TUGAS AKHIR -SM141501

RESTORASI CITRA PADA KOMPRESI SPIHT (*SET PARTITIONING IN HIERARCHICAL TREES*) MENGGUNAKAN METODE *ITERATIF LANCZOS-HYBRID REGULARIZATION*

PRISMAHARDI AJI RIYANTOKO
NRP 1212 100 051

Dosen Pembimbing
Dr. Budi Setiyono, S.Si, MT

PROGRAM STUDI S1
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

FINAL PROJECT - SM141501

IMAGE RESTORATION FOR SPIHT (SET PARTITIONING IN HIERARCHICAL TREES) COMPRESSION USING ITERATIF LANCZOS-HYBRID REGULARIZATION METHOD

PRISMAHARDI AJI RIYANTOKO
NRP 1212 100 051

Supervisor
Dr. Budi Setiyono, S.Si, MT

UNDERGRADUATE PROGRAMME
DEPARTMENT OF MATHEMATICS
FACULTY OF MATHEMATICS AND NATURAL SCIENCE
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

LEMBAR PENGESAHAN

RESTORASI CITRA PADA KOMPRESI SPIHT (SET PARTITIONING IN HIERARCHICAL TREES) MENGGUNAKAN METODE ITERATIF LANCZOS-HYBRID REGULARIZATION

IMAGE RESTORATION FOR SPIHT (SET PARTITIONING IN HIERARCHICAL TREES) COMPRESSION USING ITERATIF LANCZOS-HYBRID REGULARIZATION

Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Sains
Pada Bidang Matematika Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
PRISMAHARDI AJI RIYANTOKO
NRP. 1212 100 051

Menyetujui,
Dosen Pembimbing

Dr. Budi Setiyono, S.Si, MT
NIP. 19720207 199702 1 001

Mengetahui,
Ketua Jurusan Matematika
FMIPA ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, Juli 2016

RESTORASI CITRA PADA KOMPRESI SPIHT *(SET PARTICAL IN HIERARCHICAL TREES)* MENGGUNAKAN METODE *ITERATIF LANCZOS – HYBRID REGULARIZATION*

Nama Mahasiswa	**: Prismahardi Aji Riyantoko**
NRP	**: 1212 100 051**
Jurusan	**: Matematika**
Dosen Pembimbing	**: Dr. Budi Setiyono, S.Si, MT**

Abstrak

Restorasi citra merupakan proses merekonstruksi atau mendapatkan kembali citra asli dari sebuah citra yang terdegradasi agar dapat menyerupai citra asli. Kompresi citra merupakan salah satu proses pemampatan citra yang menyebabkan citra mengalami degradasi atau penurunan kualitas. Penurunan kualitas citra terjadi pada proses kompresi *loosy*, salah satu contoh kompresi *loosy* adalah dengan metode *Set Partitioning In Hierarchical Tress (SPIHT).* Oleh karena itu, untuk meningkatkan kembali kualitas citra agar menyerupai citra asli maka digunakan restorasi citra dengan metode *Iterative Lanczos Hybrid Regularization*. Pada tugas akhir ini menggunakan citra *grayscale* dengan beberapa variasi resolusi citra untuk data uji coba kompresi SPIHT dan restorasi citra. Pengujian restorasi citra dengan data uji coba nilai PSNR sebesar 25 dB mengalami kenaikan nilai PSNR rata-rata sebesar 0,91 dB dan waktu komputasi 187,058 detik lebih lambat dari proses kompresi. Pada data uji coba nilai PSNR sebesar 35 dB mengalami kenaikan nilai PSNR rata rata sebesar 0,57 dB dan waktu komputasi 127,418 detik lebih cepat dari proses kompresi. Hal ini menunjukkan bahwa citra hasil restorasi dengan menggunakan metode *iteratif lanczos hybrid regularization* dapat meningkatkan kualitas citra.

Kata Kunci—Kompresi, SPIHT, Restorasi, *Iterative Lanczos Hybrid Regularization*.

Image Restoration for SPIHT (Set Partitioning In Hierrarchial Trees) Compression Using Iteratif Lanczos-Hybrid Regularization Method

Name	***: Prismahardi Aji Riyantoko***
NRP	***: 1212 100 051***
Department	***: Mathematics***
Supervisor	***: Dr. Budi Setiyono, S.Si, MT***

Abstract

Image restoration is reconstructing original image from image degradation so that can be similar with original image. Image compression is one of image processing which causes image degradation or loss of quality. A decrease image quality occurs in loosy compression, an example using Set Partitioning In Hierarchical Trees. Therefore, to improve the image quality, back to resemble the original image used image restoration with iterative lanczos-hybrid regularization method. In this research using grayscale image with some variation of the image resolution for trial data compression and image restoration. Image restoration program with PSNR value by 25 dB can increased by an average of 0,91 dB and have total time elapsed about 187,058 second slowest than the compression process. At trial data PSNR value by 35 dB can increased by an average of 0,57 dB and have total time elapsed about 127,418 second fastest than the compression process. Than, this research can showed that the image restoration using iterative lanczos hybrid regularization method can increased image quality.

Keywords: *Compression, SPIHT, Restoration, Iterative Lanczos Hybrid Regularization.*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi telah semakin maju, baik dalam bidang pengembangan *hardware* maupun *software*. Banyak perusahaan asing maupun dalam negeri yang memproduksi dan menjual produk elektronik berbasis multimedia seperti *smartphone*, *notebook*, *tablet*, televisi, kamera, dan lain sebagainya. Perkembangan industri multimedia ini tidak terlepas dari citra sebagai salah satu parameter penting dalam kualitas teknologi multimedia. Selain itu, kebutuhan akan citra dimanfaatkan dalam bidang medis sebagai media pembantu diagnosa penyakit, peningkatan citra satelit, pengenalan pola, sistem keamanan berbasis citra, kompresi citra, restorasi citra dan media pembelajaran visual yang efektif dalam bidang pendidikan. Didalam penggunakaan teknologi sering dilakukan proses pengiriman citra melalui berbagai aplikasi. Saat proses pengiriman citra tentunya akan mengalami penurunan kualitas citra akibat proses kompresi citra yang dilakukan sistem. Tujuan kompresi citra sendiri untuk efektifitas pengiriman citra dengan ukuran file yang kecil dengan kualitas citra yang bagus. Tanpa disadari citra juga akan kehilangan informasi yang diakibatkan saat proses pengiriman citra.

Kompresi citra merupakan proses pemampatan citra dalam permasalahan redudansi data pada citra dengan melakukan pengurangan jumlah data yang berlebih [5]. Diperlukan teknik kompresi yang mampu mencapai rasio tinggi dengan kualitas citra yang baik, efisien dan memiliki kecepatan, salah satunya algoritma kompresi SPIHT. Algoritma kompresi SPIHT merupakan algoritma yang mengkodekan koefisien hasil transformasi wavelet secara bertahap yang diusulkan oleh Pearlman dan Said tahun 1996

[6]. Terbukti bahwa metode kompresi SPIHT lebih baik daripada kompresi JPEG. Untuk penyempurnaan citra yang terkompresi dan terdapat derau, *blurring* atau *noise* akibat proses yang terjadi, maka diperlukan restorasi citra. Restorasi citra merupakan proses merekonstruksi atau mendapatkan kembali citra asli dari sebuah citra yang terdegradasi agar dapat menyerupai citra asli dengan pengamatan secara objektif [4]. Dalam proses restorasi citra, sudah banyak teknik yang digunakan dan mampu menghasilkan citra yang baik. Metode Iteratif Lanczos – Hybrid Regularization merupakan metode yang sesuai untuk mengurangi *blurring* pada citra akibat kompresi yang teredudansi data. Metode ini menggunakan parameter regulasi *Weighted-GCV* dan iteratif lanczos bidiagonalization, untuk menyelesaikan permasalahan *blurring* dan *noise* [1]. Dengan penyelesaikan secara iteratif menunjukkan hasil yang lebih baik daripada metode Lucy R, Wiener Filter, dan Reg Filter pada penelitian terdahulu [4]. Berdasarkan latar belakang diatas maka dalam penelitian tugas akhir ini proses kompresi citra dan restorasi citra akan dimanfaatkan untuk mengembalikan citra yang mengalami redudansi data dan gangguan *noise* pada citra citra, sehingga mendapatkan citra yang baik.

Penelitian sebelumnya terkait dengan tugas akhir ini adalah penilitian yang dilakukan oleh Ahmad Saikhu, Mediana Aryuni dan Fitri Rullianti tahun 2009 yang berjudul ”*Implementasi Metoda Kompresi Set Partitioning In Hierarchical Trees Berbasis Wavelet Pada Citra*” melakukan kompresi citra berdasarkan kriteria penilain secara obyektif dan subyektif dengan menghasilkan kompresi SPIHT lebih baik daripada kompresi JPEG [7]. Secara kualitas pada metode SPIHT masih terdapat banyak derau atau *noise* sehingga diperlukan perbaikan citra untuk meningkatkan kualitas citra. Selain itu penelitian tentang kompresi SPIHT, sudah dilakukan oleh Amir Said dan

William A. Pearlman yang menggunakan dasar Transformasi Wavelet Diskrit [6]. Untuk proses restorasi citra sebelumnya sudah diteliti oleh Yudhi Purwananto, Rully Soelaiman dan Alfa Masjita Rahmat tahun 2010 yang berjudul "*Restorasi Citra Dengan Menggunakan Metode Iteratif Lanczos – Hybrid Regularization*" dengan menguji coba pengembalian kualitas citra menggunakan beberapa perbedaan dalam prosesnya dengan melakukan variasi besar nilai PSNR, variasi parameter sigma untuk peningkatan *blurring* atau *noise*, variasi ukuran citra, serta menggunakan pemberhentian kriteria untuk mendapatkan citra yang baik [4].

Berdasarkan hasil latar belakang penulis menggunakan kompresi SPIHT sebagai bahan uji coba citra. Hasil kompresi SPIHT pada penelitian sebelumnya masih kurang sempurna dikarenakan nilai PSNR citra pada *range* 20 dB – 35 dB tergolong citra yang mengandung derau atau *noise.* Dengan proses *encode* citra terkompresi setelah itu dilakukan kuantisasi untuk mendapatkan nilai matriks citra terkompresi. Untuk memunculkan citra terkompresi dilakukan *decode* agar matriks yang tersimpan dapat terlihat dalam bentuk citra. Citra yang sudah terkompresi menghasilkan citra dengan nilai PSNR 25 dB dengan banyak derau dan 35 dB dengan sedikit derau. Untuk mengurangi intensitas derau tersebut dilakukan proses restorasi citra dengan menggunakan metode *Iteratif Lanczos Hybrid Regularization.*

1.2 Rumusan Masalah

Rumusan masalah dari Tugas Akhir ini adalah :

1. Bagaimana implementasi kompresi citra menggunakan metode *Set Partitioning In Hierarchical Trees (SPIHT)*?
2. Bagaimana implementasi restorasi citra menggunakan metode *Iteratif Lanczos-Hybrid Regulazarition*?

1.3 Batasan Masalah

Batasan Masalah yang akan dibahas dalam penelitian tugas akhir ini adalah sebagai berikut :

1. Menggunakan file citra *grayscale* dengan format .**bmp*
2. *Software* yang digunakan dalam implementasi penelitian ini adalah MATLAB

1.4 Tujuan

Tujuan dari tugas akhir ini adalah :

1. Mengetaui kualitas citra pada proses kompresi citra dengan menggunakan metode *Set Partitioning In Hierarchical Trees (SPIHT).*
2. Mengetahui kualitas citra hasil kompresi SPIHT menggunakan metode restorasi citra *Iteratif Lanczos-Hybrid Regularization.*

1.5 Manfaat

Manfaat yang diperoleh dari Tugas Akhir ini adalah :

1. Sistem yang telah dibangun dapat memberikan informasi tentang kualitas kompresi citra SPIHT dan restorasi citra sehingga dapat dijadikan bahan pertimbangan bagi perusahaan pengembangan *software* terkait pengolahan citra digital.
2. Memberikan kontribusi bagi dunia penelitian khususnya dalam pengembangan perangkat lunak sistem kompresi dan restorasi.

1.6 Sistematika Penulisan Tugas Akhir

Sistematika penulisan didalam Tugas Akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang pembuatan tugas akhir, rumusan dan batasan permasalahan yang dihadapi dalam penelitian tugas akhir, tujuan dan

manfaat pembuatan tugas akhir dan sistematika penulisan tugas akhir.

BAB II KAJIAN TEORI

Bab ini menjelaskan tentang penelitian sebelumnya yang mengkaji metode kompresi citra *Set Partitioning In Hierarchical Tress*, restorasi citra *Iterative Lanczos Hybrid Regularization*, serta transformasi wavelet. Selain itu, pada bab ini akan dijelaskan kajian teori dari referensi penunjang serta penjelasan permasalahan yang dibahas dalam tugas akhir ini, meliputi Pengertian Citra Digital, Citra *grayscale*, jenis kompresi citra, kuantisasi serta kriteria penilaian kualitas citra.

BAB III METODOLOGI PENELITIAN

Bab ini berisi metodologi atau urutan pengerjaan yang dilakukan dalam menyelesaikan tugas akhir, meliputi studi literatur, pengumpulan data, analisis dan desain sistem, pembuatan program, uji coba dan evaluasi, hingga penulisan tugas akhir.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini menjelaskan analisis citra, perancangan gambaran umum sistem, perancangan proses algoritma, perancangan data, dan perancangan antar muka sistem. Sistem ini memiliki inputan berupa citra gambar *grayscale* yang selanjutnya dilakukan proses kompresi citra dengan metode SPIHT. Hasil dari proses kompresi ini berupa citra yang memiliki nilai PSNR. Hasil dari proses tersebut akan dilakukan restorasi citra dengan metode *iteratif lanczos hybrid regularization* untuk menguji citra.

BAB V PENGUJIAN DAN PEMBAHASAN HASIL

Bab ini akan menampilkan hasil uji coba, nilai *peak signal to noise ratio* (PSNR), waktu kompresi, waktu restorasi dan visual citra. Hasil pengujian inilah yang akan digunakan dalam perumusan kesimpulan dan saran.

BAB VI PENUTUP

Bab ini merupakan penutup, berisi tentang kesimpulan yang dapat diambil berdasarkan data yang ada dan saran yang selayaknya dilakukan bila tugas akhir ini dilanjutkan.

BAB II
TINJAUAN PUSTAKA

Pada bab ini menjelaskan tentang kajian teori dari referensi penunjang serta penjelasan permasalahan yang dibahas dalam tugas akhir ini, meliputi Penelitian sebelumnya terkait Tugas Akhir ini, Pengertian Citra Digital, jenis kompresi citra, Transformasi Wavelet Diskrit, kompresi citra *Set Partitioning in Hierarchical Trees* (SPIHT), Kriteria Penilaian Kualitas Citra untuk pengukuran kualitas citra terkompresi, dan Restorasi citra *Iteratif Lanczos – Hybrid Regularization.*

Penelitian Sebelumnya

Penelitian terkait tugas akhir ini tentang kompresi citra dengan metode SPIHT dilakukan oleh Amir Said dan William A. Pearlman yang berjudul *A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Tress*. Dalam penelitian ini, metode SPIHT digunakan untuk melakukan kompresi terhadap citra lena *grayscale* dengan ukuran 512x512 pixels. Berdasarkan penelitian tersebut dihasilkan nilai PSNR dengan tiga kali percobaan menghasilkan 37,2 dB, dengan laju kompresi (*rate*) 0,5 bpp, 34,1 dB dengan laju kompresi (*rate*) 0,25 bpp, dan 31,9 dB, dengan laju kompresi (*rate*) 0,15 bpp. Tetapi hasil citra tersebut masih banyak derau yang tampak pada citra. Pada percobaan dengan laju kompresi yang lebih kecil terdapat banyak derau yang muncul pada citra [6].

Penelitian terkait metode restorasi citra pada tugas akhir ini dilakukan oleh Purwananto dkk. Pada penelitian yang berjudul *Restorasi Citra dengan Menggunakan Metode Iteratif Lanczos-Hybrid Regularization* yang menggunakan *additive white Gaussian noise* pada citra untuk bahan uji coba agar citra dapat direstorasi. Hasil penelitian ini menunjukkan bahwa metode tersebut jauh lebih baik dari teknik restorasi yang telah ada sebelumnya dengan beberapa uji coba berdasarkan variasi

parameter sigma pada Gaussian PSF, variasi nilai PSNR dan variasi ukuran citra (*pixels*) [4].

Pada sub bab selanjutnya akan dijelaskan dasar-dasar teori kompresi citra dan restorasi citra.

2.1 Pengertian Citra

Citra menurut Gonzales memiliki makna rupa, gambar, atau gambaran. Sedangkan menurut kamus Webster citra adalah suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda. Citra terbagi menjadi dua yaitu citra diam dan citra bergerak. Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan, citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan pada mata kita sebagai gambar yang bergerak.

Dalam beberapa masa, citra yang dikenal manusia berbentuk citra kontinu. Suatu representasi objek yang dihasilkan dari sistem optik yang menerima sinyal analog dan dinyatakan dalam bidang dua dimensi. Nilai cahaya yang ditransmisikan pada citra kontinu memiliki rentang nilai yang tak terbatas. Contoh dari citra kontinu adalah mata manusia dan kamera analog.

Sebuah citra analog tidak dapat direpresentasikan secara langsung oleh komputer. Oleh sebab itu dilakukan sebuah proses untuk merubah nilai-nilai yang ada pada citra analog agar komputer dapat membaca dan menerjemahkan informasi yang terdapat pada citra analog. Hasil dari pemrosesan tersebut dinamakan sebagai citra digital.

2.2 Citra Digital

Citra digital merupakan fungsi dua dimensi yang dapat dinyatakan dengan fungsi $f(x,y)$, dimana x dan y merupakan titik koordinat spasial. Dan amplitudo dari fungsi f pada sembarang koordinat (x,y) merupakan nilai intensitas cahaya, yang merupakan representasi dari warna cahaya yang ada pada citra analog. Citra digital adalah suatu citra dimana (x,y) dan nilai

intensitas dari *f* terbatas (*discrete quantities*), dan telah dilakukan proses digitalisasi spasial dan digitalisisi kuantitas [5].

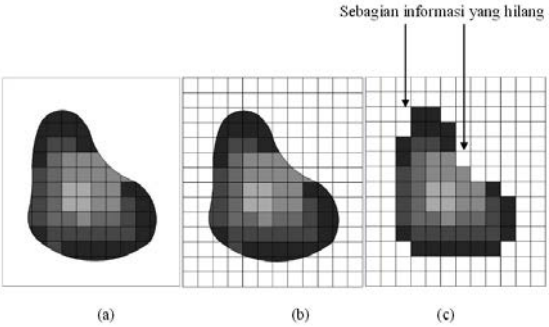
2.2.1 Digitalisasi Spasial (*Sampling*)

Sampling merupakan proses pengambilan informasi dari citra analog yang memiliki panjang dan lebar tertentu untuk membaginya ke beberapa blok kecil. Blok-blok tersebut disebut sebagai piksel. Sehingga citra digital yang lazim dinyatakan dalam bentuk matriks memiliki ukuran N x N dengan N sebagai baris dan N kolom. Bisa juga disebut sebagai citra digital yang memiliki N x N buah piksel. Notasi matriks citra digital dapat dinyatakan sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0, & N-1) \\ \vdots & \ddots & & \vdots \\ f(N-1,0) & \cdots & f(N-1,N-1) \end{bmatrix} \qquad (2.1)$$

2.2.2 Digitalisasi Intensitas (Kuantisasi)

Kuantisasi adalah proses pemberian nilai derajat keabuan di setiap titik piksel yang merupakan representasi dari warna asli dari citra analog dengan Rentang nilai keabuan adalah 0 – 255.



Gambar 2.1. Proses *sampling* dan kuantisasi. a) Citra Digital, (b) Citra Digital Disampling Menjadi 14 Baris dan 12 Kolom, (c) Citra Digital Hasil Sampling Berukuran 14 x 12 Piksel[5].

2.3 Citra Berwarna dan Citra Grayscale

Citra berwarna atau citra RGB *(Red- Green-Blue*) merupakan warna dasar yang dapat diterima oleh mata manusia. Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari ketiga warna dasar RGB. Setiap titik pada citra warna membutuhkan data sebesar 3 byte. Setiap warna dasar memiliki intensitas tersendiri dengan nilai minimum nol (0) dan nilai maksimum 255 (8 bit). RGB didasarkan pada teori bahwa mata manusia peka terhadap panjang gelombang 630nm (merah), 530 nm (hijau), dan 450 nm (biru).

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, artinya nilai dari *Red = Green = Blue*. Nilai-nilai tersebut digunakan untuk menunjukkan intensitas warna. Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra *grayscale* berbeda dengan citra ”hitam-putih”, dimana pada konteks komputer, citra hitam putih hanya terdiri atas 2 warna saja yaitu ”hitam” dan ”putih” saja. Pada citra grayscale warna bervariasi antara hitam dan putih, tetapi variasi warna diantaranya sangat banyak. Citra *grayscale* seringkali merupakan perhitungan dari intensitas cahaya pada setiap piksel pada spektrum elektromagnetik *single band.* Citra *grayscale* disimpan dalam format 8 bit untuk setiap sample piksel, yang memungkinkan sebanyak 256 intensitas.

Pada dasarnya resolusi citra secara matematis direpresentasikan dengan matriks dimana setiap nilai elemen matriksnya merupakan nilai dari intensitas warna baik itu dalam RGB atau dalam Grayscale.

2.4 Kompresi Citra

Ada dua macam teknik dalam kompresi citra, antara lain adalah :

2.3.1. Kompresi Lossy

Teknik kompresi lossy dapat mengubah warna secara detail pada file citra menjadi lebih sederhana tanpa terlihat perbedaan yang mencolok dalam pandangan secara langsung, sehingga ukurannya menjadi lebih kecil. Ukuran file citra menjadi lebih kecil dengan menghilangkan beberapa informasi dalam citra asli. Biasanya digunakan pada citra foto atau gambar lainnya yang tidak terlalu memerlukan pengamatan secara rinci pada citra, dimana kehilangan bit rate foto tidak berpengaruh pada citra.

2.3.2. Kompresi Loseless

Teknik kompresi losesless merupakan kompresi citra dimmana tidak ada satupun informasi citra yang dihilangkan. Biasanya digukanan pada citra medis dengan beberap metode diantaranya Run Length Encoding, Entropy Encoding dan Adaptive Dictionary Based.

