi V

For y:=0 to o-1 do
   For x:=0 to n-1 do
   begin
        if (x<round(j/2)) and (y<round(k/2))then begin
        a[x,y]:=(a[x,y]/factorQ );
        end
        else
        begin
               //mengubah koefisien negatif
                if (a[x,y]>200) then begin
                    a[x,y]:=a[x,y]-256;
                    a[x,y]:=(a[x,y]/factorQ );
                    end
                    else begin
               //mengubah koefisien positif
                    a[x,y]:=(a[x,y]/factorQ );
                    end;
       end;
   end;
```

Gambar 4.12. *SourceCode* konversi nilai *byte* ke *single*

Setelah didapatkan nilai yang sebenarnya, proses selanjutnya adalah transformasi balik (IHWT). Prosedur untuk melakukan IHWT ditunjukkan pada Gambar 4.13.

```
    while (j<=n) and (k<=o) do
    begin
    SetLength(Temp,k); //deklarasi ukuran array temporary
    SetLength(Temp2,j);
        //transformasi terhadap kolom
        for c:=0 to k-1 do
        begin
        //mengkopi kolom c dari array a ke array Temporary
               for r:=0 to j-1 do
               begin
               Temp2[r]:=a[r,c];
               end;
        //proses IHWT terhadap kolom
        Temp2:=IHWTstep(Temp2, j);
        //mengkopi array Temporary ke kolom c dari array a
               for r:=0 to j-1 do
               begin
               a[r,c]:=Temp2[r];
               end;
        end;

        //transformasi terhadap baris
        for r:=0 to j-1 do
        begin
        //mengkopi baris r dari array a ke array Temporary
               for c:=0 to k-1 do
               begin
               Temp[c]:=a[r,c];
               end;
         //proses IHWT terhadap baris
        Temp:=IHWTstep(Temp, k);
        //mengkopi array Temporary ke baris r dari array a
               for c:=0 to k-1 do
               begin
                   a[r,c]:=Temp[c];
               end;
        end;
    j:=2*j; //ukuran array pada level transf selanjutnya
    k:=2*k;
    end;

    For y:=0 to o-1 do
       For x:=0 to n-1 do
       begin
       a[x,y]:=a[x,y]*sqrt(n); // normalisasi matrik
       end;
end;
```

```
function IHWTstep(ATemp:ArTemp; j:integer):ArTemp;
var i,m:integer;
    b:ArTemp;
begin
     SetLength(b,j);
     //proses filtering
     for i:=0 to round(j/2)-1 do
     begin
        //invers lowpass
        b[2*i]:=(ATemp[i]+ATemp[round(j/2)+i])/sqrt(2);
        //invers highpass
        b[2*i+1]:=(ATemp[i]-ATemp[round(j/2)+i])/sqrt(2);
     end;

    ATemp:=b;  // move entire array
    result:=ATemp;
end;
```

Gambar 4.13 *SourceCode* IHWT

Prosedur ini membutuhkan masukkan berupa matrik komponen YUV *a*, *header file*, dan kedalaman transformasi *depth*. Prosedur ini menghasilkan matrik komponen warna YUV yang telah ditransformasikan balik.

4.2.3.3 Konversi Model Warna YUV ke RGB

Tahap akhir proses dekompresi adalah proses konversi warna YUV ke RGB. Proses ini membutuhkan inputan berupa ke 3 matrik komponen warna YUV (*maY,maU,maV*). Proses ini menghasilkan nilai RGB yang akan langsung ditampilkan sebagai citra rekonstruksi. Prosedur untuk melakukan RLE decoder ditunjukkan pada Gambar 4.14.

```
procedure YUVtoRGB (var Img2:TImage; maY,maU,maV:ArSingle;
                    wi,hi:integer);
var y,x,n:integer;    Clr:TColor;
    r2,g2,b2:byte;
begin
For y:= 0 to hi-1 do
    For x := 0 to wi-1 do
    begin

r2 := ByteRange((1.164383 * maY[x,y] * 0.9)+ (1.596027 *
      (maV[x,y] - 128)));
g2 := ByteRange((1.164383 * maY[x,y]*0.9)-
      (0.391762*(maU[x,y]-128))-(0.812968*(maV[x,y]-128)));
b2 := ByteRange((1.164383 * maY[x,y] * 0.9) + (2.017232 *
       (maU[x,y] - 128)));

    Img2.Canvas.Pixels[x,y] := RGB(r2,g2,b2);
    end;
end;
```

```
function ByteRange (r:single) : byte;
begin
  if r<0 then
    Result := 0
  else if r>255 then
    Result := 255
  else
    Result := Round(r);
end;
```

Gambar 4.14. *SourceCode* YUVtoRGB

4.2.4 Implementasi Rasio Kompresi

Proses rasio digunakan untuk melakukan pengujian rasio. Untuk itu, proses ini membutuhkan inputan berupa ukuran file kompresi *filesizeK* dan ukuran citra asli *filesizeA* dalam satuan bytes. Proses ini menghasilkan nilai kompresi *Rasio* beserta persentasenya (*PRasio*). Prosedur untuk menghitung rasio kompresi ditunjukkan pada Gambar 4.15.

