METODE *FUZZY-TWO STEP FILTER* UNTUK MEREDUKSI *IMPULSE NOISE* PADA CITRA BERTIPE RGB (*RED GREEN BLUE*) DAN APLIKASINYA PADA CITRA FOTOGRAFI

SKRIPSI

Diajukan Kepada Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Negeri Yogyakarta untuk Memenuhi Sebagian Persyaratan
Guna Memperoleh Gelar Sarjana Sains

Oleh:

ARIF MUNANDAR

10305141016

PROGRAM STUDI MATEMATIKA

JURUSAN PENDIDIKAN MATEMATIKA

FAKULTAS MATEMATIAK DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI YOGYAKARTA

2014

PERSETUJUAN

Skripsi yang berjudul "**METODE *FUZZY-TWO STEP FILTER* UNTUK MEREDUKSI *IMPULSE NOISE* PADA CITRA BERTIPE RGB (*RED GREEN BLUE*) DAN APLIKASINYA PADA CITRA FOTOGRAFI**" yang disusun oleh Arif Munandar, NIM 10305141016 ini telah disetujui oleh pembimbing untuk diujikan

Yogyakarta, 2 April 2014

Pembimbing

Dr. Agus Maman Abadi

NIP. 19700828 199502 1 001

PENGESAHAN

SKRIPSI DENGAN JUDUL

"METODE *FUZZY-TWO STEP FILTER* UNTUK MEREDUKSI *IMPULSE NOISE* PADA CITRA BERTIPE RGB (*RED GREEN BLUE*) DAN APLIKASINYA PADA CITRA FOTOGRAFI"

Yang Disusun Oleh

Nama : Arif Munandar
NIM : 10305141016
Prodi : Matematika

Skripsi ini telah dipertahankan di depan Dewan Penguji pada tanggal 8 April 2014 dan dinyatakan lulus

DEWAN PENGUJI

Nama	Jabatan	Tanda tangan	Tanggal
Dr. Agus Maman Abadi NIP. 19700828 199502 1 001	Ketua Penguji	.....	15-4-2014
Dwi Lestari, M.Sc. NIP. 19850513 201012 2 006	Sekertaris Penguji	.....	15-4-2014
Bambang S.H.M.M.Kom NIP. 19680210 198812 1 001	Penguji Utama	.....	14-4-2014
Musthofa, M.Sc. NIP. 19801107 200604 1 001	Penguji Pendamping	.....	15-4-2014

Yogyakarta, 17 April 2014
Fakultas Matematika dan Ilmu
Pengetahuan Alam
Dekan

Dr. Hartono
NIP. 19620329 198702 1 002

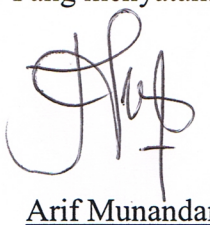
SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi ini benar-benar karya saya sendiri. Sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang ditulis atau diterbitkan orang lain kecuali sebagai acuan atau kutipan dengan mengikuti tata penulisan karya ilmiah yang telah lazim.

Tandan tangan dosen penguji yang tertera dalam halaman pengesahan adalah asli. Jika tidak asli, saya siap menerima saksi ditunda yudisium pada periode berikutnya.

Yogyakarta, 2 April 2014

Yang menyatakan



Arif Munandar

NIM 10305141016

MOTTO

Sesunguhnya bersama kesulitan ada kemudahan (Q.S., Asy-Syarh :6)

Apabila engkau telah selesai (dari suatu urusan) tetaplah bekerja keras (untuk urusan yang lain) (Q.S., Asy-Syarh: 7)

Karya kecil ini saya persembahkan untuk

Ayah, Ibu dan kakak-kakakku tercinta

Serta almamaterku

METODE *FUZZY-TWO STEP FILTER* UNTUK MEREDUKSI *IMPULSE NOISE* PADA CITRA BERTIPE RGB (*RED GREEN BLUE*) DAN APLIKASINYA PADA CITRA FOTOGRAFI

Oleh
Arif Munandar
10305141016

ABSTRAK

Fuzzy two step filter (FTSFC) adalah salah satu filter non linear yang menerapkan dasar-dasar teori *fuzzy*. Metode ini dikembangkan untuk mereduksi *impulse noise* pada citra bertipe RGB (*Red Green Blue*) dengan tetap mempertahankan tekstur dan detail dari citra. Skripsi ini ditujukan untuk menjelaskan prosedur atau tahapan-tahapan dalam metode FTSFC dan aplikasinya pada citra fotografi.

Metode ini terdiri dari dua tahapan, tahapan pertama adalah deteksi *noise* dan tahapan kedua adalah tahap filter. Tahap deteksi pada intinya didasarkan pada nilai gradien *fuzzy*. Hasil yang diperoleh pada tahap ini adalah *pixel-pixel* yang terdapat pada himpunan *fuzzy impulse noise* untuk masing-masing komponen warna. Hasil yang diperoleh pada tahap deteksi digunakan sebagai input untuk tahap filter. Tahap filter memanfaatkan perbedaan intensitas warna dari masing-masing komponen warna.

Metode FTSFC diaplikasikan pada citra RGB untuk mengetahui seberapa tingkat keakuratan dari metode tersebut. Berdasarkan pembahasan diperoleh bahwa metode ini efektif dalam mereduksi *impulse noise* pada citra fotografi bertipe RGB, terbukti dengan kecilnya nilai *Mean Square Error* (MSE) dan besarnya nilai *Peak Signal to Noise Ratio* (PSNR) yang dihasilkan. Metode ini juga lebih baik jika dibandingkan dengan metode filter linear, misalnya Median Filter. Kelemahan dari Metode FTSFC adalah *pixel-pixel* yang berada pada bagian tepi tidak dapat tereduksi *impulse noise*-nya.