2.5 Transformasi Wavelet Diskrit

Wavelet merupakan sebuah basis. Basis wavelet berassal dari fungsi penskalaan atau dikatakan juga sebagai fungsi scaling function. Menurut Gonzalez menjelaskan bahwa *scaling function* memiliki sifat yaitu dapat disusun dari sejumlah salinan dirinya yang telah ditranslasikan dan diskalakan [5]. *Scaling function* diturunkan dari persamaan dilasi, yang dianggap sebagai dasar dari teori wavelet. *Scaling function* dinyatakan dalam persamaan (2.2)

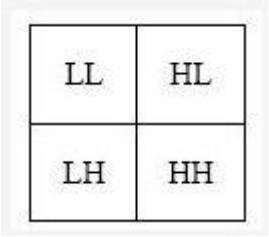
$$\varphi_{j,k}(x) = 2^{j/2}\varphi\left(2^{j}x - k\right) \qquad (2.2)$$

Dari persamaan *scaling function* dapat dibentuk persamaan wavelet yang pertaman atau disebut sebagai *mother wavelet*, dinyatakan dalam persamaan (2.3)

$$\psi_{j,k}(x) = 2^{j/2}\psi\left(2^{j}x - k\right) \qquad (2.3)$$

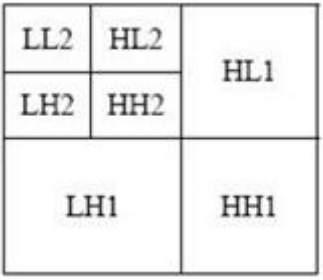
Transformasi wavelet dua dimensi mentransformasikan matrik citra terhadap garis, dilanjutkan dengan transformasi terhadap kolom. Sinyal dilewatkan pada rangkaian *filter high-pass* dan *low-pass*. Setengah dari masing-masing keluaran diambil sampel melauli operasi *sub-sampling*. Keluaran dari *filter low-pass* digunakan sebagai masukan dalam proses dekomposisi tingkat berikutnya. Proses 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. Koefisien wavelet berisi informasi sinyal hasil trnasformasi yang telah terkompresi.

Pada proses dekomposisi, citra terbagi menjadi empat buah *subband* dan dilakukan *downsampling*. Tiap koefisien merepresentasikan sebuah area spasial 2x2 piksel citra asli. *subband* pada dekomposisi citra satu tingkat direpresentasikan dalam LL, LH, HL dan HH.



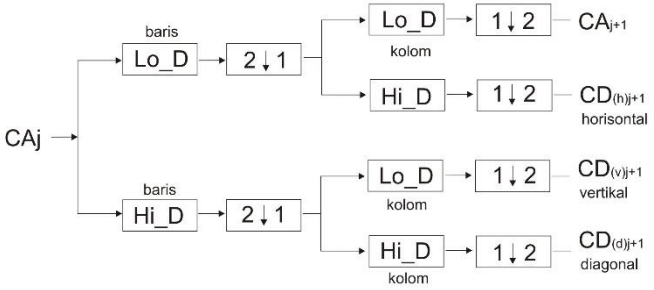
Gambar 2.2 Pemetaan Koefisien Pada Dekomposisi Citra Satu Tingkat

Bagian LL disebut sebagai bagian aproximasi (A), bagian LH disebut detail vertical (V), bagian HL disebut detail horizontal (H) dan bagian HL disebut detail diagonal (D). Untuk memperoleh skala yang lebih kasar, dilakukan proses dekomposisi terhadap *subband* LL. Proses dekomposisi dilakukan sampai pada skala tertentu. Padas skala yang lebih kasar, tiap koefisien merepresentasikan frekeunsi yang semakin sempit.



Gambar 2.3 Pemetaan Koefisien Dekomposisi Citra Dua Tingkat

Pada gambar 2.2 LH1, HL1 dan HH1 merupakan hasil dekomposisi level 1, LL1 didekomposisi lagi menjadi LL2, LH2, HL2 dan HH2. Banyaknya bagian citra hasil transformasi pada transformasi *wavelet*, ditentukan oleh level dekomposisi. Semakin besar level dekomposisi, jumlah bagian citra akan semakin banyak jika dibandingkan dengan level dekomposisi yang lebih kecil.

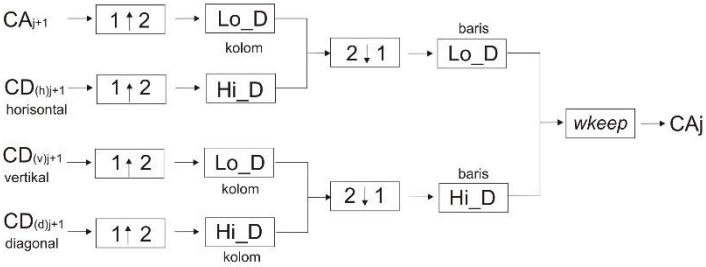


Gambar 2.4 Transformasi Wavelet Diskrit Dua Dimensi

Tiap satu tingkat proses dekomposisi, cittra dibagi menjadi empat bagian dengan resolusi yang sama. Besar pembagian ditung setengah dari resolusi citra sebelumnya. Pada gambar 2.3 citra di filter sepanjang summbu x mengggunakan *low-pass filter* H(x) dan *high-pass filter* G(x). Hasil pemfilteran di-*downsampling* factor 2, terhadap kedua keluaran *filter*. Ukuran

citra menjadi setengahnya pada sumbu x. Cara yang sama dilakukan pada sumbu y. Kedua citra difilter sepanjang sumbu y, menggunakan *low-pass filter* H(y) dan *high-pass filter* G(y). *Subband* hasil pemfilteran H(y) dan H(y) yang telah dilakukan *downsampling*, merepresentasikan bagian kasar dari citra asal. *Subband* tersebut disebut dengan komponen *lowpass* residu.

Proses dekomposisi suatu citra dilakukan dengan cara membalik tahap pemmfilteran dan melakukan proses *upsampling. Invers* DWT digunakan untuk membangun kembali sinyal hasil menjadil sinyal asli, tetapi informasi yang ada tidak ada yang hilang. Pada gambar 2.5, anak panah menunjukkan *upsampling/sampling* ke atas.



Gambar 2.5 *Invers* Transformasi Wavelet Diskrit Dua Dimensi

2.6 Algoritma SPIHT

Amir Said dan William A. Pearlman melakukan penelitian metode kompresi citra dengan menggunakan algoritma SPIHT dengan cara mengkodekan koefisien hasil transformasi wavelet dari citra secara efisien dan citra hasil rekonstruksi dapat diperoleh dengan kualitas yang berbeda tergantung dari jumlah *bit* yang diterima [6]. Kompresi SPIHT juga merupakan metode pengembangan dari algoritma EZW (*Embedded Zero Wavelet*) yang sudah diteliti oleh J. Saphiro pada penelitian sebelumny[3].

2.6.1. Algoritma *Set Partitioning*

Dalam algoritma SPIHT tidak diperlukan proses pengurutan pada semua koefisien. Algoritma yang sederhana akan memilih koefisien-koefisien $c_{i,j}$ pada interval $2^n \leq |c_{i,j}| < 2^{n+1}$, nilai n turun pada setiap proses. Jika $\leq |c_{i,j}| < 2^n$ maka koefisien tersebut signifikan dan jika $|c_{i,j}| < 2^n$ maka koefisien tidak signifikan.

Algoritma pengurutan membagi kelompok *pixel* menjadi *subset* T_m dan melakukan uji magnitude berdasarkan pertidaksamaan berikut :

$$\max_{(i,j)\epsilon \mathrm{T}_m} \{|c_{i,j}|\} \geq 2^n \qquad (2.4)$$

Jika dari uji *magnitude* diperoleh jawaban 'salah' maka semua koefisien dalam *subset* T_m tidak signifikan sedangkan jika dari uji magnitude tersebut diperoleh jawaban 'benar' maka *encoder* dan *decoder* akan membagi *subset* T_m menjadi *subset* baru $T_{m,l}$ kemudian uji magnitude kembali dilakukan pada *subset* yang baru. Pengelompokan dalam *subset* yang baru akan terus dilakukan sampai *subset* hanya memiliki satu komponen dalam satu koordinat.

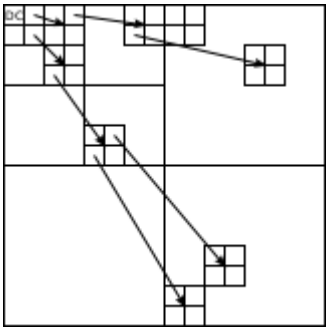
Untuk mengurangi banyaknya uji *magnitude* pada pertidaksmaan (2.4), maka didefinisikan aturan *set partitioning* secara hierarki berdasarkan metode pengkodean *subband* berhirarki. Hubungan antara uji magnitude dan pengiriman *bit* , dijelaskan dengan persamaan (2.5)

$$S_n(T) = \begin{cases} 1, & \max_{(i,j)\epsilon T}\{|c_{i,j}|\} \geq 2^n, \\ 0, & lain \end{cases} \qquad (2.5)$$

2.6.2. *Spatial Orientation Trees*

Sebuah struktur pohon yang disebut *Spatial Orientation Tree*, mendifinisikan hubungan spasial dari koefisien transformasi. *Direct Offspring* menunjukkan koefisien pada orientasi spasial yang sama pada skala yang lebih halus berikutnya. Tiap koefisien ada yang tidak mempunyai *offspring*,

tetapi ada yang mempunyai 4 buah *offspring* yang terdiri dari satu kelompok koefisien 2x2. Piksel pada level tertinggi, dikelompokkan dalam ukuran 2x2 piksel. Dimana masing masing mempunyai 4 piksel yang tidak memiliki *descendant,* ditunjukkan pada gambar 2.5



Gambar 2.6 Hubungan *Parent-Offspring* pada *Spatial Orientation Trees*

- O(i,j) : Kelompok koordinat semua *offspring* pada koordinat (i,j).
- D(i,j) : Kelompok koordinat semua *descendent* pada koordinat (i,j).
- H(i,j) : Kelompok koordinat semua akar *spatial orientation tree.*
- L(i,j) = D(i,j) - O(i,j). Semua *descendant* kecuali *offspring.*

2.6.3. Algoritma Pengkodean

Dalam algoritma SPIHT terdapat tida buah daftar urutan yaitu LIS, LIP, dan LSP. Masing masing daftar diidentifikasi dengan koordinat (i,j), dimana LIS merepresentasikan kelompok piksel D(i,j) atau L(i,j). Jika LIS merepresentasikan D(i,j) maka disebut LIS tipe A, sedangkan LIS merepresentasikan L(i,j) maka disebut LIS tipe B.

Keseluruhan algoritma, termasuk algoritma sebelumnya yang telah ditambahkan algoritma *set partitioning* adalah sebagai berikut :

1. Inisialisasi : kirm $n = [log_2(max_{(i,j)}\{|c_{i,j}|\})]$; set LSP sebagai daftar kosong, tambahkan koordinat $(i,j)\ \epsilon\ \mathcal{H}$ pada LIP yang memiliki *descendent* ke dalam LIS (tipe A).
2. *Sorting Pass* :
   A. Untuk setiap masukkan (i,j) dalam LIP, lakukan :
      - Kirim $S_n(i,j)$;
      - Jika $S_n(i,j) = 1$, maka pindahkan (i,j) ke LSP dan kirim tanda dari $c_{i,j}$.

   B. Untuk setiap masukkan (i,j) ke LIS, lakukan :
      - Jika masukkan adalah tipe A, maka
        - Kirim $S_n(\mathcal{D}(i,j))$;
        - Jika $S_n\big(\mathcal{D}(i,j)\big) = 1\ maka$
          - Untuk setiap $(k,l)\epsilon\ \mathcal{O}(i,j)$ maka
            - Kirim $S_n(k,l)$;
            - Jika $S_n(k,l) = 1$ maka tambahkan (k,l) ke LSP dan kirim tanda koefisien $c_{k,l}$;
            - Jika $S_n(k,l) = 0$ maka tambahkan (k,l) pada akhir LIP
          - Jika $\mathcal{L}(i,j) \neq \emptyset$ maka pindahkan (i,j) ke akhir LIS sebagai tipe B, dan kemudian lanjutkan langkah 2.2.2.; untuk kondisi yang lain maka hilangkan (i,j) dari LIS;
      - Jika entri bertipe B maka
        - Keluarkan $S_n(\mathcal{L}(i,j))$;
        - Jika $S_n\big(\mathcal{L}(i,j)\big) = 1$ maka
          - Tambahkan tiap $(k,l)\epsilon\ \mathcal{O}(i,j)$ diakhir LIS sebagai tipe A
          - Hilangkan (i,j)dari LIS
3. *Refinement Pass* : Untuk setiap masukkan (i,j) dalam LSP, kecuali didalam *sorting pass* terakhir. Kirim *most significant bit* ke-n dari $|c_{i,j}|$
4. *Quantization-Step Update* : Kurangi n dengan 1 dan lanjutkan ke Langkah 2.

2.7 Kriteria Obyektif

Pada kriteria obyektif, penilaian kualitas citra berdasarkan perubahan nilai tiap *pixel* pada citra rekonstruksi terhadap citra asli dengan menggunakan persamaan matematis tertentu. Dalam kompresi citra terdapat suatu standar pengukuran *error*/galat kompresi yaitu :

1. *Mean Square Error* (MSE) yaitu sigma dari jumlah *error* antara citra hasil kompresi dan citra asli. MSE direpresentasikan dalam persamaan berikut ini :

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}[\, I(x,y) - I'(x,y)]^2 \qquad (2.6)$$

- $I(x,y)$ adalah nilai piksel dicitra asli
- $I'(x,y)$ adalah nilai piksel pada citra hasil kompresi
- MN adalah dimensi citra

2. *Peak Signal to Noise Ratio* (PSNR) pada citra yang setiap *pixel*-nya dikodekan dalam cakupan nilai 0 sampai 255, persamaan kualitas citra dapat dinyatakan dalam persamaan berikut ini :

$$PSNR = 20log_{10}\frac{255}{\sqrt{MSE}} \qquad (2.7)$$

Penilaian kualitas pada citra hasil dianggap baik jika nilai perhitungan MSE yang dihasilakn rendah, sedangkan perhitungan PSNR menghasilkan nilai yang tertinggi.

Berdasarkan nilai PSNR yang dijelaskan pada buku Gonzales [5] terdapat beberapa kriteria kualitas PSNR dengan beberapa *rating* (tingkatan). Berikut tabel (2.1) berisi kriteria nilai PSNR :

Tabel 2.1 Nilai PSNR dengan Kriteria Kualitas Citra

No	Rating	Nilai PSNR	Kualitas
1	Excellent	60 dB	Kualitas sangat bagus
2	Fine	50 dB	Tanpa derau
3	Passable	40 dB	Terdapat butiran halus tapi kualitas citra masih bagus
4	Marginal	30 dB	Terdapat butiran halus didalam citra
5	Inferior	20 dB	Terdapat banyak derau
6	Unusable	10 dB	Citra tidak dapat dilihat (buruk)

2.8 Metode Iteratif Lanczos-Hybrid Regularization

Permasalahan sistem linier *inverse large-scale* biasanya ditulis dalam persamaan berikut ini :

$$b = Ax + \varepsilon \tag{2.8}$$

Untuk nilai $A \in \mathbb{R}^{mxn}, b \in \mathbb{R}^{m}, \; dan \; x \in \mathbb{R}^{n}$. Vektor $\varepsilon \in \mathbb{R}^{n}$ merepresentasikan gangguan pada data (*unknown pertubartion*) yang menyebabkan kehilangan informasi pada citra. Bentuk *Inverse problem* sering muncul dalam beberapa aplikasi diantaranya rekonstruksi, *image deblurring*, dan sebagainya. Biasanya masalah *ill-posed*, memberikan pengaruh *error* yang signifkan pada proses perhitungan aproksimasi x . Sehingga dibutuhkan bentuk regularisasi, untuk menstabilkan nilai aproksimasi x . Teknik regularisasi yang digunakan adalah Regularisasi Tikhonov [1], untuk menyelesaikan permasalahan *least square* sebagai berikut ini :

$$\min_{x} \left\| \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \lambda L \end{bmatrix} x \right\|_2 \tag{2.9}$$

Untuk L merupakan operator regularisasi, yang sering disebut sebagai matriks identitas atau diskritisasi dari operator differensiasi. λ merupakan parameter regularisasi dalam bentuk

scalar, biasanya berada diantara nilai $\sigma_n \leq \lambda \leq \sigma_1$, dimana nilai σ_n singular terkecil dari matriks A dan σ_1 nilai singular terbsesar dari matriks A.

Tidak ada metode regularisasi yang efektif tanpa pemilihan parameter yang tepat. Pada permasalahan ini digunakan metode pemilihan parameter GCV. Teknik regularisasi Tikhonov membutuhkan perhitungan *Singular Value Decomposition* (SVD) dari matriks A untuk menyelesaikan permasalahan *large scale*.

Iterasi regularisasi adalah sebuah alternatif regularisasi Tikhonov untuk menyelesaikan permasalahan *large-scale* sebagai berikut ini :

$$\min_x \|Ax - b\|_2 \tag{2.10}$$

Perilaku semikonvergen dari LSQR dapat distabilkan dengan menggunakan metode *hybrid* yang menggabungkan metode lanczos bidiagonalization dengan regularisasi Tikhonov dan pemilihan parameter GCV. Pendekatan yang digunakan untuk menyelesaikan *projected large-scale problem* adalah Krylov Subspace.

2.8.1 Regularisasi Tikhonov dan GCV

Teknik regularisasi Tikhonov membutuhkan minimalisasi dalam proses pemecahan masalah. Untuk nilai L pada pertidaksamaan (2.9) menjadi matriks identitas. Misalkan $A = U \sum V^T$ menyatakan SVD dari matriks A dimana kolom u_i dari U dan v_i dari V, masing masing vektor singular kiri dab kanan dari matriks A dan $\sum = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$ adalah matriks diagonal yang berisi nilai-nilai singular dari matriks A, dengan $\sigma_1 \geq \sigma_1 \ldots \geq \sigma_n \geq 0$.

Penggantian matriks A dengan SVD dan melakukan manipulasi aljabar, sehingga diperoleh persamaan regularisasi Tikhonov sebagai berikut ini :

$$x_\lambda = \sum_{i=1}^{n} \phi_i \frac{u_i^T b}{\sigma_i} v_i \qquad (2.11)$$

Dimana $\phi_i = \frac{\sigma_i^2}{\sigma_i^2+\lambda^2} \epsilon\, [0,1]$ merupakan *filter-factor* Tikhonov. Dengan catatan memilih $\lambda = 0$ sesuai dengan $\phi_i = 1$ untuk semua i. Parameter regularisasi digunakan untuk mencari solusi yang menghasilkan nilai berkualitas.

Metode GCV merupakan metode statistik yang berbasis prediksi yang tidak membutuhkan estimasi prioritas dari *error norm.* Pilihan λ harus memprediksi nilai dari data yang hilang, jadi tidak sembarang mengamati data yang tersisa, maka metode GCV dapat memprediksi pengamatan data yang hilang dengan cukup baik. Setiap data b_j pada nilai λ dapat meminimalisasi prediksi *error*,yang diukur dengan persamaan GCV sebagai berikut ini :

$$G_{A,b}(\lambda) = \frac{n\left\|\left(I - AA_\lambda^\dagger\right)b\right\|_2^2}{\left[trace\left(I - AA_\lambda^\dagger\right)\right]^2} \qquad (2.12)$$

Dimana $A_\lambda^\dagger = (A^T A + \lambda^2 I)^{-1} A^T$ merupakan *pseudo inverse* dari $\begin{bmatrix} A \\ \lambda L \end{bmatrix}$, dan memberikan solusi regularisasi $x_\lambda = A_\lambda^\dagger b$.

Penggantian matriks A dengan SVD dapat ditulis dengan persamaan sebagai berikut ini :

$$G_{A,b}(\lambda) = \frac{n\left(\sum_{i=1}^{n}\left(\frac{\lambda^2 u_i^T b}{\sigma_i^2+\lambda^2}\right)^2 + \sum_{i=n+1}^{m}\left(u_i^T b\right)^2\right)}{\left((m-n)+\sum_{i=1}^{n}\frac{\lambda^2}{\sigma_i^2+\lambda^2}\right)^2} \quad (2.13)$$

2.8.2 Metode Lanczos-Hybrid

Metode lanczos-hybrid dengan menggunakan GCV dapat menentukan parameter regularisasi Tikhonov secara efektif, tetapi fungsi minimisasi mensyaratkan bahwa SVD dari matriks A dihitung. Metode hybrid dapat menjadi cara efektif untuk menstabilkan perilaku semikonvergen yang merupakan karakteristik dari metode iterasi seperti LSQR. Dengan menggunakan metode iterasi seperti Lanczso-Bidiagonalization (LBD) dalam kombinasi langsung dengan metode regularisasi Tikhonov dapat menyelesaikan permasalahan *ill-posed inverse problem.*

Mengingat matriks A dan vektor b, LBD adalah skema berulang yang menghitung dekomposisi $W^T AY = B$ dimana W dan Y matriks orthonormal, dan B matriks *lower* bidiagonal. Untuk iterasi k pada LBD menghitung kolom k dari Y dan B, dan $(k+1)$ kolom dari W. Secara khusus, pada iterasi k untuk $k = 1, \dots, N$, sehingga $m(k+1)$ dari matriks W_k, $n\ x\ k$ matriks Y_k, $n\ x\ 1$ vektor y_{k+1} dan $a(k+1)\ x\ k$ matriks bidiagonal B_k seperti persamaan berikut ini :

$$A^T W_k = Y_k B_k^T + \alpha_k + 1 y_k + 1 e_{k+1}^T \quad (2.14)$$

$$AY_k = W_k B_k \quad (2.15)$$

Dimana e_{k+1} menunjukkan kolom terkahir dari matriks identitas dari dimensi $(k+1)$ dan α_{k+1} akan menjadi $(k+1)$ sebagai

masukkan untuk B_{k+1}. Matriks W_k dan Y_k merupakan kolom orthonormal dan kolom pertama dari W_k adalah $b/\|b\|$.

Mengingat hubungan tersebut, mendekati masalah *least square* dengan *projected least square* sehingga diperoleh persamaan sebagai berikut ini :

$$\begin{aligned}\min_{x \in (Y_k)} \|Ax - b\|_2 &= \min_f \left\|W_k^T b - B_k f\right\|_2 \\ &= \min_f \|\beta e_1 - B_k f\|_2\end{aligned} \tag{2.16}$$

Dimana $\beta = \|b\|$ dan memiliki nilai aproksimasi $x_k = Y_k f$. Jadi setiap iterasi metode LBD membutuhkan pemecahan *least square problem* yang melibatkan matriks bidiagonal B_k.