```
procedure Rasio(var filesizeA,filesizeK:single);
var Rasio,PRasio:real;
    R:integer;
begin
       Rasio:=(1-(filesizeK/filesizeA));
       PRasio:=Rasio*100;

FormUtama.Label14.caption:=formatfloat('0.00',Rasio);
FormUtama.Label15.caption:=formatfloat('0.00',PRasio)+'%';
end;
```

Gambar 4.15. *SourceCode* Rasio

4.2.5 Implementasi MSE

Proses MSE digunakan untuk menghitung tingkat kesalahan antara citra asli dengan citra rekonstruksi. Oleh karena itu, proses ini membutuhkan inputan berupa dimensi citra, citra asli dan citra rekonstruksi. Proses ini menghasilkan nilai MSE. Prosedur untuk melakukan proses MSE ditunjukkan pada Gambar 4.16.

```
procedure MSE(var Img:TImage; maY,maU,maV:ArSingle;
hi,wi:integer);
var x,y:integer;
    MSE:real;
    Clr:TColor;
    r,g,b:byte;
    TotY,TotU,TotV,MSEy,MSEu,MSEv,YL,U,V:real;
begin
TotY:=0;
TotU:=0;
TotV:=0;
for y:=0 to hi-1 do
       for x:=0 to wi-1 do
       begin
       Clr := Img.Canvas.Pixels[x,y];
         r := GetRValue (Clr);
         g := GetGValue (Clr);
         b := GetBValue (Clr);

 YL:=((0.256788 * r) + (0.504129 * g) + (0.097906 * b))+ 16;
 U:=((-0.148223 * r) - (0.290993 * g) + (0.439216 * b))+128;
 V:=((0.439216 * r) - (0.367788 * g) - (0.071427 * b))+ 128;

       MSEy:=sqr(YL-maY[x,y]);
        TotY:=TotY+MSEy;
        MSEu:=sqr(U-maU[x,y]);
        TotU:=TotU+MSEu;
       MSEv:=sqr(V-maV[x,y]);
        TotV:=TotV+MSEv;
       end;
```

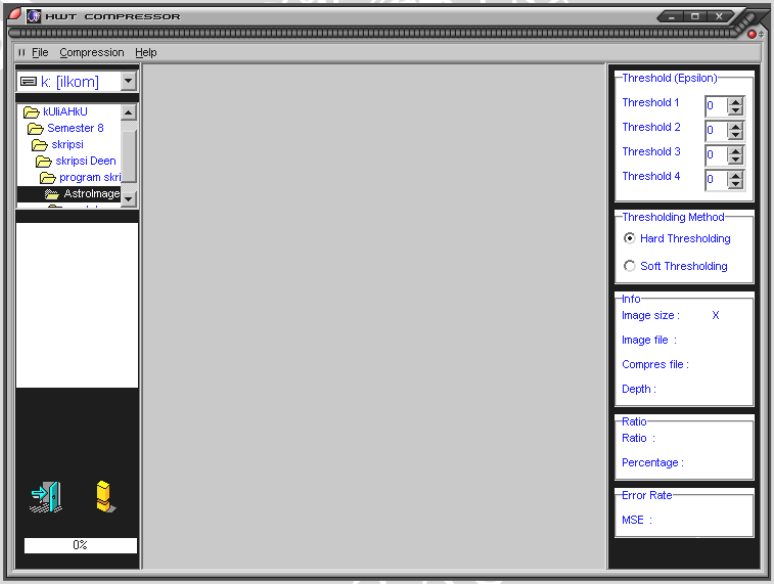
```
MSE:=(TotY+TotU+TotV)/3;
MSE:=MSE/(wi*hi);