Kata Kunci: *fuzzy two step filter*, *impulse noise*, reduksi *noise*, MSE, PSNR

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, yang telah melimpahkan segala karunia, rahmat dan hidayah-Nya sehingga tugas akhir skipsi ini dapat terselesaikan. Skripsi ini dapat terselesaikan dengan baik dan lancar berkat bantuan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ayah dan Ibu serta kakak-kakakku, yang telah memberikan begitu banyak curahan kasing sayangnya.
2. Bapak Dr. Agus Maman Abadi selaku Ketua Program Studi Matematika dan Dosen Pembimbing serta Penasehat Akademik.
3. Bapak Prof. Dr. Etienne Kerre dan Stefan Schulte, Ph.D. yang telah begitu banyak membantu dalam penyusunan skripsi ini.
4. Guru-guru penulis terutama Bapak Sukirman, M.Pd, yang telah banyak memberikan pelajaran hidup.
5. K.H. Rosim Al Fatih, Lc dan Ibu Hj. Anita Durrotul Yatimah Al Hafidzoh selaku pengasuh pesantren Al- Barakah dan K.H. Ya'qub Mashuri, M.Pd dan Ibu Hj. Mahmmudah selaku pengasuh pesantren Fajar Sa'adah yang telah memberikan banyak doa restunya.
6. Segenap santri putra dan putri pondok persantren Al-Barakah, khususnya santri diniyah kelas 'Ulya' dan santri komplek Al-Fatih, terimakasih atas kebersamaan dan rasa kekeluargaan yang kalian berikan.

7. Teman-teman seperjuanganku: Aziz Ali, Ikhfan, Yoga, Jaka, dan Udhi. Kabar keberhasilan kalian membuatku semangat untuk terus berkarya.
8. Teman-teman organisasi HIMATIKA 2011, BEM FMIPA 2012, dan BEM FMIPA 2013 yang telah memberikan begitu banyak pengalaman dan kenangan tak terlupakan.
9. Seluruh Mahasiswa Matematika UNY Angkatan 2010 serta semua pihak yang telah membantu secara langsung maupun tidak langsung.

Penulis menyadari bahwa dalam pembuatan skripsi ini masih banyak kekurangan. Oleh karena itu, kritik dan saran sangat diharapkan sebagai koreksi. Semoga skripsi ini bermanfaat bagi penulis dan bagi akademisi yang tertarik dengan "*fuzzy image processing*" khususnya bidang reduksi *noise* serta dapat menjadi referensi untuk adik angkatan yang akan menulis skripsi nantinya.

Yogyakarta, 2 April 2014

Penulis

Arif Munandar

NIM. 10305141016

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

DAFTAR SIMBOL

$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix}$	Bentuk umum Matriks Citra Digital berukuran $N \times M$
(i,j)	Vektor posisi pada citra berwarna (dua dimensi)
$f(i,j)$	Derajat keabuan pada posisi (i,j)
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \dots$	Matriks
(i,j,k)	Vektor posisi pada citra berwarna (tiga dimensi)
m	Banyaknya bit
Z	Indek dari map
O	Citra berwarna asli
$O(i,j,1), O(i,j,2), O(i,j,3)$	Komponen warna merah, hijau dan biru asli
p_k, p_k', p_k''	Peluang sebuah *pixel* terkorupsi *noise* pada masing-masing komponen (diskrit)
$\eta_{G(x,y)}$	*Noise* berdistribusi *Gaussian*
$g(x,y)$	Citra terkorupsi *Gaussian noise*
L	Derajat keabuan
h_i	Histogram pada derajat keabuan i
n	Banyaknya *pixel* pada citra
n_i	Banyaknya *pixel* dengan derajat keabuan sama
A	Himpunan

$\mu_A(x)$	Fungsi keanggotaan himpunan *fuzzy A*
$\widetilde{A}$	Himpunan *fuzzy A*
$\overline{A}$	Komplemen dari himpunan *fuzzy A* (standar)
$A \cap B$	Irisan himpunan *fuzzy* A dan B (standar)
$\mu_{A\cap B}(x)$	Definisi Operasi standar irisan 2 himpunan *fuzzy*
$A \cup B$	Gabungan himpunan *fuzzy* A dan B (standar)
$\mu_{A\cup B}(x)$	Definisi Operasi standar gabungan 2 himpunan *fuzzy*
c	Operasi komplemen pada himpunan *fuzzy*
c_λ	Sugeno Class (komplemen)
s	*s-norm*
$s_\lambda(a,b)$	Dombi class (*s-norm*)
$s_\alpha(a,b)$	Dubois-Prade class (*s-norm*)
$s_\omega(a,b)$	Yager Class (*s-norm*)
t	*t-norm*
$t_\lambda(a,b)$	Dombi class (*t-norm*)
$t_\alpha(a,b)$	Dubois-Prade class (*t-norm*)
$t_\omega(a,b)$	Yager Class (*t-norm*)
T_n	Himpunan nilai-nilai kebenaran
X	Nama variabel (Variabel lingustik)
$T(x)$	Himpunan istilah dari x
U	Domain dimana variabel linguistik didefinisikan

M	aturan semantik untuk menghubung-kan setiap nilai *x* dengan artinya
Org	Citra asli sebelum dikenai filter
Img	Citra hasil filter
N	Parameter ukuran persekitaran
$\nabla_{(k,l)}A_R(i,j)$	Nilai gradient
$D = \{BL, U, TL, T, TG, S, BD, B\}$	Himpunan delapan arah
$\nabla_D A_R(i,j), \nabla'_D A_R(i,j), \nabla''_D A_R(i,j)$	Nilai gradien pertama kedua dan ketiga untuk komponen merah dan arah *D*
$\nabla_D^F A_R(i,j)$	Nilai gradien *fuzzy* untuk komponen merah dan arah *D*
$\mu_{Impulse}^R$, $\mu_{Impulse}^G$ dan $\mu_{Impulse}^B$	Himpunan *fuzzy impulse noise* pada komponen merah, hijau dan biru
$\max_{deteksi}$	Nilai *pixel* pada citra yang paling besar
$HISTR(q)$	*Pixel* bernoise dengan nilai q
σ	Etimasi standar deviasi
$RG(i,j), GR(i,j), BG(i,j), GB(i,j), RB(i,j), BR(i,j)$	Matrik perbedaan intensitas warna
$F_R(i,j)$	Komponen warna merah setelah proses filter
$A_R(i,j), A_G(i,j), A_B(i,j)$	Nilai *pixel* pada posisi (i,j) untuk komponen merah hijau dan biru
$\Delta_{RB}(i,j)$	Estimasi intensitas perbedaan warna merah dan biru
$\Delta_{RG}(i,j)$	Estimasi intensitas perbedaan warna

	merah dan hijau
$\#_e^R$	Banyaknya *pixel* pada himpunan *fuzzy impulse noise*

BAB I

PENDAHULUAN

A. LATAR BELAKANG

Menurut Munir (2002 :2), Citra adalah gambar pada bidang dua dimensi (dwimantra). Dilihat dari sudut pandang matematis citra didefinisikan sebagai fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Proses diperolehnya citra ini dimulai dengan adanya sumber cahaya yang menerangi suatu objek, kemudian objek memantulkan kembali sebagian berkas cahaya tersebut. Pantulan cahaya ini kemudian ditangkap oleh alat-alat optik seperti mata manusia, kamera, *scanner*, dan alat optik lainnya, sehingga bayangan objek yang disebut sebagai citra tersebut dapat terekam.

