TUGAS AKHIR – K141502

IMPLEMENTASI *CLUSTERING* K-MEANS DALAM DOMAIN *WAVELET* MENGGUNAKAN ADAPTIF *SOFT-THRESHOLDING* UNTUK *DENOISING* CITRA

BIANDINA MEIDYANI
NRP 5112100218

Dosen Pembimbing I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Dosen Pembimbing II
Rully Soelaiman, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – K141502

IMPLEMENTASI *CLUSTERING* K-MEANS DALAM DOMAIN *WAVELET* MENGGUNAKAN ADAPTIF *SOFT-THRESHOLDING* UNTUK *DENOISING* CITRA

BIANDINA MEIDYANI
NRP 5112100218

Dosen Pembimbing I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Dosen Pembimbing II
Rully Soelaiman, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF K-MEANS *CLUSTERING* IN WAVELET DOMAIN USING ADAPTIVE *SOFT-THRESHOLDING* FOR IMAGE *DENOISING*

BIANDINA MEIDYANI
NRP 5112100218

Supervisor I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Supervisor II
Rully Soelaiman, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI *CLUSTERING* K-MEANS DALAM DOMAIN WAVELET MENGGUNAKAN ADAPTIF *SOFT-THRESHOLDING* UNTUK *DENOISING* CITRA

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
BIANDINA MEIDYANI
NRP : 5112 100 218

Disetujui oleh Dosen Pembimbing tugas akhir :

Arya Yudhi Wijaya, S.Kom., M.Kom. .....
NIP: 197906262005012002 (pembimbing 1)

Rully Soelaiman, S.Kom., M.Kom. .....
NIP: 197002131994021001 (pembimbing 2)

SURABAYA
JANUARI 2017

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI *CLUSTERING* K-MEANS DALAM DOMAIN WAVELET MENGGUNAKAN ADAPTIF *SOFT-THRESHOLDING* UNTUK *DENOISING* CITRA

Nama Mahasiswa	:	Biandina Meidyani
NRP	:	5112100218
Jurusan	:	Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	:	Arya Yudhi Wijaya, S.Kom., M.Kom.
Dosen Pembimbing 2	:	Rully Soelaiman, S.Kom., M.Kom.

ABSTRAK

Noise adalah gangguan yang menyebabkan sebuah nilai intensitas piksel tidak mencerminkan nilai yang sebenarnya dihasilkan. Kerusakan yang ditimbulkan dapat berupa menurunnya kualitas gambar atau hilangnya beberapa informasi detail citra. Oleh karena itu, dibutuhkan proses denoising gambar yang bertujuan untuk mereduksi noise yang terdapat pada citra sehingga kualitas citra menjadi lebih baik.

Metode yang akan digunakan pada tugas akhir ini untuk memperbaiki citra yang mengandung noise adalah pemilihan parameter threshold berdasarkan clustering K-Means. Ekstraksi fitur denoising citra ini menggunakan Transformasi Wavelet Diskrit 2-dimensi (DWT). DWT akan mendekomposisikan suatu ruang vektor ke dalam sekumpulan ruang vektor yang berbeda, sehingga dapat dilakukan analisis. Kemudian untuk menghilangkan noise dibantu dengan Adaptif Soft-thresholding. Metode ini akan diterapkan pada citra berwarna yang ditambahkan noise Gaussian. Output dari tugas akhir ini yaitu citra berwarna yang telah berkurang noisenya.

Uji coba dilakukan dengan 8 citra sebagai data tes dan 512 citra sebagai data training. Setiap data tes akan diberikan noise dengan nilai varian yang berbeda-beda. Kemudian akan dihitung PSNR input dan PSNR output beserta MSE. Hasil uji coba

menunjukkan bahwa metode ini dapat digunakan untuk proses denoising citra pada citra berwarna hingga nilai varian 0,06 dan mengalami peningkatan nilai PSNR sekitar 2-6 dB.

Kata kunci: ***Transformasi Wavelet Diskrit 2D, Clustering K-Means, denoising citra, soft-thresholding.***

IMPLEMENTATION OF K-MEANS *CLUSTERING* IN WAVELET DOMAIN USING ADAPTIVE *SOFT-THRESHOLDING* FOR IMAGE *DENOISING*

Student Name	: Biandina Meidyani
Student NRP	: 5112100218
Major	: Teknik Informatika FTIf-ITS
Supervisor I	: Arya Yudhi Wijaya, S.Kom., M.Kom.
Supervisor II	: Rully Soelaiman, S.Kom., M.Kom.

ABSTRACT

Noise is a disorder that causes an intensity value of pixel cannot reflect the resulted real value because of noise. The breakage was in form of the decrease of image quality or the loss of information in image detail. Therefore, image-denoising process was needed to reduce noise on the image so that the quality of it might be better.

The method of this study which was used to fix the image was Threshold-parameter based on clustering K-Means. The feature extraction of image- denoising exerted 2D-Discrete Wavelet Transform (DWT). DWT would decompose a vektor room into different number of vektor rooms so that analysis could be done. Afterward, to lose the noise was helped by Adaptive Soft-Thresholding. The method was implemented in colored image which noise Gaussian was added. The output of this study was the colored image which the noise has been decreased. Eight images were exerted as data to test and 512 images as training data. Each testing data were given noise with different varian value. Furthermore, PSNR input and PSNR output were calculated along with MSE.

Experimental results show that this method can be used to process the image denoising in color images up to 0.06 variance value and increased value of PSNR about 2-6 dB.

***Keywords**: 2D-Discrete Wavelet Transform, image denoising, K-Means Clustering, soft-thresholding.*

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji dan syukur bagi Allah SWT, yang telah melimpahkan rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"IMPLEMENTASI *CLUSTERING* K-MEANS DALAM DOMAIN WAVELET MENGGUNAKAN ADAPTIF *SOFT-THRESHOLDING* UNTUK *DENOISING* CITRA".**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang berharga bagi penulis. Dengan pengerjaan Tugas Akhir, penulis dapat memperdalam, meningkatkan, serta menerapkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya Tugas Akhir ini tidak lepas dari bantuan dan dukungan dari berbagai pihak. Dan dalam kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Nabi Muhammad SAW, yang telah memberikan banyak sekali teladan dan inspirasi bagi penulis.
3. Abah, Mama, serta Aa penulis yang senantiasa memberikan banyak semangat, dukungan, doa, dan perhatian kepada penulis. Buku ini penulis persembahkan secara khusus untuk Abah, Mama, dan Aa.
4. Bapak Rully Soelaiman, S.Kom., M.Kom. selaku pembimbing II. Terima kasih atas kesabaran bapak terhadap penulis dalam menerima curahan hati, membimbing, membantu, memotivasi, memberikan semangat.
5. Bapak Arya Yudhi Wijaya, S.Kom.,M.Kom. selaku pembimbing I Tugas Akhir penulis yang telah membimbing, membantu, dan memotivasi penulis sehingga dapat menyelesaikan Tugas Akhir.

6. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah membagikan ilmu kepada penulis.
7. Teman-teman rantau, Atika Faradina, Lidra Trifidya, Natasha Bunga, Uti Solichah, Maharani, Hendy, Izdihar Farahdina, Kelly Rossa, Leli Maharani, Ratih , Nur Mei, Linggar, Afina, Sinta, teman-teman di KCV dan DTK, dan semua teman-teman TC 2012 yang telah berteman dengan penulis selama menjalani perkuliahan di ITS.
8. Keluarga dan teman-teman di Banjar yang selalu mendukung dan mendoakan penulis selama ini.
9. Teman-teman TC serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Surabaya, Januari 2017

Biandina Meidyani

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Ketidaksempurnaan pengambilan gambar dapat menimbulkan adanya *noise*. *Noise* sendiri dapat menyebabkan sebuah nilai intensitas piksel tidak mencerminkan nilai yang sebenarnya dihasilkan. Intensitas *noise* berbanding lurus dengan kerusakan pada gambar yang dihasilkan, semakin tinggi nilai *noise* maka semakin tinggi pula resiko kerusakan pada gambar dan kebalikannya. Kerusakan yang di timbulkan dapat berupa menurunnya kualitas gambar atau hilangnya beberapa informasi detail citra. Oleh karena itu, dibutuhkan proses *denoising* citra yang bertujuan untuk mereduksi *noise* yang terdapat pada citra sehingga kualitas citra menjadi lebih baik. Beberapa algoritma telah diusulkan beberapa tahun terakhir ini. Sebagian besar algoritma dibagi menjadi dua kategori utama, yaitu domain spasial dan transformasi domain filtering. Ada dua jenis penyaringan domain spasial yaitu, a) filter Linear dan b) filter Non-Linear.

Filter linear yang banyak digunakan untuk *denoising* citra adalah spatial averaging filter dan wiener filter. Hasil dari averaging filter yaitu dapat menghapus transisi tajam dari gambar berdasarkan perhitungan rata-rata tetangga piksel dan mengganti piksel tengah dengan nilai rata-rata. Wiener filter memanfaatkan pengetahuan tentang *noise* tambahan dan minimum *Mean Squared Error* (MSE). Filter Non-Linear digunakan untuk menghilangkan *noise Salt and Pepper* pada gambar. Transform filtering domain sebagian besar terdiri dari penyaringan gambar di domain yang berbeda seperti Fourier, Wavelet, Ridgelet, dll. Dalam beberapa

tahun terakhir, *denoising* citra menggunakan metode *clustering* K-*Mean*s.

K-Means merupakan algoritma *clustering* yang sederhana dan sangat cepat karena mengelompokkan data menjadi beberapa kelompok berdasarkan jarak terdekat secara iteratif [1]. Metode ini mampu menghasilkan *clustering* yang bagus. Metode yang akan digunakan pada tugas akhir ini untuk memperbaiki citra yang mengandung *noise* adalah pemilihan parameter threshold berdasarkan *clustering* K-Means. Ekstraksi fitur *denoising* citra ini menggunakan Transformasi Wavelet Diskrit 2-dimensi (DWT). DWT akan mendekomposisikan suatu ruang vektor ke dalam sekumpulan ruang vektor yang berbeda, sehingga dapat dilakukan analisis. Kemudian untuk menghilangkan *noise* dibantu dengan Adaptif *Soft-thresholding* [2]. Metode ini akan diterapkan pada citra berwarna yang ditambahkan *noise* Gaussian.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Memahami konsep *noise* pada citra.
2. Memahami konsep *denoising* citra
3. Memahami kualitas gambar yang baik menurut perhitungan *Peak Signal to Noise Ratio* (PSNR).
4. Mengimplementasikan sistem yang dirancang untuk mengurangi *noise* pada citra.
5. Menyusun uji coba serta melakukan uji coba terhadap citra ber-*noise* menggunakan *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding*.

1.3. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Implementasi dilakukan dengan menggunakan perangkat lunak MATLAB R2014a.

2. Data yang digunakan untuk data latih dan data uji berupa gambar liburan yang diperoleh dari http://lear.inrialpes.fr/people/jegou/data.php.
3. Jenis *noise* yang digunakan yaitu *noise* Gaussian.
4. Input citra yang digunakan berupa citra RGB dengan piksel 256×256.

1.4. Tujuan

Adapun beberapa tujuan dari pembuatan Tugas Akhir ini, yakni sebagai berikut:

1. Mengetahui penerapan metode Transformasi Wavelet Diskrit 2-dimensi, *clustering* K-Means dan *soft-thresholding* untuk *denoising* citra.
2. Mengimplementasikan metode *Clustering* K-*Mean*s dalam domain wavelet menggunakan adaptif *soft-thresholding* dapat digunakan untuk menghilangkan *noise* Gaussian pada citra berwarna.

1.5. Manfaat

Dengan dibuatnya Tugas Akhir ini, metode *denoising* ini dapat menjadi metode yang efektif untuk mengurangi *noise* pada citra berwarna.

1.6. Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini, yaitu Implementasi *clustering* K-Means dalam

domain wavelet menggunakan adaptif *soft-thresholding* untuk *denoising* pada citra berwarna.

2. Studi literatur

   Pada tahap ini dilakukan pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literatur yang diperlukan untuk pembuatan Implementasi *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* untuk *denoising* pada citra berwarna. Informasi dan literatur didapatkan dari literatur buku dan sumber-sumber informasi lain yang berhubungan.

3. Perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Kemudian dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan dari tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahapan ini dilakukan uji coba pada data yang telah dikumpulkan. Pengujian dan evaluasi akan dilakukan dengan menggunakan MATLAB R2014a. Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian data dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk *pseudocode*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

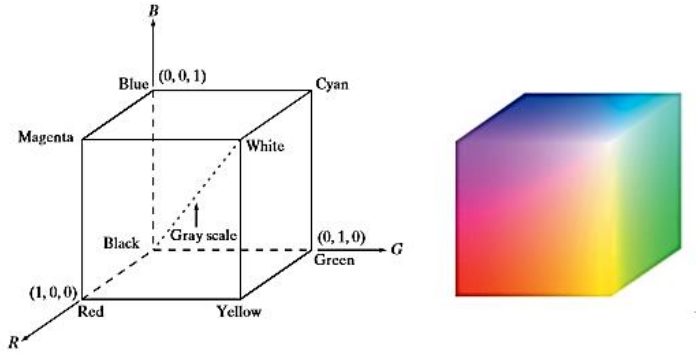
Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
DASAR TEORI

Bab ini berisi penjelasan dasar teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1. Citra Berwarna

Citra berwarna adalah citra digital yang setiap pikselnya mengadung informasi warna. Dalam model RGB, setiap warna yang muncul adalah kombinasi dari warna primer merah (R), hijau (G) dan biru (B). Bagian ruang warna dapat dilihat pada Gambar 2.1.