FormUtama.Label16.caption:=formatfloat('0.00',MSE);
end;
```

Gambar 4.16 *SourceCode* MSE

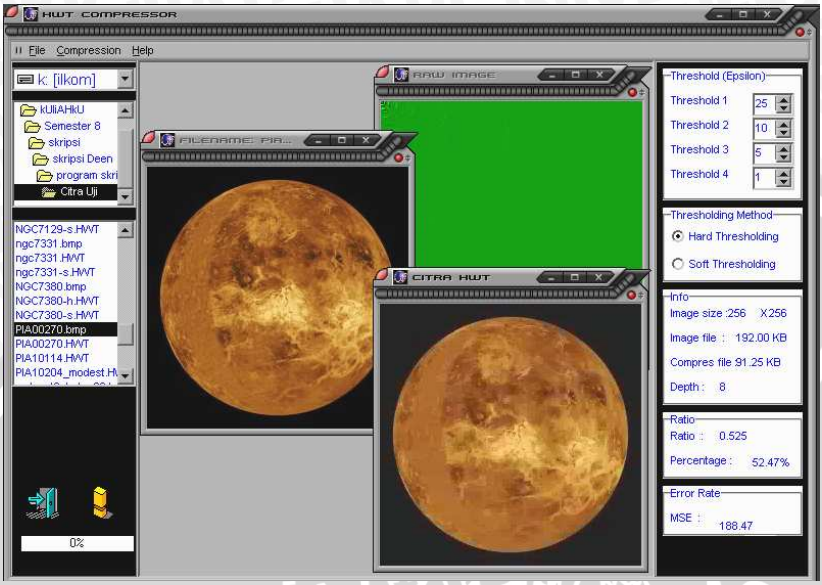
4.3 Implementasi Antarmuka (*interface*)

Berdasarkan rancangan antarmuka pada Sub bab 3.5, dihasilkan antarmuka untuk kompresi citra. Gambar 4.17 adalah tampilan utama.



Gambar 4.17 Tampilan Utama

Pada proses kompresi, *user* perlu menginputkan, citra inputan, nilai *threshold*, dan metode *threshold* yang digunakan. Tampilan pada saat proses kompresi ditunjukkan pada Gambar 4.18.



Gambar 4.18. Tampilan Proses kompresi

4.4 Implementasi Uji Coba

4.4.1 Hasil Uji Rasio

Pengujian rasio kompresi citra astronomi dilakukan sebanyak tiga kali dengan nilai *threshold* 25, 10, 5, dan 1 yang akan dipilih secara *random* pada saat akan melakukan *thresholding*. Selain itu, setiap citra astronomi akan diuji menggunakan metode *hard* dan *soft thresholding*.

Hasil uji rasio menggunakan metode *hard thresholding* ditunjukkan pada Tabel 4.2. Sedangkan hasil uji rasio menggunakan metode *soft thresholding* ditunjukkan pada Tabel 4.3. Data mengenai citra uji dapat dilihat pada lampiran.

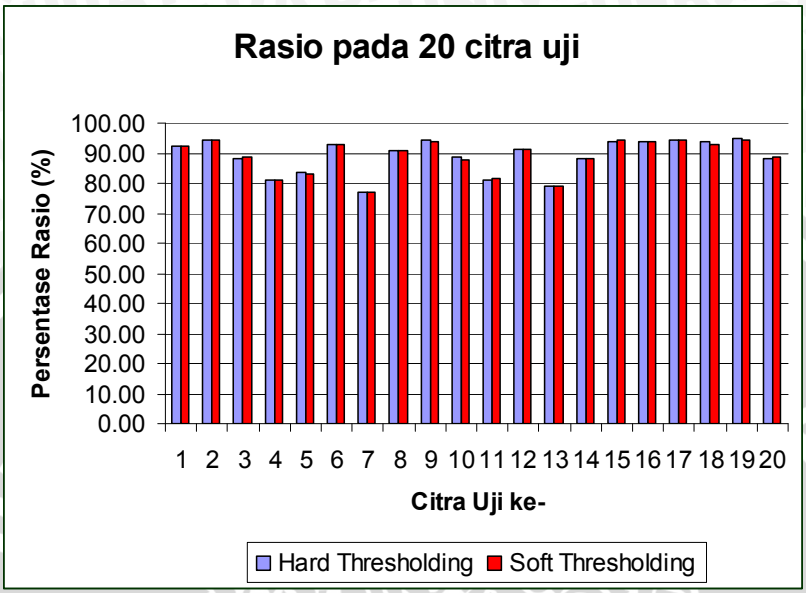
Tabel 4.2. Rasio Citra Uji menggunakan *Hard Thresholding*

No	Nama citra	Rasio Kompresi				
		Kompresi			Rata-rata	Persentase(%)
		1	2	3		
1	Coronet Cluster.bmp	0.924	0.921	0.92	0.921	92.16
2	Holmes-f4.bmp	0.945	0.944	0.941	0.943	94.33
3	horseregion.bmp	0.893	0.889	0.87	0.884	88.4
4	M5.bmp	0.809	0.809	0.814	0.810	81.06
5	M45WF.bmp	0.834	0.832	0.84	0.835	83.53
6	M51_003.bmp	0.924	0.931	0.927	0.927	92.73
7	Mira.bmp	0.766	0.766	0.771	0.767	76.76
8	Moon.bmp	0.91	0.91	0.91	0.91	91
9	NGC2841.bmp	0.943	0.942	0.943	0.942	94.26
10	NGC6962.bmp	0.886	0.885	0.89	0.887	88.7
11	NGC7129.bmp	0.814	0.812	0.809	0.811	81.16
12	NGC7331.bmp	0.911	0.911	0.914	0.912	91.2
13	NGC7380.bmp	0.791	0.785	0.787	0.787	78.76
14	NGC1291.bmp	0.892	0.879	0.882	0.884	88.43
15	NGC4258.bmp	0.943	0.941	0.938	0.940	94.06
16	NGC4569.bmp	0.943	0.941	0.938	0.940	94.06
17	Redspot2.bmp	0.945	0.945	0.942	0.944	94.4
18	Rho Ophiuchi.bmp	0.93	0.944	0.937	0.937	93.7
19	Saturntitan.bmp	0.947	0.946	0.948	0.947	94.7
20	Venus.bmp	0.883	0.878	0.885	0.882	88.2
rata-rata					0.890	89.08

Tabel 4.3. Rasio Citra Uji menggunakan *Soft Thresholding*

No.	Nama citra	Rasio Kompresi				
		Kompresi			Rata-rata	Persentase (%)
		1	2	3		
1	Coronet Cluster.bmp	0.919	0.922	0.923	0.921	92.13
2	Holmes-f4.bmp	0.939	0.944	0.941	0.941	94.13
3	horseregion.bmp	0.88	0.894	0.891	0.888	88.83
4	M5.bmp	0.811	0.809	0.813	0.811	81.1
5	M45WF.bmp	0.825	0.833	0.83	0.829	82.93
6	M51_003.bmp	0.924	0.924	0.929	0.925	92.56
7	Mira.bmp	0.77	0.766	0.769	0.768	76.83
8	Moon.bmp	0.91	0.91	0.91	0.91	91
9	NGC2841.bmp	0.942	0.941	0.94	0.941	94.1
10	NGC6962.bmp	0.87	0.886	0.879	0.878	87.83
11	NGC7129.bmp	0.817	0.812	0.814	0.814	81.43
12	NGC7331.bmp	0.914	0.911	0.921	0.915	91.53
13	NGC7380.bmp	0.789	0.782	0.791	0.787	78.73
14	NGC1291.bmp	0.88	0.883	0.879	0.880	88.06
15	NGC4258.bmp	0.936	0.947	0.941	0.941	94.13