Pepatah mengatakan bahwa citra dapat lebih bermakna dibanding seribu ungkapan. Hal ini mungkin saja benar dalam beberapa hal, sebagai contoh, misalkan ada seorang penjual rumah menawarkan rumahnya dengan mendeskripsikannya dalam kata-kata. Sebanyak apapun kata yang disampaikan, pembeli hanya dapat membayangkan seperti apa rumah yang ditawarkannya tanpa mengerti sepenuhnya tentang kondisi nyata dari rumah tersebut. Namun jika penjual rumah membawa foto dari rumah yang akan dijualnya, maka pembeli akan mengerti sepenuhnya tentang kondisi dari rumah tersebut. Hal ini mengindikasikan bahwa citra kaya akan informasi.

Meskipun citra kaya akan informasi, namun seringkali citra mengalami penurunan kualitas, seperti munculnya *noise* pada citra, warnanya terlalu kontras,

warnanya kurang tajam, warnanya kabur (*blurring*) dan masalah yang lainnya. Hal ini tentunya dapat menyebabkan kesulitan dalam menginterpretasi citra sehingga informasi yang akan disampaikan oleh citra tersebut menjadi berkurang.

Skripsi ini akan fokus untuk mengatasi masalah yang terjadi pada citra yang disebabkan oleh munculnya *noise* pada citra. Lebih khususnya terhadap salah satu jenis *noise* yaitu *impulse noise*. Jenis *noise* ini dapat ditemukan pada citra hasil fotografi ataupun sinar–X. *Noise* jenis ini dapat disebabkan oleh proses pengambilan citra, pengkompresian citra, transmisi atau pemindahan citra, penyimpanan dan proses lainnya (Gonzalez, 2002). Karakteristik utama dari *noise* ini adalah munculnya bintik-bintik berwarna hitam atau putih pada citra. Karakteristik ini yang akan digunakan sebagai dasar untuk menentukan *pixel-pixel* yang terkorupsi oleh *impulse noise*.

Banyak penelitian sebelumnya yang sudah membahas mengenai reduksi *impulse noise* yang terjadi pada citra digital khususnya citra jenis RGB. Berikut ini adalah metode-metode yang sudah dikemukakan oleh peneliti terdahulu yang sudah memakai dasar-dasar teori *fuzzy*;

1. *The fuzzy inference rule by else-action filters* (FIRE) disajikan dalam jurnal "*Removal of impulse noise using* FIRE *filter*" ditulis oleh F. Russo (1996),
2. *Fuzzy filter based on median filter* (FMF), disajikan dalam jurnal "*Median filter based on fuzzy rule and its application to image restoration*" ditulis oleh K Arakawa (1996) yang kemudian juga ditulis dalam jurnal "*Review*

of impulse noise reduction technique using fuzzy logic for image processing" ditulis oleh Kaur, J. (2012),

3. *Fuzzy filter based on average filter* (FAF), disajikan dalam jurnal "*Weighted fuzzy mean filter for image processing*" ditulis oleh C.S Lee dan Y.H Kuo (1997),
4. *The fuzzy based similarity*, ditulis dalam jurnal "*Real-time image noise cancellationbased on fuzzy similarity*", oleh I. Kalaykov dkk (2003),
5. *The adaptive fuzzy switching filter* (AFSF), ditulis dalam jurnal berjudul "*Adaptive fuzzy switching filter for image corrupted by impulse noise*" oleh H. Xu, dkk (2004),

dan masih banyak metode lain yang disampaikan dalam jurnal yang berbeda, baik yang sudah memakai dasar-dasar teori *fuzzy* maupun yang belum.

Berdasarkan hasil analisis yang diberikan oleh Schulte, Stefan, dkk dalam jurnalnya "*fuzzy two-step filter impulse noise redction from color image*", metode *fuzzy two-step filter* (FTSFC) memberikan hasil yang paling memuaskan jika dibandingkan dengan semua metode-metode yang disampaikan di atas. Oleh karenanya penulis mencoba untuk menulis ulang dan menjabarkan jurnal tersebut dalam bahasa penulis sendiri dan menerapkannya pada citra fotografi. Sehingga penulis memberi judul skripsi ini dengan "Metode *fuzzy two-step fiter* untuk mereduksi *impulse noise* pada citra bertipe RGB (Red Green Blue) dan aplikasinya pada citra fotografi". Citra fotografi dipilih sebagai media untuk mengaplikasikan metode ini karena *noise* tipe ini dapat terjadi pada citra-citra fotografi.

Kualitas metode tersebut dalam mereduksi *noise* pada citra diukur dengan dua besaran, yaitu MSE (*Mean Square Error*) dan PSNR (*Peak Signal to Noise Ratio*). MSE (*Mean Square Error*) menyatakan tingkat kesalahan kuadrat rata-rata dari *output* yang dihasilkan terhadap *input*. Semakin kecil nilai MSE menunjukkan semakin sesuainya *output* dengan *input*. Parameter PSNR bernilai sebaliknya, semakin besar parameter PSNR semakin sesuai *output* dengan *input*.

B. Batasan Masalah

Batasan masalah yang digunakan dalam penulisan skripsi ini adalah penggunaan filter non linear metode *fuzzy two-step filter* pada citra bertipe RGB dengan tingkat *impulse noise* yang beragam. Hasil dari metode akan diuji tingkat keakuratannya dengan menggunakan MSE (*Mean Square Error*) dan PSNR (*Peak Signal to Noise Ratio*). Adapun citra yang akan digunakan sebagai sampel untuk aplikasi metode ini adalah citra fotografi. Citra fotografi yang dimaksud dalam skripsi ini adalah citra hasil kamera digital yang bertipe RGB. Lebih spesifiknya adalah citra berupa foto yang dihasilkan oleh fotografer profesional. Adapun impulse noise yang terdapat pada citra dihasilkan dengan menggunakan program MATLAB.