Sebuah elemen penting dari LBD adalah nilai singular dari B_k untuk nilai terkecil k mendekati nilai terbesar dan terkecil nilai matriks A. Oleh karena itu, regularisasi harus digunakan untuk menguhitung persamaan berikut ini :

$$f_\lambda = \beta B_{k,\lambda}^{\dagger} e_1 \tag{2.17}$$

Pada bagian berikutnya dapat digambarkan seberaoa baik metode LBD dapat bekerja untuk regularisasi Tikhonov menggunakan fungsi GCV, sehingga menggunakan persamaan GCV sebagai berikut ini :

$$G_{B_k,\beta e_1}(\lambda) = \frac{k\left\|\left(I - BB_{k,\lambda}^{\dagger}\right)\beta e_1\right\|_2^2}{\left[trace\left(I - BB_{k,\lambda}^{\dagger}\right)\right]^2} \tag{2.18}$$

Untuk memilih parameter regularisasi pada setiap iterasi sesuai persamaan (2.16). Perhatikan bahwa SVD didefinisikan dengan $(k+1)x\ k$ dari matriks B_k sebagai berikut ini :

$$B_k = P_k \begin{bmatrix}\Delta_k \\ 0^T\end{bmatrix} Q_k^T \tag{2.19}$$

Sehingga $G_{B_k,\beta e_1}(\lambda)$ sesuai persamaan (2.18) dapat ditulis sebagai berikut ini :

$$G_{B_k,\beta e_1}(\lambda) = \frac{k\beta^2\left(\sum_{i=1}^{k}\left(\frac{\lambda^2}{\delta_i^2+\lambda^2}[P_k^T e_1]_i\right)^2 + \sum_{i=n+1}^{m}\left([P_k^T e_1]_{k+1}\right)^2\right)}{\left(1+\sum_{i=1}^{k}\frac{\lambda^2}{\delta_i^2+\lambda^2}\right)^2} \quad (2.20)$$

Dimana $\left[P_k^T e_1\right]_i$ merupakan komponen dari vektor $P_k^T e_1$ dan δ_i merupakan nilai singular terbesar dari martiks B_k.

2.8.3 Weighted-GCV

Pada metode GCV mengalami kesulitan pada *smoothing* pada iterasi Lanczos-Hybrid, sehingga menggunakan perbedaan bobot pada metode Weighted GCV.

Persamaan Tikhonov GCV sudah didefinisikan pada persamaan (2.12), sehingga dengan mempertimbangkan penambahan bobot diperoleh persamaan W-GCV sebagai berikut ini :

$$G_{A,b}(\omega,\lambda) = \frac{n\left\|\left(I-AA_\lambda^\dagger\right)b\right\|^2}{\left[trace\left(I-\omega AA_\lambda^\dagger\right)\right]^2} \quad (2.21)$$

Dengan memperhatikan ketergantungan persamaan (2.21) terdapat parameter baru yaitu nilai ω . Pemilihan $\omega = 1$ memberikan perubahan fungsi GCV standar sesuai persamaan (2.12). Untuk pemilihan nilai $\omega > 1$, maka hasilnya lebih halus, sedangkan untuk pemilihan nilai $\omega < 1$, maka hasilnya kurang halus.

Untuk persamaan WGCV yang digunakan pada metode Lanczos Bidiagonal didapatkan dari hasil modifikasi persamaan (2.18) dengan penambahan nilai parameter ω adalah sebagai berikut ini :

$$G_{B_k,\beta e_1}(\omega,\lambda) = \frac{k\left\|\left(I - BB_{k,\lambda}^{\dagger}\right)\beta e_1\right\|_2^2}{\left[trace\left(I - BB_{k,\lambda}^{\dagger}\right)\right]^2} = \frac{k\beta^2\left(\sum_{i=1}^{k}\left(\frac{\lambda^2}{\delta_i^2+\lambda^2}\left[P_k^T e_1\right]_i\right)^2 + \left(\left[P_k^T e_1\right]_{k+1}\right)^2\right)}{\left(1+\sum_{i=1}^{k}\frac{(1-\omega)\delta_i^2+\lambda^2}{\delta_i^2+\lambda^2}\right)^2} \quad (2.21)$$

Pemilihan nilai ω sangat berpengaruh pada keberhasilan metode Lanczos-Hybrid Regularisasi. Pada setiap iterasi dari metode Lanczos Hybrid, dapat menyelesaikan *projected* LS *problem* menggunakan Regularisasi Tikhonov.

Parameter regularisasi yang optimal pada iterasi ke-k disimbolkan dengan $\lambda_{k,opt}$, maka nilai k yang kecil akan memenuhi kondisi berikut:

$$0 \le \lambda_{k,opt} \le \sigma_{min}(B_k)$$

dimana σ_{min} menunjukkan nilai singular terkecil dari matriks. Jika pada iterasi k, diasumsikan $\lambda_{k,opt}$ diketahui, maka dapat mencari nilai ω dengan meminimalkan fungsi GCV yang berhubungan dengan nilai ω. Maka dapat menggunakan fungsi turunun sebagai berikut ini :

$$\frac{\partial}{\partial\lambda}[G(\omega,\lambda)]\Big|_{\lambda=\lambda_{k,opt}} = 0$$

Karena $\lambda_{k,opt}$ tidak diketahui maka, maka nilai $\sigma_{min}(B_k)$ akan mendekati nilai 0 akibat mengalami kondisi *ill-conditioning.*

Cara lain untuk mendapatkan nilai ω adalah dengan menggunakan pendekatakan adaptive berikut ini :

$$\omega_k = mean\{\omega_1, \omega_2, \cdots, \omega_k\}$$

Berdasarkan pendekatan adaptive, bagian yang digunakan masih bersifat *well-conditioned*. Namun, ada hal lain yang membatasi pencarian nilai ω yaitu solusinya bersifat *undersmooth* untuk nilai k yang semakin besar.

2.8.4 Penentuan Kriteria Pemberhentian pada Lanczos Bidiagonalization

Selanjutnya melakukan pendekatan dengan menentukan titik yang tepat untuk menghentikan iterasi. Pada penentuan kriteria pemberhentian akan dijelaskan pendekatan yang sama untuk Regularisasi Tikhonov. Dimulai dengan mendifinisikan perhitungan pada iterasi lanczos hybrid sebagai berikut ini :

$$x_k = Y_k f_{\lambda k} = Y_k\left((B_k^T B_k + \lambda_k^T I)\right)^{-1} B_k^T W_k^T b = A_\lambda^\dagger b \qquad (2.22)$$

Dengan menggunakan penyelesaian GCV untuk menentukan kriteria pemberhentian iterasi, k, untuk meminimalkan iterasi, maka menggunakan persamaan berikut ini :

$$\hat{G}(k) = \frac{n\left\|\left(I - AA_\lambda^\dagger\right)b\right\|_2^2}{\left[trace\left(I - \omega AA_\lambda^\dagger\right)\right]^2} \qquad (2.23)$$

Menggunakan persamaan (2.22) dan (2.15), dengan pembilang persamaan (2.23) maka didapat persamaan sebagai berikut ini :

$$n\left\|\left(I - \omega AA_\lambda^\dagger\right)b\right\|^2 = n\|(I - B_k(B_k^T B_k + \lambda_k^2 I)^{-1} B_k^T)\beta e_1\|^2 \quad (2.24)$$

Pada persamaan (2.24) matriks B_k diganti dengan SVD, sehingga didapat persamaan berikut ini :

$$n\left\|\left(I - AA_\lambda^\dagger\right)b\right\|_2^2 = n\beta^2 \left\| \begin{bmatrix} \frac{\lambda_k^2}{\delta_i^2 + \lambda_k^2} & & \\ & \frac{\lambda_k^2}{\delta_i^2 + \lambda_k^2} & \\ & & 1 \end{bmatrix} P_k^T e_1 \right\|_2^2$$

$$= n\beta^2 \left(\sum_{i=1}^{k} \left(\frac{\lambda_k^2}{\delta_i^2 + \lambda_k^2} [P_k^T e_1]_i \right)^2 + \left([P_k^T e_1]_{k+1} \right)^2 \right) \quad (2.25)$$

Demikian dengan penyebut persamaaan diapat ditulis sebagai berikut ini :

$$\left[trace\left(I - \omega AA_\lambda^\dagger\right) \right]^2 = \left((m-k) + \sum_{i=1}^{k} \frac{\lambda_k^2}{\delta_i^2 + \lambda_k^2} \right)^2 \quad (2.26)$$

Maka, dengan mensubstitusikan persamaan (2.25) dan (2.26) kedalam persamaan (2.23) diperoleh persamaan berikut ini :

$$\hat{G}(k) = \frac{n\beta^2 \left(\sum_{i=1}^{k} \left(\frac{\lambda_k^2}{\delta_i^2 + \lambda_k^2} [P_k^T e_1]_i \right)^2 + \left([P_k^T e_1]_{k+1} \right)^2 \right)}{\left((m-k) + \sum_{i=1}^{k} \frac{\lambda_k^2}{\delta_i^2 + \lambda_k^2} \right)^2} \quad (2.27)$$

Persamaan (2.27) digunakan untuk menentukan iterasi pemberhentian dalam proses implementasi metode iterative lanczos hybrid regularization.

Dalam kondisi ideal konvergensi metode lanczos hybrid dapat stabil sempurna, untuk λ_k konvergen sesuai nilai parameter regularisasi. Untuk $\hat{G}(k)$ memiliki nilai konvergen tetap. Oleh karena itu, untuk mengakhiri iterasi saat nilai mendekati angka 0, agar nilai tidak sama dengan 0 maka menggunakan pertidaksamaan berikut ini :

$$\left|\frac{\hat{G}(k+1) - \hat{G}(k)}{\hat{G}(1)}\right| < tol$$

Namun, kondisi pertidaksamaan diatas sangat jarang ditemui. Karena sebagian besar akan mengalami perilaku semikonvergen dari iterasi lanczos hybrid regularization. Perilaku tersebut tentunya juga berdampak pada penentuan parameter pada fungsi GCV. Oleh karena itu, pemberhentian iterasi dengan kriteria lain dapat menggunakan persamaan berikut ini :

$$k_0 = \arg\min_k \hat{G}(k)$$

BAB III
METODOLOGI

Bab ini membahas mengenai metodologi sistem yang digunakan untuk menyelesaikan tugas akhir. Pembahasan metodologi sistem diawali dengan penjelasan tentang objek penelitian, peralatan yang digunakan, dan tahap penelitian.

3.1 Objek Penelitian

Objek penelitian yang akan digunakan pada tugas akhir adalah citra *grayscale* diantaranya citra babbon, lena dan pepper.

3.2 Tahap Penelitian

Adapun tahap-tahap yang dilakukan dalam penyusunan Tugas Akhir ini adalah sebagai berikut :

1. Studi Literatur
   Pada tahap ini akan dikaji tentang kompresi citra dan restorasi citra yang diimplementasikan pada citra dengan jenis *grayscale*. Studi ini dilakukan dengan membaca jurnal metode kompresi citra *set partitioning in hierarchical trees*, metode restorasi citra *iteratif lanczos – hybrid regularization*, buku mengenai transformasi wavelet.

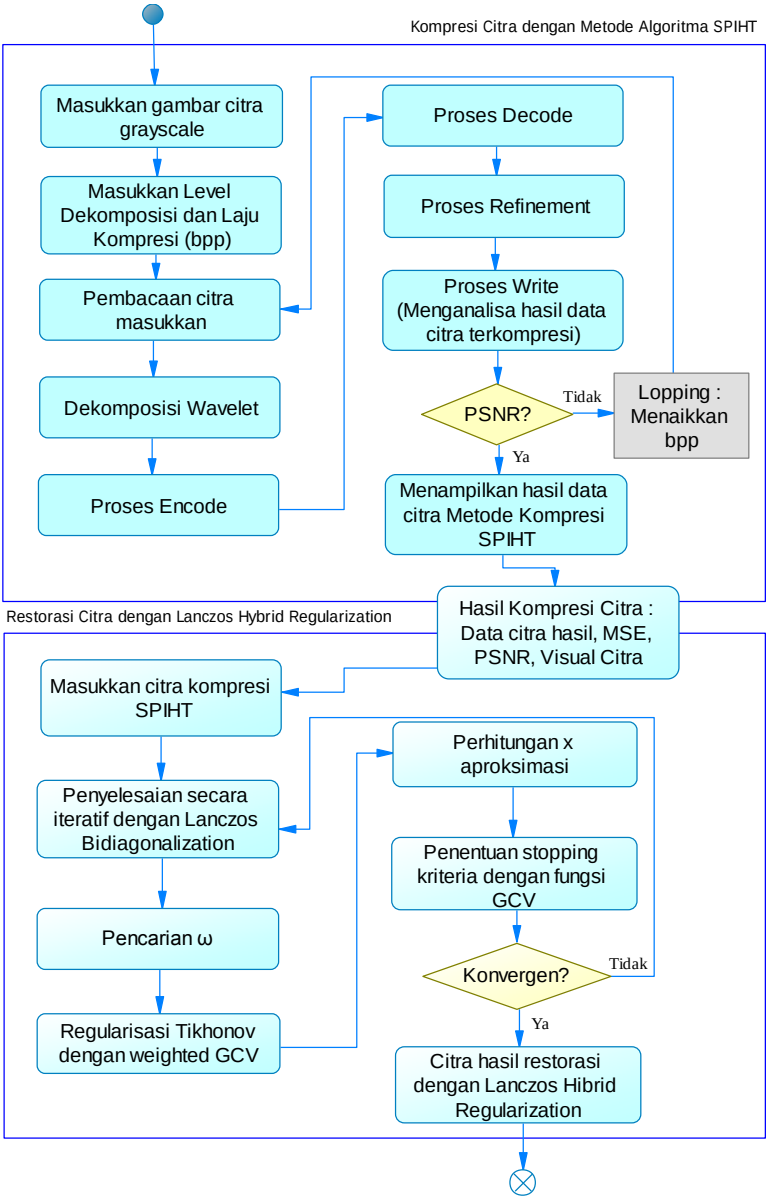
2. Analisis dan Perancangan Sistem
   Pada tahapan ini akan dilakukan implementasi citra *grayscale*, perancangan gambaran umum sistem, perancangan algoritma, perancangan data, dan perancangan antar muka sistem. Sistem ini memiliki inputan berupa citra *grayscale*. Citra tersebut dikompresi dengan menggunakan metode SPIHT. Hasil citra gambar tersebut diselesaikan menggunakan restorasi citra secara iteratif dengan *Lanczos Bidiagonalization.* Tahap selanjutnya dilakkukan penghitungan omega ω, regularisasi *Tikhonov* dengan *Weighted-GCV*, melakukan perhitungan aproksimasi x, serta penentuan *stopping criteria* dengan

fungsi GCV hingga diperoleh hasil yang konvergen. Diagram proses kompresi citra dan restorasi citra dapat dilihat pada gambar 3.1.

3. Pengujian dan Evaluasi Sistem
   Pada tahap ini akan dilakukan pengujian terhadap program yang telah dibangun. Pengujian ini juga akan dihitung nilai PSNR, waktu kompresi, waktu restorasi dan kenaikkan PSNR agar diperoleh nilai representasi tingkat keberhasilan program. Diagram uji dapat dilihat pada gambar 3.2.

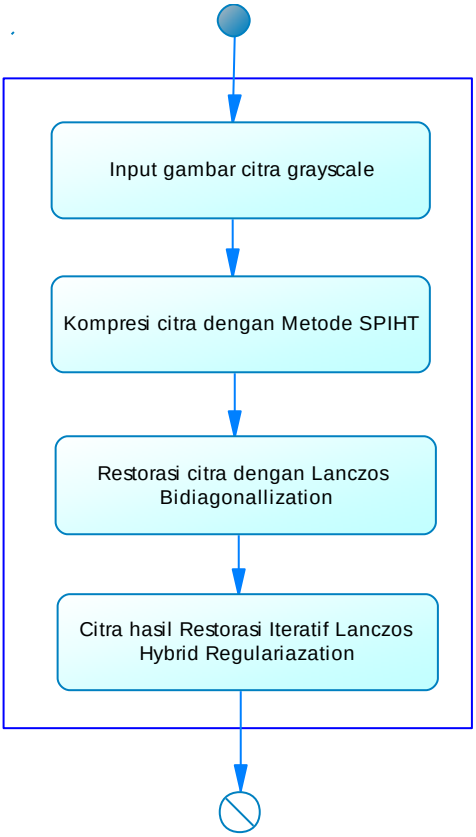
4. Penarikan Kesimpulan
   Tahap penarikan kesimpulan merupakan tahap akhir dalam proses penelitian Tugas Akhir ini, dimana pada tahap ini dilakukan penarikan kesimpulan terhadap hasil yang telah dicapai.

Tahap-tahap pengerjaan Tugas Akhir yang telah dijelaskan di atas digambarkan dalam diagram alir pada Gambar 3.1.



Gambar 3.1. Diagram Metodologi Penelitian

Berikut merupakan Diagram tahap Pengujian Kompresi dan Restorasi Citra :



Gambar 3.2. Diagram Proses Tahap Pengujian Kompresi dan Restorasi Citra

BAB IV
PERANCANGAN DAN IMPLEMENTASI SISTEM

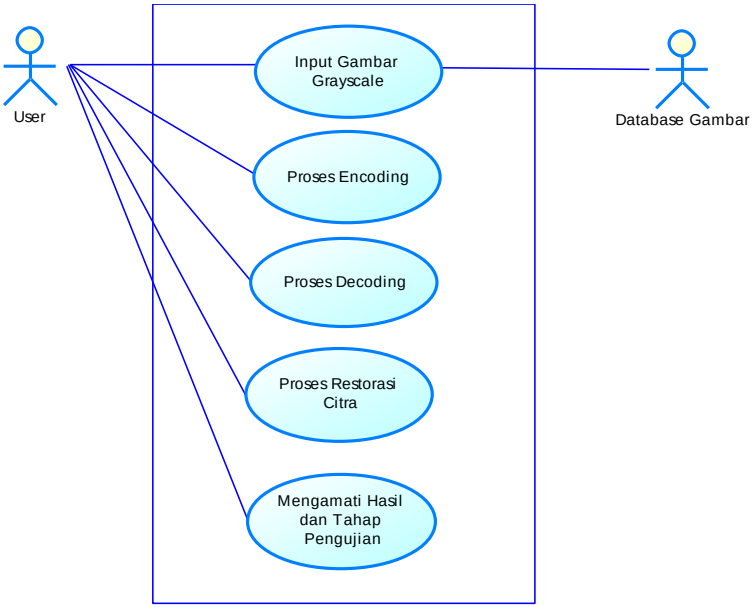
Pada bab ini akan dibahas mengenai perancangan dan implementasi sistem dimulai dari pembahasan proses input citra *grayscale*, kompresi citra menggunakan metode *Set Partitioning in Hierarchical Trees*, proses restorasi citra menggunakan metode *Iterative Lanczos – Hybrid Regularization*, serta penjelasan mengenai cara untuk mendapatkan data keluaran yang sesuai dengan tujuan dari penelitian Tugas Akhir ini.

4.1 Analisa Sistem

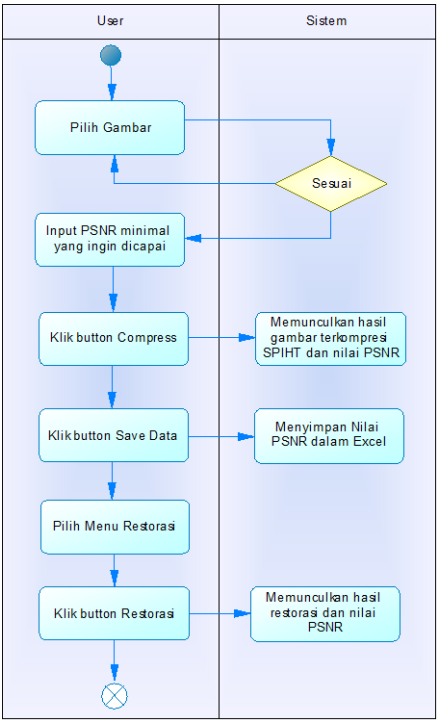
Dalam subab ini akan dijelaskan proses analisis dalam membangun perangkat lunak restorasi citra berbasis kompresi citra, mulai dari *Use Case Diagram*, *Activity Diagram*, proses kompresi citra menggunakan *Set Partitioning in Hierarchical Tress* yang meliputi proses *encoding* dan *decoding*, serta proses restorasi citra menggunakan *Iteratif Lanczos-Hybrid Regularization*, serta pengujian hasil dari sistem.

4.1.1 Analisis Sistem Perangkat Lunak

Perangkat lunak yang akan dibangun dapat membantu pihak-pihak dalam menganalisis citra dengan kompresi citra dan restorasi citra dalam memaksimalkan hasil citra. Misal analisis citra *grayscale* dengan proses kompresi yang menghasilkan citra bernoise disempurnakan dengan restorasi citra agar citra lebih memilliki kemirikan hasil yang lebih bagus dengan gambar aslinya. Secara umum *Use Case Diagram* dari sistem ini disajikan pada Gambar 4.1, *Swimlane Diagram* pada Gambar 4.2 menunjukkan jalannya perangkat lunak.



Gambar 4.1. *Use Case Diagram* Perangkat Lunak Restorasi Citra berbasis Kompresi Citra



Gambar 4.2. *Swimlane Diagram* Perangkat Lunak Restorasi Citra berbasis Kompresi Citra

Sistem perangkat lunak yang dibangun ini memiliki beberapa tahapan sebagai berikut :

a. Input citra *grayscale*

Data masukan berupa file citra dengan format *.bmp*. Citra yang menjadi bahan penelitian diantaranya, citra Lena, citra Babbon dan citra Pepper dengan kualitas *grayscale.*

Data *input* citra tersebut ditentukan kriterianya sebagai berikut :

- Citra memiliki resolusi 512 x 512 pixels, 320 x 320 pixels, 256 x 256 pixels dan 180 x 180 pixels (Batas resolusi maksimal program ini adalah citra berukuran 1024x1024 piksel)
- Citra berjenis *grayscale*

b. Proses *Encode*

Proses *encode* merupakan proses dimana dilakukan pengecekan terhadap jenis wavelet yang digunakan. Pada proses ini menggunakan wavelet *bior6.8*. Setelah itu proses *encode* menghasilkan citra yang disimpan dalam bentuk *bit stream* di *file temporary*. Pada proses *encode* memasukkan paremeter laju kompresi *(bpp)* dan level dekomposisi untuk penggunaan wavelet secara otomatis.

c. Penyimpanan Citra Hasil Encode

Hasil proses *encode* tersimpan didalam *file temporary* yang berhasil dilakukan proses dengan wavelet *bior6.8*. Selanjutnya hasil penyimpanan ini digunakan untuk masukan data pada proses *decode*.

d. Proses *Decode*

Proses *decode* mengambil data dari *file temporary* yang digunakan untuk menyimpan hasil *encode*. Setelah itu proses tersebut mengembalikan citra kedalam bentuk semula dengan beberapa proses menggunakan *invers discret wavelet transformation*. Selanjutnya citra keluar dengan menghasilkan data *Mean Square Error, Peak Noise to Ratio*, untuk digunakan sebagai parameter dalam proses restorasi citra.

e. Proses Restorasi Citra

Proses restorasi citra dengan menggunakan *Iteratif Lanczos-Hybrid Regularization* mengambil citra berdasarkan hasil kompresi citra. Untuk teknik Regulasi pada penelitian ini menggunakan teknik Tikhonov dengan metode pemilihan parameter Weighted GCV.