16	NGC4569.bmp	0.931	0.943	0.941	0.938	93.83
17	Redspot2.bmp	0.945	0.942	0.94	0.942	94.23
18	Rho Ophiuchi.bmp	0.93	0.924	0.927	0.927	92.7
19	Saturntitan.bmp	0.947	0.946	0.944	0.945	94.56
20	Venus.bmp	0.885	0.885	0.889	0.886	88.63
	rata-rata				0.889	88.96

Pada Tabel 4.2 dan 4.3 telihat bahwa nilai MSE citra astronomi pada pengujian 1, 2, dan 3 tidaklah sama. Hal ini dikarenakan pemilihan nilai *threshold* secara *random* Hasil perhitungan rasio rata-rata pada Tabel 4.2 dan 4.3 dapat disajikan dalam bentuk grafik seperti pada Gambar 4.19.



Gambar 4.19. Grafik perbandingan rasio antara *Hard* dan *Soft thresholding*.

4.4.2 Hasil Uji MSE

Pengujian MSE kompresi citra astronomi dilakukan sebanyak tiga kali dengan nilai *threshold* 25, 10, 5, dan 1 yang akan dipilih secara *random* pada saat akan melakukan *thresholding*. Selain itu, setiap citra astronomi akan diuji menggunakan metode *hard* dan *soft thresholding*.

Hasil uji MSE menggunakan metode *hard thresholding* ditunjukkan pada Tabel 4.4. Sedangkan hasil uji rasio menggunakan metode *soft thresholding* ditunjukkan pada Tabel 4.5.

Tabel 4.4. MSE Citra Uji menggunakan *Hard Thresholding*

No	Nama citra	MSE			
		Kompresi			Rata-rata
		1	2	3	
1	Coronet Cluster.bmp	23.83	22.76	21.32	22.63
2	Holmes-f4.bmp	4.04	4.16	3.99	4.06
3	Horseregion.bmp	13.07	14.01	12.87	13.31

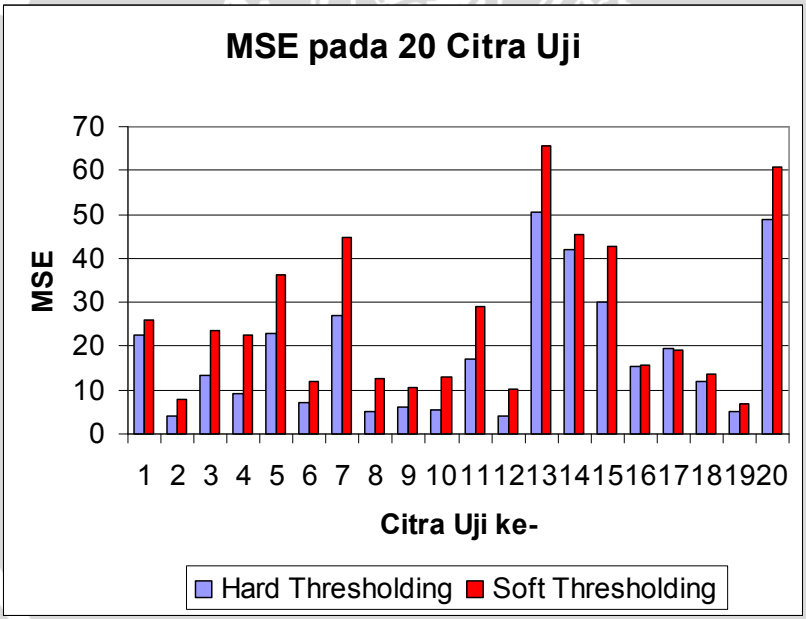
4	M5.bmp	9.35	9.65	8.56	9.18
5	M45WF.bmp	23.16	23.76	21.45	22.79
6	M51_003.bmp	7.81	6.89	7.23	7.31
7	Mira.bmp	27.71	27.54	25.64	26.96
8	Moon.bmp	5.07	6.03	4.61	5.23
9	NGC2841.bmp	6.14	5.79	6.09	6
10	NGC6962.bmp	5.81	5.77	5.24	5.6
11	NGC7129.bmp	17.15	16.66	17.37	17.06
12	NGC7331.bmp	4.14	4.11	3.79	4.01
13	NGC7380.bmp	40.09	54.36	57.21	50.55
14	NGC1291.bmp	42.34	41.09	42.07	41.83
15	NGC4258.bmp	29.44	29.08	31.76	30.09
16	NGC4569.bmp	15.17	15.78	14.99	15.31
17	Redspot2.bmp	20.54	19.87	18.24	19.55
18	Rho Ophiuchi.bmp	12.13	11.76	12.34	12.07
19	Saturntitan.bmp	5.06	4.67	5.71	5.14
20	Venus.bmp	54.65	46.21	46.01	48.95
rata-rata					18.38

Tabel 4.5. MSE Citra Uji menggunakan *Soft Thresholding*

No	Nama citra	MSE			
		Kompresi			Rata-rata
		1	2	3	
1	Coronet Cluster.bmp	24.77	25.76	26.87	25.8
2	Holmes-f4.bmp	7.61	8.02	8.12	7.916
3	Horseregion.bmp	22.84	23.01	24.71	23.52
4	M5.bmp	22.69	23.33	21.99	22.67
5	M45WF.bmp	34.74	36.99	37.01	36.24
6	M51_003.bmp	11.12	12.32	12.56	12
7	Mira.bmp	44.34	45.05	45.18	44.85
8	Moon.bmp	11.93	12.87	13.07	12.62
9	NGC2841.bmp	9.95	10.98	11.28	10.73
10	NGC6962.bmp	12.27	13.76	12.99	13.00
11	NGC7129.bmp	28.62	29.54	29.35	29.17
12	NGC7331.bmp	9.24	10.32	10.99	10.18
13	NGC7380.bmp	64.59	65.77	66.37	65.57

14	NGC1291.bmp	42.28	47.89	46.21	45.46
15	NGC4258.bmp	43.34	42.12	42.69	42.71
16	NGC4569.bmp	15.03	15.99	16.24	15.75
17	Redspot2.bmp	18.66	19.31	19.57	19.18
18	Rho Ophiuchi.bmp	12.84	13.65	13.83	13.74
19	Saturntitan.bmp	6.63	7.35	7.01	6.99
20	Venus.bmp	57.14	63.55	62.08	60.92
	rata-rata				25.95