Gambar 2.1 Bagian ruang warna

Dimana RGB berada di tiga sudut, warna sekunder cyan, magenta dan kuning di tiga sudut lainnya, hitam berada di titik asal dan putih berada di sudut terjauh dari titik asal. Dalam model ini, skala abu-abu memanjang dari hitam menjadi putih sepanjang garis yang menghubungkan dua titik tersebut. Seperti pada Gambar 2.1

diasumsikan nilai-nilai warna telah dinormalisasikan sehingga semua nilai R, G, B memiliki kisaran [0,1] [3].

Jika setiap proses perhitungan dilakukan menggunakan tiga *channel*, berarti dilakukan tiga perhitungan yang sama. Untuk memproses tiap komponen dapat dilakukan dengan mengambil rata-rata dari nilai R, G dan B yang dapat dituliskan dengan persamaan (2.1) :

$$S = \frac{r+g+b}{3} \quad (2.1)$$

2.2. *Noise*

Noise dapat diartikan sebagai sinyal yang tidak diinginkan pada suatu pemrosesan yang mengandung informasi. Pada citra, *noise* adalah nilai yang mengganggu kualitas citra. Timbulnya *noise* dapat disebabkan oleh faktor kesengajaan dan ketidaksengajaan. Faktor ketidaksengajaan dapat terjadi karena pengambilan gambar yang tidak sempurna sedangkan faktor kesengajaan terjadi dengan cara menambahkan *noise* pada citra. Ketika sebuah citra terkena *noise*, maka akan timbul bintik hitam atau putih yang muncul secara acak dan tidak beraturan yang tidak diinginkan pada citra. Untuk menghilangkan *noise* tersebut perlu dilakukan pemrosesan citra agar mendapatkan citra yang diinginkan. Pada beberapa pengolahan citra, terkadang untuk menguji suatu algoritma yang bertujuan untuk menghilangkan *noise*, maka *noise* dihasilkan dari proses pembangkitan *noise*. Dari studi yang telah dilakukan para ahli, ada berbagai macam *noise* yang dapat ditambahkan ke dalam citra [4]. Masing-masing *noise* memilliki karakteristik dan parameter sebagai berikut :

1. Gaussian

   Gaussian merupakan model *noise* yang mengikuti distribusi normal standard dengan rata-rata 0 (nol) dan standard deviasi 1. Efek dari *noise* ini adalah munculnya titik-titik berwarna yang jumlahnya sama dengan prosentase *noise*.

   Memiliki parameter rata-rata dan variansi.

2. Speckle
   Speckle merupakan model *noise* yang memberikan warna hitam pada titik yang terkena *noise*.
   Memiliki parameter variansi.
3. Salt & Pepper
   Salt & pepper memberikan *noise* seperti taburan garam, akan memberikan warna putih pada titik yang terkena *noise*.
   Memiliki parameter kepadatan (*density*).

2.3. Gaussian *Noise*

Noise ini memiliki intensitas yang sesuai dengan distribusi normal yang memiliki rata-rata (*mean*) dan nilai standar deviasi atau yang disebut *Probability Density Function* (PDF). Rata-rata dan standar deviasi merupakan suatu konstanta real. Nilainya dapat berupa positif maupun negatif. Makin besar nilai konstantanya maka citra akan semakin kabur, sebaliknya semakin kecil konstantanya efek pada citra makin tidak terlihat. Nilai default untuk *mean* adalah nol (0) sedangkan nilai varian dapat bervariasi sesuai dengan semakin banyaknya titik-titik *noise* yang terdapat pada citra. Disebut *white noise* karena pada saat nilai rata-rata dan standar deviasinya besar maka citra seolah-olah hanya terlihat seperti citra putih saja. Pada citra, Gaussian *noise* terlihat seperti titik-titik yang tersebar di seluruh citra [2] [4]. Fungsi citra yang terkena *noise* dapat dilihat pada persamaan (2.2):

$$x_{(i,j)} = f_{(i,j)} + \sigma z_{(i,j)} \qquad (2.2)$$

Dimana :

$x_{(i,j)}$ = citra yang terkena *noise*
$f_{(i,j)}$ = citra asli
σ = standar deviasi
$z_{(i,j)}$ = *noise*

Distribusi Gaussian dengan pdf dapat dilihat pada persamaan (2.3) :

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2} \quad (2.3)$$

Dimana :
z = nilai keabuan
σ = standar deviasi
μ = rata-rata (*mean*)

2.4. *Denoising* Citra

Denoising citra adalah salah satu upaya untuk mereduksi *noise* yang terdapat pada citra sehingga citra terlihat lebih baik. Dalam pengolahan citra, pengurangan *noise* dan pemulihan citra bertujuan untuk memperbaiki kualitas dari suatu gambar. Tujuan utama dari *denoising* gambar adalah untuk mengembalikan detail dari gambar asli sebisa mungkin. Kriteria masalah penghapusan *noise* tergantung pada jenis *noise* yang akan merusak gambar [5]. Untuk citra yang mengalami perubahan karena *noise* dapat dimodelkan pada persamaan (2.4) :

$$y(i,j) = x(i,j) + n(i,j) \quad (2.4)$$

Dimana :
y(i,j) = nilai citra yang terkena *noise*
x(i,j) = nilai citra sebelum terkena *noise*
n(i,j) = nilai *noise*

2.5. Transformasi Wavelet

Wavelet adalah matematika yang memotong-motong data menjadi kumpulan-kumpulan frekuensi yang berbeda, sehingga masing masing komponen tersebut dapat dipelajari dengan menggunakan skala resolusi yang berbeda. Wavelet merupakan sebuah fungsi variabel real t, diberi notasi Ψt dalam ruang fungsi $L^2(R)$. Fungsi ini dihasilkan oleh parameter dilatasi dan translasi yang dinyatakan dalam persamaan (2.5) dan persamaan (2.6):

$$\Psi_{a,b}(t) = a^{-1/2}\Psi\left(\frac{t-b}{a}\right); a > 0, b \epsilon \Re \quad (2.5)$$

$$\Psi_{j,k}(t) = a^{j/2}\Psi\left(2^j t - k\right); j, k \epsilon Z \quad (2.6)$$

Dimana :
a = parameter dilatasi
b = parameter translasi
$\Re$ = mengkondisikan nilai a dan b bernilai real
2j = parameter dilatasi
k = parameter waktu atau lokasi ruang
Z = mengkondisikan nilai j dan k bernilai integer

Fungsi persamaan yang pertama dikenalkan pertama kali oleh Grossman dan Morlet, sedangkan persamaan yang kedua dikenalkan oleh Daubechies. Transformasi wavelet menggunakan dua komponen penting dalam melakukan transformasi yakni fungsi skala (*scaling function*) dan fungsi wavelet (*wavelet function*). Fungsi skala (*scaling function*) disebut juga sebagai *Lowpass Filter*, sedangkan fungsi wavelet (*wavelet function*) disebut juga sebagai *Highpass Filter*. Kedua fungsi ini digunakan pada saat transformasi wavelet dan invers transformasi wavelet [3].

1. Fungsi wavelet

disebut juga *highpass filter* yang mengambil citra dengan gradiasi intensitas yang tinggi dan perbedaan intensitas yang rendah akan dikurangi atau dibuang.

2. Fungsi skala

disebut juga *lowpass filter* yang mengambil citra dengan gradiasi intensitas yang halus dan perbedaan intensitas yang tinggi akan dikurangi atau dibuang. Kedua komponen diatas dapat disebut sebagai *mother* wavelet yang harus memenuhi kondisi seperti persamaan (2.7):

$$\int_{-\infty}^{\infty} \Psi(x)dx = 0 \quad (2.7)$$

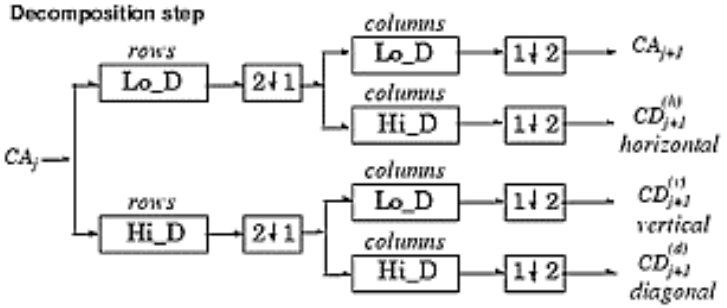
Formula di atas menunjukkan sifat ortogonalitas vektor.

2.6. Transformasi Wavelet Diskrit

Wavelet merupakan metode yang bisa digunakan untuk mendefinisikan ruang multiresolusi. Dengan menggunakan wavelet, suatu ruang vektor dapat didekomposisi ke dalam sekumpulan ruang vektor bersarang dengan resolusi yang berbeda-beda, sehingga memungkinkan dilakukannya anaisis terhadap fungsi baik pada domain waktu maupun frekuensi pada resolusi yang berbeda. Wavelet terpilih sebagai metode untuk *denoising* citra [2].

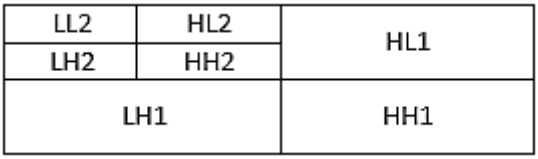
Transformasi wavelet diskrit secara umum merupakan dekomposisi citra pada frekuensi subband citra tersebut dimana komponennya dihasilkan dengan cara penurunan level dekomposisi. Implementasi transformasi wavelet diskrit dapat dilakukan dengan cara melewatkan sinyal frekuensi tinggi atau *highpass filter* dan frekuensi rendah atau *lowpass filter*.

Ada berbagai jenis transformasi wavelet, akan tetapi pada bagian ini lebih menitikberatkan pada transformasi wavelet dua dimensi. Tranformasi wavelet diskrit dua dimensi merupakan hasil pemfilteran baris dan kolom. Transformasi Wavelet Diskrit dua dimensi dengan level dekomposisi satu dapat dilihat pada Gambar 2.2.



Gambar 2.2 Transformasi Wavelet Diskrit 2-dimensi dengan level dekomposisi satu

Transformasi Wavelet Diskrit (DWT) diimplementasikan sebagai rangkaian proyeksi ke fungsi penskalaan di L^2(R). DWT adalah proses dekomposisi. Data dua dimensi diganti dengan empat blok yang bersesuaian dengan subband yang mewakili *low pass filtering* dan *high pass filtering* di setiap arah. Prosedur untuk dekomposisi wavelet terdiri dari operasi berturut-turut pada baris dan kolom citra. Pertama, dilakukan transformasi pada semua baris yang menghasilkan matriks, dimana sisi kiri berisi koefisien *low pass down sample* dari setiap baris, dan sisi kanan berisi koefisien *high pass*. Kemudian dekomposisi diterapkan untuk semua kolom. Dari proses DWT, akan didapatkan koefisien-koefisien wavelet berupa subband LL, subband LH, subband HL, dan subband HH seperti pada Gambar 2.3.



Gambar 2.3 Dekomposisi citra level 2 menggunakan DWT

Dari subband LL, LH, HL, dan HH, yang dilakukan proses *denoising* pada tugas akhir ini LL yaitu disebut sebagai koefisien aproksimasi. Pada proses dekomposisi multilevel wavelet, frekuensi rendah dibagi kembali menjadi frekuensi tinggi dan rendah dengan kata lain koefisien aproksimasi level 1 dipecah menjadi koefisien aproksimasi level 2 dan koefisien detail level 2 dan seterusnya. Proses diulang sampai citra tidak dapat didekomposisi lagi atau sampai pada level yang memungkinkan. Hasil analisis tranformasi wavelet adalah koefisien aproksimasi dan detail. Kelebihan dari analisis citra menggunakan wavelet adalah bahwa dapat dipelajarinya karakteristik sinyal secara lokal dan detail, sesuai dengan skala-nya.

Pada transformasi DWT terdapat proses pengembalian kembali komponen-komponen yang telah kita gunakan. *Inverse*

Discrete Wavelet Transform (IDWT) merupakan kebalikan dari transformasi wavelet diskrit (DWT). Pada transformasi ini dilakukan proses rekonstruksi sinyal, yaitu mengembalikan komponen frekuensi menjadi komponen sinyal semula. Transformasi dilakukan dengan proses *up sampling* dan pemfilteran dengan koefisien filter invers. Sehingga dalam satu sistem transformasi wavelet menggunakan empat macam filter, yaitu *low-pass filter* dan *high-pass filter* dekomposisi, dan *lowpass filter* dan *high-pass filter* rekonstruksi. *Low-pass filter* dan *high-pass filter* dekomposisi digunakan pada saat proses dekomposisi atau pemecahan sedangkan *low-pass filter* dan *highpass filter* rekonstruksi digunakan untuk proses rekonstruksi wavelet [3].

2.7. Clustering K-Means

K-Means adalah algoritma *clustering* yang khas dalam data mining dan secara luas digunakan untuk mengelompokkan data yang besar. Menurut MacQueen, K-*Mean*s adalah salah satu algoritma *clustering* yang paling sederhana [1]. *Clustering* K-Means membagi komponen dataset dengan cara mengelompokkannya berdasarkan sifat serupa yang mana nantinya akan dikelompokkan ke kelompok yang sama. Dalam *clustering* K-Means, kita ingin mengelompokkan obyek ke dalam k kelompok atau *cluster*. Untuk melakukan *clustering* ini, nilai k harus ditentukan terlebih dahulu. Biasanya pengguna telah mempunyai informasi awal tentang obyek yang sedang dipelajari, termasuk berapa jumlah cluster yang paling tepat. Konsep dasar K-Means adalah pencarian pusat *cluster* secara iteratif. Pusat *cluster* ditetapkan berdasarkan jarak setiap data ke pusat *cluster*. Proses *clustering* dimulai dengan mengidentifikasi data yang akan di-cluster. Pada awal iterasi, pusat setiap *cluster* ditetapkan secara bebas. Kemudian dihitung jarak antara setiap data dengan setiap pusat cluster. Untuk dapat mengelompokkan data, langkah penting yang perlu dilakukan adalah menghitung jarak Euclidean, yang digunakan untuk menentukan jarak terdekat antara masing-masing

objek data dengan *center cluster*. Jarak Euclidean antara satu vector $x = (x_1, x_2, \dots, x_n)$ dan vector $y = (y_1, y_2, \dots, y_n)$. Jarak Euclidean $d = (x_i, y_i)$ dapat diperoleh pada persamaan (2.8).

$$d(x_i, y_i) = \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{1/2} \quad (2.8)$$

Suatu data akan menjadi anggota dari *cluster* ke-i apabila jarak data tersebut ke pusat *cluster* ke-i bernilai paling kecil jika dibandingkan dengan jarak ke pusat *cluster* lainnya. Selanjutnya, kelompokkan data-data yang menjadi anggota pada setiap *cluster*.