4.1.2 Analisis Kebutuhan Sistem

Perangkat lunak ini dibangun menggunakan software MATLAB R2010a baik dari desain interface, sistem *toolbox, computer vision* dan pengolahan video digital.

Tabel 4.1. Data Kebutuhan Sistem

Perangkat Keras	Prosesor	: AMD A10-5750M APU with Radeon ™ HD Graphics (4CPUs), ~2.5GHz
	RAM Memory	: 4 GB
Perangkat Lunak	Sistem Operasi	: Windows 8.1 Enterprise 64-bit (6.3, Build 9600)
	Tools	: MATLAB R2010a

4.2 Perancangan Sistem

Setelah analisis sistem, kemudian dilanjutkan dengan perancangan sistem. Perancangan sistem tersebut meliputi perancangan data sistem, perancangan *class* sistem, perancangan proses algoritma sistem, dan perancangan antarmuka sistem.

4.2.1 Gambaran Sistem

Pada subbab ini akan disajikan diagram alir sistem secara umum berupa proses berjalannya sistem sesuai alur pada gambar 3.1 pada bab III Metodologi yang secara umum terbagi menjadi 2 tahapan yaitu kompresi citra dengan metode algoritma *Set Partitioning in Hierarchical Tress* dan restorasi citra dengan metode *Iteratif Lanczos-Hybrid Regularization.*

4.2.2 Penjelasan Umum Sistem

Proses algoritma ini dimulai dengan pemilihan citra *grayscale* yang digunakan dalam proses kompresi citra. Setelah itu menentukan level dekomposisi citra dan laju kompresi (*bpp*), yang dilanjutkan dengan proses pembacaan file citra *.bmp* citra *grayscale*. Pada tahap dekomposisi dengan wavelet *bior6.8* didapatkan beberapa bagian matrik citra yaitu bagian

approximation, horizontal, *vertical* dan *diagonal*. Pada tahap encode inilah citra yang dihasilkan disimpan dalam *file temporaray* dalam bentuk bit stream. Sehingga pada proses decode citra dikembalikan bentuk gambar ke bentuk semula. Hasil dari kompresi SPIHT berupa gambar dengan berbagai macam PSNR dan gambar visual citra. Hasil tersebut yang nantinya direstorasi citra menggunakan *Iterative Lanczos-Hybrid Regularization* .

Pada sub bab selanjutnya akan dijelaskan algoritma tiap proses dalam metode yang digunakan.

4.2.3 Perancangan Algoritma

Perancangan proses algoritma ini secara garis besar di bagi menjadi dua tahapan, yaitu proses kompresi citra dan restorasi citra. Pada awalnya citra pada kompresi citra dilakukan proses *encode* dan *decode*. Selanjutnya gambar disimpan untuk proses restorasi dengan menggunakan teknik regulasi Tikhonov dan pemilihan parameter Weighted GCV.

4.2.3.1 Perancangan Kompresi Citra

Pada sub bab ini akan dijelaskPembaan mengenai desain sistem untuk implementasi metode SPIHT pada kompresi citra. Desain proses menjelaskan tentang proses dekomposisi, *encode, decode,* rekonstruksi citra dan menampilkan data keluaran.

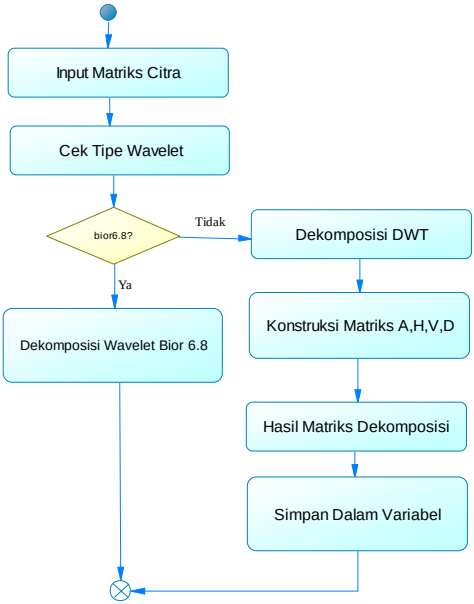
4.2.3.1.1 Pembacaan Citra Masukan

Gambar 4.3 menunjukkan proses pembacaan citra masukkan pada sistem kompresi. Tahap pertama yang dilakukan adalah identifikasi file dengan menghitung jumlah baris dan kolom pada citra. Tahap selanjutnya file dibaca dan disimpan dalam sebuah matrik. Format citra menggunakan .**bmp* dalam citra *grayscale*. File citra disimpan dalam *file temporary*.

Gambar 4.3. Diagram Alir Pembacaan Citra Masukan

4.2.3.1.2 Dekomposisi dengan Wavelet

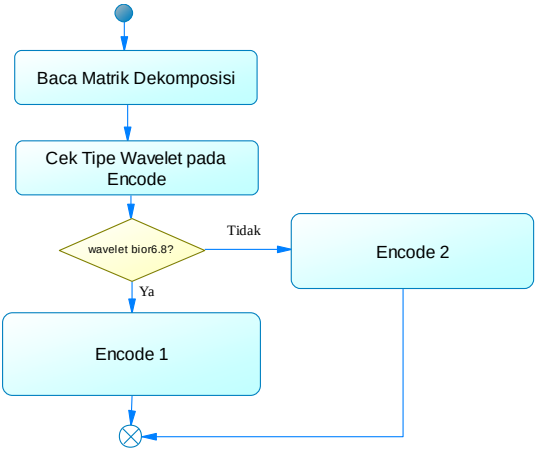
Tahap dekomposisi ditunjukkan pada gambar 4.4. Pada tahap dekomposisi digunakan beberapa tipe wavelet yang mempunyai karakteristik wavelet berbeda-beda. Pada tipe wavelet *bior6.8* diimplementasikan dalam fungsi tersendiri. Pemfilteran dilakukan terhadap baris, dilanjutkan terhadap kolom. Dekomposisi pada sinyal citra akan berhenti sesuai dengan level dekomposisi yang diberikan. Dari proses dekomposisi didapatkan beberapa bagian matrik yaitu bagian *aproksimasi*, horizontal, vertical dan diagonal. Semakin tinggi level dekomposisi, maka area spasial pada bagian-bagian tersebut semakin kecil.



Gambar 4.4. Diagram Alir Dekomposisi Wavelet

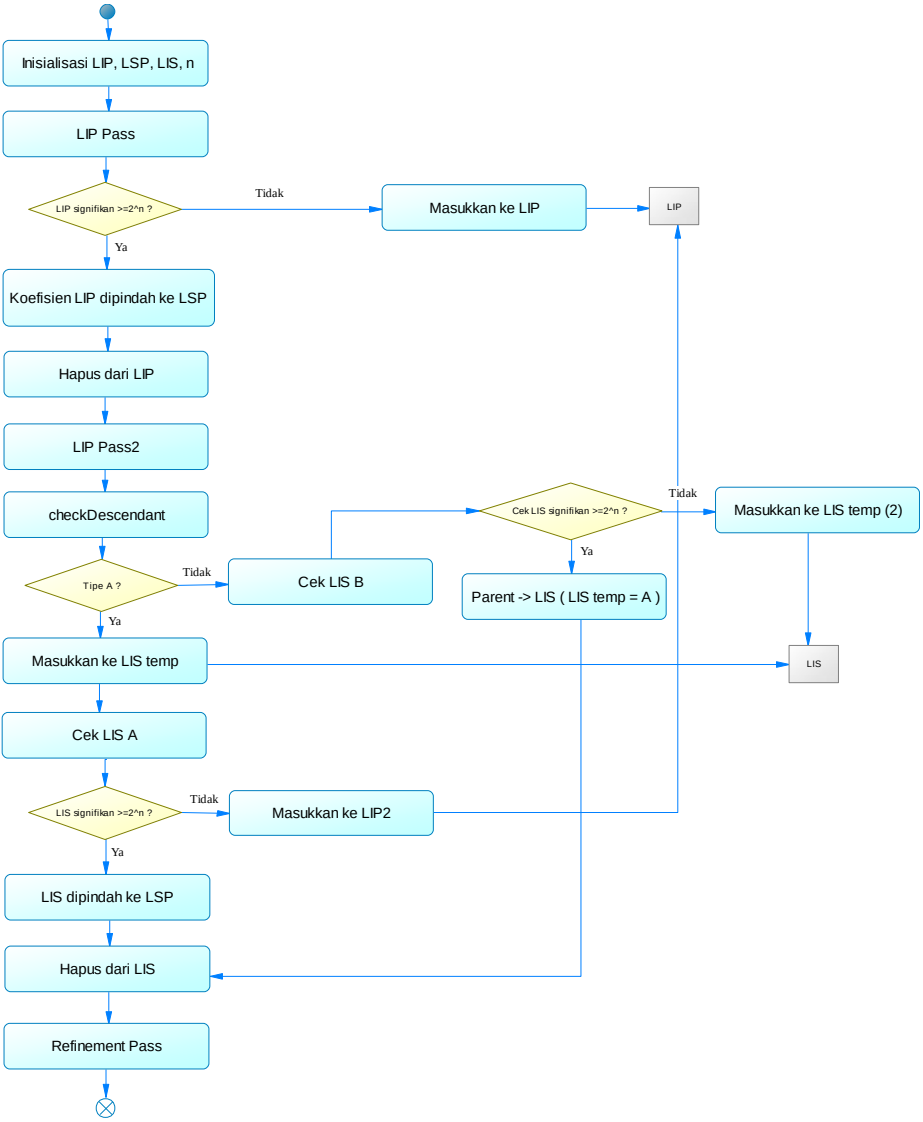
4.2.3.1.3 Proses Encode

Pada tahap *encode* yang ditunjukkan pada gambar 4.5, dilakukan pengecekan terhadap tipe wavelet yang digunakan. Khusus tipe wavelet *bior6.8*, digunakan fungsi tertentu. Proses *encode* diimplementasikan lebih detail dengan menggunakan fungsi fungsi tertentu, yang ditunjukkan pada gambar 4.5. *Encode* dengan menggunakan fungsi wavelet *bior6.8* hampir sama dengan proses *encode* yang dilakukan oleh wavelet lainnya, hanya dibedakan oleh variable masukkannya.



Gambar 4.5. Diagram Alir Proses *Encode I*

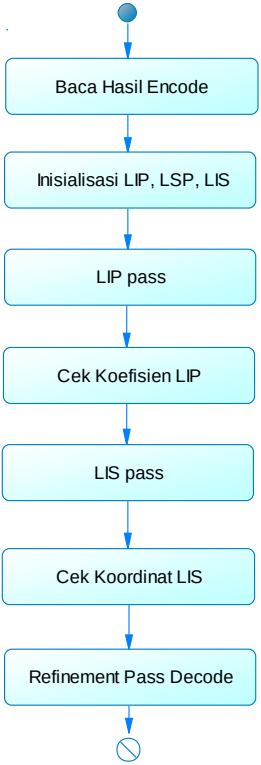
Pada proses *encode* dilakukan inisialisasi terhadap nilai n, LIP, LSP, dan LIS. Koefisien n didapatkan dari penghitungan *log 2* terhadap koefisien terbesar dalam matrik citra. LSP pada awal iterasi, diasumsikan kosong. Selanjutnya dilakukan *sorting pass* dan *refinement pass* sehingga hasil LIP, LSP dan LIS diketahui.



Gambar 4.6. Diagram Alir Proses *Encode II*

4.2.3.1.4 Proses Decode

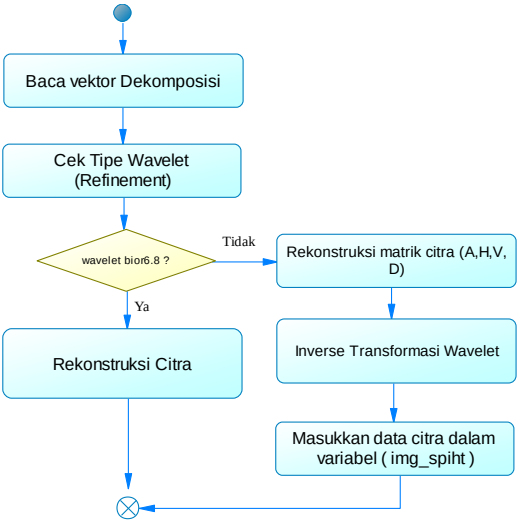
Gambar 4.7 menunjukkan proses *decode* pada kompresi citra. Tahap *decode* merupakan *invers* dari tahap *encode*. Data citra hasil proses *decode* berfungsi sebagai masukan untuk kemudian dilakukan pengecekan terhadap masing masing variable.



Gambar 4.7. Diagram Alir Proses *Decode*

4.2.3.1.5 Proses Refinement

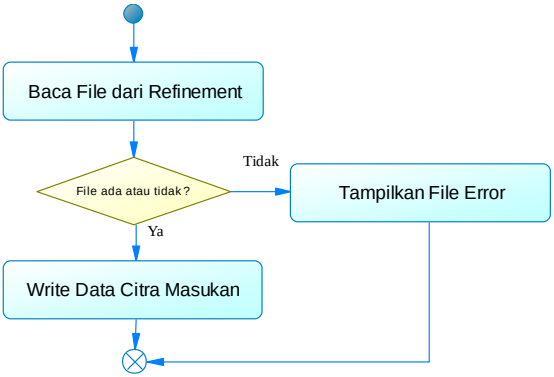
Proses *refinement* yang ditunjukkan dalam gambar 4.8, merupakan proses rekonstruksi dengan menggunakan wavelet. Pada wavelet *bior6.8*, terdapat fungsi tersendiri untuk melakukan *refinement.* Data hasil *refinement* disimpan dalam sebuah variable untuk selanjutnya dilakukan proses *write* terhadap hasil.



Gambar 4.8. Diagram Alir Proses *Refinement*

4.2.3.1.6 Proses Write

Proses *write* ditunjukkan pada gambar 4.9, yang merupakan tahap untuk menampilkan hasil data citra hasil kompresi metode SPIHT.



Gambar 4.9. Diagram Alir Proses *Write*

4.2.3.1.7 Proses Transfornasi Wavelet

Transformasi wavelet menggunakan dua komponen penting dalam melakukan transformasi yaitu fungsi skala dan fungsi wavelet. Berikut persamaan (2.2) mengenai fungsi skala :

$$\varphi_{j,k}(x) = 2^{j/2}\varphi\left(2^{j}x - k\right)$$

Fungsi skala (2.1) disebut juga *lowpass filter* yang mengambil citra dengan gradasi intensitas yang halus dan perbedaan intensitas yang tinggi akan dikurangi.

$$\psi_{j,k}(x) = 2^{j/2}\psi\left(2^{j}x - k\right)$$

Fungsi wavelet pada persamaan (2.3) disebut juga *highpass filter* yang mengambil citra dengan gradasi intensitas yang tingggi dan perbedaan intensitas yang rendah akan dikurangi.

Berikut contoh transformasi wavelet diskrit dua dimensi dengan level dekomposisi satu

Gambar 4.10. Contoh Perhitungan Matriks Pada Citra Lena 5x5 Piksel

L	H
L	H

=

C_A	C_V
C_H	C_D

Diketahui matriks A pada citra berdimensi 5x5 piksel dengan menggunakan warna *grayscale* 8 bit, maka matriks piksel x dapat dibuat sebagai berikut :

$$A = \begin{bmatrix} 33 & 25 & 17 & 10 & 12 \\ 12 & 7 & 7 & 20 & 1 \\ 11 & 17 & 10 & 9 & 40 \\ 27 & 43 & 45 & 2 & 22 \\ 8 & 8 & 8 & 12 & 10 \end{bmatrix}$$

Untuk contoh yang penulis gunakan merupakan dekomposisi wavelet dengan tipe Haar maka filter *lowpass* dan *highpass* sebagai berikut :

$$L = \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \end{bmatrix}$$

$$H = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \end{bmatrix}$$

Maka matriks A akan dikonvolusi baris dengan L menjadi AA sebagai berikut :

$$AA = A * L$$

$$AA = \begin{bmatrix} 33 & 25 & 17 & 10 & 12 \\ 12 & 7 & 7 & 20 & 1 \\ 11 & 17 & 10 & 9 & 40 \\ 27 & 43 & 45 & 2 & 22 \\ 8 & 8 & 8 & 12 & 10 \end{bmatrix} * \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \end{bmatrix}$$

$$AA = \begin{bmatrix} 46 & 41 & 29 & 19 & 15 & 16 \\ 22 & 16 & 9 & 19 & 14 & 1 \\ 15 & 20 & 19 & 13 & 34 & 56 \\ 38 & 49 & 62 & 33 & 16 & 31 \\ 11 & 11 & 11 & 14 & 15 & 14 \end{bmatrix}$$

Nilai AA digunakan untuk menghitung nilai C_A. Tetapi sebelum dilakukan perhitungan terlebih dahulu dilakukan *downsampling* terhadap baris dari matriks AA, sehingga :

$$AA' = \begin{bmatrix} 41 & 19 & 16 \\ 16 & 19 & 1 \\ 19 & 13 & 56 \\ 49 & 33 & 31 \\ 11 & 14 & 14 \end{bmatrix}$$

Tahap selanjutnya proses konvolusi kolom matriks AA' dengan L.

$$AA'' = A' * L$$

$$AA'' = \begin{bmatrix} 41 & 19 & 16 \\ 16 & 19 & 1 \\ 19 & 13 & 56 \\ 49 & 33 & 31 \\ 11 & 14 & 14 \end{bmatrix} * \begin{bmatrix} 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \end{bmatrix}$$

$$AA'' = \begin{bmatrix} 58 & 27 & 24 \\ 40 & 27 & 13 \\ 25 & 23 & 41 \\ 49 & 33 & 62 \\ 43 & 33 & 32 \\ 16 & 20 & 20 \end{bmatrix}$$

Baris pada matriks AA'' kemudian di downsampling sehingga nilai C_A sekarang adalah sebagai berikut :

$$C_A = \begin{bmatrix} 40 & 27 & 13 \\ 49 & 33 & 62 \\ 16 & 20 & 20 \end{bmatrix}$$

Untuk penghitungan nilai C_H dilakukan dengan cara melakukan konvolusi baris matriks A dengan L. Setelah itu menghasilkan matriks AA, dan di-*downsampling* menjadi matriks AA'. Matriks AA' dilakukan konvolusi kolom dengan matriks H sehingga diperoleh matriks AA''. Matriks AA'' di-*downsampling* menghasilkan nilai matriks C_H.

$$C_H = \begin{bmatrix} 17 & 0 & 11 \\ -21 & -14 & 18 \\ 0 & 0 & 0 \end{bmatrix}$$

Untuk penghitungan nilai C_V dilakukan dengan cara melakukan konvolusi baris matriks A dengan H. Setelah itu menghasilkan matriks AA, dan di-*downsampling* menjadi matriks AA'. Matriks AA' dilakukan konvolusi kolom dengan matriks L sehingga diperoleh matriks AA''. Matriks AA'' di-*downsampling* menghasilkan nilai matriks C_V.

$$C_V = \begin{bmatrix} 8 & -3 & 0 \\ -11 & 22 & 0 \\ 0 & -4 & 0 \end{bmatrix}$$

Untuk penghitungan nilai C_D dilakukan dengan cara melakukan konvolusi baris matriks A dengan H. Setelah itu menghasilkan matriks AA, dan di-*downsampling* menjadi matriks AA'. Matriks AA' dilakukan konvolusi kolom dengan matriks H sehingga diperoleh matriks AA''. Matriks AA'' di-*downsampling* menghasilkan nilai matriks C_D.

$$C_D = \begin{bmatrix} 0 & 10 & 0 \\ 5 & -21 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Matriks $C_A, C_H, C_V \; dan \; C_D$ masing masing disusun menjadi satu kolom dan ditranspose sehingga diperoleh :

$C_A = [40\ 49\ 16\ 27\ 33\ 20\ 13\ 62\ 20]$
$C_H = [17\ -21\ 0\ 0\ -14\ 0\ 11\ 18\ 0]$
$C_V = [8\ -11\ 0\ -3\ 22\ -4\ 0\ 0\ 0]$
$C_D = [0\ 5\ 0\ 10\ -21\ 0\ 0\ 0\ 0]$

Maka nilai dari matriks $C_A, C_H, C_V \; dan \; C_D$ disusun sebagai berikut ini :

$C = [40\ 49\ 16\ 27\ 33\ 20\ 13\ 62\ 20\ 17\ -21\ 0\ 0\ -14\ 0\ 11\ 18\ 0\ 8$
$-11\ 0\ -3\ 22\ -4\ 0\ 0\ 0\ 0\ 5\ 0\ 10\ -21\ 0\ 0\ 0\ 0]$

Matriks C dalam proses kompresi disimpan dalam file temporary dalam bentuk bit stream. Apabila ingin memunculakn nilai matriks C maka perlu dilakukan kuantisasi terlebih dahulu setelah itu dilakukan *decode* untuk meunculkan citra dalam bentuk gambar.

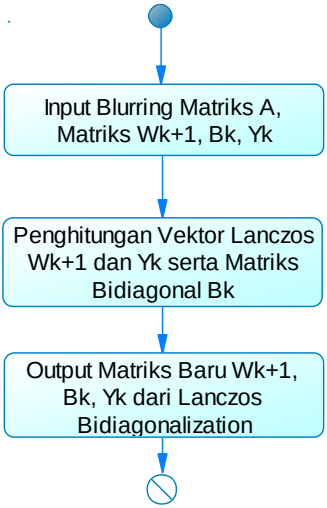
4.2.3.2 Perancangan Restorasi Citra

Pada sub bab ini akan dijelaskan mengenai desain sistem untuk implementasi metode *iterative lanczos-hybrid regularization* pada restorasi citra. Desain proses menjelaskan tentang proses proses lanczos bidiagonalization, pencarian ω, Regularisasi Tikhonov dan penghitungan parameter dengan

Weighted GCV, perhitungan x aproksimasi dan penentuan stopping criteria berdasarkan nilai fungsi GCV.

4.2.3.2.1 Proses Metode Lanczos Bidiagonalization

Proses Lanczos Bidiagonalization dijelaskan pada gambar 4.11. Input proses dari proses ini adalah blurring matriks A, kemudian matriks B_k, Y_k, W_{k+1}. Khusus pada matriks B_k, Y_k, W_{k+1} nilai inputannya berasal dari nilai pada iterasi sebelumnya. Kemudian sesuai persamaan, (2.13, 2.15, 2.20), nilai B_k, Y_k, W_{k+1} segera dicari. Hasil output dari proses ini adalah matriks B_k, Y_k, W_{k+1} yang baru pada iterasi tertentu.