C. Rumusan Masalah

Permasalahan yang akan dibahas dalam skripsi ini adalah

1. Bagaimana menjelaskan prosedur penggunaan metode *fuzzy two-step filter* untuk mereduksi *impulse noise* pada citra bertipe RGB?

2. Berapa tingkat keakuratan metode *fuzzy two-step filter* dalam mereduksi citra fotografi jenis RGB yang terkena *impulse noise*?

D. Tujuan

Tujuan penulisan skripsi ini adalah

1. Menjelaskan prosedur penggunaan metode *fuzzy two-step filter* untuk mereduksi *impulse noise* pada citra bertipe RGB.
2. Menganalisis keakuratan metode jika diterapkan pada citra fotografi jenis RGB dengan menggunakan MSE dan PSNR.

E. Manfaat

1. Bagi mahasiswa, mampu memahami secara mendalam mengenai metode *fuzzy two step filter* dan dapat mengaplikasikannya pada citra fotografi.
2. Bagi jurusan, mampu memberikan referensi yang bermanfaat tentang metode *fuzzy two-step filter*.
3. Bagi pembaca, mampu memberikan bacaan yang bermanfaat tentang metode *fuzzy two-step filter* untuk mereduksi *impulse noise* pada citra digital bertipe RGB.

BAB II

KAJIAN TEORI

Pada bab ini akan dibahas teori-teori yang mendukung dalam penulisan tugas akhir ini. Teori-teori tersebut adalah teori reduksi *noise* pada citra, teori himpunan *fuzzy*, logika *fuzzy*, *Mean Square Error* (MSE), dan *Peak Signal to Noise Ratio* (PSNR).

A. REDUKSI NOISE PADA CITRA DIGITAL

1. Definisi Citra

Menurut Munir (2002: 2) dari sudut pandang matematis citra adalah fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimantra (dua dimensi). Secara matematis fungsi intensitas cahaya pada bidang dua dimensi disimbolkan dengan $f(x, y)$, yang dalam hal ini (x, y) adalah koordinat pada bidang dua dimensi dan $f(x, y)$ adalah intensitas cahaya (*brightness*) pada titik (x, y).

Cahaya merupakan bentuk energi, dan energi tak berhingga maka intensitas cahaya bernilai dari 0 sampai tidak berhingga,

$$0 \leq f(x, y) \leq \infty$$

Nilai $f(x, y)$ ini sebenarnya adalah hasil kali dari $i(x, y)$ yang merupakan jumlah cahaya yang berasal dari sumbernya (*illumination*), nilainya dari 0 sampai tidak berhingga, dan *r*(*x*, *y*) adalah derajat kemampuan obyek memantulkan cahaya (*reflection*), nilainya dari 0 sampai 1. (Munir, 2002: 16).

Nilai *i*(*x*, *y*) ditentukan oleh sumber cahaya, sedangkan *r*(*x*, *y*) ditentukan oleh karakteristik objek di dalam gambar. Nilai *r*(*x*,*y*) = 0 mengindikasikan penerapan total, sedangkan *r*(*x*,*y*) = 1 menyatakan pemantulan total.

2. **Representasi Citra**

Agar dapat diolah dengan komputer digital, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dari fungsi kontinu menjadi nilai-nilai diskrit disebut digitalisasi. Pada umumnya citra digital berbentuk persegi panjang, dan dimensi ukurannya dinyatakan sebagai tinggi x lebar (atau lebar x panjang).

Citra digital yang berukuran $N \times M$ lazim dinyatakan dengan matriks yang berukuran N baris dan M kolom sebagai berikut:

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

Indeks baris (i) dan indeks kolom (j) menyatakan suatu koordinat titik pada citra, sedangkan $f(i,j)$ merupakan intensitas (derajat keabuan) pada titik (i,j) (Gonzalez, 2002: 55). Bentuk matriks dari persamaan (2.1) di atas dapat dituliskan sebagai berikut:

$$\boldsymbol{A} = \begin{bmatrix} a_{0.0} & a_{0.1} & \cdots & a_{0.M-1} \\ a_{1.0} & a_{1.1} & \dots & a_{1.M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1.0} & a_{N-1.1} & \cdots & a_{N-1.M-1} \end{bmatrix} \quad (2.2)$$

dengan $a_{i,j} = f(i,j)$, sehingga matriks pada persamaan (2.1) sama dengan matriks pada persamaan (2.2) (Gonzalez, 2002: 55).

Definisi 2.1 *Pixel* (Munir, 2002: 19)

Pixel, *image element*, atau *pixture element* didefinisikan sebagai elemen-elemen pada matriks citra digital.

Berdasarkan matriks pada persamaan (2.1) dan (2.2) maka yang dimaksud *pixel* adalah elemen-elemen dari matriks pada persamaan tersebut.

3. Jenis-jenis Citra Digital

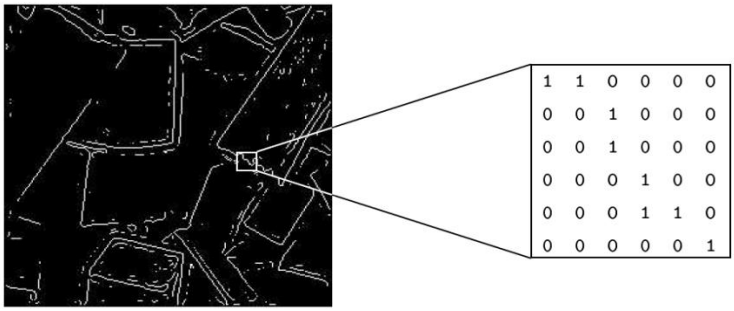
Ada empat tipe dasar dari citra (McAndraw, 2004: 12) yaitu:

a. Biner

Masing-masing *pixel* pada jenis citra ini hanya berupa unsur hitam dan putih. Karena hanya ada dua nilai yang mungkin untuk masing-masing *pixel* maka hanya dibutuhkan satu bit untuk satu *pixel*. Citra jenis ini sangat efisien dalam hal sedikitnya memori yang dibutuhkan untuk menyimpan citra. Citra yang direpresentasikan secara biner ini cocok untuk sidik jari atau rancangan arsitek.