2.8. *Soft-Thresholding*

Konsep yang dilakukan dalam *denoising* citra adalah men-threshold atau menghilangkan terhadap komponen berfrekuensi tinggi dari wavelet yang disebut koefisien detail. Koefisien wavelet sangat sensitif terhadap *noise* [2]. Dalam *soft-thresholding*, nilai *thresholding* dihitung berdasarkan persamaan (2.9).

$$Threshold = \frac{1}{2^{(j-1)}}\left(\frac{\sigma}{\mu}\right)M \quad (2.9)$$

Dimana :
j = level dari wavelet
M = *median* dari jarak Euclidean *centroid* dengan data tes
σ = standar deviasi dari jarak Euclidean *Centroid* dengan data tes
μ = *mean* dari jarak Euclidean c*entroid* dengan data tes

Untuk Fungsi *soft-thresholding* dapat digambarkan seperti persamaan (2.10).

$$W_{soft} = sign(W)(|W| - T)_+ \quad (2.10)$$

Dimana :
sign(W) = fungsi signum dari koefisien wavelet

$$sign(W) = \begin{cases} +1 \ \ if \ W > 0 \\ 0 \ \ if \ W = 0 \\ -1 \ \ if \ W < 0 \end{cases} \quad (2.11)$$

W = koefisien wavelet
T = nilai *thresholding*

2.9. *Mean Squared Error* (MSE)

MSE merupakan ukuran kontrol kualitas yang digunakan untuk mengetahui kualitas dari sebuah proses. MSE menghitung seberapa besar pergeseran data antara sinyal sumber dan sinyal hasil keluaran, dimana sinyal hasil keluaran dan sinyal sumber memiliki ukuran yang sama. Sebuah gambar dinyatakan buruk apabila MSE bernilai besar [6]. MSE dapat didefinisikan pada persamaan (2.12).

$$MSE = \frac{1}{M^2} \sum_{i,j=1}^{M} \left(z(i,j) - s(i,j) \right)^2 \quad (2.12)$$

Dimana :
$M \times M$ = ukuran gambar
$z(i,j)$ = gambar dengan *noise*
$s(i,j)$ = gambar asli tanpa *noise*

2.10. *Peak Signal to Noise Ratio* (PSNR)

Peak Signal to Noise Ratio (PSNR) merupakan nilai perbandingan antara nilai maksimum dari citra hasil *filtering* dengan *noise*, yang dinyatakan dalam satuan desibel (dB). Sebuah gambar dapat dikatakan berkualitas tinggi jika memiliki nilai PSNR yang kecil [6]. Dalam tugas akhir ini, gambar memiliki performa yang bagus jika nilai PSNR berkisar di atas 65 dB [7]. PSNR dapat didefinisikan pada persamaan (2.13).

$$PSNR = \left[10 \ LOG \ \frac{255^2}{MSE} \right] \quad (2.13)$$

BAB III
PERANCANGAN

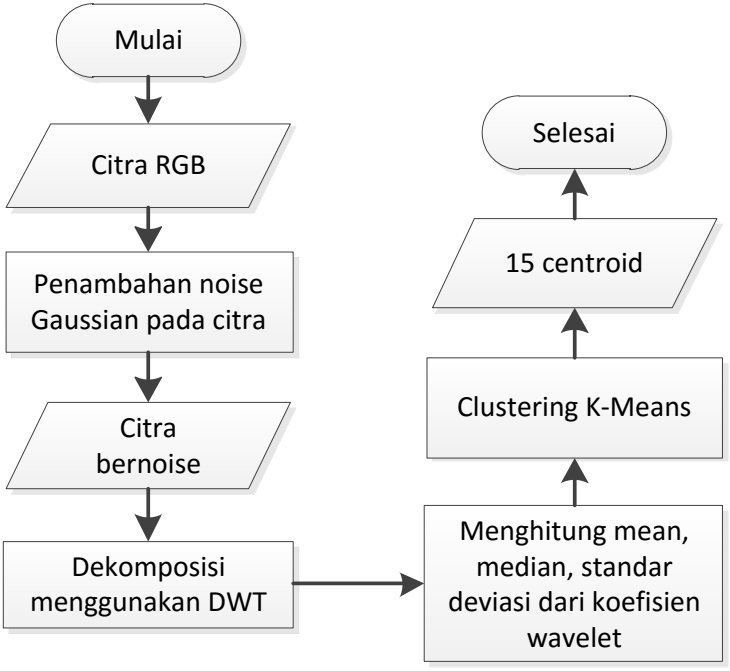
Pada bab ini akan dijelaskan perancangan perangkat lunak *denoising* citra berwarna. Perancangan dan desain sistem yang akan dibahas dalam bab ini meliputi desain secara umum dari sistem, perancangan data, data-data yang digunakan, diagram alir (*flowchart*) dan *pseudocode*.

3.1. Desain Secara Umum

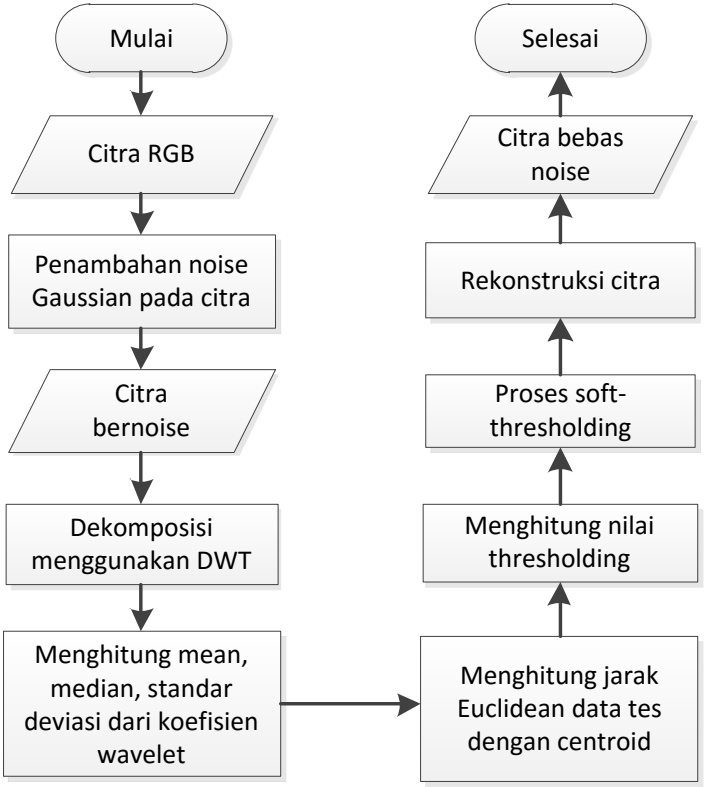
Pada subbab ini akan dijelaskan mengenai perancangan proses yang bertujuan menghilangkan *noise* beserta langkah-langkahnya pada setiap proses dalam membangun perangkat lunak *denoising* pada citra RGB dengan metode *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding*. Hasil akhir yang diharapkan dari proses ini yaitu suatu citra yang telah ter*denoising* sesuai dengan alur proses yang dijalankan.

Terdapat dua fase yaitu fase *training* dan fase *testing*. Fase *training* akan membuat dataset dari 512 citra yang akan diproses dengan metode tugas akhir ini dan fase *testing* akan dilakukan proses denoising citra. Fase *training* dimulai dari input berupa citra RGB. Masing-masing komponen warna akan ditambahkan *noise* Gaussian sebagai simulasi citra ber*noise*, kemudian dilakukan proses dekomposisi Transformasi Wavelet Diskrit 2-dimensi. Hasil DWT ini akan dihitung *mean*, *median*, dan standar deviasi yang akan di*clustering* menggunakan *clustering* K-Means. Didapatkan 15 *centroid* untuk proses denoising. Kemudian dilanjutkan dengan fase *testing*. Fase *testing* akan memproses 1 citra data tes menjadi citra ber*noise*. Data tes akan diproses sama seperti *training* yaitu di tambahan *noise* Gaussian, kemudian di dekomposisi wavelet dan dihitung *mean*, *median* dan standar deviasi. Proses *denoising* meliputi perhitungan jarak euclidean 15 c*entroid* dengan hasil proses data tes tadi dan *soft-thresholding* sebagai nilai ambang penentu *noise*. Langkah terakhir yaitu merekonstruksi kembali

koefisien wavelet menggunakan Invers Transformasi Wavelet Diskrit 2-dimensi, sehingga citra akhir diharapkan telah ter*denoising*. Citra berwarna yang diproses akan dikembalikan sesuai channel RGB. Untuk mengetahui kualitas citra hasil dengan citra asli maka dilakukan perhitungan *Peak Signal to Noise Ratio* (PSNR). Perhitungan PSNR dilakukan dengan menghitung nilai *Mean Square Error (MSE)* dari citra input dan citra output. Proses *denoising* citra dengan metode *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* dapat dilihat pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Diagram alir fase *training*

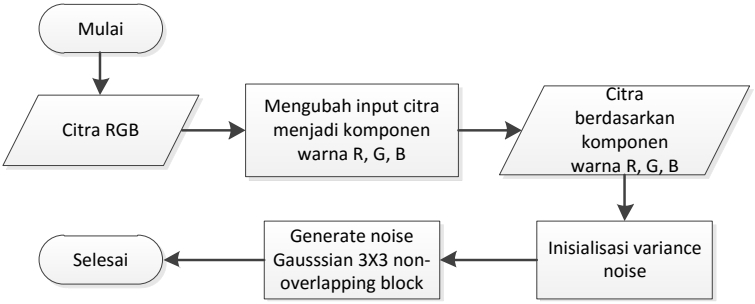


Gambar 3.2 Diagram alir fase *testing*

3.2. Perancangan Pemberian *Noise* Gaussian pada Citra

Pada proses ini dilakukan penambahan *noise* terhadap citra. Proses ini sengaja dilakukan untuk mengetahui seberapa besar metode *denoising* efektif menghilangkan *noise*. Jenis *noise* yang ditambahkan yaitu *noise* Gaussian. Dalam penambahan *noise* ini parameter yang digunakan adalah nilai varian *noise*, sehingga tiap *noise* memiliki nilai varian yang berbeda. Semakin besar varian *noise* semakin banyak mengandung *noise*. Dalam tugas akhir ini, diberikan nilai varian dari 0,01 hingga nilai varian yang

menyebabkan kualitas gambar menjadi jelek. Jika proses pemberian *noise* telah dilakukan maka akan dilanjutkan dengan proses transformasi wavelet diskrit 2-dimensi. Diagram alir proses ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram alir proses pemberian *noise*

Daftar variabel dan fungsi yang digunakan untuk proses pemberian *noise* Gaussian pada kasus citra berwarna ditunjukkan pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Daftar variabel untuk proses pemberian *noise* pada kasus citra berwarna

No	Nama variabel	Tipe	Penjelasan
1	h	Integer	Jumlah baris dari matriks citra input
2	w	Integer	Jumlah kolom dari matriks citra input
3	image_noise	Array of double	Array yang menampung hasil citra yang telah ditambahkan *noise*
4	mean_noise	Integer	Parameter *mean* untuk *noise* Gaussian
5	std_noise	Double	Parameter varian untuk *noise* Gaussian
6	img_input	Array of double	Matriks citra input

Tabel 3.2 Daftar fungsi untuk proses pemberian *noise* pada kasus citra berwarna

No	Nama fungsi	Penjelasan
1	imnoise	Fungsi untuk menggenerate *noise* ke citra

Pseudocode proses pemberian *noise* pada kasus citra berwarna ini dapat dilihat pada Gambar 3.4.