Pada Tabel 4.4 dan 4.5 telihat bahwa nilai MSE citra astronomi pengujian 1, 2, dan 3 tidaklah sama. Hal ini dikarenakan pemilihan nilai *threshold* secara *random*. Hasil perhitungan MSE rata-rata pada Tabel 4.4 dan 4.5 dapat disajikan dalam bentuk grafik seperti pada Gambar 4.20.



Gambar 4.20. Grafik perbandingan MSE antara *Hard* dan *Soft thresholding*.

4.4.3 Analisa Hasil

Hasil pengujian rasio di subbab 4.4.1 pada ke 20 citra astronomi menunjukkan nilai yang bervariasi. Baik menggunakan metode *hard* ataupun *soft thresholding*. Citra Mira.bmp memiliki rasio kompresi yang terkecil yaitu 76,76 % dan 76,83%. Sedangkan citra Saturntitan.bmp memiliki rasio kompresi terbesar yaitu 94,7% dan 94,56%. Perbedaan ini dikarenakan perbedaan objek yang terdapat pada citra. Citra Mira.bmp memiliki banyak objek yang menyebar, sehingga menghasilkan data transformasi yang tidak banyak memiliki nilai nol.

Berdasarkan Tabel 4.1 dan Tabel 4.2, diketahui bahwa rata-rata total rasio kompresi menggunakan metode *hard thresholding* adalah 89,08 %. Sedangkan rata-rata total rasio kompresi menggunakan metode *soft thresholding* adalah 88,96 %. Data perbandingkan rasio kompresi tersebut dapat diperjelas dengan melihat grafik Gambar 4.19. Berdasarkan grafik tersebut terlihat bahwa rasio kompresi citra yang menggunakan metode *hard thresholding* tidak memiliki perbedaan yang berarti dibandingkan kompresi citra yang menggunakan metode *Soft thresholding,* bahkan cenderung sama.

Hasil pengujian MSE di subbab 4.4.2 pada ke 20 citra astronomi menunjukkan nilai yang bervariasi. Pada metode *hard thresholding.* Citra NGC7331.bmp memiliki MSE kompresi yang terkecil yaitu 4,01. Sedangkan citra NGC7380.bmp memiliki MSE kompresi terbesar yaitu 50,55. Pada metode *soft thresholding,* citra Saturntitan.bmp memiliki MSE kompresi yang terkecil yaitu 6,99. Sedangkan citra NGC7380.bmp memiliki MSE kompresi terbesar yaitu 65,57. Perbedaan nilai MSE disebabkan perbedaan data yang dihilangkan pada saat proses *thresholding* pada citra.

Berdasarkan Tabel 4.3 dan Tabel 4.4, diketahui bahwa rata-rata total MSE kompresi menggunakan metode *hard thresholding* adalah 18,38. Sedangkan rata-rata total MSE kompresi menggunakan metode *soft thresholding* adalah 25,95. Data perbandingkan MSE kompresi tersebut dapat diperjelas dengan melihat grafik Gambar 4.20. Berdasarkan grafik tersebut terlihat bahwa MSE kompresi citra yang menggunakan metode *hard thresholding* memiliki perbedaan yang berarti dibandingkan kompresi citra yang menggunakan metode *Soft thresholding.* Metode *Soft* menghasilkan MSE yang lebih besar dari pada metode *hard.* Hal ini dikarenakan prinsip kerja metode *soft*

adalah meredam / mengurangi jangkauan koefisien *wavelet*. Sehingga memodifikasi energi sinyal.

Contoh perbandingan kompresi citra astronomi menggunakan metode *hard* dan *soft* dengan nilai ε=25,10,5,1 ditunjukkan pada Gambar 4.21.

(a) RedSpot2.bmp (b) Rekonstruksi *Hard* (c) Rekonstruksi *Soft*

Gambar 4.21. Perbandingan citra rekonstruksi *Hard* dan *Soft*