Contoh 2.1

Citra dengan tipe biner direpresentasikan dalam Gambar 2.1, dalam citra tersebut hanya terdapat dua jenis warna yaitu hitam direpresentasikan dengan angka 0 sebagai *background* dan putih sebagai tepi citra yang direpresentasikan dengan angka 1.



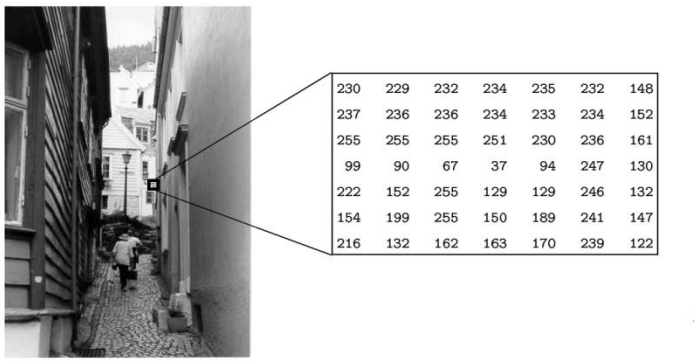
Gambar 2.1 Citra Bertipe Biner (McAndraw, 2004: 13)

b. Grayscale

Masing-masing *pixel* pada citra jenis *grayscale* mengambarkan derajat keabuan, secara normal dari 0 (warna hitam) sampai 255 (warna putih). *Range* nilai ini menunjukkan bahwa masing-masing *pixel* dapat direpresentasikan oleh 8-bit atau tepatnya 1 byte. Secara alami ini adalah range yang cocok untuk banyak jenis gambar. Contoh dari citra bertipe *grayscale* adalah hasil citra yang dihasilkan oleh sinar –X.

Contoh 2.2

Contoh dari citra bertipe *grayscale* direpresentasikan dalam Gambar 2.2, dalam citra dengan tipe ini maksimal terdapat 256 derajat keabuan atau tingkatan warna.

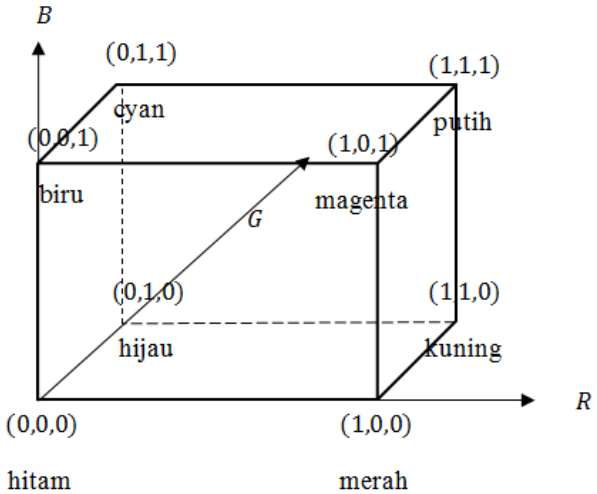


Gambar 2.2 Citra Bertipe *Grayscale* (McAndraw, 2004: 14)

c. RGB

Citra digital berwarna dimodelkan dalam ruang warna RGB. Ruang warna RGB diperoleh dengan mencampurkan tiga jenis warna yaitu merah, hijau dan biru. Berdasarkan hal ini maka citra dengan jenis warna RGB dimodelkan dalam vektor 3 dimensi dengan merah sebagai vektor pertama, hijau sebagai vektor kedua dan biru sebagai vektor ke tiga, dalam hal ini setiap posisi pada citra digital berwarna dapat dinyatakan dalam (i, j, k), dengan i untuk vektor warna merah, j adalah vektor warna hijau dan k adalah vektor warna biru. Masing-masing nilai vektor berada pada rentang 0 sampai $2^m - 1$, dengan m adalah banyaknya bit, dalam citra jenis RGB yang umum digunakan adalah $m = 8$.

Model standar untuk warna RGB adalah dengan menggunakan kubus warna dengan range 0 sampai 1. Model ini direpresentasikan dengan gambar berikut



Gambar 2.3 Kubus Warna untuk Model Warna Tipe RGB (McAndraw, 2004:20)

Contoh 2.3

Citra bertipe RGB direpresentasikan dalam Gambar 2.3, dalam citra dengan tipe ini terdapat 3 komponen warna yaitu komponen warna merah, hijau, dan biru dengan setiap komponen maksimal terdapat 256 tingkatan warna. Tiga matriks yang ditampilkan pada gambar tersebut adalah nilai *pixel* untuk gambar tersebut pada masing-masing komponen warna yaitu komponen warna merah, hijau dan biru. Dengan demikian warna jenis ini mempunyai variasi kombinasi warna sebanyak 256^3.

Gambar 2.4 Citra Bertipe RGB (McAndraw, 2004:15)

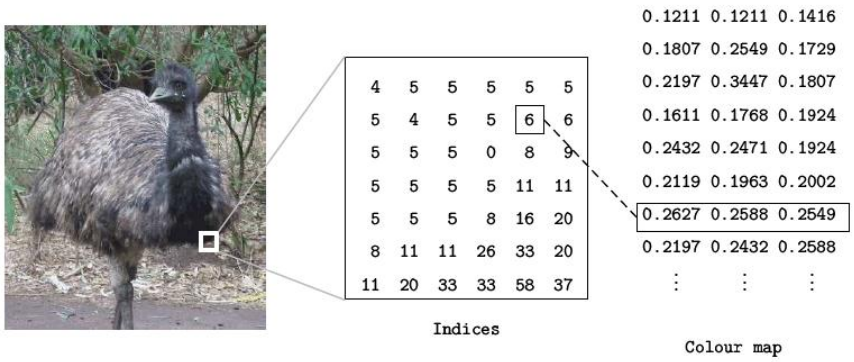
d. Index

Kebanyakan citra warna hanya memiliki sebagian kecil dari 16 juta warna yang mungkin. Untuk kenyamanan dalam menyimpanan berkas

file, citra warna bertipe index mempunyai sebuah peta warna yang terkait indeks warna, yang hanya menyimpan daftar semua warna yang digunakan pada citra tersebut. Setiap piksel pada citra warna berindeks mempunyai nilai yang tidak mewakili warna yang diberikan (seperti pada citra warna RGB), tetapi nilai tersebut hanya mewakili sebuah indeks warna ,yang mana representasi warna tersebut tersimpan pada peta warna. Berikut contoh sebuah citra warna berindeks.