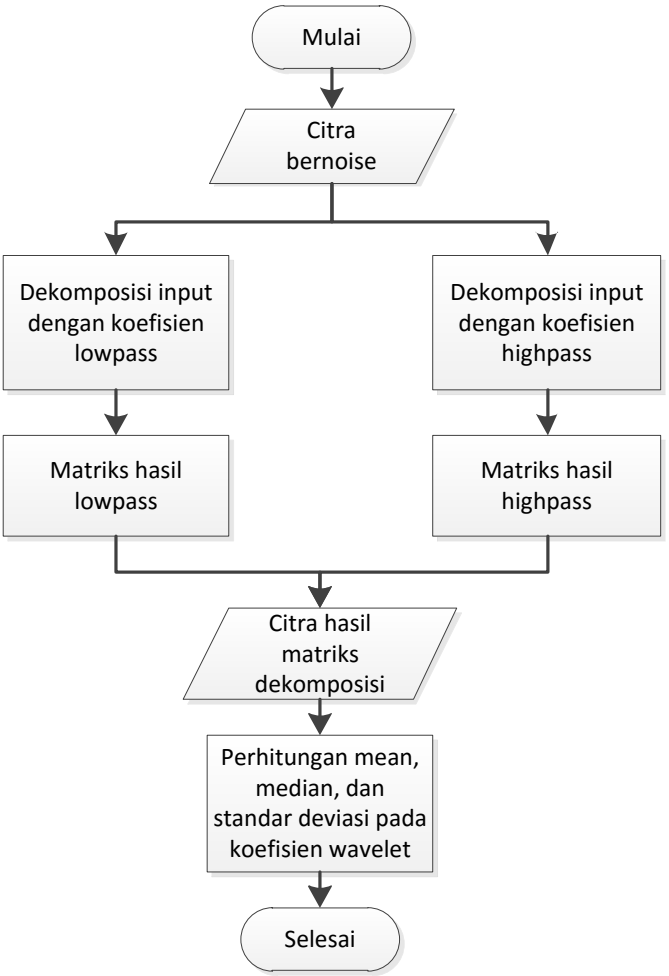
Masukan	Citra input
Keluaran	Citra ber*noise*

```
1. [h, w] = size(IR);
2. IN = [];
3. for i = 1:3 : h
4.     for j = 1:3 : w
5.         mean_noise = 0;
6.          std_noise = 0.04;
7.         IN=imnoise(IR,'gaussian',mean_noise,
8.         std_noise);
9.     end
10. end
```

Gambar 3.4 *Pseudocode* proses pemberian *noise*

3.3. Perancangan Dekomposisi Wavelet

Pada proses ini dilakukan dekomposisi citra menggunakan Transformasi Wavelet Diskrit 2D untuk mendapatkan koefisien-koefien wavelet. Proses dekomposisi ini dilakukan untuk setiap baris dan kolom dari matriks input citra. Data yang dihasilkan dari proses dekomposisi ini adalah matriks 2D (m baris, n kolom). Untuk setiap baris dan kolom, data yang dihasilkan terdiri dari dua jenis yaitu hasil matriks dekomposisi dengan koefisiesn highpass dan lowpass. Pada proses ini terdapat perkalian matriks citra dengan matriks koefisien. Data yang dihasilkan dari proses ini digunakan sebagai data masukan untuk menghitung *mean*, *median*, dan standar deviasi. Hasil ini akan menjadi data masukan untuk proses *denoising*. Langkah-langkah dari proses dekomposisi wavelet digambarkan pada diagram alir pada Gambar 3.5.



Gambar 3.5 Diagram alir proses dekomposisi

Daftar variabel dan fungsi yang digunakan untuk proses dekomposisi wavelet pada kasus citra berwarna ditunjukkan pada Tabel 3.3 dan Tabel 3.4.

Tabel 3.3 Daftar variabel untuk proses dekomposisi wavelet pada kasus citra berwarna

No	Nama variabel	Tipe	Penjelasan
1	no_level	uint8	Jumlah level dekomposisi wavelet
2	a1	Array of double	Output dari dekomposisi wavelet
3	a2	Array of integer	Indeks hasil dekomposisi
4	IN	Double	Matriks citra ber-*noise*
5	arr	Array of integer	Array untuk menampung hasil perhitungan statistik citra dari koefisien wavelet
6	approks	Array of double	Komponen hasil koefisien approksimasi
7	final	Array of double	Matriks hasil perhitungan statistik citra dari hasil dekomposisi
8	temp	Array of double	Variabel temporary

Tabel 3.4 Daftar fungsi untuk proses dekomposisi wavelet pada kasus citra berwarna

No	Nama fungsi	Penjelasan
1	wavedec2	Fungsi untuk analisis dekomposisi dari 2D sinyal
2	appcoef2	Fungsi untuk mendekomposisi koefisien approksimasi dar 2D sinyal
3	mean	Fungsi untuk menghitung rata-rata dari data
4	median	Fungsi untuk menghitung nilai tengah dari data
5	std2	Fungsi untuk menghitung standar deviasi dari data

3.3.1. Transformasi Wavelet Diskrit 2D

Proses ini melakukan dekomposisi dari citra ber*noise* menggunakan metode Transformasi Wavelet Diskrit 2D. Pada tugas akhir ini, digunakan 5 level dekomposisi dan tipe *filter* wavelet Daubechies 6 (db6). *Pseudocode* program ini ditunjukkan pada Gambar 3.6.

Masukan	Matrik citra RGB dengan *noise* img_*noise*, jumlah level, jenis filter wavelet
Keluaran	Matrik citra hasil dekomposisi koefisien approksimasi approks

```
1. no_level = jumlah level;
2. [a1,a2]=wavedec2(IN,no_level,'wname');
3. approks = appcoef2(a1,a2,'wname',no_level);
```

Gambar 3.6 *Pseudocode* Transformasi Wavelet Diskrit 2D

3.3.2. Perhitungan Statistika Citra

Pada proses ini dilakukan perhitungan *mean*, *median*, dan standar deviasi tiap level koefisien wavelet. Koefisien yang digunakan yaitu koefisien aproksimasi. Karena dekomposisi dilakukan sebanyak 5 level maka proses perhitungan statistik citra dilakukan sebanyak 5 kali. Semua hasil dari proses tersebut dimasukkan ke dalam suatu array dengan variabel *final*. *Pseudocode* perhitungan statistik citra ditunjukkan pada Gambar 3.7.

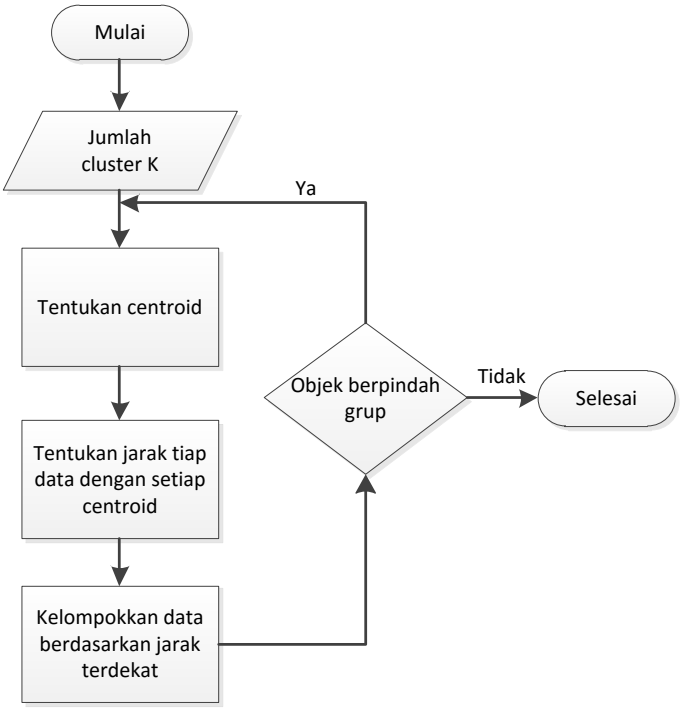
Masukan	Matrik hasil dekomposisi koefisien approksimasi approks
Keluaran	Array final

```
1. arr = 1
2. final(arr) = mean2(approks); arr = arr + 1;
3. temp = median(approks);
4. final(arr) = median(temp); arr = arr + 1;
5. final(arr) = std2(approks); arr = arr + 1;
```

Gambar 3.7 *Pseudocode* perhitungan statistik citra

3.4. Perancangan Clustering K-Means

Hasil dekomposisi di atas selanjutnya akan di kelompokkan sesuai kemiripan data menggunakan metode clustering K-means. Pertama kita harus menentukan jumlah *centroid* yang diinginkan. Pada tugas akhir ini, ditentukan 15 *centroid* yang akan dihitung jaraknya dengan tiap data menggunakan jarak Euclidean. Untuk melihat diagram alir dari proses ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 Diagram alir proses *clustering*

Daftar variabel dan fungsi yang digunakan untuk proses *clustering* K-Means pada kasus citra berwarna ditunjukkan pada Tabel 3.5 dan Tabel 3.6.

Tabel 3.5 Daftar variabel untuk proses *clustering* pada kasus citra berwarna

No	Nama variabel	Tipe	Penjelasan
1	opts	Array	Pilihan untuk mengontrol algoritma iterative untuk meminimalkan kriteria
2	idx	Integer	Indeks cluster
3	C	Array of integer	*Centroid* yang dihasilkan
4	Distance	Array of integer	Jarak tiap data dengan *Centroid*
5	denoise.final	Array of double	Hasil dekomposisi data *training*
6	k	Integer	Banyaknya cluster yang diinginkan
7	jumlah_iterasi	Integer	Banyaknya iterasi yang akan dilakukan

Pada proses ini dilakukan *clustering* K-Means untuk mengelompokkan citra yang memiliki kesamaan data. Pengguna menentukan nilai k atau jumlah cluster yang diinginkan. Kemudian hitung jarak setiap data yang ada terhadap masing-masing c*entroid* menggunakan rumus jarak Euclidean hingga ditemukan jarak yang paling dekat dari setiap data dengan *centroid*. Kelompokkan setiap data berdasarkan kedekatannya dengan *centroid* hingga nilai c*entroid* tidak berubah (stabil). *Pseudocode clustering* K-Means ditunjukkan pada Gambar 3.9.

Tabel 3.6 Daftar fungsi untuk proses *clustering* pada kasus citra berwarna

No	Nama fungsi	Penjelasan
1	statset	Fungsi untuk mengontrol algoritma iterative untuk meminimalkan kriteria
2	kmeans	Fungsi untuk mengelompokkan data menggunakan metode *clustering* K-*Means*
3	Sqeuclidean	Fungsi untuk menghitung jarak Euclidean
4	Replicates	Fungsi untuk mengetahui banyaknya perulangan *clustering* menggunakan posisi yang baru dengan *centroid* awal
5	Options	Fungsi untuk mengontrol algoritma iterative untuk meminimalkan kriteria

Masukan	Komponen hasil dekomposisi citra data *training*
Keluaran	*Centroid* hasil *clustering*

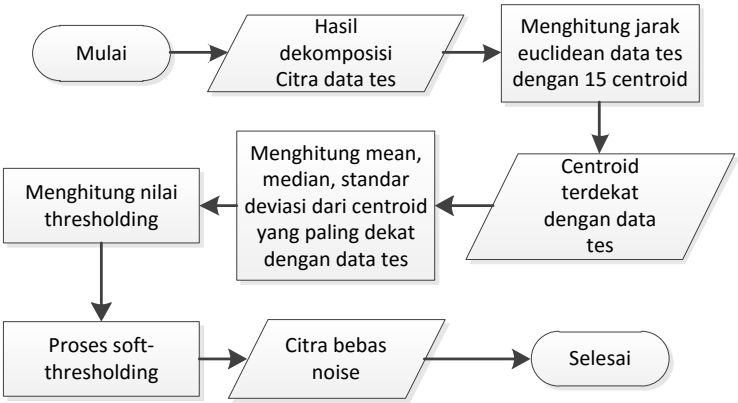
```
1. denoisee = load('trainingR52n0.05.mat');
2. opts = statset('Display','final');
3. k = banyak cluster yang diinginkan;
4. jumlah_iterasi = banyaknya iterasi;
5. [idx,C]=kmeans(denoisee.final,k,'Distance',
6. 'sqeuclidean',...
7. 'Replicates',jumlah_iterasi,'Options',opts);
```

Gambar 3.9 *Pseudocode clustering* K-Means

3.5. Perancangan Proses *Denoising*

Pada proses ini mulai masuk ke fase *testing*. Dimulai dari input citra RGB sebagai data tes. Kemudian dilakukan proses penambahan *noise*, dekomposisi citra, dan perhitungan statistik citra seperti fase *training* yang dijelaskan pada subbab 3.2 da subbab 3.3. Dari proses dekomposisi data tes ini, hasilnya akan di hitung jarak terdekat dengan 15 *centroid* yang telah didapatkan

menggunakan jarak Euclidean. Kemudian akan didapatkan *centroid* yang paling dekat dengan data tes. *Centroid* tersebut akan menjadi parameter menentukan nilai thresholding. Nilai thresholding ini sebagai ambang batas koefisien wavelet yang akan di*denoising* dengan *soft-thresholding*. Untuk melihat diagram alir dari proses ini dapat dilihat pada Gambar 3.10.



Gambar 3.10 Diagram alir proses *denoising*

Daftar variabel dan fungsi yang digunakan untuk proses *denoising* pada kasus citra berwarna ditunjukkan pada Tabel 3.7 dan Tabel 3.8.

3.5.1. Jarak Euclidean

Konsep jarak Euclidean ini memperlakukan semua peubah adalah bebas. Jarak Euclidean adalah besarnya jarak suatu garis lurus yang menghubungkan antar objek. Misalkan ada dua objek yaitu, A dengan koordinat (x_1, y_1) dan B dengan koordinat (x_2, y_2) maka jarak antar objek tersebut dapat diukur dengan rumus persamaan (3.1):

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (3.1)$$

Kemudian dihitung jarak Euclidean menggunakan rumus pada persamaan (3.1) untuk mengetahui jarak terdekat tiap data tes dengan *centroid* yang telah didapatkan. *Pseudocode* untuk proses jarak Euclidean dapat dilihat pada Gambar 3.11.