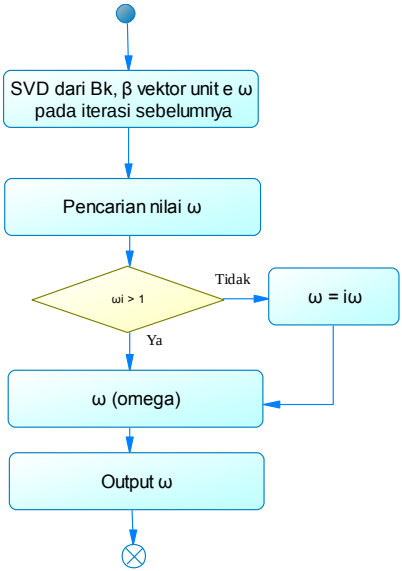
Gambar 4.11. Diagram Alir Lanczos Bidiagonalization

4.2.3.2.2 Proses Pencarian ω

Langkah berikutnya yaitu pencarian nilai ω sebagai parameter tambahan fungsi Weighted GCV seperti gambar 4.12. Input pada proses ini adalah nilai SVD dari matriks B_k, β, e_1 dan ω pada iterasi sebelumnya. Persamaan (2.21) menunjukkan cara

pencarian nilai ω berdasarkan iterasi melalui derivasi dari fungsi Weighted GCV.

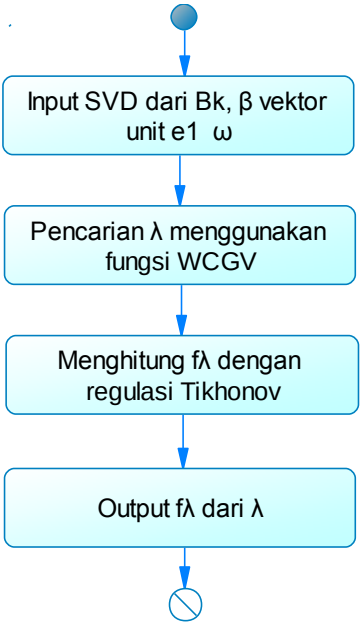
Setelah nilai ω ditemukan, nilai tersebut harus berkisar $\omega \leq 1$. Bila tidak, maka nilai ω adalah 1. Kemudian, nilai ω pada iterasi tersebut dijumlahkan dengan nilai ω pada iterasi-iterasi sebbelumnya dan dihitung nilai rata-ratanya. Hasil perhitungan tersebut adalah nilai ω yang dipakai oleh regularisasi Weighted GCV.



Gambar 4.12. Diagram Alir Pencarian Nilai ω

4.2.3.2.3 Proses Regularisasi Tikhonov dan WGCV

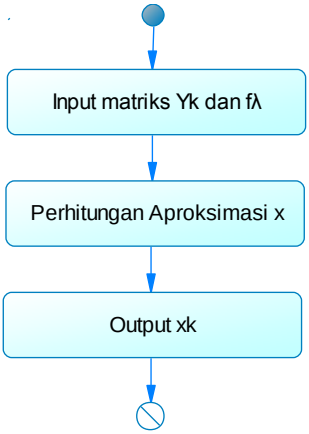
Penjelasan diagram alir untuk proses metode regularisasi ini dapat dilihat pada gambar 4.13. Input yang dibutuhkan yaitu SVD dari matriks B_k, β, e_1 dan ω hasil proses (3.2.2). Proses dimulai dengan menghitung fungsi WGCV sesuai persamaan (2.21) untuk menghasilkan regularisasi parameter (λ). Langkah selanjutnya menghitung f_λ. Output dari proses ini adalah f_λ dan λ.



Gambar 4.13. Diagram Alir Regularisasi Tikhonov dan WGCV

4.2.3.2.4 Proses Perhitungan x Aproksimasi

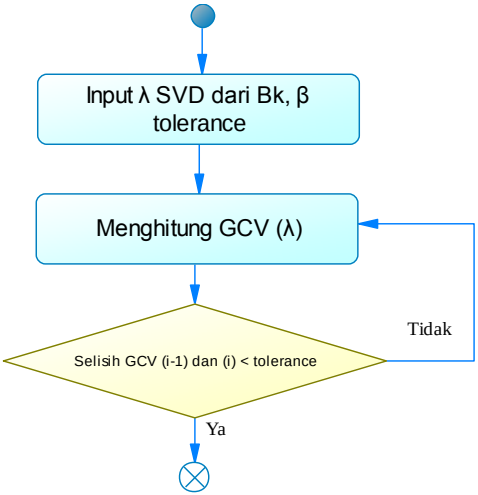
Diagram alir yang menjelaskan proses ini adalah gambar 4.14. Input pada proses ini adalah matriks Y_k dan f_λ. Output yang dihasilkan adalah aproksimasi nilai x. Nilai tersebut dihitung berdasarkan nilai-nilai yang telah didapat pada proses sebelumnya. Persamaan (2.22) menjelaskan cara menghitung nilai aproksimasi x.



Gambar 4.14. Diagram Alir Perhitungan x Aproksimasi

4.2.3.2.5 Proses Penentuan Stopping Criteria berdasarkan nilai Fungsi GCV

Langkah berikutnya menentukan pemberhentian iterasi pada sistem. Diagram alir yang menjelaskan proses ini adalah gambar 4.15. Input proses ini antara lain nilai λ, SVD dari B_k, β, dan dimensi dari *blurring* matriks. Perhitungan kriteria tersebut berdasarkan nilai dari GCV seperti pada persamaan (2.27) dengan parameter lambda pada tiap iterasi dengan WGCV. Kemudian, iterasi dinyatakan berhenti apabila memenuhi dua kondisi, yaitu selisih antara nilai GCV pada iterasi sebelah nya sangat kecil atau grafik nilai GCV sudah menunjukkan tanda-tanda semi konvergence. Output yang dihasilkan pada proses ini adalah penanda pemberhentian iterasi.



Gambar 4.15. Diagram Alir Penentuan Stopping Criteria berdasarkan nilai Fungsi GCV

4.2.4 Perancangan Data

Terdapat tiga macam data yang digunakan oleh sistem ini antara lain data masukan, data proses, dan data keluaran. Pada Tugas Akhir ini data masukkan berupa gambar *grayscale.* Data proses merupakan data yang berisi parameter parameter yang akan digunakan oleh Algoritma *Set Partitioning in Hirarchical Tress* dan proses restorasi citra *Iterative Lanczos-Hybrid Regularization.* Sedangkan data keluaran adalah data citra hasil restorasi citra berbasis kompresi citra.

4.2.4.1 Data Masukan

Data masukan sistem ini berupa citra gambar *grayscale* yang diambil oleh peniliti. File citra yang diambil memiliki spesifikasi yaitu :

a. Memiliki resolusi citra 512 x 512 piksel, 320 x 320 piksel, 256 x 256 piksel, dan 180 x 180 piksel dalam gambar *grayscale* dengan resolusi maksimal 1024x1024 piksel.
b. Format ekstensi citra *grayscale* dapat berupa : .**bmp*.

4.2.4.2 Data Proses

Data proses merupakan data yang digunakan dalam proses penngolahan data masukan. Data proses ini diperoleh dari hasil pengolahan data masukan sesuai dengan tahapan algoritma dan metode yang telah disusun. Tabel 4.2 menjelaskan tahapan dari data proses.

Tabel 4.2. Data Proses

No	Tahapan	*Input*	*Output*
1.	*Input* Awal	Citra Citra *grayscale*	Ukuran matriks baris dan kolom
2.	Proses *Encode*	Matriks dekomposisi, laju kompresi	*Encode* Data dalam file *bit stream*
3.	Proses *Decode*	*Encode* Data dalam file *bit stream*, level dekomposisi	Citra kompresi, MSE, dan PSNR
4.	Proses metode Lanczos Bidiagonalization	Blurring matriks A, B_k, Y_k, W_{k+1}	Matriks baru B_k, Y_k, W_{k+1}
5.	Proses Pencarian ω	Nilai SVD dari matriks B_k, β, e_1 dan ω	Nilai ω
6.	Regularisasi Tikhonov dan WGCV	Nilai SVD dari matriks B_k, β, e_1 dan ω	Nilai f_λ dan λ
7.	Proses perhitungan x aproksimasi	Nilai Y_k dan f_λ	Nilai aproksimasi x
8.	Proses Penentuan Stopping Criteria berdasarkan nilai Fungsi GCV	Nilai λ, SVD dari B_k, β, dan dimensi dari *blurring* matriks	PSNR

4.2.4.3 Data Keluaran

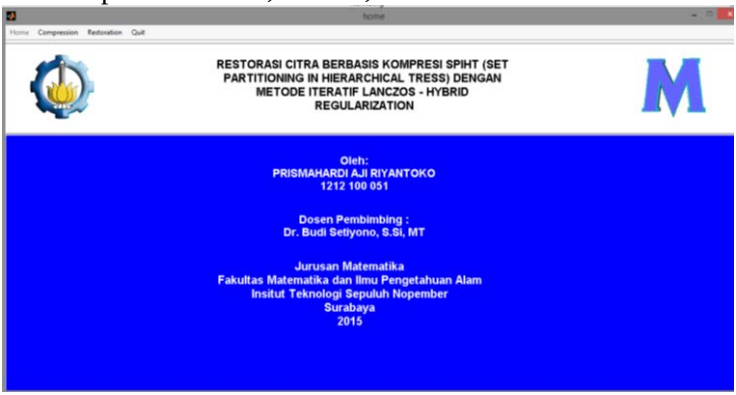
Data keluaran berupa citra hasil Restorasi citra berbasis Kompresi citra dan nilai PSNR. Program ini juga menghasilkan keluaran berupa citra terkompresi, nilai MSE, PSNR, dan compression ratio.

4.2.5 Perancangan Antar Muka Sistem

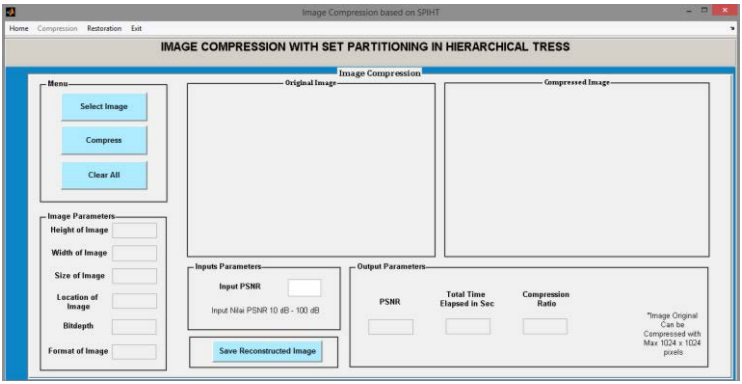
Desain antar muka sistem dibutuhkan agar pengguna dengan mudah mengoperasikan perangkat lunak yang dibangun.

4.2.5.1 Perancangan Halaman Utama

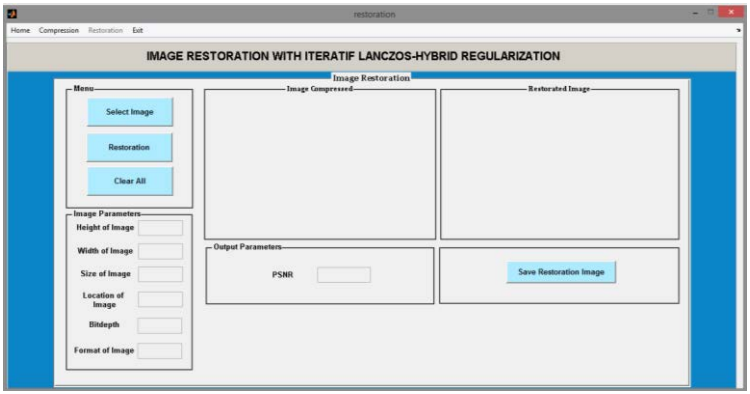
Halaman utama ini memiliki desain antar muka yang ditunjukkan pada gambar 4.16, 4.17, dan 4.18. Pada halaman ini dilakukan proses *encode, decode,* dan restorasi citra.



Gambar 4.16. Halaman Utama Program



Gambar 4.17. Halaman Kompresi SPIHT



Gambar 4.18. Halaman Restorasi Lanczos – Hybrid Regularization

4.3 Implementasi Sistem

4.3.1 Implementasi Input Citra

Proses input citra menggunakan fungsi uigetfile dengan format .**bmp*. Citra masukkan disimpan dalam sebuah data matriks baris dan kolom.

```
[file path]=uigetfile('*.bmp');
Orig_I=imread(file);
[r c p]=size(Orig_I);
```

```
[row1 col1]=size(Orig_I);
Orig_I=imresize(Orig_I,[512 512]);
```

4.3.2 Implementasi Dekomposisi Wavelet

Proses dekomposisi wavelet pada tipe wavelet *bior6.8* diimplementasikan dalam fungsi wavelet *bior6.8.*

```
type = 'bior6.8';
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(type);
```

Hasil pembacaan terhadap citra masukkan, dihasilkan koefisien matrik citra dalam ukuran tertentu. Proses dekomposisi dilakukan dengan melewatkan dinyal pada filter *low pass* dan *high pass.* Kedua filter dalam proses dekomposisi, dikodekan sebagai *Lo_D* dan *Hi_D.* Filter *Lo_D* dan *Hi_D* mempunyai panjang yang sama. Dekomposisi dilakukan sampai level tertentu sesuai inputan. Proses pembagian koefisien menjadi *subband*, diimplementasikan dalam fungsi *dwt2.* Detail koefisien *approximation*, horizontal, vertical dan diagonal disimpan dalam sebuah variable.

```
for i=1:n
[x,h,v,d] = dwt2(x,Lo_D,Hi_D,'mode','per');
% decomposition
c = [h(:)' v(:)' d(:)' c];
s = [size(x);s];
end
```

Dalam fungsi *func_DWT* terdapat fungsi *Mywavedec2* yang digunakan sebagai anlisa dekomposisi wave;et dua dimensi. Vektor C digunakan untuk menyimpan koefisien *aproxmiation*, horizontal, vertical dan diagonal.

```
[C,S] = func_Mywavedec2(I,level,Lo_D,Hi_D);
```

4.3.3 Implementasi *Encoding*

Proses *encode* tipe wavelet *bior6.8* diimplementasikan pada fungsi *My_Encode_SPIHT*. Dengan parameter inputan berupa citra masukan, level dekomposisi dan laju kompresi.

```
img_enc = func_SPIHT_Enc(I_W, max_bits,
nRow*nColumn, 5);
```

Tahap *encode* dimulai dari inisialisasi terhadap LIP, LSP dan LIS dan dilanjutkan *sorting pass.* Dalam *sorting pass*, pengecekan terhadap *offspring* dan descendant digunakan untuk mencari koefisien dalam LIP, LSP dan LIS. Pencarian terhadap koefisien *offspring* dan *descendant* dimplementasikan dalam fungsi *checkDescendant.* Koefisien hasil selanjutnya diolah dalam *refinement pass* dengan menggunakan *floor*.

```
max_d =
func_MyDescendant(LIStemp(i,1),LIStemp(i,2)
,LIStemp(i,3),m);
```

Hasil dari proses *encode* berupa file out stream, yang dijasikan variable masukan pada proses *decode*.

4.3.4 Implementasi *Decoding*

Tahap decode pada metode SPIHT diimplementasikan dalam fungsi *func_SPIHT_Dec.* Data masukan pada tahap decode berupa file out stream.

```
img_dec = func_SPIHT_Dec(img_enc);
```

4.3.5 Implementasi Program Utama Restorasi

Input proses ini terdiri dari 4 parameter, antara lain A, b, dan P. Berikut merupakan program utama dengan fungsi Hybr.

```
function [x_out, output] = HyBR (A, b, P)
%HyBR didefinisikan sebagai Metode Lanczos
```

```
- Hybrid Regularization
%1. Metode Iteratif Lanczos
Bidiagonalization (LBD)
%2. Regularisasi Tikhonov dengan WGCV

%Inisialisasi
if nargin < 2
    error ('HyBR: Not Enough Input')
elseif nargin < 3
    P = [];
end
[m,n] = size (A);
maxiter = 300;
regstart = 2;
degflat = 10^-6;

bSize = size(b);
b = b(:);
A.imsize = bSize;

outputparams = nargout > 1;
if outputparams
    output.iterations = maxiter;
    output.GCVstop = [];
    output.B = [];
    output.flag = 3;
end

if isempty (P)
    beta = norm (b); U = (1/beta)*b;
    handle = @LBD;
else
    U=P\b;
    beta = norm(U); U = U/beta;
    handle = @PLBD;
end

%maincode
```

```
B = []; V = []; GCV = []; Omega = [];
omegaTemp = 0;
terminate = 1; warning = 0;

h = waitbar (0, 'Beginning iterations :
please wait . . .');

for i = 1:maciter + 1
    [U,B,V] = feval(handle, A, U, B, V, P);
    vector = (beta(1)*eye(size(U,2),1));

    if i >= 2 %Mulai iterasi Lanczos
        [Ub, Sb, Vb] = svd (B);

omega(i-1) = min(1, findomega(Ub'*vector,
diag(Sb)));
omegaTemp = mean(Omega);

%Solve the projected problem dengan
Tikhonov
[f, alpha] = Tikhonovsolver (Ub, diag(Sb),
Vb, vector, omegaTemp);

%Hitung stopping kriteria dengan fungsi GCV
GCV(i-1) = GCVstopfun(alpha,
Ub(1,:)',diag(Sb),beta,m,n);

%Penghentian iterasi
if i>2 && terminate

%Flatness pada fungsi GCV
if abs((GCV(i-1)-GCV(i-2)))/GCV(1) <
degflat
    x_out = V*f;
    if outputparams
        output.B = B;
        output.GCVstop = GCV(:);
        output.iterations = i-1;
```

```
        output.flag = 1;
    end
    close (h)

    x_out = reshape (x_out, bSize);
    return;
    terminate = 0;

%Perilaku semi konvergensi maka harus
dihentikan
elseif warning && lenght GCV >
iterations_save + 3
%Grafiknya mengalami perilaku semi
konvergen, sehingga harus stop
if GCV (iterations_save) < GCV
(iterations_save+1:end)
    x_out = x_save;
    if outputparams
        output.B = B;
        output.GCVstop = GCV (:);
        output.iterations =
iterations_save;
        output.flag = 2;
    end
    close (h)
    x_out = reshape (x_out, bSize);
    return;
    terminate = 0; %Solution is already
found

else %Hanya sedikit naikan (bump)
    warning = 0;
    x_out = [];
    iterations_save = maxiter;
end

%Tidak ada warning
elseif ~warning
    %Pengecekan bila nilai GCV sebelumnya
```

```
lebih kecil daripada sekarang
    if GCV (i-2) < GCV (i-1)
        warning = 1;
        x_save = V*f;
        iterations_save = i-1;
    end
end
end
 x = V*f;

    end
    waitbar (i/(maxiter+1), h)
end
close (h)
```

4.3.6 Implementasi Proses metode Lanczos Bidiagonalization

```
function [U, B, V] = LBD (A, U, B, V ,P)
%menghitung Lanczos Bidiagonalization tiap
iterasi

[m,k] = size(U);
if k == 1
    v = A'*U(:,k);
else
    v = A'*U(:,k) - B(k, k-1)*V(:,k-1);
end

alpha = norm(v);
v = v/alpha;
u = A*v - alpha*U(:,k);

beta = norm(u);
u = u/beta;

U=[U,u]; %U dan V orthogonal matrix dari
iterasi Lanczos Bidiogonalization
```

```
V = [V,v];

%matriz bidiagonalization dari iterasi
Lanczos Bidiagonalization
B=[B,[zeros(k-1,1);alpha];[zeros(1,k-
1),beta]];
```

4.3.7 Implmentasi Proses Pencarian ω

```
function omega = findomega (bhat,s)
alpha = s(end);
m = length (bhat);
n = length (s);
t0=sum(abs(bhat(n+1:m)).^2);
s2=abs(s).^2;
alpha2= alpha^2;
tt = 1./(s2+alpha2);
t1=sum(s2.*tt);
t2=abs(bhat(1:n).*alpha.*s).^2;
t3=sum(t2.*abs((tt.^3)));
t4=sum((s.*tt).^2);
t5=sum((abs(alpha2*bhat(1:n).*tt)).^2);
v1=abs(bhat(1:n).*s).^2;
v2=sum(v1.*abs((tt.^3)));

omega= (m*alpha2*v2)/(t1*t3 + t4*(t5+t0));
```

4.3.8 Implementasi Regularisasi Tikhonov dan WGCV

```
function [f,alpha] = Tikhonovsolver(U , S,
V, b, omega)
%Menghitung projected problem dengan
regularisasi tikhonov
%Input : U,S,V - SVD dari matrix Bk. b -
Beta*el
%omega - parameter "weight" pada WGCV
```

```
bhat = U'*b;
bhat = bhat (:);

%parameter regularisasi
alpha = fminbnd ('TikWGCV', 0, 1, [], bhat,
S, omega);

D = abs(S).^2 + alpha^2;
bhat = conj(S) .* bhat(1:length (S));
xhat = bhat./D;
f =  V*xhat; %hasil dari projected problem
(f lambda)
```

```
function G = TikGCCV (alpha, bhat, s,
omega)
%Menghitung Fungsi WGCV untuk Regularisasi
Tikhonov
%bhat = vector U'*b
%s = vektor yang mengandung nilai singular
if nargin == 3
    omega = [];
end
if isempty (omega)
    omega = 1;
end

m = length(bhat);
n = length (s);
t0 = sum(abs(bhat(n+1:m)).^2);

s2 = abs(s).^2;
alpha2 = alpha^2;
tt= 1./ (s2 + alpha2);
t1 = alpha2.*tt;
t2 = abs(bhat/(1:n) .* t1) .^2;
t3 = t1 + (1-omega)*s2 .* tt;
```

```
G = n*(sum(t2) +t0) / (sum(t3) + m - n)^2;
```

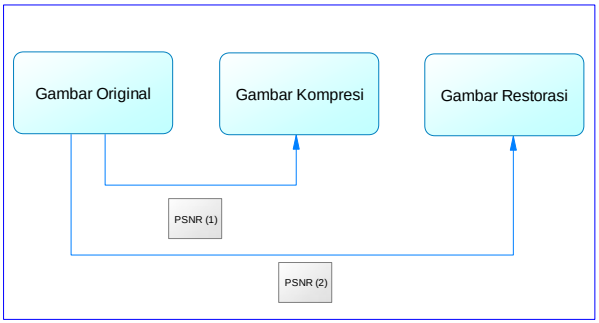
4.3.9 Implementasi Proses Penentuan Stopping Criteria berdasarkan nilai Fungsi GCV

```
function G = GCVstopfun (alpha, u, s, beta,
m, n)
k = length (s);
beta2 = beta^2;

s2 = abs(s) .^ 2;
alpha2 = alpha^2;

t1 = 1 ./ (s2 +alpha2);
t2 = abs(alpha2*u(1:k) .* t1) .^2;
t3 = s2 .* t1;

num = n*beta*(sum(t2) + abs(u(k+1))^2);
den = (m -k + sum(alpha2*t1))^2;
G = num / den;
```

BAB V
PENGUJIAN DAN PEMBAHASAN

Pada bab ini akan dilakukan pengujian serta pembahasan terhadap program yang telah dibuat. Pengujian sistem ini merupakan pengujian kompresi dan restorasi berdasarkan data gambar *grayscale* yang tersedia dalam penyimpanan (direktori) komputer. Pengujian dilakukan pada citra *grayscale* babbon, lena dan pepper untuk mendapatkan nilai PSNR.