Contoh 2.4

Di bawah ini adalah contoh citra yang bertipe index.



Gambar 2.5 Citra Bertipe Index (McAndraw, 2004:16)

4. *Impulse Noise* pada Citra RGB

Noise didefinisikan sebagai gangguan yang terjadi pada citra. *Noise* pada citra tidak hanya terjadi karena ketidak-sempurnaan dalam proses pengambilan gambar, tetapi dapat juga disebabkan oleh kotoran-kotoran yang terjadi pada citra. Berdasarkan bentuk dan karakteristiknya, *noise* pada citra dibedakan menjadi 4 jenis yaitu; *Gaussian Noise, speckle noise, impulse noise (salt & pepper noise)* dan *periodic noise* (McAndrew, 2004: 111). Dari

sumber yang lain *noise* pada citra dibedakan menjadi 3 yaitu; *Gaussian noise*, *speckle noise* dan *impulse noise (salt & pepper noise)* (Haris: 47). Penjelasan selanjutnya akan lebih dititik beratkan pada pengertian dari *impulse noise* dan juga *Gaussian noise* yang terjadi pada citra berwarna jenis RGB.

Selain dengan menggunakan model standar yang berupa vektor 3 dimensi, warna jenis RGB dapat pula dimodelkan dalam bentuk vektor 2 dimensi. Berikut adalah definisi dari model *impulse noise* dalam vektor 2 dimensi,

Definisi 2.2 Model *Impulse Noise* dalam Vektor 2 dimensi (Schulte, 2006: 1)

Terjadinya *impulse noise* pada citra dimodelkan sebagai berikut;

$$\textit{Pixel yang bernoise}=\begin{cases}[p_k \;\; O(i,j,2) \;\; O(i,j,3)] & \text{atau}\\ [O(i,j,1) \;\; p'_k \;\; O(i,j,3)] & \text{atau}\\ [O(i,j,1)\; O(i,j,2)\; p''_k\,] & \text{atau}\\ [p_k\, p'_k\, O(i,j,3)\,] & \text{atau}\\ [p_k \;\; O(i,j,2)\; p''_k] & \text{atau}\\ [O(i,j,1) \;\; p_k\, p''_k] & \text{atau}\\ [p_k\, p'_k\, p''_k] & \text{atau}\end{cases} \tag{2.3}$$

dengan $O(i,j,1), O(i,j,2)$, dan $O(i,j,3)$ berturut-turut adalah *pixel* komponen warna merah, hijua dan biru yang tidak terkorupsi oleh *impulse noise*, $k \in \{1,2,\dots,n\}$ dengan $n \leq 2^m - 1$, dan m adalah banyaknya bit pada citra.

Dalam persamaan di atas notasi p_k adalah *pixel* pada komponen merah dengan posisi (i,j) yang mengalami *noise,* notasi p'_k adalah *pixel* pada komponen hijau dengan posisi (i,j) yang mengalami *noise* dan notasi p''_k adalah *pixel* pada komponen biru dengan posisi (i,j) yang mengalami *noise*.

Berdasarkan persamaan (2.3), baris pertama dari persamaan tersebut menyatakan bahwa *pixel* pada posisi (i, j) dalam komponen merah terkorupsi *impulse noise*, baris kedua menyatakan bahwa komponen hijau terkorupsi *impulse noise*, dan seterusnya hingga baris terakhir dari persamaan tersebut yang menyatakan bahwa semua komponen mengalami *impulse noise*.

Akibat dari *impulse noise*, nilai *pixel* yang terkorupsi *noise* sangat berbeda dari nilai *pixel* di sekelilingnya. Hal ini mengakibatkan bintik-bintik warna yang sangat mencolok dan muncul pada citra. Ini adalah karakteristik utama dari *impulse noise*, dan karakteristik inilah yang akan banyak digunakan sebagai dasar untuk mereduksi *impulse noise*.

Skripsi ini juga membahas tentang citra yang terkorupsi oleh *noise* campuran antara *Gaussian noise* dan *impulse noise*. *Gaussian noise* adalah salah satu jenis *additive noise*, berikut adalah definisi dari *Gaussian noise*.

Definsi 2.3 *Gaussian Noise* (Krishnan, 2013: 2)

Gaussian *noise* didefinisikan dengan persamaan berikut

$$g(x, y) = f(x, y) + \eta_{G(x,y)} \tag{2.4}$$

dengan $g(x, y)$ adalah citra yang terkorupsi *Gaussian noise*, $f(x, y)$ adalah citra asli dan $\eta_{G(x,y)}$ adalah *noise* berdistribusi *Gaussian.*

Contoh 2.5

Berikut ini adalah contoh citra bertipe RGB, citra RGB yang terkorupsi oleh *impulse noise* sebesar 10% dan citra yang terkorupsi oleh *Gaussian noise* dengan *mean* 0.01 dan variansi 0.05.

(a) (b) (c)

Gambar 2.6 (a) Citra "*global fisheris crisis*.jpg", (b) Citra "*global fisheris crisis*.jpg" yang Terkorupsi oleh *Impulse Noise* Sebesar 10%, (c) Citra "global fisheris crisis.jpg" yang Terkorupsi oleh *Gaussian Noise* dengan *Mean* 0.01 dan *Variance* 0.05 (Citra diakses dari http://national-geographic-wallpapers.en.lo4d.com/details)

5. Reduksi *noise*

Reduksi *noise* adalah proses untuk menghilangkan *noise* yang terjadi pada citra. Secara umum metode untuk mereduksi *noise* dapat dilakukan dengan cara melakukan operasi pada *pixel* citra digital dengan menggunakan suatu jendela ketetanggaan (*neighborhood window*). Proses penerapan jendela ketetanggan tersebut sering disebut sebagai proses *filtering*.