Tabel 3.7 Daftar variabel untuk proses *denoising* pada kasus citra berwarna

No	Nama variabel	Tipe	Penjelasan
1	final2	Array of double	Hasil dekomposisi data tes
2	euc	Array of double	Hasil perhitungan jarak Euclidean *Centroid* dengan data tes
3	sorted	Array of double	Hasil yang telah diurutkan
4	index	Integer	Indeks data yang telah diurutkan
5	sort_*Centroid*	Array of double	Mengambil data *Centroid* keseluruhan yang sudah diurutkan
6	sort_*Centroid* 1	Array of double	Mengambil *Centroid* yang paling dekat dengan data tes
7	*mean_Centroid*	Double	Rata-rata dari *Centroid*
8	*median_Centroid*	Double	Nilai tengah dari *Centroid*
9	Std_*Centroid*	Double	Standar deviasi dari *Centroid*
10	Thresh	Double	Menghitung nilai thresholding
11	Thesholdval	Double	Menghitung nilai thresholding
12	ytsoft	Array of double	Proses *soft-thresholding*

Tabel 3.8 Daftar fungsi untuk proses *denoising* pada kasus citra berwarna

No	Nama fungsi	Penjelasan
1	sqrt	Fungsi untuk perhitungan akar (operasi matematika)
2	size	Fungsi untuk mengukur ukuran suatu data
3	sum	Fungsi untuk menghitung jumlah keseluruhan data
4	sign	Fungsi untuk mengembalikan lambang angka. Mengembalikan 1 jika angkanya positif, nol (0) jika angkanya 0 dan -1 jika angkanya negatif
5	abs	Fungsi untuk nilai absolut

Masukan	*Centroid* hasil *clustering* dan komponen hasil dekomposisi data tes
Keluaran	*Centroid* yang paling dekat dengan data tes

```
1. for m=1 : size(final2,1)
2.     for n=1 : size(C,1)
3.         euc(n)      =      sqrt(sum((final2(m)-
   C(n)) .^2));
4.     end
5.     [sorted, index] = sort(euc);
6.     sort_Centroid = C(index,:);
7.     sort_Centroid1 = f(1,:);
8. end
```

Gambar 3.11 *Pseudocode* proses jarak Euclidean

3.5.2. Perhitungan Nilai Thresholding

Proses ini memerlukan nilai *mean*, *median*, dan standar deviasi dari c*entroid* yang paling dekat dengan data tes. Kemudian nilai-nilai tersebut dihitung menurut rumus *thresholding* pada persamaan (2.9). Thresholding ini bertujuan untuk melewatkan koefisien tersebut ke suatu ambang batas yang telah ditentukan, sehingga koefisien yang tidak sesuai ambang tersebut tidak

digunakan. *Pseudocode* perhitungan nilai *thresholding* dapat dilihat pada Gambar 3.12.

Masukan	*Centroid* yang paling dekat dengan data tes
Keluaran	Nilai thresholding
1.mean_centroid = *mean*(sort_*Centroid*1); 2.med_centroid = *median*(sort_*Centroid*1); 3.std_centroid = std(sort_*Centroid*1); 4.Thresh=(std_centroid/mean_centroid)*med_*centroid*; 5.Thresholdval = Thresh/2^(no_level-1);	

Gambar 3.12 *Pseudocode* perhitungan nilai *thresholding*

3.5.3. Proses *Soft-thresholding*

Pada proses ini diharapkan nilai *noise* yang tidak sesuai dengan nilai *thresholding* dapat hilang sehingga menjadi citra yang bebas *noise*. *Thresholding* yang digunakan yaitu *soft-thresholding*. Proses ini akan membuat semua koefisien mengalami modifikasi. Koefisien yang memiliki nilai absolut di atas nilai threshold akan dikurangi nilainya sedangkan koefisien lainnya akan dibuat nol (0). Fungsi signum bertujuan untuk mengemballikan lambang angka. Mengembalikan 1 jika angkanya positif, nol (0) jika angkanya 0 dan -1 jika angkanya negatif. *Pseudocode* proses *soft-thresholding* ini dapat dilihat pada Gambar 3.13.

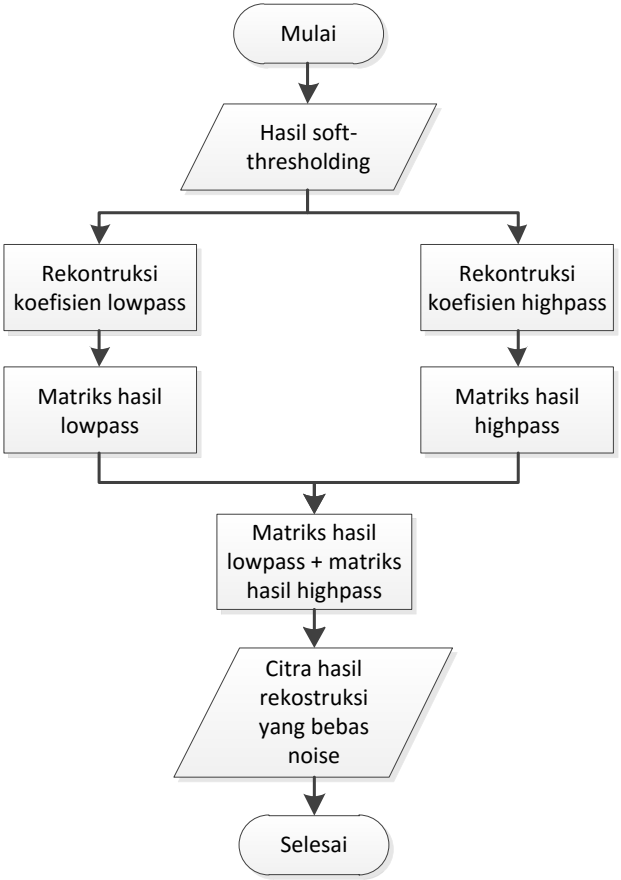
Masukan	Nilai thresholding dan array hasil dekomposisi
Keluaran	Matriks hasil *soft-thresholding*
1. ytsoft = sign(at1).*(((abs(at1)-Thresholdval)>=0).*(abs(at1)-Thresholdval));	

Gambar 3.13 *Pseudocode* proses *soft-thresholding*

3.6. Perancangan Rekonstruksi Citra

Rekonstruksi bertujuan agar citra yang telah didekomposisi kembali memiliki piksel seperti citra awal. Proses

ini menginversikan transformasi wavelet dari koefisien wavelet yang dithreshold untuk mendapatkan sinyal yang terdenoising. Proses rekonstruksi ini akan dilakukan untuk setiap baris dan setiap kolom dari matriks citra de*noise*. Diagram alir dari proses ini dapat dilihat pada Gambar 3.14.



Gambar 3.14 Diagram alir proses rekonstruksi citra

Pada transformasi ini dilakukan proses rekonstruksi sinyal, yaitu mengembalikan komponen frekuensi menjadi komponen sinyal semula. Transformasi dilakukan dengan proses *up sampling* dan pemfilteran dengan koefisien *filter* invers. Sehingga dalam satu sistem transformasi wavelet menggunakan empat macam *filter*, yaitu *low-pass filter* dan *high-pass filter* dekomposisi, dan *lowpass filter* dan *high-pass filter* rekonstruksi. *Low-pass filter* dan *high-pass filter* dekomposisi digunakan pada saat proses dekomposisi atau pemecahan sedangkan *low-pass filter* dan *highpass filter* rekonstruksi digunakan untuk proses rekonstruksi wavelet.

Daftar variabel dan fungsi yang digunakan untuk proses rekonstruksi citra pada kasus citra berwarna ditunjukkan pada Tabel 3.9 dan Tabel 3.10.

Pseudocode proses rekonstruksi citra pada kasus citra berwarna ini dapat dilihat pada Gambar 3.15.

Tabel 3.9 Daftar variabel untuk proses rekonstruksi citra pada kasus citra berwarna

No	Nama variabel	Tipe	Penjelasan
1	rekonstruksi_ citra	Array of double	Matriks output citra hasil rekonstruksi
2	at2	Array of integer	Indeks hasil dekomposisi data tes

Tabel 3.10 Daftar fungsi untuk proses rekonstruksi pada kasus citra berwarna

No	Nama fungsi	Penjelasan
1	waverec2	Fungsi untuk rekonstruksi citra

3.7. Perhitungan PSNR

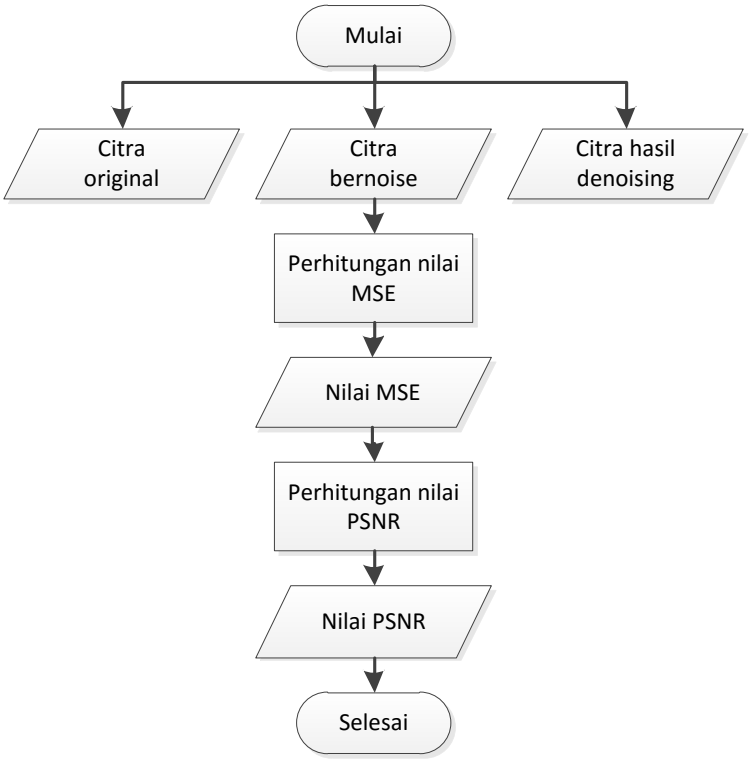
Pada proses ini dilakukan perhitungan MSE yang kemudian nilai MSE digunakan untuk menghitung nilai PSNR dengan persamaan (2.13). Nilai PSNR digunakan untuk mengukur keberhasilan dari metode *denoising* pada citra. Semakin besar nilai

PSNR maka semakin berkurang juga *noise* pada citra tersebut. Diagram alir dari proses ini ditunjukkan pada Gambar 3.16.

Masukan	Hasil proses *soft-thresholding* ytsoft dan indeks dekomposisi data tes at2
Keluaran	Citra hasil rekonstruksi

```
1.  rekonstruksi_citra = waverec2 (ytsoft, at2,
    'wname');
```

Gambar 3.15 *Pseudocode* proses rekonstruksi citra



Gambar 3.16 Diagram alir perhitungan PSNR

Daftar variabel dan fungsi yang digunakan untuk perhitungan PSNR pada kasus citra berwarna ditunjukkan pada Tabel 3.11, Tabel 3.12 dan Tabel 3.13.

Pseudocode perhitungan PSNR pada kasus citra berwarna ini dapat dilihat pada Gambar 3.17 dan Gambar 3.18.

Tabel 3.11 Daftar variabel untuk perhitungan PSNR pada kasus citra berwarna (bagian pertama)

No	Nama variabel	Tipe	Penjelasan
1	N	Integer	Ukuran dari citra input
2	M	Integer	Ukuran baris citra
3	NN	Integer	Ukuran kolom citra
4	mseRImagei	Double	Perhitungan nilai MSE untuk citra input chanel red (R)
5	mseGImagei	Double	Perhitungan nilai MSE untuk citra input chanel green (G)
6	mseBImagei	Double	Perhitungan nilai MSE untuk citra input chanel blue (B)
7	mseRi	Double	Perhitungan nilai MSE untuk citra input chanel red (R)
8	mseGi	Double	Perhitungan nilai MSE untuk citra input chanel green (G)
9	mseBi	Double	Perhitungan nilai MSE untuk citra input chanel blue (B)
10	msei	Double	Nilai MSE input
11	PSNR_valuei	Double	Nilai PSNR input
12	mseRImage	Double	Perhitungan nilai MSE untuk citra output chanel red (R)
13	mseGImage	Double	Perhitungan nilai MSE untuk citra output chanel green (G)
14	mseBImage	Double	Perhitungan nilai MSE untuk citra output chanel blue (B)

Tabel 3.12 Daftar variabel untuk perhitungan PSNR pada kasus citra berwarna (bagian kedua)

No	Nama variabel	Tipe	Penjelasan
15	mseR	Double	Perhitungan nilai MSE untuk citra output chanel red (R)
16	mseG	Double	Perhitungan nilai MSE untuk citra output chanel green (G)
17	mseB	Double	Perhitungan nilai MSE untuk citra output chanel blue (B)
18	mse	Double	Nilai MSE output
19	PSNR_value	Double	Nilai PSNR output
20	ori	Array of integer	Matriks citra original
21	imgrgb	Array of double	Matriks citra hasil *denoising*

Tabel 3.13 Daftar fungsi untuk perhitungan PSNR pada kasus citra berwarna

No	Nama fungsi	Penjelasan
1	log	Fungsi untuk menghitung logaritma

Masukan	Matriks citra original, matriks citra ber*noise*, matriks citra hasil *denoising*
Keluaran	Nilai PSNR

```
1.  n = size(ori);
2.  M = nnn(1);
3.  NN = nnn(2);
4.  %input
5.  mseRImagei = (double(ori(:,:,1)) -
6.  double(img_noise_(:,:,1))) .^ 2;
7.  mseGImagei = (double(ori(:,:,2)) -
8.  double(img_noise_(:,:,2))) .^ 2;
9.  mseBImagei = (double(ori(:,:,3)) -
```

Gambar 3.17 *Pseudocode* perhitungan PSNR (bagian pertama)

Masukan	Matriks citra original, matriks citra ber*noise*, matriks citra hasil *denoising*
Keluaran	Nilai PSNR

```
10. double(img noise(:,:,3))) .^ 2;
11. mseRi = sum(sum(mseRImagei)) / (MM * NNN);
12. mseGi = sum(sum(mseGImagei)) / (MM * NNN);
13. mseBi = sum(sum(mseBImagei)) / (MM * NNN);
14. msei = (mseRi + mseGi + mseBi)/3;
15. PSNR_Valuei = 10 * log10( 255^2 / msei);
16. %output
17. mseRImage = (double(ori(:,:,1)) -
    double(imgrgb(:,:,1))) .^ 2;
18. mseGImage      =      (double(ori(:,:,2))      -
    double(imgrgb(:,:,2))) .^ 2;
19. mseBImage = (double(ori(:,:,3)) -
    double(imgrgb(:,:,3))) .^ 2;
20. mseR = sum(sum(mseRImage)) / (M * NN);
21. mseG = sum(sum(mseGImage)) / (M * NN);
22. mseB = sum(sum(mseBImage)) / (M * NN);
23. mse = (mseR + mseG + mseB)/3;
24. PSNR_Value = 10 * log10( 255^2 / mse);
```

Gambar 3.18 *Pseudocode* perhitungan PSNR (bagian kedua)

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1. Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi adalah MATLAB R2014a yang diinstal pada sistem operasi *Windows* 10 64-bit.