Secara visual Gambar 4.21 (b) lebih baik dari pada Gambar 4.15 (c) . Dengan demikian data citra asli yang berubah pada citra hasil rekonstruksi *hard thresholding* lebih sedikit dibandingkan pada citra rekonstruksi *soft thresholding*.

UNIVERSITAS BRAWIJAYA

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, kesimpulan yang diperoleh antara lain:

1. Model transformasi citra menggunakan *Modified Haar Wavelet Transform* yang dapat menghasilkan matrik tranformasi sesuai dengan nilai dan metode *thresholding* yang digunakan telah dirancang dan diimplementasikan.
2. Model kompresi *RLE* untuk mengkompres matrik *Haar Wavelet Transform* telah dirancang dan diimplementasikan.
3. Berdasarkan pengujian yang telah dilakukan, didapatkan nilai rata-rata rasio dari ke-20 citra uji menggunakan metode *hard thresholding* sebesar 89,08 %. Sedangkan nilai rata-rata rasio menggunakan *soft thresholding* sebesar 88,96 %. Sehingga dapat disimpulkan bahwa rasio kedua metode tidak berbeda jauh.
4. Pada pengujian tingkat kesalahan yang diukur menggunakan nilai MSE, didapatkan nilai rata-rata menggunakan metode *hard thresholding* sebesar 18,38. Sedangkan nilai rata-rata MSE menggunakan *soft thresholding* sebesar 25,95. Sehingga dapat diambil kesimpulan bahwa metode *hard thresholding* lebih layak digunakan untuk mengkompresi citra astronomi, karena menghasilkan MSE yang lebih kecil.
5. Modifikasi energi sinyal yang merupakan prinsip kerja metode *soft thresholding* menyebabkan reduksi data yang lebih besar dari pada metode *hard thresholding*. Hal inilah yang menyebabkan nilai MSE metode *soft thresholding* cenderung lebih besar dari nilai MSE metode *hard thresholding.*

5.2 Saran

Saran yang diberikan untuk mengembangkan penelitian ini antara lain:

1. Melakukan transformasi citra menggunakan metode *wavelet* yang lain dan membandingkannya. Misalnya *Daubechies D4, D5,* dan *B-SpinLine*.
2. Melakukan kompresi menggunakan nilai *threshold* yang lain.

3. Menggunakan metode kompresi *lossless* yang lain. Misalnya *Shannon, Aritmathic, Huffman*.
4. Menggunakan metode thresholding yang lain. Misalnya *universal thresholding*.
5. Menerapkan *Modified Haar Wavelet* untuk mengkompresi citra digital pada bidang lain seperti citra kedokteran.

DAFTAR PUSTAKA

Ames, Greg. 2002. *Image Compression.* Jurnal fall2002.

Balza, Achmad dan Kartika Firdausy. 2004. *Teknik Pengolahan Citra Digital Menggunakan Delphi*. Yogyakarta : Ardi Publishing.

Dhani. 2002. *Aneka Instrumen Untuk Teleskop Modern* http://dhani.singcat.com/. 14 Maret 2008

Gonzalez, Rafael C. dan Richard E. Woods. 2005. *Digital Image Processing*,2nd ed., Prentice-Hall Inc., New Jersey.

Handoko, dkk. 2005. *Kompresi Citra Berwarna Dengan OBDD.* Jurnal Informatika Vol. 6, No. 1,: 17 – 23.

Hartman, William K. 1991. *Astronomy: The Cosmic Journey*. San Francisco : Wadsworth Publishing.

Http://id.wikipedia.org/wiki/ Astronomi. 5 April 2008

Kaplan, Ian. 2004. *Applying the Haar Wavelet Transform to Time Series Information*. http://www.bearcave.com/misl/misl_tech/wavelets/haar.html . 14 Maret 2008