Diagram Blok Hasil Uji Coba Program



Gambar 5.1. Diagram Blok Hasil Uji Coba Program

Diagram blok pada Gambar 5.1 menjelaskan mengenai alur untuk mendapatkan nilai PSNR. Sebagai bahan uji coba mengggunakan nilai PSNR (1) dengan membandingkan gambar original dan gambar terkompresi. Nilai PSNR pada data uji coba sebesar 25 dB termasuk dalam rating Inferior dengan banyak derau dan 35 dB termasuk dalam rating Marginal dengan banyak butiran halus. PSNR (2) merupakan hasil perbandingan nilai piksel gambar original dan gambar restorasi. Sehingga, secara analisis dapat dikatakan mengalami kenaikkan kualitas citra apabila nilai PSNR (2) > PSNR (1).

5.1 Data Uji Coba

Uji coba pada program dalam penelitian tugas akhir ini dilakukan terhadap citra *grayscale* sebanyak tiga citra diantaranya babbon, lena dan pepper.

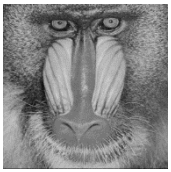
Tabel 5.1 Data Citra Uji Coba

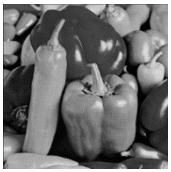
No	Nama	Gambar
1	Lena	
2	Babbon	
3	Pepper	

5.2 Pengujian Tahap Kompresi Citra

Proses ini bertujuan untuk menentukan nilai PSNR yang digunakan untuk diuji pada proses restorasi dan ukuran dimensi citra dengan spesifikasi 180x180, 256x256, 320x320 dan 512x512. Citra dan ukuran dari setiap citra objek yang akan dikompresi dapat dilihat pada Gambar 5.2.

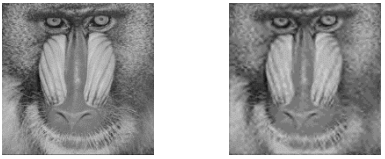

(a)


(b)


(c)

Gambar 5.2. (a) Citra Lena, (b) Citra Babbon, (c) Citra Pepper

1. Proses uji coba pertama dilakukan dengan PSNR sebesar 25 dB, dengan iterasi laju kompresi antara 0,1 sampai 0,9 dengan penambahan nilai setiap iterasi sebesar 0,1. Hasil kompresi citra yang telah dicapai adalah sebagai berikut :



(a) (b)

Gambar 5.3. (a) Citra Babbon Original, (b) Citra Babbon terkompresi. Citra diatas menggunakan ukuran 180x180 piksel

Tabel 5.2 Citra Kompresi Ukuran 180x180 pixels (PSNR 25dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	95 KB	33 KB	29,15 dB	20,2856 *detik*
Babbon	95 KB	33 KB	25,64 dB	22,5813 *detik*
Peppers	95 KB	33 KB	29,20 dB	21,0535 *detik*

(a) (b)

Gambar 5.4. (a) Citra Lena Original, (b) Citra Lena terkompresi. Citra diatas menggunakan ukuran 256x256 piksel

Tabel 5.3 Citra Kompresi Ukuran 256x256 pixels (PSNR 25dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	192 KB	64 KB	28,16 dB	20,8841 *detik*
Babbon	192 KB	64 KB	26,64 dB	105,659 *detik*
Peppers	192 KB	64 KB	28,24 dB	21,0556 *detik*

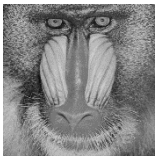
(a)

(b)

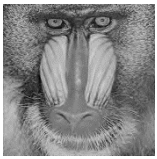
Gambar 5.5. (a) Citra Peppers Original, (b) Citra Peppers terkompresi. Citra diatas menggunakan ukuran 320x320 piksel

Tabel 5.4 Citra Kompresi Ukuran 320x320 pixels (PSNR 25dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	301 KB	102 KB	27,84 dB	19,4528 *detik*
Babbon	301 KB	102 KB	25,27 dB	111,624 *detik*
Peppers	301 KB	102 KB	27,64 dB	21,0353 *detik*



(a)



(b)

Gambar 5.6. (a) Citra Babbon Original, (b) Citra Babbon terkompresi. Citra diatas menggunakan ukuran 512x512 piksel

Tabel 5.5 Citra Kompresi Ukuran 512x512 pixels (PSNR 25dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	767 KB	257 KB	27,26 dB	8,3081 *detik*
Babbon	767 KB	257 KB	25,10 dB	326,827 *detik*
Peppers	767 KB	257 KB	29,20 dB	20,1444 *detik*

2. Proses uji coba kedua dilakukan dengan PSNR sebesar 35 dB, dengan iterasi laju kompresi antara 0,1 sampai 0,9 dengan penambahan nilai setiap iterasi sebesar 0,1. Hasil kompresi citra yang telah dicapai adalah sebagai berikut :


(a) (b)

Gambar 5.7. (a) Citra Peppers Original, (b) Citra Peppers terkompresi. Citra diatas menggunakan ukuran 180x180 piksel

Tabel 5.6 Citra Kompresi Ukuran 180x180 pixels (PSNR 35dB)

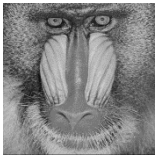
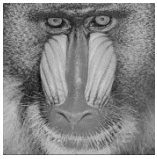
Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	95 KB	33 KB	36,13 dB	115,036 *detik*
Babbon	95 KB	33 KB	35,33 dB	569,648 *detik*
Peppers	95 KB	33 KB	36,37 dB	104,300 *detik*


(a) (b)

Gambar 5.8. (a) Citra Lena Original, (b) Citra Lena terkompresi. Citra diatas menggunakan ukuran 256x256 piksel

Tabel 5.7 Citra Kompresi Ukuran 256x256 pixels (PSNR 35dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	192 KB	64 KB	35,69 dB	128,387 *detik*
Babbon	192 KB	64 KB	35,16 dB	570,712 *detik*
Peppers	192 KB	64 KB	35,53 dB	105,288 *detik*

(a) (b)

Gambar 5.9. (a) Citra Babbon Original, (b) Citra Babbon terkompresi. Citra diatas menggunakan ukuran 320x320 piksel

Tabel 5.8 Citra Kompresi Ukuran 320x320 pixels (PSNR 35dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	301 KB	102 KB	36,31 dB	109,541 *detik*
Babbon	301 KB	102 KB	35,08 dB	580,831 *detik*
Peppers	301 KB	102 KB	36,33 dB	261,440 *detik*


(a) (b)

Gambar 5.10. (a) Citra Peppers Original, (b) Citra Peppers terkompresi. Citra diatas menggunakan ukuran 512x512 piksel

Tabel 5.9 Citra Kompresi Ukuran 512x512 pixels (PSNR 35dB)

Nama Citra	Ukuran File Observasi	Ukuran File Terkompresi	PSNR Kompresi	Waktu Komputasi
Lena	767 KB	257 KB	35,27 dB	280,433 *detik*
Babbon	767 KB	257 KB	35,01 dB	607,834 *detik*
Peppers	767 KB	257 KB	36,26 dB	498,690 *detik*

5.3 Pengujian Tahap Restorasi Citra

Proses ini bertujuan untuk meningkatkan nilai PSNR yang digunakan pada proses restorasi dan ukuran dimensi citra dengan spesifikasi 180x180, 256x256, 320x320 dan 512x512. Tahap juga menampilkan data waktu komputasi untuk mengetahui kecepatan program. Hasil proses restorasi citra yang telah dicapai adalah sebagai berikut :

1. Pengujian Citra dengan PSNR awal 25 dB

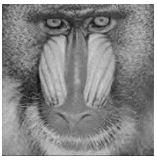
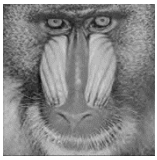
(a)

(b)

Gambar 5.11. (a) Citra Peppers terkompresi, (b) Citra Peppers terestorasi. Citra diatas menggunakan ukuran 180x180 piksel

Tabel 5.10 Citra Restorasi Ukuran 180x180 pixels (PSNR 25dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	29,15 dB	29,88 dB	237,187 *detik*
Babbon	25,64 dB	27,14 dB	232,405 *detik*
Peppers	29,20 dB	30,23 dB	348,174 *detik*

(a) (b)

Gambar 5.12. (a) Citra Babbon terkompresi, (b) Citra Babbon terestorasi. Citra diatas menggunakan ukuran 256x256 piksel

Tabel 5.11 Citra Restorasi Ukuran 256x256 pixels (PSNR 25dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	28,16 dB	28,86 dB	220,091 *detik*
Babbon	26,64 dB	28,35 dB	221,583 *detik*
Peppers	28,24 dB	29,21 dB	220,571 *detik*


(a) (b)

Gambar 5.13. (a) Citra Lena terkompresi, (b) Citra Lena terestorasi. Citra diatas menggunakan ukuran 320x320 piksel

Tabel 5.12 Citra Restorasi Ukuran 320x320 pixels (PSNR 25dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	27,84 dB	28,19 dB	266,174 *detik*
Babbon	25,27 dB	26,66 dB	341,697 *detik*
Peppers	27,64 dB	28,51 dB	219,096 *detik*


(a) (b)

Gambar 5.14. (a) Citra Peppers terkompresi, (b) Citra Peppers terestorasi. Citra diatas menggunakan ukuran 512x512 piksel

Tabel 5.13 Citra Restorasi Ukuran 512x512 pixels (PSNR 25dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	27,26 dB	27,59 dB	217,269 *detik*
Babbon	25,10 dB	26,22 dB	213,779 *detik*
Peppers	27,20 dB	27,37 dB	231.311 *detik*

2. Pengujian Gambar dengan PSNR awal 35 dB


(a) (b)

Gambar 5.15. (a) Citra Peppers terkompresi, (b) Citra Peppers terestorasi. Citra diatas menggunakan ukuran 180x180 piksel

Tabel 5.14 Citra Restorasi Ukuran 180x180 pixels (PSNR 35dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	36,13 dB	36,59 dB	223,088 *detik*
Babbon	35,33 dB	35,69 dB	226.380 *detik*
Peppers	36,37 dB	36,87 dB	192,049 *detik*

(a) (b)

Gambar 5.16. (a) Citra Babbon terkompresi, (b) Citra Babbon terestorasi. Citra diatas menggunakan ukuran 256x256 piksel

Tabel 5.15 Citra Restorasi Ukuran 256x256 pixels (PSNR 35dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	35,69 dB	36,31 dB	193,008 *detik*
Babbon	35,16 dB	36,11 dB	188,531 *detik*
Peppers	35,53 dB	36,30 dB	201,861 *detik*

(a) (b)

Gambar 5.17. (a) Citra Lena terkompresi, (b) Citra Lena terestorasi. Citra diatas menggunakan ukuran 320x320 piksel

Tabel 5.16 Citra Restorasi Ukuran 320x320 pixels (PSNR 35dB)

Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	36,31 dB	36,94 dB	191,163 *detik*
Babbon	35,08 dB	35,64 dB	204,089 *detik*
Peppers	36,33 dB	37,18 dB	231,191 *detik*

(a)

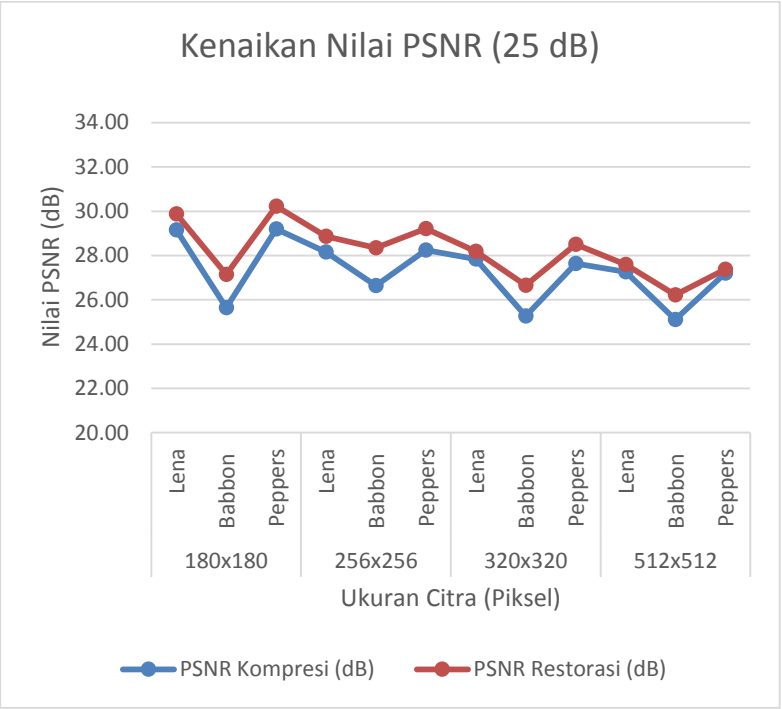
(b)

Gambar 5.18. (a) Citra Peppers terkompresi, (b) Citra Peppers terestorasi. Citra diatas menggunakan ukuran 512x512 piksel

Tabel 5.17 Citra Restorasi Ukuran 512x512 pixels (PSNR 35dB)

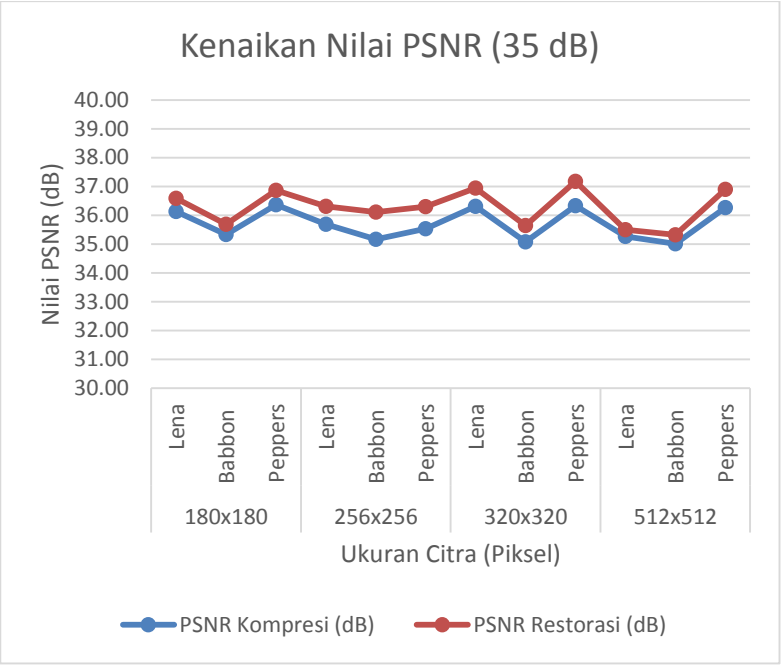
Nama Citra	PSNR Kompresi	PSNR Restorasi	Waktu Komputasi
Lena	35,27 dB	35,50 dB	178.638 *detik*
Babbon	35,01 dB	35,32 dB	187,711 *detik*
Peppers	36,26 dB	36,90 dB	185,688 *detik*

5.4 Pembahasan Hasil Uji Coba



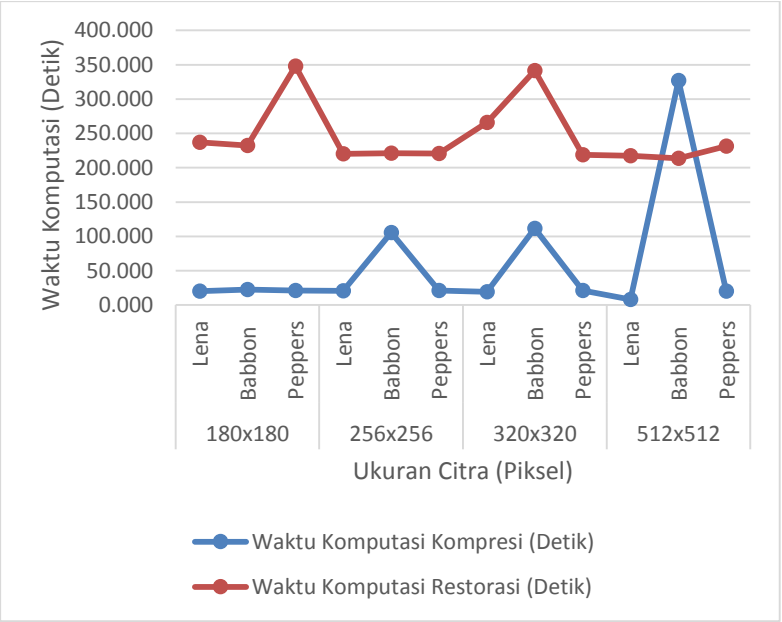
Gambar 5.19 Grafik Kenaikan Nilai PSNR (25 dB)

Berdasarkan gambar 5.19 nilai *Peak Signal Noise to Ratio* (PSNR) tidak begitu mengalami kenaikan secara signifikan. Nilai PSNR pada target uji coba 25 dB mengalami kenaikan rata-rata 0,91 dB. Sehingga kualitas citra terestorasi memiliki kualitas hampir sama dengan citra terkompresi. Tetapi secara teori proses restorasi dengan menggunakan metode *iterative lanczos hybrid regularization* berhasil mengurangi derau pada citra yang diperoleh dari hasil kompresi dengan metode *Set Partitioning In Hierarchical Trees* (SPIHT).



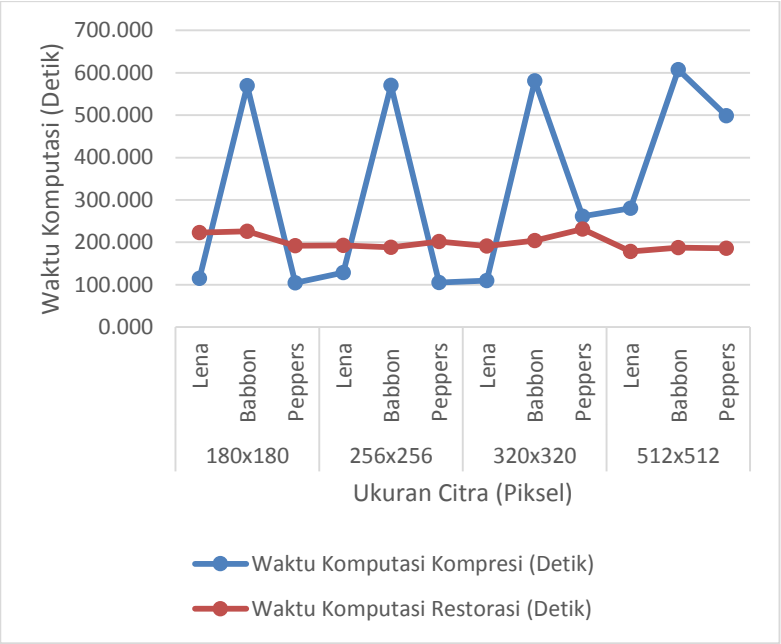
Gambar 5.20 Grafik Kenaikan Nilai PSNR (35 dB)

Berdasarkan gambar 5.20 nilai *Peak Signal Noise to Ratio* (PSNR) tidak begitu mengalami kenaikan secara signifikan. Nilai PSNR pada target uji coba 35 dB mengalami kenaikan rata-rata 0,57 dB. Sehingga kualitas citra terestorasi memiliki kualitas hampir sama dengan citra terkompresi. Tetapi secara teori proses restorasi dengan menggunakan metode *iterative lanczos hybrid regularization* berhasil mengurangi derau pada citra yang diperoleh dari hasil kompresi dengan metode *Set Partitioning In Hierarchical Trees* (SPIHT)



Gambar 5.21 Grafik Waktu Komputasi pada Uji Coba Citra dengan besar PSNR 25 dB

Berdasarkan gambar 5.21 waktu komputasi yang diperlukan saat merestorasi citra lebih lama dikarenakan citra mengandung banyak derau. Untuk nilai PSNR 25 dB yang tergolong citra *Inferior* atau banyak derau. Waktu komputasi yang diperlukan untuk menyelesaikan proses restorasi citra dengan nilai PSNR 25 dB (*Inferior*) rata-rata sekitar 187,058 detik lebih lambat dari proses kompresi citra. Tetapi hasil citra terestorasi mengalami sedikit kenaikan PSNR sehingga kualitas citra hampir mirip dengan citra terkompresi.



Gambar 5.22 Grafik Waktu Komputasi pada Uji Coba Citra dengan besar PSNR 35 dB

Berdasarkan gambar 5.22 waktu komputasi yang diperlukan saat merestorasi citra lebih cepat dibandingkan saat kompresi dikarenakan citra mengandung sedikit derau berupa butiran halus. Untuk nilai PSNR 35 dB tergolong citra *Marginal* atau terdapat sedikit butiran halus pada citra. Waktu komputasi yang diperlukan untuk menyelesaikan proses restorasi citra dengan nilai PSNR 35 dB (*Marginal*) rata-rata sekitar 127,418 detik lebih cepat dari proses kompresi citra. Tetapi hasil citra terestorasi mengalami sedikit kenaikan PSNR sehingga kualitas citra hampir mirip dengan citra terkompresi.