4.2. Implementasi

Pada subbab ini akan dijelaskan implementasi dari setiap subbab yang terdapat pada bab sebelumnya yaitu bab perancangan perangkat lunak. Pada bagian implementasi ini dilakukan 2 fase, yaitu fase *training* dan fase *testing*. Fase *training* dimulai dari pemberian *noise*, dekomposisi wavelet dan proses *clustering*. Fase *testing* juga akan diberlakukan sama seperti *training*, yaitu dimulai dari pemberian *noise*, dekomposisi wavelet. Namun hanya sampai proses itu saja. Kemudian langsung masuk ke proses *denoising*, rekonstruksi citra dan perhitungan PSNR.

4.2.1. Implementasi Pemberian *Noise* Gaussian pada Citra

Proses Penambahan *Noise* mengimplementasikan desain pada subbab 3.2. Pertama citra RGB diubah menurut komponen warnanya yaitu, *channel red, green,* dan *blue*. Untuk menambahkan *noise* menggunakan fungsi im*noise* dan pilih ‘gaussian’. *Noise* Gaussian terpilih untuk dimasukkan ke dalam citra dengan menetukan standar deviasi dari tiap citra. Implementasi Proses Penambahan *Noise* ditunjukkan pada Kode Sumber 4.1.

```
1.    let IN[0..MAXR] be a new array
2.    for i = 1 to 3 to h
3.        for j = 1 to 3 to w
4.            IN = imnoise(input,'type noise', mean,
      varian);
```

Kode Sumber 4.1 Implementasi pemberian *noise* pada citra

4.2.2. Implementasi Dekomposisi Wavelet

Proses Wavelet Dekomposisi telah dijelaskan pada subbab 3.3. Proses ini bertujuan untuk mendapatkan koefisien wavelet dari sinyal *noise*.

4.2.2.1. Transformasi Wavelet Diskrit 2D

Parameter yang digunakan yaitu input dari citra ber*noise*, tipe filter wavelet, level dekomposisi. Implementasi dari Proses Wavelet Dekomposisi ditunjukkan pada Kode Sumber 4.2.

```
1.  no_level = jumlah level;
2.  [a1,a2]=wavedec2(IN,no_level, 'wname');
3.  approks = appcoef2(a1,a2, 'wname',no_level);
```

Kode Sumber 4.2 Implementasi Transformasi Wavelet Diskrit 2D

4.2.2.2. Implementasi Perhitungan Statistika Citra

Proses Statistik Citra akan menghitung *mean*, *median* dan standar deviasi dari koefisien wavelet yang dijelaskan pada subbab 3.3.2. Implementasi dari Proses Statistik Citra dapat dilihat pada Kode Sumber 4.3.

```
1.  arr = 1
2.  final(arr) = mean2(approks); arr = arr + 1;
3.  temp = median(approks);
4.  final(arr) = median(temp); arr = arr + 1;
5.  final(arr) = std2(approks); arr = arr + 1;
```

Kode Sumber 4.3 Implementasi perhitungan statistika citra

4.2.3. Implementasi *Clustering* K-Means

Pada subbab ini akan dijelaskan implementasi dari proses *clustering*. Metode yang digunakan yaitu *clustering* K-*Mean*s. Proses ini akan melakukan *clustering* terhadap hasil dari dekomposisi citra yang telah dilakukan. Hasil *clustering* ini akan menjadi parameter untuk proses selanjutnya. Fungsi ini merupakan implementasi dari desain pada subbab 3.4. Implementasi dari Proses K-Means dapat dilihat pada Kode Sumber 4.4.

```
1.  denoisee = load('datatraining.mat');
2.  k = banyak cluster yang diinginkan;
3.  opts = statset('Display','final');
4.  jumlah_iterasi = banyaknya iterasi;
5.  [idx,C]=kmeans(denoisee.final,k,'Distance',
    'sqeuclidean',...
    'Replicates',jumlah_iterasi, 'Options',opts);
```

Kode Sumber 4.4 Implementasi *clustering* K-Means

4.2.4. Implementasi Proses *Denoising*

Pada subbab ini akan dijelaskan implementasi dari proses *denoising* yang dimulai dari memasukkan data tes dan dilakukan fase *testing*. Data tes akan diberlakukan sama seperti data *training*, yaitu diberi *noise* dan didekomposisi menggunakan wavelet. Kemudian hasil dekomposisi tersebut akan dilakukan perhitungan jarak Euclidean, perhitungan nilai *thresholding* dan proses *soft-thresholding*.

4.2.4.1. Jarak Euclidean

Proses jarak Euclidean akan menghitung jarak antara 15 *centroid* dengan data tes dan akan diambil jarak terdekatnya untuk menentukan *centroid* mana yang paling dekat dengan data tes. Fungsi ini merupakan implementasi dari desain pada subbab 3.5.1. Implementasi dari Proses jarak Euclidean dapat dilihat pada Kode Sumber 4.5.

```
1.  for m=1 : size(final2,1)
2.     for n=1 : size(C,1)
3.        euc(n) = sqrt(sum((final2(m)-C(n)) .^2));
4.     End
5.     [sorted, index] = sort(euc);
6.     sort_Centroid = C(index,:);
7.     sort_Centroid1 = f(1,:);
8.  End
```

Kode Sumber 4.5 Implementasi jarak *Euclidean*

4.2.4.2. Perhitungan Nilai *Thresholding*

Fungsi ini merupakan implementasi dari desain proses *Thresholding* pada subbab 3.5.2. Implementasi dari Proses *Thresholding* dapat dilihat pada Kode Sumber 4.6.

```
1.  mean_Centroid = mean(sort_Centroid1);
2.  median_Centroid = median(sort_Centroid1);
3.  std_Centroid = std(sort_Centroid1);
4.  Thresh=(std_Centroid/mean_Centroid)*
    med_Centroid;
5.  Thresholdval = Thresh/2^(no_level-1);
6.  Thresholdval = Thresh/2^(no_level-1);
```

Kode Sumber 4.6 Implementasi perhitungan nilai *thresholding*

4.2.4.3. Proses *Soft-Thresholding*

Proses *Soft-thresholding* akan menjelaskan proses yang sudah dijelaskan pada subbab 3.5.3. Implementasi dari Proses *Soft-thresholding* dapat dilihat pada Kode Sumber 4.7.

```
1.  ytsoft=sign(at1).*(((abs(at1)-
    Thresholdval)>=0).* (abs(at1)-Thresholdval));
```

Kode Sumber 4.7 Implementasi proses *soft-thresholding*

4.2.5. Implementasi Rekonstruksi Citra

Proses ini merupakan implementasi dari desain proses Rekonstruksi Citra pada subbab 3.6. Rekonstruksi dilakukan sesuai

dengan level dekomposisi. Pada proses ini hasil bagian *lowpass filter* dijumlahkan dengan bagian *highpass filter*. Sehingga hasil penjumlahan ini akan membentuk sinyal output 1 dimensi. Matrik 1 dimensi ini digabung menjadi satu dengan matrik 1 dimensi lainnya menjadi citra hasil *denoising* 2 dimensi. Implementasi dari Proses Rekonstruksi Citra dapat dilihat pada Kode Sumber 4.8.

```
1.  rekonstruksi_citra = waverec2 (ytsoft, at2,
    'wname');
```

Kode Sumber 4.8 Implementasi rekonstruksi citra

4.2.5.1. Implementasi Perhitungan PSNR

Untuk menghitung nilai PSNR diperlukan nilai MSE. Perhitungan MSE menggunakan formula pada subbab 2.9 dan PSNR dengan formula di subbab 2.10. Input MSE didapatkan dari dari nilai citra original dan citra ber-*noise*, yang selanjutkan akan digunakan untuk menghitung nilai input PSNR. Sedangkan untuk output MSE didapatkan dari nilai citra original dan citra hasil *denoising*, yang selanjutnya digunakan untuk menghitung nilai output PSNR.Fungsi ini merupakan implementasi dari desain proses PSNR yang terdapat pada subbab 3.7. Implementasi dari Proses PSNR dapat dilihat pada Kode Sumber 4.9 dan Kode Sumber 4.10.

```
1.  n = size(ori);
2.  M = nnn(1);
3.  NN = nnn(2);
4.  %input
5.  mseRImagei     =     (double(ori(:,:,1))     -
    double(img_noise(:,:,1))) .^ 2;
6.  mseGImagei     =     (double(ori(:,:,2))     -
    double(img_noise(:,:,2))) .^ 2;
7.  mseBImagei     =     (double(ori(:,:,3))     -
    double(img_noise(:,:,3))) .^ 2;
8.  mseRi = sum(sum(mseRImagei)) / (MM * NNN);
```

Kode Sumber 4.9 Implementasi perhitungan PSNR (bagian pertama)

```
9.  mseGi = sum(sum(mseGImagei)) / (MM * NNN);
10. mseBi = sum(sum(mseBImagei)) / (MM * NNN);
11. msei = (mseRi + mseGi + mseBi)/3;
12. PSNR_Valuei = 10 * log10( 255^2 / msei);
13. %output
14. mseRImage     =     (double(ori(:,:,1))     -
    double(imgrgb(:,:,1))) .^ 2;
15. mseGImage     =     (double(ori(:,:,2))     -
    double(imgrgb(:,:,2))) .^ 2;
16  mseBImage     =     (double(ori(:,:,3))     -
    double(imgrgb(:,:,3))) .^ 2;
17. mseR = sum(sum(mseRImage)) / (M * NN);
18. mseG = sum(sum(mseGImage)) / (M * NN);
19. mseB = sum(sum(mseBImage)) / (M * NN);
20. mse = (mseR + mseG + mseB)/3;
21. PSNR_Value = 10 * log10( 255^2 / mse);
```

Kode Sumber 4.10 Implementasi perhitungan PSNR (bagian kedua)

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada sistem yang dikembangkan. Pembahasan pada bab ini meliputi lingkungan uji coba, hasil uji coba, dan evaluasi.

5.1. Lingkungan Pengujian

Lingkungan uji coba yang digunakan dalam pembuatan Tugas Akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk melakukan uji coba *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding*. Lingkungan pengujian merupakan komputer tempat uji coba sistem dengan spesifikasi yang dijelaskan sebagai berikut:

Prosesor : Intel® Core™ i3-4030U CPU @ 1.90GHz.
Memori : 2.00 GB
Jenis *Device* : *Notebook*
Sistem Operasi : Microsoft Windows 10

5.2. Data Uji Coba

Citra yang digunakan dalam tugas akhir ini berupa gambar yang diambil ketika melakukan liburan seperti pemandangan alam, perkotaan, perumahan, dan lain sebagainya. Data yang digunakan berformat *.jpg* dan berupa citra RGB. Data dibagi menjadi data *training* dan data test. Untuk data *training* digunakan 512 citra dari dataset yang telah disediakan. Uji coba yang dilakukan menggunakan citra 101101.jpg, 101401.jpg, 102001.jpg, 105401.jpg, 108501.jpg, 117101.jpg, 117601.jpg, dan 117801.jpg dengan ukuran yang diperkecil menjadi 256x256 piksel. Untuk masing-masing citra akan memberikan hasil output MSE dan PSNR yang berbeda. Daftar citra yang digunakan untuk uji coba ditunjukkan pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Daftar citra untuk uji coba (bagian pertama)

No	Data	Citra Berwarna
1.	Gambar 1	 101101.jpg
2.	Gambar 2	 101401.jpg
3.	Gambar 3	 102001.jpg
4.	Gambar 4	 105401.jpg

Tabel 5.2 Daftar citra untuk uji coba (bagian kedua)

No	Data	Citra Berwarna
5.	Gambar 5	 108501.jpg
6.	Gambar 6	 117101.jpg
7.	Gambar 7	 117601.jpg
8.	Gambar 8	 117801.jpg

Hasil uji coba akan memberikan hasil MSE dan PSNR. Hasil MSE digunakan untuk menghitung beda (kesalahan) antara citra input dengan citra output. Sedangkan hasil PSNR digunakan untuk menghitung rasio citra ouput terhadap *noise*.

Pada tiap uji coba, hasil Analisa ditampilkan dalam bentuk grafik. Hasil uji coba pada bab ini hanya ditampilkan berupa table nilai MSE dan PSNR sedangkan hasil uji coba berupa citra dapat dilihat pada lampiran buku tugas akhir.

5.3. Hasil Uji Coba

Berikut ini adalah seluruh hasil uji coba yang menunjukkan kinerja algoritma *clustering* k-*mean*s dalam domain wavelet menggunakan adapftif *soft-thresholding* untuk perbaikan citra ber-*noise*. Hasil uji coba akan menunjukkan hasil MSE dan PSNR yang berbeda-beda untuk gambar yang berbeda-beda. Hasil dari uji coba ini akan digunakan untuk menarik kesimpulan pada tugas akhir ini.

5.3.1. Uji Coba pada Gambar 1

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 1 dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil uji coba gambar 1

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0087	68.69	0.0052	70.90	143
0.02	0.0167	65.90	0.0068	70.13	146
0.03	0.0241	64.32	0.0088	68.84	154
0.04	0.0308	63.20	0.0111	67.99	145
0.05	0.0374	62.39	0.0115	67.40	143
0.06	0.0438	61.72	0.0132	66.91	140
0.07	0.0494	61.16	0.0217	66.27	148
0.08	0.0550	60.71	0.0197	65.17	146
0.09	0.0601	60.29	0.0195	65.09	152
0.1	0.0654	59.95	0.0275	63.94	154

Berdasarkan Tabel 5.3, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 2-5 dB dan metode ini dapat diterapkan hingga nilai varian 0.09 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini antara 143-154 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.1.