Khalid, Sayood. 2000. *Introduction to Data Compression*, Second Edition, Morgan Kaufmann Publishers.

Linda S., Agustina. 2005. *Penerapan Region of Interest (ROI) pada Metode Kompresi JPEG2000*. Departemen Teknik Informatika, Institut Teknologi Bandung.

Murni, Aniati dan Dina Cahyati. 2003. *Pengolahan Citra Digital*. Fakultas Ilmu Komputer Universitas Indonesia.

Porwik, Piotr,dan Agnieszka Lisowska. 2004. *The Haar–Wavelet Transform in Digital Image Processing: Its Status and Achievements*. Jurnal Machine Graphics & Vision.

Raviraj, P. and M.Y. Sanavullah. 2007. *The Modified 2D-Haar Wavelet Transformation in Image Compression*. Middle-East Journal of Scientific Research 2 (2): 73-78.

Saha, Subhasis.2004. *Image Compression - from DCT to Wavelets : A Review*. http://www.acm.org/crossroads/xrds6-3 /sahaimgcoding. html. 14 Maret 2008

Salomon, David. 2004. *Data Compression: The Complete Reference*. New York: Springer-Verlag, Inc.

Sandberg, Kristian. 2000. *The Haar wavelet transform*. http://amath.colorado.edu/courses/4720/2000Spr/Labs/haar.html. 14 Maret 2008

Suhendra, Adang. 2007. *Catatan Kuliah Pengantar Pengolahan Citra.* Fakultas Teknik Informatika Universitas Kristen Duta Wacana.

Walidainy, Hubbul dan Nazlun. 2004. *Simulasi Menghapus Derau Pada Sinyal Suara*. Jurusan Elektro Fakultas Teknik Universitas Syiah Kuala.

White, Richard L dan Jeffrey W. Percival. 1995. *Compression and progressive transmission of astronomical images.* Space Telescope Science Institute. Baltimore.

Lampiran

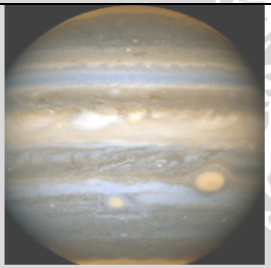
Data Citra Uji

No	Citra	Keterangan
1	 Coronet Cluster.bmp	Mission: Spitzer Space Telescope (SST) Spacecraft: Spitzer Space Telescope (SST) Instrument: Infrared Array Camera (IRAC) Product Size: 3300 samples x 3300 lines Produced By: California Institute of Technology Full-Res TIFF: PIA09926.tif (32.67 MB) Full-Res JPEG: PIA09926.jpg (583.5 kB)
2	 Holmes-f4.bmp	The comet brightened extremely rapidly from extremely dim to naked-eye visibility, around magnitude 2.5; then almost as rapidly began to dim again. It is now about 2 astronomical units (1 au = radius of Earth's orbit) from the sun.
3	 Horseregion_ssro2.bmp	To bring out details of the Horsehead's pasture, amateur astronomers at the Star Shadow Remote Observatory in New Mexico, USA fixed a small telescope on the region for over seven hours filtering out all but a very specific color of red light emitted by hydrogen.

4	 M5.bmp	Scope: Astro-Physics 130 EDF at f4.5 on A-P 1200GTO mount; unguided Meade 416XTE CCD camera with 616 color wheel controlled with MaxIm DL/CCD v3.05
5	 M45WF.bmp	M45 "The Pleiades" and IC353 + IC1995 TeleVue 101 IS f/5.3 SBIG STL-11K A/P 1200 GTO Taken Nov. 2007 from Foresthill, CA
6	 M51_003.bmp	Scope: RCOS Ritchey-Chretien 12.5 inch scope on an A-P 1200GTO mount with an A-P 0.75 x reducer for approximately f/6.8. Camera: SBIG ST-10XME with CFW-8A color wheel. MaxIm CCD camera control. Guiding was via the built-in guide chip.