LAMPIRAN A

Kode Program Proses Antarmuka Awal

```
function varargout = home(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename,
...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@home_OpeningFcn, ...
                   'gui_OutputFcn',
@home_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function home_OpeningFcn(hObject, eventdata,
handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
ITS=imread('ITS.jpg');
axes(handles.axes1);
imshow(ITS);

MAT=imread('MAT.jpg');
axes(handles.axes2);
imshow(MAT);
```

LAMPIRAN A (LANJUTAN)

```
function varargout = home_OutputFcn(hObject,
eventdata, handles)
varargout{1} = handles.output;
function home_Callback(hObject, eventdata,
handles)
home
function Compression_gui_Callback(hObject,
eventdata, handles)
Compression_gui
function dekompresi_Callback(hObject, eventdata,
handles)
dekompresi
function restoration_Callback(hObject,
eventdata, handles)
restoration
function keluar_Callback(hObject, eventdata,
handles)
respon=keluar('Title','Konfirmasi Keluar');
switch lower(respon)
case 'tidak'
%tidak ada aksi
case 'ya'
close
end

function varargout = Compression_gui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename,
...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@Compression_gui_OpeningFcn, ...
                   'gui_OutputFcn',
@Compression_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
```

LAMPIRAN A (LANJUTAN)

```
                    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function Compression_gui_OpeningFcn(hObject,
eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout =
Compression_gui_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;

function sel_img_Callback(hObject, eventdata,
handles)
global Orig_I;
[file path]=uigetfile('*.bmp');
Orig_I=imread(file);
[r c p]=size(Orig_I);
 if(p==3)
     Orig_I=rgb2gray(Orig_I);
 else
     Orig_I=Orig_I;
 end
my_name = cat(2,path,file);
propertise=imfinfo(my_name);

set(handles.edit6,'string',propertise.Height)
set(handles.edit7,'string',propertise.Width)
```

LAMPIRAN A (LANJUTAN)

```
set(handles.edit8,'string',propertise.FileSize)
set(handles.edit9,'string',propertise.Filename)
set(handles.edit10,'string',propertise.BitDepth)
set(handles.edit11,'string',propertise.Format)


axes(handles.axes1)
imshow(Orig_I)
[row1 col1]=size(Orig_I);

Orig_I=imresize(Orig_I,[512 512]);

handles.Orig_I=Orig_I;
handles.row1=row1;
handles.col1=col1;
guidata(hObject, handles);

function compress_Callback(hObject, eventdata,
handles)
set(handles.figure1, 'pointer', 'watch')
drawnow;

tic
Orig_I=handles.Orig_I;

Pt=str2double(get(handles.edit1,'string'));
P=0;
    OrigSize = size(Orig_I, 1);
    [nRow, nColumn] = size(Orig_I);
    type = 'bior6.8';
    [Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(type);
    row1=handles.row1;
    col1=handles.col1;

    for rate=0.1:0.1:0.9
```

LAMPIRAN A (LANJUTAN)

```
    for level=1:9

    max_bits = floor(rate * OrigSize^2);

    [I_W, S] = func_DWT(Orig_I, level, Lo_D,
Hi_D);

    img_enc = func_SPIHT_Enc(I_W, max_bits,
nRow*nColumn, 5);

    [No_of_bits,len_seq]=dec2bin_con(img_enc);

    comp_ratio=(nRow*nColumn*8)/len_seq;
    comp_ratio1=num2str(comp_ratio);
    set(handles.edit12,'string',comp_ratio1)

    img_dec = func_SPIHT_Dec(img_enc);
    img_ebcot = func_InvDWT(img_dec, S, Lo_R,
Hi_R, level);

    hasilPSNR = PSNRCom(Orig_I,img_ebcot);
    P=hasilPSNR;
    P
    if P>Pt
        break;
    end
    end
    if P>Pt
        break;
    end

    end

p1=num2str(P);
set(handles.edit15,'string',p1)
```

LAMPIRAN A (LANJUTAN)

```
img_ebcot=imresize(img_ebcot,[row1 col1]);

axes(handles.axes2)
imshow(img_ebcot,[])

t=toc;
set(handles.edit19,'string',t)
set(handles.figure1, 'pointer', 'arrow')
handles.img_ebcot=img_ebcot;

guidata(hObject, handles);

function clear_all_Callback(hObject, eventdata,
handles)
a=[];
set(handles.edit1,'string',a)
set(handles.edit6,'string',a)
set(handles.edit7,'string',a)
set(handles.edit8,'string',a)
set(handles.edit9,'string',a)
set(handles.edit10,'string',a)
set(handles.edit11,'string',a)
set(handles.edit12,'string',a)
set(handles.edit15,'string',a)
set(handles.edit19,'string',a)

axes(handles.axes1);cla
axes(handles.axes2);cla
clc

function pushbutton8_Callback(hObject,
eventdata, handles)
img_ebcot=handles.img_ebcot;
[file path]=uiputfile('*.bmp');
imwrite(uint8(img_ebcot),[path file])
```

LAMPIRAN A (LANJUTAN)

```
function pushbutton6_Callback(hObject,
eventdata, handles)
 [file path]=uigetfile('*.txt');


fiddec=fopen(file,'r');
img_enc=fscanf(fiddec,'%d');
fclose(fiddec);
img_enc=img_enc';

handles.img_enc=img_enc;
guidata(hObject, handles);

function pushbutton7_Callback(hObject,
eventdata, handles)
a=[];
set(handles.edit14,'string',a)
set(handles.edit15,'string',a)
set(handles.edit16,'string',a)
set(handles.edit17,'string',a)
set(handles.edit18,'string',a)
set(handles.edit19,'string',a)

axes(handles.axes2);cla
clc;
clear all;

function keluar_Callback(hObject, eventdata,
handles)
respon=keluar('Title','Konfirmasi Keluar');
switch lower(respon)
case 'tidak'
%tidak ada aksi
case 'ya'
close
```

LAMPIRAN A (LANJUTAN)

```
end
function varargout = restoration(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename,
...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @restoration_OpeningFcn,
...
    'gui_OutputFcn',  @restoration_OutputFcn,
...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function restoration_OpeningFcn(hObject,
eventdata, handles, varargin)
handles.output = hObject;

% deklarasi
path(path,'RestoreTools/Classes')
path(path,'RestoreTools/IterativeMethods')
path(path,'RestoreTools/DirectMethods')

guidata(hObject, handles);

function pushbutton2_Callback(hObject,
eventdata, handles)
```

LAMPIRAN A (LANJUTAN)

```
[file path] = uigetfile('*.bmp');
Orig_R = imread(file);
my_name = cat(2, path, file);
propertise = imfinfo(my_name);

set(handles.edit5, 'string', propertise.Height)
set(handles.edit6,'string', propertise.Width)
set(handles.edit7,'string', propertise.FileSize)
set(handles.edit8,'string', propertise.Filename)
set(handles.edit9,'string', propertise.BitDepth)
set(handles.edit10,'string',propertise.Format)

axes(handles.axes1)
imshow(Orig_R)
[row1 col1]=size(Orig_R);

Orig_R=imresize(Orig_R,[512 512]);
handles.Orig_R = Orig_R;

guidata(hObject, handles);

function pushbutton3_Callback(hObject,
eventdata, handles)
a=[];
set(handles.edit5,'string',a)
set(handles.edit6,'string',a)
set(handles.edit7,'string',a)
set(handles.edit8,'string',a)
set(handles.edit9,'string',a)
set(handles.edit10,'string',a)
set(handles.edit11,'string',a)
set(handles.edit19,'string',a)
axes(handles.axes1);
cla
axes(handles.axes2);
cla
```

LAMPIRAN A (LANJUTAN)

```
clc
function pushbutton5_Callback(hObject,
eventdata, handles)
thresh = handles.Y_I;
[file, path]=uiputfile('*.bmp');
A(:,:,1) = uint8(thresh);
A(:,:,2) = uint8(thresh);
A(:,:,3) = uint8(thresh);

imwrite(A, [path file]);

guidata(hObject, handles);
```

LAMPIRAN B

Kode Program Kompresi Citra

```
function [I_W , S] = func_DWT(I, level, Lo_D,
Hi_D)
[C,S] = func_Mywavedec2(I,level,Lo_D,Hi_D);

S(:,3) = S(:,1).*S(:,2);
L = length(S);

I_W = zeros(S(L,1),S(L,2));

I_W( 1:S(1,1) , 1:S(1,2) ) =
reshape(C(1:S(1,3)),S(1,1:2));

for k = 2 : L-1
    rows = [sum(S(1:k-1,1))+1:sum(S(1:k,1))];
    columns = [sum(S(1:k-1,2))+1:sum(S(1:k,2))];
    % horizontal part
    c_start = S(1,3) + 3*sum(S(2:k-1,3)) + 1;
    c_stop = S(1,3) + 3*sum(S(2:k-1,3)) +
S(k,3);
    I_W( 1:S(k,1) , columns ) = reshape(
C(c_start:c_stop) , S(k,1:2) );

    % vertical part
    c_start = S(1,3) + 3*sum(S(2:k-1,3)) +
S(k,3) + 1;
    c_stop = S(1,3) + 3*sum(S(2:k-1,3)) +
2*S(k,3);
    I_W( rows , 1:S(k,2) ) = reshape(
C(c_start:c_stop) , S(k,1:2) );

    % diagonal part
    c_start = S(1,3) + 3*sum(S(2:k-1,3)) +
2*S(k,3) + 1;
    c_stop = S(1,3) + 3*sum(S(2:k,3));
```

LAMPIRAN B (LANJUTAN)

```
    I_W( rows , columns ) = reshape(
C(c_start:c_stop) , S(k,1:2) );
end
-------------------------------------------------
function im_rec = func_InvDWT(I_W, S, Lo_R,
Hi_R, level)

L = length(S);

m = I_W;

C1 = zeros(1,S(1,3)+3*sum(S(2:L-1,3)));

% approx part
C1(1:S(1,3)) = reshape( m( 1:S(1,1) , 1:S(1,2)
), 1 , S(1,3) );

for k = 2:L-1
    rows = [sum(S(1:k-1,1))+1:sum(S(1:k,1))];
    columns = [sum(S(1:k-1,2))+1:sum(S(1:k,2))];
    % horizontal part
    c_start = S(1,3) + 3*sum(S(2:k-1,3)) + 1;
    c_stop = S(1,3) + 3*sum(S(2:k-1,3)) +
S(k,3);
    C1(c_start:c_stop) = reshape( m( 1:S(k,1) ,
columns ) , 1, c_stop-c_start+1);
    % vertical part
    c_start = S(1,3) + 3*sum(S(2:k-1,3)) +
S(k,3) + 1;
    c_stop = S(1,3) + 3*sum(S(2:k-1,3)) +
2*S(k,3);
    C1(c_start:c_stop) = reshape( m( rows ,
1:S(k,2) ) , 1 , c_stop-c_start+1 );
    % diagonal part
    c_start = S(1,3) + 3*sum(S(2:k-1,3)) +
2*S(k,3) + 1;
```

LAMPIRAN B (LANJUTAN)

```
    c_stop = S(1,3) + 3*sum(S(2:k,3));
    C1(c_start:c_stop) = reshape( m( rows ,
columns ) , 1 , c_stop-c_start+1);
end

if (( L - 2) > level)   %set those coef. in
higher scale to 0
    temp = zeros(1, length(C1) -
(S(1,3)+3*sum(S(2:(level+1),3))));
    C1(S((level+2),3)+1 : length(C1)) = temp;
end

S(:,3) = [];

im_rec = func_Mywaverec2(C1,S, Lo_R, Hi_R);
---------------------------------------------------
function a = func_Myappcoef2(c,s,varargin)
if
errargn(mfilename,nargin,[3:5],nargout,[0:1]),
error('*'), end
rmax = size(s,1);
nmax = rmax-2;
if ischar(varargin{1})
    [Lo_R,Hi_R] = wfilters(varargin{1},'r');
next = 2;
else
    Lo_R = varargin{1}; Hi_R = varargin{2};
next = 3;
end
if nargin>=(2+next) , n = varargin{next}; else,
n = nmax; end

if (n<0) | (n>nmax) | (n~=fix(n))
    errargt(mfilename,'invalid level
value','msg'); error('*');
end
```

LAMPIRAN B (LANJUTAN)

```
nl   = s(1,1);
nc   = s(1,2);
a    = zeros(nl,nc);
a(:) = c(1:nl*nc);

rm   = rmax+1;
for p=nmax:-1:n+1
    [h,v,d] = detcoef2('all',c,s,p);
    a = idwt2(a,h,v,d,Lo_R,Hi_R,s(rm-
p,:),'mode','per');
end
-------------------------------------------------
function value = func_MyDescendant(i, j, type,
m)

s = size(m,1);

S = [];

index = 0; a = 0; b = 0;

while ((2*i-1)<s & (2*j-1)<s)
    a = i-1; b = j-1;

    mind = [2*(a+1)-1:2*(a+2^index)];
    nind = [2*(b+1)-1:2*(b+2^index)];


    chk = mind <= s;
    len = sum(chk);
    if len < length(mind)
        mind(len+1:length(mind)) = [];
    end
```

LAMPIRAN B (LANJUTAN)

```
    chk = nind <= s;
    len = sum(chk);
    if len < length(nind)
        nind(len+1:length(nind)) = [];
    end

    S = [S reshape(m(mind,nind),1,[])];

    index = index + 1;
    i = 2*a+1; j = 2*b+1;
end

if type == 1
    S(:,1:4) = [];;
end

value = max(abs(S));
-------------------------------------------------
function [c,s] = func_Mywavedec2(x,n,varargin)
if
errargn(mfilename,nargin,[3:4],nargout,[0:2]),
error('*'), end
if errargt(mfilename,n,'int'), error('*'), end
if nargin==3
    [Lo_D,Hi_D] = wfilters(varargin{1},'d');
else
    Lo_D = varargin{1};   Hi_D = varargin{2};
end

% Initialization.
s = [size(x)];
c = [];

for i=1:n
```

LAMPIRAN B (LANJUTAN)

```
    [x,h,v,d] = dwt2(x,Lo_D,Hi_D,'mode','per');
% decomposition
    c = [h(:)' v(:)' d(:)' c];     % store
details
    s = [size(x);s];               % store size

end

% Last approximation.
c = [x(:)' c];
s = [size(x);s];
-------------------------------------------------
function x = func_Mywaverec2(c,s,varargin)
if
errargn(mfilename,nargin,[3:4],nargout,[0:1]),
error('*'), end
x = func_Myappcoef2(c,s,varargin{:},0);
-------------------------------------------------
function m = func_SPIHT_Dec(in)
m = zeros(in(1,1));
n_max = in(1,2);
level = in(1,3);
ctr = 4;

%-----------   Initialize LIP, LSP, LIS   ------
----------
temp = [];
bandsize = 2.^(log2(in(1,1)) - level + 1);
temp1 = 1 : bandsize;
for i = 1 : bandsize
    temp = [temp; temp1];
end
LIP(:, 1) = temp(:);
temp = temp';
LIP(:, 2) = temp(:);
```

LAMPIRAN B (LANJUTAN)

```
LIS(:, 1) = LIP(:, 1);
LIS(:, 2) = LIP(:, 2);
LIS(:, 3) = zeros(length(LIP(:, 1)), 1);
pstart = 1;
pend = bandsize / 2;
for i = 1 : bandsize / 2
    LIS(pstart : pend, :) = [];
    pdel = pend - pstart + 1;
    pstart = pstart + bandsize - pdel;
    pend = pend + bandsize - pdel;
end
LSP = [];

%-----------   coding   ----------------
n = n_max;
while (ctr <= size(in,2))

    %Sorting Pass
    LIPtemp = LIP; temp = 0;
    for i = 1:size(LIPtemp,1)
        temp = temp+1;
        if ctr > size(in,2)
            return
        end
        if in(1,ctr) == 1
            ctr = ctr + 1;
            if in(1,ctr) > 0
                m(LIPtemp(i,1),LIPtemp(i,2)) =
2^n + 2^(n-1);
            else
                m(LIPtemp(i,1),LIPtemp(i,2)) = -
2^n  - 2^(n-1);
            end
            LSP = [LSP; LIPtemp(i,:)];
            LIP(temp,:) = []; temp = temp - 1;
        end
        ctr = ctr + 1;
```

LAMPIRAN B (LANJUTAN)

```
    end

    LIStemp = LIS; temp = 0; i = 1;
    while ( i <= size(LIStemp,1))
        temp = temp + 1;
        if ctr > size(in,2)
            return
        end
        if LIStemp(i,3) == 0
            if in(1,ctr) == 1
                ctr = ctr + 1;
                x = LIStemp(i,1); y =
LIStemp(i,2);

                if ctr > size(in,2)
                    return
                end
                if in(1,ctr) == 1
                    LSP = [LSP; 2*x-1 2*y-1];
                    ctr = ctr + 1;
                    if in(1,ctr) == 1
                        m(2*x-1,2*y-1) = 2^n +
2^(n-1);
                    else
                        m(2*x-1,2*y-1) = -2^n  -
2^(n-1);
                    end
                    ctr = ctr + 1;
                else
                    LIP = [LIP; 2*x-1 2*y-1];
                    ctr = ctr + 1;
                end

                if ctr > size(in,2)
                    return
                end
```

LAMPIRAN B (LANJUTAN)

```
                if in(1,ctr) == 1
                    ctr = ctr + 1;
                    LSP = [LSP; 2*x-1 2*y];
                    if in(1,ctr) == 1;
                        m(2*x-1,2*y) = 2^n +
2^(n-1);
                    else
                        m(2*x-1,2*y) = -2^n  -
2^(n-1);
                    end
                    ctr = ctr + 1;
                else
                    LIP = [LIP; 2*x-1 2*y];
                    ctr = ctr + 1;
                end

                if ctr > size(in,2)
                    return
                end
                if in(1,ctr) == 1
                    ctr = ctr + 1;
                    LSP = [LSP; 2*x 2*y-1];
                    if in(1,ctr) == 1
                        m(2*x,2*y-1) = 2^n +
2^(n-1);
                    else
                        m(2*x,2*y-1) = -2^n  -
2^(n-1);
                    end
                    ctr = ctr + 1;
                else
                    LIP = [LIP; 2*x 2*y-1];
                    ctr = ctr + 1;
                end

                if ctr > size(in,2)
                    return
```

LAMPIRAN B (LANJUTAN)

```
                end
                if in(1,ctr) == 1
                    ctr = ctr + 1;
                    LSP = [LSP; 2*x 2*y];
                    if in(1,ctr) == 1
                        m(2*x,2*y) = 2^n + 2^(n-
1);
                    else
                        m(2*x,2*y) = -2^n  -
2^(n-1);
                    end
                    ctr = ctr + 1;
                else
                    LIP = [LIP; 2*x 2*y];
                    ctr = ctr + 1;
                end

                if ((2*(2*x)-1) < size(m) &
(2*(2*y)-1) < size(m))
                    LIS = [LIS; LIStemp(i,1)
LIStemp(i,2) 1];
                    LIStemp = [LIStemp;
LIStemp(i,1) LIStemp(i,2) 1];
                end
                LIS(temp,:) = []; temp = temp-1;

            else
                ctr = ctr + 1;
            end
        else
            if in(1,ctr) == 1
                x = LIStemp(i,1); y =
LIStemp(i,2);
                LIS = [LIS; 2*x-1 2*y-1 0; 2*x-1
2*y 0; 2*x 2*y-1 0; 2*x 2*y 0];
                LIStemp = [LIStemp; 2*x-1 2*y-1
0; 2*x-1 2*y 0; 2*x 2*y-1 0; 2*x 2*y 0];
```

LAMPIRAN B (LANJUTAN)

```
                LIS(temp,:) = []; temp = temp -
1;
            end
            ctr = ctr + 1;
        end
        i = i+1;
    end

    % Refinement Pass
    temp = 1;
    value = m(LSP(temp,1), LSP(temp,2));
    while (abs(value) >= 2^(n+1) & (temp <=
size(LSP,1)))
        if ctr > size(in,2)
            return
        end

        value = value + ((-1)^(in(1,ctr) + 1)) *
(2^(n-1))*sign(m(LSP(temp,1),LSP(temp,2)));
        m(LSP(temp,1),LSP(temp,2)) = value;
        ctr = ctr + 1;
        temp = temp + 1;
        if temp <= size(LSP,1)
            value = m(LSP(temp,1),LSP(temp,2));
        end
    end

    n = n-1;
end
-------------------------------------------------
function out = func_SPIHT_Enc(m, max_bits,
block_size, level)
%-----------   Initialization  -----------------
bitctr = 0;
out = 2*ones(1,max_bits);
n_max = floor(log2(abs(max(max(m)'))));
```

LAMPIRAN B (LANJUTAN)

```
Bits_Header = 0;
Bits_LSP = 0;
Bits_LIP = 0;
Bits_LIS = 0;

%-----------   output bit stream header   ------
----------
% image size, number of bit plane, wavelet
decomposition level should be
% written as bit stream header.
out(1,[1 2 3]) = [size(m,1) n_max level]; bitctr
= bitctr + 24;
index = 4;
Bits_Header = Bits_Header + 24;

%-----------   Initialize LIP, LSP, LIS   ------
----------
temp = [];
bandsize = 2.^(log2(size(m, 1)) - level + 1);
temp1 = 1 : bandsize;
for i = 1 : bandsize
    temp = [temp; temp1];
end
LIP(:, 1) = temp(:);
temp = temp';
LIP(:, 2) = temp(:);
LIS(:, 1) = LIP(:, 1);
LIS(:, 2) = LIP(:, 2);
LIS(:, 3) = zeros(length(LIP(:, 1)), 1);
pstart = 1;
pend = bandsize / 2;
for i = 1 : bandsize / 2
    LIS(pstart : pend, :) = [];
    pdel = pend - pstart + 1;
    pstart = pstart + bandsize - pdel;
    pend = pend + bandsize - pdel;
end
```

LAMPIRAN B (LANJUTAN)

```
LSP = [];

n = n_max;

%-----------   coding   ----------------
while(bitctr < max_bits)

    % Sorting Pass
    LIPtemp = LIP; temp = 0;
    for i = 1:size(LIPtemp,1)
        temp = temp+1;
        if (bitctr + 1) >= max_bits
            if (bitctr < max_bits)
                out(length(out))=[];
            end
            return
        end
        if abs(m(LIPtemp(i,1),LIPtemp(i,2))) >=
2^n % 1: positive; 0: negative
            out(index) = 1; bitctr = bitctr + 1;
            index = index +1; Bits_LIP =
Bits_LIP + 1;
            sgn =
m(LIPtemp(i,1),LIPtemp(i,2))>=0;
            out(index) = sgn; bitctr = bitctr +
1;
            index = index +1; Bits_LIP =
Bits_LIP + 1;
            LSP = [LSP; LIPtemp(i,:)];
            LIP(temp,:) = []; temp = temp - 1;
        else
            out(index) = 0; bitctr = bitctr + 1;
            index = index +1;
            Bits_LIP = Bits_LIP + 1;
        end
    end
```

LAMPIRAN B (LANJUTAN)

```
    LIStemp = LIS; temp = 0; i = 1;
    while ( i <= size(LIStemp,1))
        temp = temp + 1;
        if LIStemp(i,3) == 0
            if bitctr >= max_bits
                return
            end
            max_d =
func_MyDescendant(LIStemp(i,1),LIStemp(i,2),LISt
emp(i,3),m);
            if max_d >= 2^n
                out(index) = 1; bitctr = bitctr
+ 1;
                index = index +1; Bits_LIS =
Bits_LIS + 1;
                x = LIStemp(i,1); y =
LIStemp(i,2);

                if (bitctr + 1) >= max_bits
                    if (bitctr < max_bits)
                        out(length(out))=[];
                    end
                    return
                end
                if abs(m(2*x-1,2*y-1)) >= 2^n
                    LSP = [LSP; 2*x-1 2*y-1];
                    out(index) = 1; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    sgn = m(2*x-1,2*y-1)>=0;
                    out(index) = sgn; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                else
```