5.3.2. Uji Coba pada Gambar 2

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 2 dapat dilihat pada Tabel 5.4.

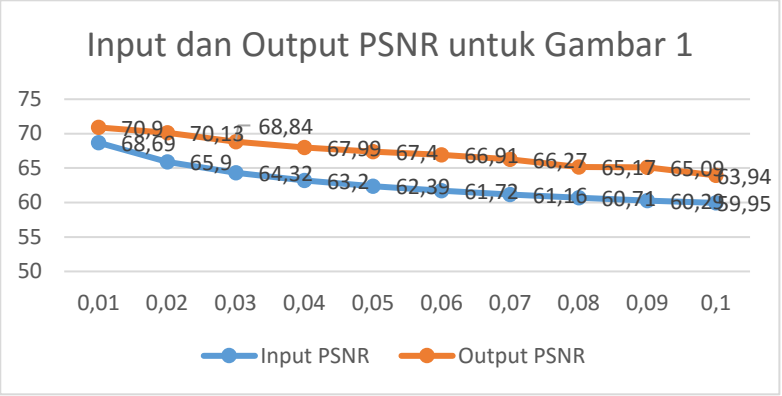
Berdasarkan Tabel 5.4, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 4-6 dB dan metode ini dapat diterapkan hingga nilai varian 0.08 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini antara 143-157 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.2.

5.3.3. Uji Coba pada Gambar 3

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 3 dapat dilihat pada Tabel 5.5.

Berdasarkan Tabel 5.5, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 3-5 dB dan metode ini dapat diterapkan hingga nilai varian 0.06 karena menghasilkan nilai PSNR lebih besar sama

dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini antara 139-148 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.3.



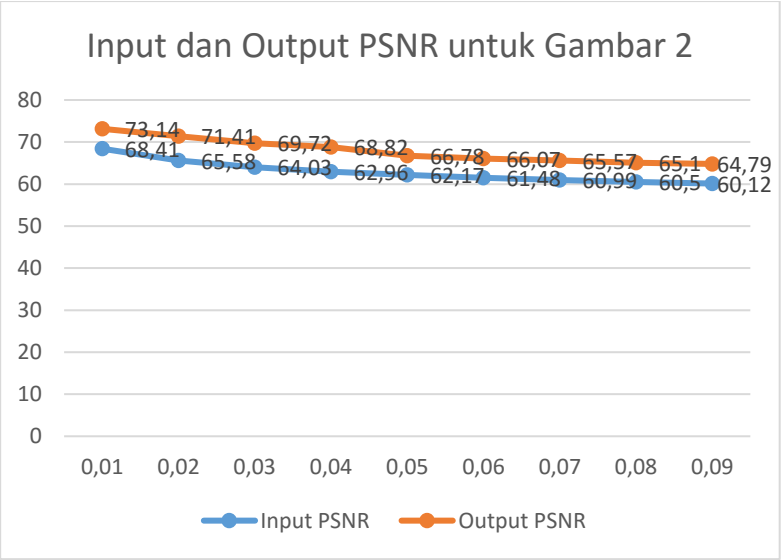
Gambar 5.1 Grafik PSNR gambar 1

Tabel 5.4 Hasil uji coba gambar 2

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0093	68.41	0.0024	73.14	143
0.02	0.0178	65.58	0.0045	71.41	157
0.03	0.0255	64.03	0.0061	69.72	147
0.04	0.0328	62.96	0.0096	68.82	144
0.05	0.0394	62.17	0.0113	66.78	150
0.06	0.0462	61.48	0.0149	66.07	143
0.07	0.0521	60.99	0.0210	65.57	146
0.08	0.0579	60.50	0.0287	65.10	148
0.09	0.0629	60.12	0.0211	64.79	149

5.3.4. Uji Coba pada Gambar 4

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 4 dapat dilihat pada Tabel 5.6.



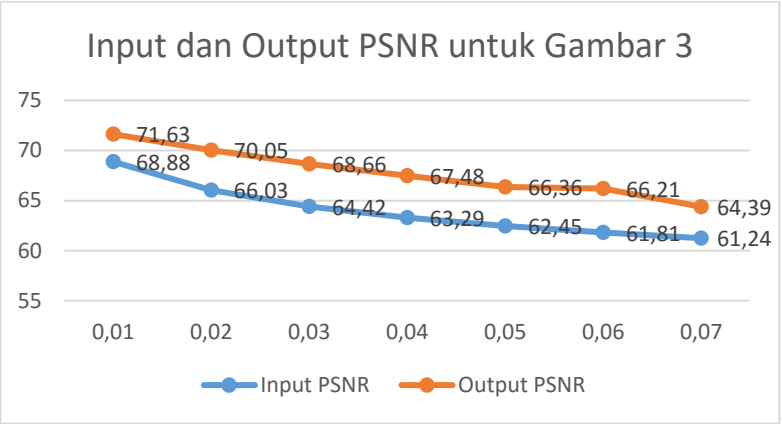
Gambar 5.2 Grafik PSNR gambar 2

Tabel 5.5 Hasil uji coba gambar 3

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0084	68.88	0.0040	71.63	147
0.02	0.0161	66.03	0.0069	70.05	148
0.03	0.0234	64.42	0.0092	68.66	145
0.04	0.0303	63.29	0.0110	67.48	147
0.05	0.0366	62.45	0.0149	66.36	143
0.06	0.0428	61.81	0.0164	66.21	139
0.07	0.0490	61.24	0.0189	64.39	147

Berdasarkan Tabel 5.6, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 3-5 dB dan metode ini dapat diterapkan hingga nilai varian 0.08 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini

antara 141-150 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.4.



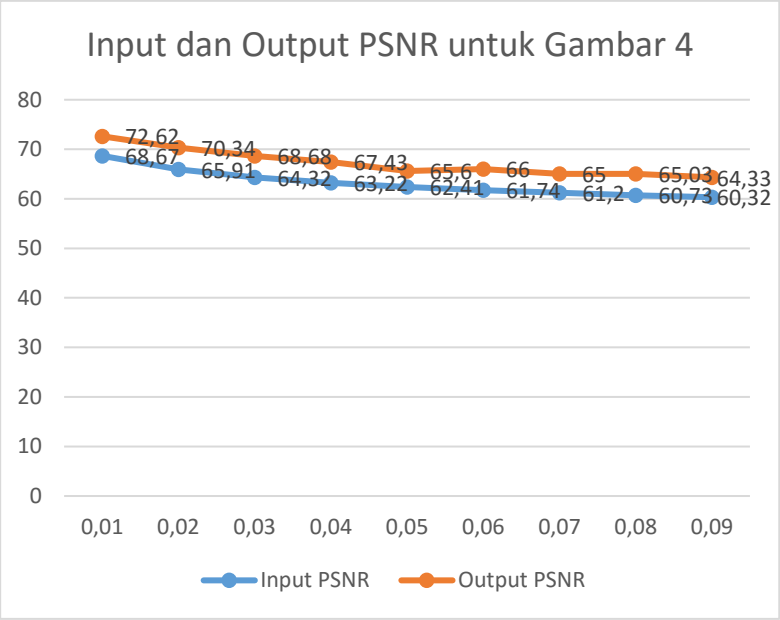
Gambar 5.3 Grafik PSNR gambar 3

Tabel 5.6 Hasil uji coba gambar 4

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0088	68.67	0.0039	72.62	147
0.02	0.0167	65.91	0.0053	70.34	145
0.03	0.0241	64.32	0.0069	68.68	144
0.04	0.0311	63.22	0.0108	67.43	141
0.05	0.0375	62.41	0.0113	65.60	141
0.06	0.0434	61.74	0.0163	66	150
0.07	0.0493	61.20	0.0192	65	147
0.08	0.0552	60.73	0.0271	65.03	146
0.09	0.0608	60.32	0.0247	64.33	149

5.3.5. Uji Coba pada Gambar 5

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 5 dapat dilihat pada Tabel 5.7.



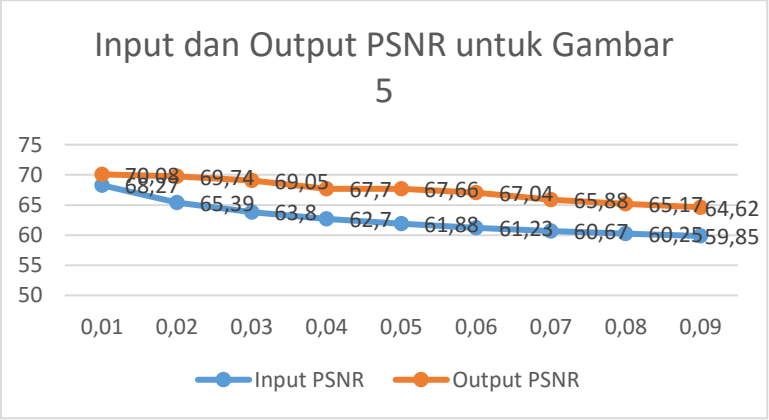
Gambar 5.4 Grafik PSNR gambar 4

Tabel 5.7 Hasil uji coba gambar 5

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0097	68.27	0.0058	70.08	149
0.02	0.0187	65.39	0.0068	69.74	151
0.03	0.0270	63.80	0.0080	69.05	153
0.04	0.0349	62.70	0.0099	67.70	148
0.05	0.0424	61.88	0.0120	67.66	152
0.06	0.0493	61.23	0.0149	67.04	140
0.07	0.0553	60.67	0.0209	65.88	148
0.08	0.0615	60.25	0.0223	65.17	145
0.09	0.0669	59.85	0.0223	64.62	148

Berdasarkan Tabel 5.7, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet

menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 2-6 dB dan metode ini dapat diterapkan hingga nilai varian 0.08 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini antara 140-153 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.5.



Gambar 5.5 Grafik PSNR gambar 5

5.3.6. Uji Coba pada Gambar 6

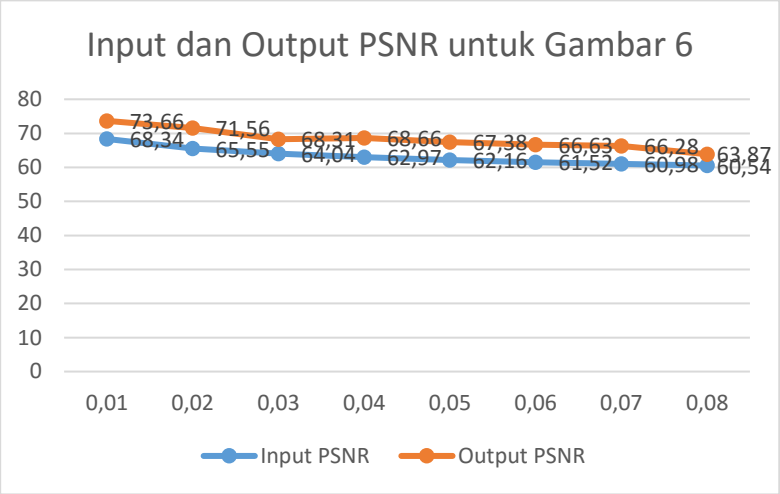
Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 6 dapat dilihat pada Tabel 5.8**.**

Berdasarkan Tabel 5.8, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 3-6 dB dan metode ini dapat diterapkan hingga nilai varian 0.07 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini

antara 141-160 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.6.

Tabel 5.8 Hasil uji coba gambar 6

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0095	68.34	0.0029	73.66	141
0.02	0.0180	65.55	0.0045	71.56	154
0.03	0.0256	64.04	0.0058	68.31	154
0.04	0.0328	62.97	0.0080	68.66	146
0.05	0.0395	62.16	0.0135	67.38	157
0.06	0.0458	61.52	0.0104	66.63	141
0.07	0.0515	60.98	0.0151	66.28	147
0.08	0.0571	60.54	0.0205	63.87	146



Gambar 5.6 Grafik PSNR gambar 6

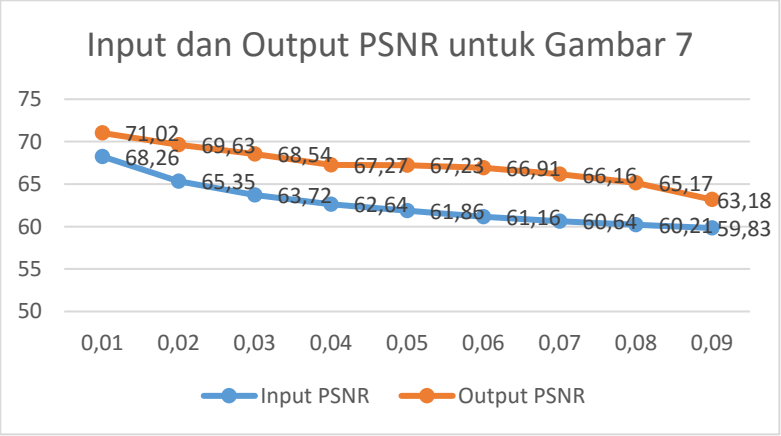
5.3.7. Uji Coba pada Gambar 7

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 7 dapat dilihat pada Tabel 5.9.

Berdasarkan Tabel 5.9, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 3-5 dB dan metode ini dapat diterapkan hingga nilai varian 0.08 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini antara 142-151 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.7.