Menurut Koli (2012: 1) dalam jurnalnya "*review of impulse noise reduction techniques",* reduksi *noise* dapat dibagi menjadi dua tipe yaitu reduksi *noise* dengan filter linear dan reduksi *noise* dengan filter non linear. Reduksi *noise* dengan filter linear menerapkan filter pada semua *pixel* dalam citra tanpa mengklasifikasikan *pixel* yang terkorupsi *noise* ataupun yang tidak. Hal ini tentunya dapat menyebabkan *pixel* yang tidak terkorupsi *noise* menjadi terkorupsi *noise* setelah proses reduksi *noise* berlangsung. Filter non linear menggunakan dua tahap, tahap pertama adalah proses deteksi terhadap *pixel-pixel* yang terkena *noise*, dan tahap kedua adalah proses filtering terhadap *pixel-pixel* yang sudah terdeteksi pada tahap pertama.

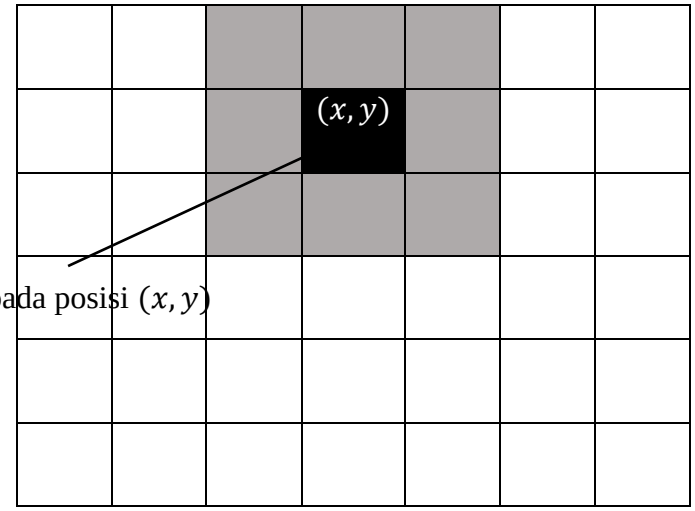
Terdapat banyak jenis filter linear dan filter non linear yang sudah ditemukan. Lebih lanjut Koli (2012: 3) mengemukakan beberapa jenis filter linear yang sudah dikenal, diantaranya *average filter, median filter,* dan *mean filter*. Sementara jenis filter non linear antara lain *min-max median filter, center weighted median filter, adaptive median filter, progressive switching median filter, tri-state median filter, decision based median filter.* Sementara filter yang sudah menggunakan dasar-dasar *fuzzy* yang dikenal diantaranya adalah *Arakawa's Fuzzy Median Filter* (Kaur, 2012: 4) dan metode yang akan dibahas yaitu *fuzzy two step filter* yang disampaikan oleh Schulte (2006: 1-12) dalam jurnalnya *fuzzy two step filter to reduction impulse noise in color image*.

6. *Neighborhood Processing*

Citra dapat dimanipulasi dengan merubah *pixel* dari citra. Proses manipulasi dapat dilakukan dengan menggunakan fungsi- fungsi tertentu yang diterapkan pada masing-masing *pixel*. *Neighborhood processing* adalah salah satu hal yang dapat diterapkan untuk melakukan manipulasi tersebut, proses ini dilakukan dengan membuat persekitaran pada masing-masing *pixel* dan kemudian fungsi tertentu diaplikasikan pada setiap persekitaran tersebut.

Idenya adalah dengan membuat semacam persekitaran berbentuk segi empat (biasanya persegi dengan ukuran ganjil) pada masing-masing *pixel*. Kombinasi dari persekitaran dan fungsi yang dibentuk pada persekitaran tersebut kemudian disebut sebagai filter. Jika fungsi yang dibentuk adalah fungsi yang linear maka filternya disebut sebagai filter linear.

Filter linear dapat diperoleh dengan mengalikan, menambah, mengurangi ataupun membagi setiap elemen pada persekitaran dengan nilai tertentu. Misalkan terdapat persekitaran dengan ukuran 3x3 yang diilustrasikan pada Gambar 2.7 yaitu pada bagian yang berwarna abu-abu muda, maka filter dapat dibuat dengan mengoperasikan elemen-elemen yang terdapat pada persekitaran tersebut (McAndraw, 2004: 60).



Gambar 2.7.Persekitaran dengan Ukuran 3x3

Salah satu filter linear yang banyak digunakan adalah *average filter* dengan menggunakan jendela ketetanggaan berukuran 3×3. Berikut adalah definisi dari *average filter*.

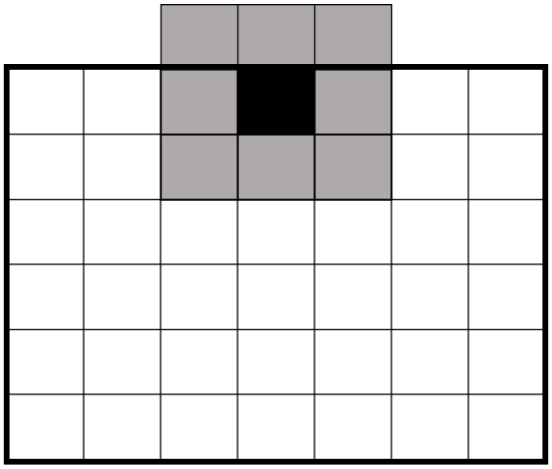
Definisi 2.4 *Average filter* (Mcandrew, 2004: 50)

Average filter dengan menggunakan persekitaran 3×3 didefinisikan dengan persamaan berikut;

$$f'(x,y) = \frac{1}{9}\big(f(x-1,y-1) + f(x-1,y) + f(x-1,y+1) + f(x,y-1) + f(x,y) + f(x,y+1) + f(x+1,y-1) + f(x+1,y) + f(x+1,y+1)\big) \quad (2.5)$$

dengan $f(x,y)'$ adalah *pixel* pusat baru untuk $f(x,y)$, dan $f(x-i,y-j)$, $i,j \in \{-1,0,1\}$ adalah *pixel* - *pixel* yang berada pada persekitaran *pixel* pusat.