LAMPIRAN B (LANJUTAN)

```
                    out(index) = 0; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    LIP = [LIP; 2*x-1 2*y-1];
                end

                if (bitctr + 1) >= max_bits
                    if (bitctr < max_bits)
                        out(length(out))=[];
                    end
                    return
                end
                if abs(m(2*x-1,2*y)) >= 2^n
                    LSP = [LSP; 2*x-1 2*y];
                    out(index) = 1; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    sgn = m(2*x-1,2*y)>=0;
                    out(index) = sgn; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                else
                    out(index) = 0; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    LIP = [LIP; 2*x-1 2*y];
                end

                if (bitctr + 1) >= max_bits
                    if (bitctr < max_bits)
                        out(length(out))=[];
                    end
                    return
```

LAMPIRAN B (LANJUTAN)

```
                end
                if abs(m(2*x,2*y-1)) >= 2^n
                    LSP = [LSP; 2*x 2*y-1];
                    out(index) = 1; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    sgn = m(2*x,2*y-1)>=0;
                    out(index) = sgn; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                else
                    out(index) = 0; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    LIP = [LIP; 2*x 2*y-1];
                end

                if (bitctr + 1) >= max_bits
                    if (bitctr < max_bits)
                        out(length(out))=[];
                    end
                    return
                end
                if abs(m(2*x,2*y)) >= 2^n
                    LSP = [LSP; 2*x 2*y];
                    out(index) = 1; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    sgn = m(2*x,2*y)>=0;
                    out(index) = sgn; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
```

LAMPIRAN B (LANJUTAN)

```
                else
                    out(index) = 0; bitctr =
bitctr + 1;
                    index = index +1; Bits_LIS =
Bits_LIS + 1;
                    LIP = [LIP; 2*x 2*y];
                end

                if ((2*(2*x)-1) < size(m) &
(2*(2*y)-1) < size(m))
                    LIS = [LIS; LIStemp(i,1)
LIStemp(i,2) 1];
                    LIStemp = [LIStemp;
LIStemp(i,1) LIStemp(i,2) 1];
                end
                LIS(temp,:) = []; temp = temp-1;

            else
                out(index) = 0; bitctr = bitctr
+ 1;
                index = index +1; Bits_LIS =
Bits_LIS + 1;
            end
        else
            if bitctr >= max_bits
                return
            end
            max_d =
func_MyDescendant(LIStemp(i,1),LIStemp(i,2),LISt
emp(i,3),m);
            if max_d >= 2^n
                out(index) = 1; bitctr = bitctr
+ 1;
                index = index +1;
                x = LIStemp(i,1); y =
LIStemp(i,2);
```

LAMPIRAN B (LANJUTAN)

```
                LIS = [LIS; 2*x-1 2*y-1 0; 2*x-1
2*y 0; 2*x 2*y-1 0; 2*x 2*y 0];
                LIStemp = [LIStemp; 2*x-1 2*y-1
0; 2*x-1 2*y 0; 2*x 2*y-1 0; 2*x 2*y 0];
                LIS(temp,:) = []; temp = temp -
1;
            else
                out(index) = 0; bitctr = bitctr
+ 1;
                index = index +1; Bits_LIS =
Bits_LIS + 1;
            end
        end
        i = i+1;
    end

    % Refinement Pass
    temp = 1;
    value = floor(abs(2^(n_max-
n+1)*m(LSP(temp,1),LSP(temp,2))));
    while (value >= 2^(n_max+2) & (temp <=
size(LSP,1)))
        if bitctr >= max_bits
            return
        end
        s = bitget(value,n_max+2);
        out(index) = s; bitctr = bitctr + 1;
        index = index +1; Bits_LSP = Bits_LSP +
1;
        temp = temp + 1;
        if temp <= size(LSP,1)
            value = floor(abs(2^(n_max-
n+1)*m(LSP(temp,1),LSP(temp,2))));
        end
    end
    n = n - 1;
end
```

LAMPIRAN B (LANJUTAN)

```
------------------------------------------------
function psnr= PSNRCom(Orig_I,img_ebcot)
%Calculates the Peak-to-peak Signal to Noise
Ratio of two images X and Y
[M,N]=size(Orig_I);
m=double(0);
Orig_I=cast(Orig_I,'double');
img_ebcot=cast(img_ebcot,'double');
for i=1:M
    for j=1:N
        m=m+((Orig_I(i,j)-img_ebcot(i,j))^2);
    end
end
m=m/(M*N);
psnr=(10*log10(255*255/m));
```

LAMPIRAN B (LANJUTAN)

LAMPIRAN C

Kode Program Restorasi Citra

```
function [x_out, output] = HyBR (A, b, P)
%HyBR didefinisikan sebagai Metode Lanczos -
Hybrid Regularization
%1. Metode Iteratif Lanczos Bidiagonalization
(LBD)
%2. Regularisasi Tikhonov dengan WGCV

%Inisialisasi
if nargin < 2
    error ('HyBR: Not Enough Input')
elseif nargin < 3
    P = [];
end
[m,n] = size (A);
maxiter = 300;
regstart = 2;
degflat = 10^-6;

bSize = size(b);
b = b(:);
A.imsize = bSize;

outputparams = nargout > 1;
if outputparams
    output.iterations = maxiter;
    output.GCVstop = [];
    output.B = [];
    output.flag = 3;
end

if isempty (P)
    beta = norm (b); U = (1/beta)*b;
    handle = @LBD;
else
    U=P\b;
```

LAMPIRAN C (LANJUTAN)

```
    beta = norm(U); U = U/beta;
    handle = @PLBD;
end

%maincode
B = []; V = []; GCV = []; Omega = []; omegaTemp
= 0;
terminate = 1; warning = 0;

h = waitbar (0, 'Beginning iterations : please
wait . . .');

for i = 1:maciter + 1
    [U,B,V] = feval(handle, A, U, B, V, P);
    vector = (beta(1)*eye(size(U,2),1));

    if i >= 2 %Mulai iterasi Lanczos
        [Ub, Sb, Vb] = svd (B);

omega(i-1) = min(1, findomega(Ub'*vector,
diag(Sb)));
omegaTemp = mean(Omega);

%Solve the projected problem dengan Tikhonov
[f, alpha] = Tikhonovsolver (Ub, diag(Sb), Vb,
vector, omegaTemp);

%Hitung stopping kriteria dengan fungsi GCV
GCV(i-1) = GCVstopfun(alpha,
Ub(1,:)',diag(Sb),beta,m,n);

%Penghentian iterasi
if i>2 && terminate

%Flatness pada fungsi GCV
if abs((GCV(i-1)-GCV(i-2)))/GCV(1) < degflat
    x_out = V*f;
```

LAMPIRAN C (LANJUTAN)

```
    if outputparams
        output.B = B;
        output.GCVstop = GCV(:);
        output.iterations = i-1;
        output.flag = 1;
    end
    close (h)

    x_out = reshape (x_out, bSize);
    return;
    terminate = 0;

%Perilaku semi konvergensi maka harus dihentikan
elseif warning && lenght GCV > iterations_save +
3
%Grafiknya mengalami perilaku semi konvergen,
sehingga harus stop
if GCV (iterations_save) < GCV
(iterations_save+1:end)
    x_out = x_save;
    if outputparams
        output.B = B;
        output.GCVstop = GCV (:);
        output.iterations = iterations_save;
        output.flag = 2;
    end
    close (h)
    x_out = reshape (x_out, bSize);
    return;
    terminate = 0; %Solution is already found

else %Hanya sedikit naikan (bump)
    warning = 0;
    x_out = [];
    iterations_save = maxiter;
end
```

LAMPIRAN C (LANJUTAN)

```
%Tidak ada warning
elseif ~warning
    %Pengecekan bila nilai GCV sebelumnya lebih
kecil daripada sekarang
    if GCV (i-2) < GCV (i-1)
        warning = 1;
        x_save = V*f;
        iterations_save = i-1;
    end
end
end
 x = V*f;

    end
    waitbar (i/(maxiter+1), h)
end
close (h)
-------------------------------------------------
function [U, B, V] = LBD (A, U, B, V ,P)
%menghitung Lanczos Bidiagonalization tiap
iterasi

[m,k] = size(U);
if k == 1
    v = A'*U(:,k);
else
    v = A'*U(:,k) - B(k, k-1)*V(:,k-1);
end

alpha = norm(v);
v = v/alpha;
u = A*v - alpha*U(:,k);

beta = norm(u);
u = u/beta;
```

LAMPIRAN C (LANJUTAN)

```
U=[U,u]; %U dan V orthogonal matrix dari iterasi
Lanczos Bidiogonalization
V = [V,v];

%matriz bidiagonalization dari iterasi Lanczos
Bidiagonalization
B=[B,[zeros(k-1,1);alpha];[zeros(1,k-1),beta]];
-------------------------------------------------
function G = TikGCCV (alpha, bhat, s, omega)
%Menghitung Fungsi WGCV untuk Regularisasi
Tikhonov
%bhat = vector U'*b
%s = vektor yang mengandung nilai singular
if nargin == 3
    omega = [];
end
if isempty (omega)
    omega = 1;
end

m = length(bhat);
n = length (s);
t0 = sum(abs(bhat(n+1:m)).^2);

s2 = abs(s).^2;
alpha2 = alpha^2;
tt= 1./ (s2 + alpha2);
t1 = alpha2.*tt;
t2 = abs(bhat/(1:n) .* t1) .^2;
t3 = t1 + (1-omega)*s2 .* tt;

G = n*(sum(t2) +t0) / (sum(t3) + m - n)^2;
-------------------------------------------------
function [f,alpha] = Tikhonovsolver(U , S, V, b,
omega)
```

LAMPIRAN C (LANJUTAN)

```
%Menghitung projected problem dengan
regularisasi tikhonov
%Input : U,S,V - SVD dari matrix Bk. b - Beta*el
%omega - parameter "weight" pada WGCV

bhat = U'*b;
bhat = bhat (:);

%parameter regularisasi
alpha = fminbnd ('TikWGCV', 0, 1, [], bhat, S,
omega);

D = abs(S).^2 + alpha^2;
bhat = conj(S) .* bhat(1:length (S));
xhat = bhat./D;
f =  V*xhat; %hasil dari projected problem (f
lambda)
-------------------------------------------------
function omega = findomega (bhat,s)
alpha = s(end);
m = length (bhat);
n = length (s);
t0=sum(abs(bhat(n+1:m)).^2);
s2=abs(s).^2;
alpha2= alpha^2;
tt = 1./(s2+alpha2);
t1=sum(s2.*tt);
t2=abs(bhat(1:n).*alpha.*s).^2;
t3=sum(t2.*abs((tt.^3)));
t4=sum((s.*tt).^2);
t5=sum((abs(alpha2*bhat(1:n).*tt)).^2);
v1=abs(bhat(1:n).*s).^2;
v2=sum(v1.*abs((tt.^3)));

omega= (m*alpha2*v2)/(t1*t3 + t4*(t5+t0));
-------------------------------------------------
```

LAMPIRAN C (LANJUTAN)

```
function G = GCVstopfun (alpha, u, s, beta, m,
n)
k = length (s);
beta2 = beta^2;

s2 = abs(s) .^ 2;
alpha2 = alpha^2;

t1 = 1 ./ (s2 +alpha2);
t2 = abs(alpha2*u(1:k) .* t1) .^2;
t3 = s2 .* t1;

num = n*beta*(sum(t2) + abs(u(k+1))^2);
den = (m -k + sum(alpha2*t1))^2;
G = num / den;
-------------------------------------------------
function psnr= PSNRNew(Image_Res, Orig_I)
%Calculates the Peak-to-peak Signal to Noise
Ratio of two images X and Y
[M,N]=size(Image_Res);
m=double(0);
Image_Res=cast(Image_Res,'double');
Orig_I=cast(Orig_I,'double');
for i=1:M
    for j=1:N
        m=m+((Image_Res(i,j)-Orig_I(i,j))^2);
    end
end
m=m/(M*N);
psnr=(10*log10(255*255/m));
```

LAMPIRAN C (LANJUTAN)

LAMPIRAN D

Data Hasil Kompresi dan Restorasi Citra

Tabel Rekap Nilai Kenaikan PSNR dan Waktu Komputasi pada Proses Uji Coba dengan Nilai PSNR 25 dB

No	Target PSNR (dB)	Ukuran Citra (Piksel)	Nama Citra	Ukuran File Observasi (KB)	Ukuran File Terkompresi (KB)	Selisih Nilai PSNR (dB)	PSNR Kompresi (dB)	PSNR Restorasi (dB)	Kenaikan PSNR (dB)	Waktu Komputasi Kompresi (Detik)	Waktu Komputasi Restorasi (Detik)	Lama Waktu Komputasi (Detik)	Keterangan Waktu Komputasi
1	25 dB	160x160	Lena	95 KB	33 KB	4.15	29.15	29.88	0.73	20.286	237.187	216.901	Lama
2			Baboon	95 KB	33 KB	0.64	25.64	27.14	1.50	22.581	232.405	209.824	Lama
3			Peppers	95 KB	33 KB	4.20	29.20	30.23	1.03	21.054	348.174	327.121	Lama
4		256x256	Lena	192 KB	64 KB	3.16	28.16	28.86	0.70	20.884	220.091	199.207	Lama
5			Baboon	192 KB	64 KB	1.64	26.64	28.35	1.71	105.659	221.258	115.599	Lama
6			Peppers	192 KB	64 KB	3.24	28.24	29.21	0.97	21.056	220.571	199.515	Lama
7		320x320	Lena	301 KB	102 KB	2.84	27.84	28.19	0.35	19.453	266.174	246.721	Lama
8			Baboon	301 KB	102 KB	0.27	25.27	26.66	1.39	111.624	341.697	230.073	Lama
9			Peppers	301 KB	102 KB	2.64	27.64	28.51	0.87	21.035	219.096	198.061	Lama
10		512x512	Lena	767 KB	257 KB	2.26	27.26	27.59	0.33	8.308	217.269	208.961	Lama
11			Baboon	767 KB	257 KB	0.10	25.10	26.22	1.12	326.827	213.779	-113.048	Cepat
12			Peppers	767 KB	257 KB	2.20	27.20	27.37	0.17	20.144	231.311	211.167	Lama
	Rata Selisih Nilai PSNR Citra Asli dengan Terkompresi					2.28	Rata Kenaikan PSNR			0.91	Rata Waktu Komputasi		187.508

LAMPIRAN D (LANJUTAN)

Tabel Rekap Nilai Kenaikan PSNR dan Waktu Komputasi pada Proses Uji Coba dengan Nilai PSNR 35 dB

No	Target PSNR (dB)	Ukuran Citra (Piksel)	Nama Citra	Ukuran File Observasi (KB)	Ukuran File Terkompresi (KB)	Selisih Nilai PSNR (dB)	PSNR Kompresi (dB)	PSNR Restorasi (dB)	KenaiKan PSNR (dB)	Waktu Komputasi Kompresi (Detik)	Waktu Komputasi Restorasi (Detik)	Lama Waktu Komputasi (Detik)	Keterangan Waktu Komputasi
1	35 dB	180x180	Lena	95 KB	33 KB	1.13	36.13	36.59	0.46	115.036	223.088	108.052	Lama
2			Baboon	95 KB	33 KB	0.33	35.33	35.69	0.36	569.648	226.380	-343.268	Cepat
3			Peppers	95 KB	33 KB	1.37	36.37	36.87	0.50	104.300	192.049	87.749	Lama
4		256x256	Lena	192 KB	64 KB	0.69	35.69	36.31	0.62	128.387	193.008	64.621	Lama
5			Baboon	192 KB	64 KB	0.16	35.16	36.11	0.95	570.712	188.531	-382.181	Cepat
6			Peppers	192 KB	64 KB	0.53	35.53	36.30	0.77	105.288	201.861	96.573	Lama
7		320x320	Lena	301 KB	102 KB	1.31	36.31	36.94	0.63	109.541	191.163	81.622	Lama
8			Baboon	301 KB	102 KB	0.08	35.08	35.64	0.56	580.831	204.089	-376.742	Cepat
9			Peppers	301 KB	102 KB	1.33	36.33	37.18	0.85	261.440	231.191	-30.249	Cepat
10		512x512	Lena	767 KB	257 KB	0.27	35.27	35.50	0.23	280.433	178.368	-102.065	Cepat
11			Baboon	767 KB	257 KB	0.01	35.01	35.32	0.31	607.834	187.711	-420.123	Cepat
12			Peppers	767 KB	257 KB	1.26	36.26	36.90	0.64	498.690	185.688	-313.002	Cepat
Rata Selisih Nilai PSNR Citra Asli dengan Terkompresi						0.71	Rata KenaiKan PSNR			0.57	Rata Waktu Komputasi		-127.418

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan penelitian yang telah dilaksanakan. Di samping itu, pada bab ini juga dimasukkan beberapa saran yang dapat digunakan jika penelitian ini ingin dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian yang telah dilakukan terhadap sistem restorasi *iterative lanczos hybrid regularization* berbasis kompresi *set partitioning in hierarchical trees*, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Rata rata dari selisih nilai target PSNR awal 25 dB dengan nilai hasil kompresi SPIHT adalah 2.28 dB. Sedangkan rata rata dari selisih nilai target PSNR awal 35 dB dengan nilai hasil kompresi SPIHT adalah 0.71 dB.
2. Kenaikan nilai PSNR berdasarkan data uji coba nilai PSNR citra terkompresi awal 25 dB diperoleh rata-rata sebesar 0,91 dB. Sedangkan, data uji coba nilai PSNR citra terkompresi awal 35 dB mengalami kenaikan nilai PSNR rata-rata sebesar 0,57 dB.
3. Waktu komputasi yang dibutuhkan untuk menyelesaikan proses restorasi citra dengan data uji coba nilai PSNR awal 25 dB memiliki rata rata 187,058 detik lebih lambat dari proses kompresi citra. Sedangkan, proses restorasi citra dengan data uji coba nilai PSNR citra sebesar 35 dB memiliki rata rata 127,418 detik lebih cepat dari proses kompresi citra.

6.2 Saran

Berdasarkan hasil yang dicapai pada penelitian ini, ada beberapa hal yang penulis sarankan untuk pengembangan selanjutnya yaitu:

1. Hasil dari proses restorasi citra pada sistem ini masih berupa citra grayscale. Untuk penelitian selanjutnya dapat menggunakan citra RGB.
2. Pada proses uji coba, hanya menggunakan variasi ukuran citra dan nilai PSNR. Untuk penelitian selanjutnya data uji coba lebih bervariasi agar hasil restorasi citra lebih signifikan.
3. Pada penelitian ini menggunakan metode kompresi *Set Partitioning In Hierarchical Trees* (SPIHT) sebagai data uji coba. Untuk penelitian kedapannya bisa menggunakan variasi metode kompresi jenis *loosy* untuk membandingkan hasil restorasi citra dengan menggunakan metode *iterative lanczos hybrid regularization.*

DAFTAR PUSTAKA

[1] Chung, J, James G. Nagy dan Dianne P. O'leary. 2008. ***A Weighted-GCV Method For Lanczos-Hybrid Regularization.*** Eelectronic Transaction On Numerical Analysis. Kent State University. Volume 28 pp 149-167

[2] Daubechies, Y. Meyer, and S. Mallat. ***Basicd of Wavelets***. Ten Lectures on Wavelet, Orthonormal Bases of Compactly Supported Wavelets.

[3] J.M. Shapiro. 1993. ***Embedded Image Coding Using Zerotrees of Wavelet Coefficients.*** IEEE Transactions On Signal Processing, Vol. 41, No. 12.

[4] Purwananto, Y, Rully Soelaiman dan Alfa Masjita Rahmat. 2010. ***Restorasi Citra Dengan Menggunakan Metode Iteratif Lanczos-Hybrid Regularization***. Jurusan TeknikInformatika, ITS. RSIf 006.42 Rah r.

[5] Rafael C. Gonzalez, Richard E. Woods. 2003. ***Digital Image Processing (Second Edition)***. Prentice Hall.

[6] Rema, NR, Binnu Ani Oommen, dan Mythili P. 2014. ***Image Compression Using SPIHT with Modified Spatial Orientation Trees.*** Division of Electronics Engineering, School of Engineering CUSAT Kochi-682022. India. Procedia Computer Science 46 (2015) 1732 – 1738.

[7] Said, A Member, IEEE and William A. Pearlman, Senior Member, IEE. 1996. ***A New, Fast, and Efficient Image Codes Based on Set Partitioning in Hierarchical Tress***. IEEE Transactions on Circuit and System for Video Technology, Vol. 6, No. 3, June 1996.

[8] Saikhu, A, Mediana Aryuni dan Fitri Rullianti. 2009. ***Implementasi Metoda Kompresi Set Partitioning In***

Hierarchical Trees Berbasis Wavelet Pada Citra. Undergraduate Thesis, Jurusan Teknik Informatika ITS Surabaya, RSIf 006.042 Rul i.

[9] Zainuddin, Agus dan Aries Pratiarso. 2007. ***Pengkodean Gabungan SPIHT-LPDC Menggunakan Teknik Diversitas***. Jurusan Telekomunikasi, PENS Surabaya.

[10] Zhang H and Shimei DAI. 2012. ***Image Inpainting Based on Wavelet Decompotition***. International Workshop on Information and Electronics Engineering. Procedia Engineering 29, 3674-3678.

[11] Ibrahim, T, Ali Melit dan Fouad Khelifi. ***An improved SPIHT algorithm for Lossless Image Coding***. Departement of Electeonics Enginneering, Faculty of Engineering Science, University of Jijel, B.P. 98 Ouled Aissa, Jijel 18000, Algeria Queen's University Belfast, Northern Ireland, UK. Digital Signal Processing 19 (2009) 220-228.

BIODATA PENULIS

Penulis memiliki nama lengkap Prismahardi Aji Riyantoko, lahir di Lumajang pada tanggal 11 Oktober 1993. Penulis berasal dari Kabupaten Lumajang, bertempat tinggal di Dusun Sarirejo I RT 003 RW 001, Kebonsari, Sumbersuko, Lumajang. Pendidikan formal yang pernah ditempuh yaitu SDN Kebonsari 01, SMPN 1 Lumajang, dan SMAN 2 Lumajang. Kemudian, penulis melanjutkan studi di jurusan Matematika ITS, dengan bidang minat ilmu komputer. Dalam bidang minat ini penulis mulai mengenal bahasa pemrograman diantaranya adalah C, C++, Java, PHP-MySQL, dan MATLAB. Semasa menempuh jenjang pendidikan S-1, penulis juga aktif dalam kegiatan non-akademis diantaranya aktif di organisasi kemahasiswaan Matematika ITS sebagai Staff Departemen Hubungan Luar periode 2013/2014 , Ketua Himpunan Mahasiswa Matematika ITS periode 2014/2015 dan mengikuti kepanitiaan acara besar yang ada di ITS yaitu OMITS 2014 sebagai PJ Regional Website. Selama penulisan Tugas Akhir ini Penulis tidak lepas dari kekurangan, untuk itu penulis mengharapkan kritik, saran, dan pertanyaan mengenai Tugas Akhir ini yang dapat dikirimkan melalui *e-mail* ke prismahardi@gmail.com.