7	 Mira.bmp	Target Name: Mira Mission: Galaxy Evolution Explorer (GALEX) Spacecraft: GALEX Orbiter Instrument: Ultraviolet/Visible Camera Product Size: 1569 samples x 800 lines Produced By: California Institute of Technology Full-Res TIFF: PIA09959.tif (3.772 MB)

		Full-Res JPEG: PIA09959.jpg (275.3 kB)
8	 Moon.bmp	A film image of the moon; E200 film, 1/500 at f12, contrast enhanced and converted to gray scale in Photoshop.
9	 NGC2841.bmp	This sharp view of the gorgeous island universe shows off a striking yellow nucleus and galactic disk with tightly wound spiral arms.
10	 NGC6962.bmp	
11	 NGC7129.bmp	This panorama was made from two frames using the RCOSCC 14.5" f/8 and the SBIG STL-11K camera.

12	 NGC7331.bmp	Scope: 12.5 inch RCOS Ritchey-Chretien at f/4.5 (native f/9 x 0.75 x 0.67 A-P reducers) on an A-P 1200GTO mount. Camera: SBIG ST-10MXE and CFW-8a color wheel, controlled with MaxIm.
13	 NGC7380.bmp	NGC 7380 in Cepheus Taken with 14.5" RCOSCC and SBIG ST 11K. LRGB 100:80:60:60
14	 NGC1291.bmp	Mission: Galaxy Evolution Explorer (GALEX) Spacecraft: GALEX Orbiter Instrument: Ultraviolet/Visible Camera Product Size: 1012 samples x 1012 lines Produced By: California Institute of Technology Full-Res TIFF: PIA10114.tif (3.077 MB) Full-Res JPEG: PIA10114.jpg (145.5 kB)
15		Target Name: NGC 4258 Mission: Spitzer Space Telescope (SST) Spacecraft: Spitzer Space Telescope (SST) Instrument: Infrared Array Camera

	NGC4258.bmp	(IRAC) Product Size: 3000 samples x 2501 lines Produced By: California Institute of Technology Full-Res TIFF: PIA10204.tif (22.51 MB) Full-Res JPEG: PIA10204.jpg (238.8 kB)
16	 NGC4569.bmp	Mission: Galaxy Evolution Explorer (GALEX) Spacecraft: GALEX Orbiter Instrument: Ultraviolet/Visible Camera Product Size: 1122 samples x 1122 lines Produced By: California Institute of Technology Full-Res TIFF: PIA10115.tif (3.781 MB) Full-Res JPEG: PIA10115.jpg (167.7 kB)
17	 Redspot2.bmp	This sharp Hubble Space Telescope image showing the two salmon colored Jovian storms was recorded in April. About half the size of the original Red Spot, Red Spot Jr. is similar in diameter to planet Earth.

18	 Rho Ophiuchi.bmp	Target Name: Rho Ophiuchi Mission: Spitzer Space Telescope (SST) Spacecraft: Spitzer Space Telescope (SST) Instrument: Infrared Array Camera (IRAC) Product Size: 6020 samples x 2905 lines Produced By: California Institute of Technology Full-Res TIFF: PIA10182.tif (52.46 MB) Full-Res JPEG: PIA10182.jpg (1.065 MB)
19	 Saturntitan.bmp	Pictured above, magnificent Saturn and enigmatic Titan were imaged together in true color by Cassini earlier this year.
20	 Venus.bmp	Target Name: Venus Is a satellite of: Sol (our sun) Mission: Magellan Spacecraft: Magellan Instrument: Imaging Radar Product Size: 4096 samples x 4096 lines Produced By: JPL

		Producer ID: P42383 Addition Date: 1996-09-23 Primary Data Set: Magellan MIDRs Full-Res TIFF: PIA00270.tif (7.793 MB) Full-Res JPEG: PIA00270.jpg (1.615 MB)