Tabel 5.9 Hasil uji coba gambar 7

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0097	68.26	0.0066	71.02	144
0.02	0.0189	65.35	0.0087	69.63	148
0.03	0.0274	63.72	0.0087	68.54	149
0.04	0.0352	62.64	0.0135	67.27	144
0.05	0.0426	61.86	0.0118	67.23	145
0.06	0.0495	61.16	0.0141	66.91	142
0.07	0.0560	60.64	0.0188	66.16	146
0.08	0.0619	60.21	0.0190	65.17	151
0.09	0.0676	59.83	0.0272	63.18	150



Gambar 5.7 Grafik PSNR gambar 7

5.3.8. Uji Coba pada Gambar 8

Uji coba dilakukan dengan memberi nilai varian *noise* yang berbeda-beda. Hasil uji coba pada gambar 8 dapat dilihat pada Tabel 5.10**.**

Berdasarkan Tabel 5.10, hasil uji coba implementasi algoritma *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* menghasilkan nilai output PSNR yang lebih baik daripada citra ber-*noise*. Peningkatan output PSNRnya antara 3-6 dB dan metode ini dapat diterapkan hingga nilai varian 0.07 karena menghasilkan nilai PSNR lebih besar sama dengan 65. Waktu yang dibutuhkan untuk proses *denoising* ini antara 141-191 detik. Grafik perbandingan input PSNR dan output PSNR dapat dilihat pada Gambar 5.8.

Tabel 5.10 Hasil uji coba gambar 8

Nilai Varian	Input MSE	Input PSNR (dB)	Output MSE	Output PSNR (dB)	Waktu (s)
0.01	0.0098	68.18	0.0045	71.34	150
0.02	0.0192	65.21	0.0058	70.44	191
0.03	0.0280	63.65	0.0077	68.61	172
0.04	0.0363	62.53	0.0087	67.68	145
0.05	0.0438	61.71	0.0126	65.66	152
0.06	0.0512	61.04	0.0204	67.07	141
0.07	0.0574	60.52	0.0194	65.46	158
0.08	0.0635	60.08	0.0224	64.02	146

5.4. Evaluasi Perbandingan Nilai Varian *Noise*

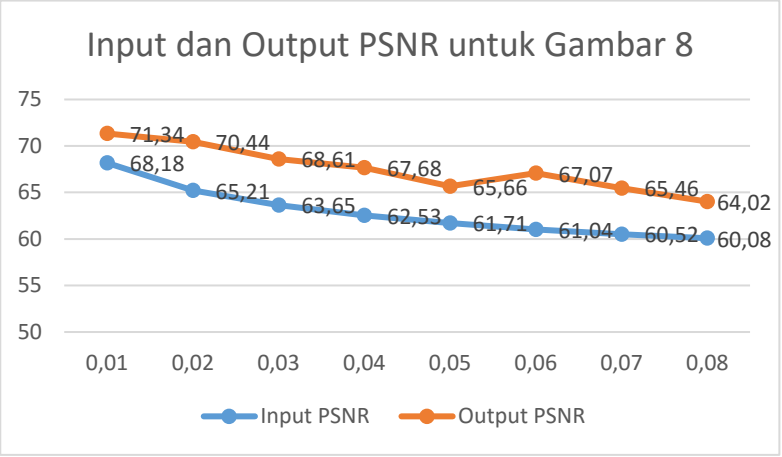
Berdasarkan hasil uji coba yang telah dilakukan pada data uji coba maka dapat disimpulkan bahwa metode tugas akhir ini terbukti untuk memperbaiki citra yang mengandung Gaussian *noise* pada citra berwarna. Hal ini dibuktikan dengan nilai output PSNR yang lebih besar dari nilai input PSNR serta menurunnya nilai output MSE dari nilai input MSE yang dapat dilihat pada

Tabel 5.3 sampai dengan Tabel 5.10 yang artinya citra dapat dikatakan telah berkurang *noise*nya. Metode ini dapat melakukan *denoising* hingga nilai varian rata-rata 0.06 pada setiap citra.

Gambar 6 memiliki rata-rata nilai output PSNR lebih besar dibandingkan dengan gambar yang lain yaitu sebesar 69.36 dB. Hal ini dapat dilihat pada Tabel 5.11.

Hasil reduksi *noise* dipengaruhi oleh nilai varian *noise* yang diberikan ke citra. Semakin kecil nilai varian *noise* maka semakin besar nilai PSNR yang berarti kualitas hasil reduksi *noise* semakin tinggi.

Pada Tabel 5.12 dan Tabel 5.13 dapat dilihat bahwa semakin besar varian *noise* yang diberikan kepada citra maka rata-rata output PSNR semakin kecil dan memiliki kompleksitas waktu yang berbeda-beda. Dengan kata lain pengurangan *noise* citra hasil proses *denoising* akan semakin kecil ketika nilai *noise* bertambah. Grafik pengaruh nilai varian *noise* terhadap output PSNR dapat dilihat pada Gambar 5.9.



Gambar 5.8 Grafik PSNR gambar 8

Tabel 5.11 Pengaruh nilai varian terhadap output PSNR dan waktu komputasi

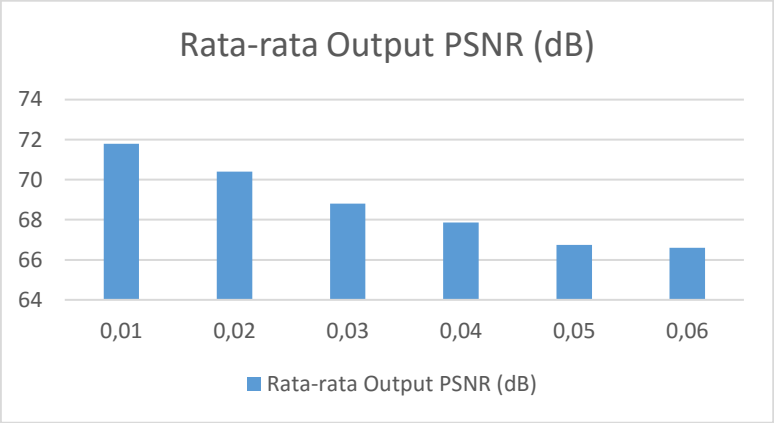
Nilai Varian-ce	Output PSNR					
	101101	Waktu (s)	101401	Waktu (s)	102001	Waktu (s)
0.01	70.90	143	73.14	143	71.63	147
0.02	70.13	146	71.41	157	70.05	148
0.03	68.84	154	69.72	147	68.66	145
0.04	67.99	145	68.82	144	67.48	147
0.05	67.40	143	66.78	150	66.36	143
0.06	66.91	140	66.07	143	66.21	139
Rata-rata	68.44	145	69.32	147	68.39	144
Nilai Varian-ce		Output PSNR				
	105401	Waktu (s)	108501	Waktu (s)	117101	Waktu (s)
0.01	72.62	147	70.08	149	73.66	141
0.02	70.34	145	69.74	151	71.56	154
0.03	68.68	144	69.05	153	68.31	154
0.04	67.43	141	67.70	148	68.66	146
0.05	65.60	141	67.66	152	67.38	157
0.06	66	150	67.04	140	66.63	141
Rata-rata	68.44	144	68.54	148	**69.36**	148
Nilai Varian-ce	Output PSNR					
	117601		Waktu (s)	117801		Waktu (s)
0.01	71.02		144	71.34		150
0.02	69.63		148	70.44		191
0.03	68.54		149	68.61		172
0.04	67.27		144	67.68		145
0.05	67.23		145	65.66		152
0.06	66.91		142	67.07		141
Rata-rata	68.43		145	68.46		158

Tabel 5.12 Pengaruh Nilai Varian Terhadap Output PSNR dan Waktu Komputasi (bagian pertama)

Nilai Varian-ce	Rata-rata Output PSNR (dB)	Rata-rata Waktu Komputasi (s)
0.01	71.79	145
0.02	70.41	155
0.03	68.80	152

Tabel 5.13 Pengaruh Nilai Varian Terhadap Output PSNR dan Waktu Komputasi (bagian kedua)

Nilai Vari-ance	Rata-rata Output PSNR (dB)	Rata-rata Waktu Komputasi (s)
0.04	67.87	145
0.05	66.75	147
0.06	66.60	142



Gambar 5.9 Grafik pengaruh nilai varian terhadap output PSNR

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dari hasil selama proses perancangan, implementasi, serta pengujian dapat diambil kesimpulan sebagai berikut:

1. Metode tugas akhir ini terbukti dapat memperbaiki citra yang mengandung Gaussian *noise* pada citra berwarna. Hal ini dibuktikan dengan nilai output PSNR yang lebih besar dari nilai input PSNR serta menurunnya nilai output MSE dari nilai input MSE. Peningkatan output PSNR dari input PSNR tiap citra berkisar 2-6 dB pada varian 0.01-0.06.
2. Metode ini dapat melakukan *denoising* hingga nilai varian rata-rata 0.06 dengan peningkatan PSNR sekitar 2-6 dB. Hal ini menunjukkan citra dianggap baik dengan nilai PSNR lebih dari sama dengan 65 dB.
3. Nilai varian *noise* mempengaruhi PSNR, namun waktu komputasi tidak terlalu berpengaruh.
4. Gambar 6 memiliki rata-rata nilai PSNR lebih besar dibandingkan dengan citra yang lain yaitu sebesar 69.36 dB.

6.2. Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Diantaranya adalah sebagai berikut:

1. Untuk selanjutnya perlu dikembangkan dengan menggunakan UI.

2. Perlu dilakukan uji coba yang lebih mendalam untuk mengetahui efektivitas hasil perbaikan dengan metode *clustering* K-Means dalam domain wavelet menggunakan adaptif *soft-thresholding* dalam citra RGB.

DAFTAR PUSTAKA

[1] P. Chatterjee and P. Milanfar, "Clustering-based Denoising with Locally Learned Dictionaries," *IEEE Transactions on Image Processing*, vol. 18, no. No.7, July 2009.

[2] A. Khare and U. S. Tiwary, "Soft-Thresholding Denoising of Medical Images-A Multiresolution Approach," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 3, no. No.4, pp. 477-496, 2005.

[3] R. C. Gonzales and R. E. Woods, Digital Image Processing, 2008.

[4] P. Kamboj and V. Rain, "A Brief Study of Various Noise Model and Filtering Techniques," *Journal of Global Research In Computer Science*, vol. 4, April 2013.

[5] J. Saedi, M. H. Moradi and K. Faez, "A New Wavelet-Based Fuzzy Single and Multi-Channel Image Denoising," *Elvesier Journal, Image and Vision Computing*, vol. 28, pp. 1161-1623, 2010.

[6] H. Om and M. Biswas, "An Improved Image Denoising Method Based On Wavelet Thresholding," *Journal of Signal and Information Processing*, pp. 109-116, 2012.

[7] A. Utkarsh, U. S. Tiwary, R. S. Kumar and P. D. S, "K-Means Clustering for Adaptive Wavelet Based Image Denoising," *International Conference on Advances in Computer Engineering and Applications (ICACEA)*, 2015.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Table A Perbandingan hasil *denoising* gambar 1 dengan nilai varian = 0.01

Citra Asli Tanpa *Noise*	
	
Citra Ber-*noise*	**Citra Hasil *Denoising***
	

Table B Perbandingan hasil *denoising* gambar 2 dengan nilai varian = 0.02 (bagian pertama)

Citra Asli Tanpa *Noise*


Table C Perbandingan hasil *denoising* gambar 2 dengan nilai varian = 0.02 (bagian kedua)

Citra Ber-*noise*	Citra Hasil *Denoising*
	

Table D Perbandingan hasil *denoising* gambar 3 dengan nilai varian = 0.03

Citra Asli Tanpa *Noise*	
	
Citra Ber-*noise*	**Citra Hasil *Denoising***
	

Table E Perbandingan hasil *denoising* gambar 4 dengan nilai varian = 0.04

Citra Asli Tanpa *Noise*	
	
Citra Ber-*noise*	**Citra Hasil *Denoising***
	

Table F Perbandingan hasil *denoising* gambar 5 dengan nilai varian = 0.05 (bagian pertama)

Citra Asli Tanpa *Noise*


Table G Perbandingan hasil *denoising* gambar 5 dengan nilai varian = 0.05 (bagian kedua)

Citra Ber-*noise*	Citra Hasil *Denoising*
	

Table H Perbandingan hasil *denoising* gambar 6 dengan nilai varian = 0.06

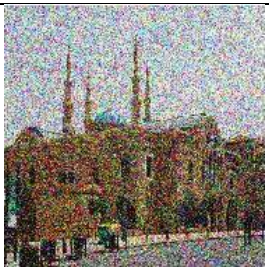
Citra Asli Tanpa *Noise*	
	
Citra Ber-*noise*	**Citra Hasil *Denoising***
	

Table I Perbandingan hasil *denoising* gambar 7 dengan nilai varian = 0.08

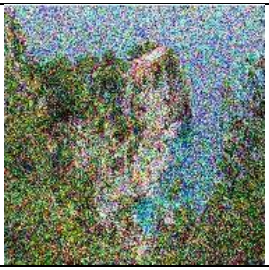
Citra Asli Tanpa *Noise*	
	
Citra Ber-*noise*	**Citra Hasil *Denoising***
	

Table J Perbandingan hasil *denoising* gambar 8 dengan nilai varian = 0.07 (bagian pertama)

Citra Asli Tanpa *Noise*


Table K Perbandingan hasil *denoising* gambar dengan nilai varian = 0.07 (bagian kedua)

Citra Ber-*noise*	Citra Hasil *Denoising*
	

BIODATA PENULIS

Biandina Meidyani, lahir di Banjarmasin pada tanggal 04 Mei 1994. Penulis telah menempuh pendidikan di SDN Banjarbaru Utara 1 (2000-2006), SMPN 1 Banjarbaru (2006-2009), SMAN 1 Banjarbaru (2009-2012). Pada tahun 2012, penulis diterima di S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Di jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Cerdas dan Visualisasi (KCV). Selama masa kuliah, penulis aktif dalam pada organisasi mahasiswa tingkat jurusan, yaitu Himpunan Mahasiswa Teknik Computer-Informatika (HMTC). Di HMTC penulis pernah menjabat sebagai staff departemen kesejahtraan masyarakat. Penulis juga mengikuti beberapa Latihan Keterampilan Manajemen Mahasiswa (LKMM) yaitu praTD dan TD. Penulis dapat dihubungi melalui *email*: dinabian1@gmail.com.