Jendela ketetanggaan memungkinkan *pixel* – *pixel* yang berada pada tepi tidak dapat dibuat persekitaran. Hal ini diilustrasikan sebagai berikut:



Gambar 2.8 Persekitaran dari *Pixel* yang Berada pada Tepi

Gambar di atas adalah ilustrasi dari jendela ketetanggaan (persekitaran) yang berukuran 3×3 dan terletak pada tepi citra. McAndraw (2004:62) menyampaikan bahwa kasus di atas dapat diatasi dengan dua solusi yaitu;

1. Abaikan *pixel* yang berada pada tepi

   Solusi pertama adalah dengan mengabaikan *pixel-pixel* yang berada pada bagian tepi, dengan demikian proses filter hanya dilakukan untuk *pixel-pixel* yang berada secara penuh dalam jangkauan persekitaran yang dibuat. Berdasarkan hal ini maka metode ini akan menghasilkan ukuran citra yang lebih kecil dibandingkan dengan ukuran aslinya. Akibatnya semakin besar persekitaran yang dibuat maka akan semakin banyak *pixel* tepi yang

hilang, sehingga beberapa informasi mungkin akan ikut hilang. Dengan demikian jika metode ini diterapakan maka ada baiknya jika persekitaran yang dibuat berukuran kecil.

Contoh 2.6

Misalkan *average filter* dengan ukuran jendela ketetanggaan 3×3 akan diterapkan pada matriks berikut;

```
>> x=uint8(10*magic(5))
x =
  170  240   10   80  150
  230   50   70  140  160
   40   60  130  200  220
  100  120  190  210   30
  110  180  250   20   90
```

Proses filter pertama diterapkan pada matriks baris pertama sampai ketiga dan kolom pertama sampai ketiga yaitu

```
170  240   10   80   150
230   50   70  140   160
 40   60  130  200  220
100  120  190  210   30
110  180  250   20   90
```

Sehingga *pixel* yang bernilai 50 akan diganti dengan rata-rata dari sembilan *pixel* tersebut yaitu

$$\frac{1}{9}(170 + 240 + 10 + 230 + 50 + 70 + 40 + 60 + 130) = 111{,}1111$$

Proses dilanjutkan dengan mengeser jendela ketetanggan kesebelah kanan, sehingga akan *pixel* 70 akan diganti dengan 108.8889, dan seterusnya sehingga diperoleh hasil akhirya adalah

111.1111 108.8889 128.8889

110.0000 130.0000 150.0000

131.1111 151.1111 148.8889

2. Menambahkan *pixel* nol diluar citra

Solusi kedua dilakukan dengan cara mengasumsikan nilai pada *pixel* diluar citra adalah nol. Dengan metode ini maka ukuran citra akan tetap seperti semula, namun metode ini memungkinkan perubahan pada *pixel* yang berada pada tepi.

Contoh 2.7

Misalkan matriks **X** pada contoh sebelumnya akan diproses dengan menggunakan *average filter* dengan menambahkan nol diluar gambar, maka matriks **X** akan menjadi

0	0	0	0	0	0	0
0	170	240	10	80	150	0
0	230	50	70	140	160	0
0	40	60	130	200	220	0
0	100	120	190	210	30	0
0	110	180	250	20	90	0
0	0	0	0	0	0	0

Dengan menggunakan metode yang sama akan diperoleh hasilnya sebagai berikut

76.6667 85.5556 65.5556 67.7778 58.8889

87.7778 111.1111 108.8889 128.8889 105.5556

66.6667 110.0000 130.0000 150.0000 106.6667

67.7778 131.1111 151.1111 148.8889 85.5556

56.6667 105.5556 107.7778 87.7778 38.8889

7. Histogram Citra

Informasi penting mengenai kandungan yang terdapat pada citra digital dapat diketahui dengan membuat histogram citra. Berikut adalah definisi dari histogram citra.

Defensi 2.5 Histogram citra (Munir, 2002: 83)

Histogram citra adalah grafik yang menggambarkan penyebaran nilai-nilai intensitas *pixel* dari suatu citra atau bagian tertentu di dalam citra. Secara matematis dirumuskan sebagai berikut

$$h_i = \frac{n_i}{n}, i = 1,2,3 \dots, L-1 \quad (2.6)$$

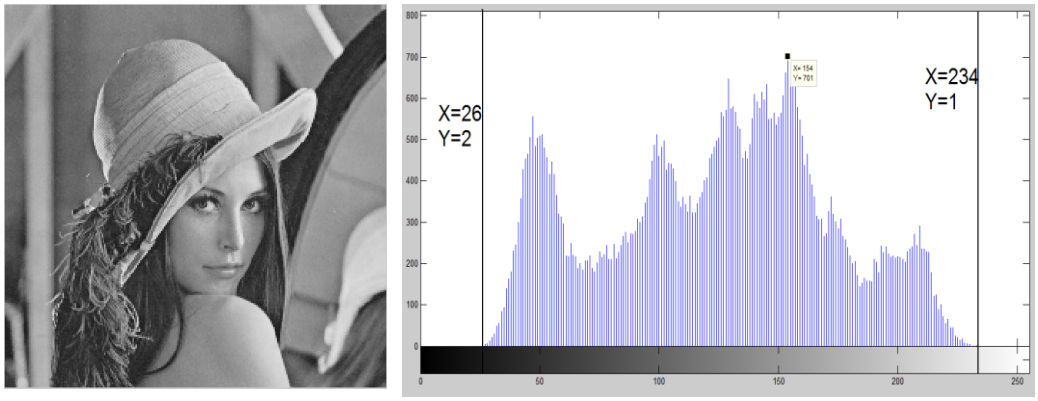
Dengan L adalah derajat keabuan, n_i adalah banyaknya *pixel* pada citra yang mempunyai derajat keabuan sama dan n adalah banyaknya *pixel* dalam citra.

Dari sebuah histogram dapat diketahui frekuensi kemunculan nisbi (*relative*) dari intensitas pada citra tersebut. Histogram juga dapat menunjukkan banyak hal tentang kecerahan (*brightness*) dan kekontasan dari sebuah gambar. Karena itu, menurut Munir (2002: 83), histogram adalah alat bantu yang berharga dalam pekerjaan pengolahan citra baik secara kualitatif maupun kuantitatif.

Cara lain untuk merumuskan histogram citra adalah dengan mendata semua *pixel-pixel* pada citra yang mempunyai derajat keabuan yang sama tanpa dibagi dengan banyaknya *pixel* yang ada pada citra tersebut. Metode ini yang digunakan dalam MATLAB.

Contoh 2.8

Berikut adalah citra bertipe *grayscale* dan histogram dari citra tersebut dengan menggunakan bantuan MATLAB.



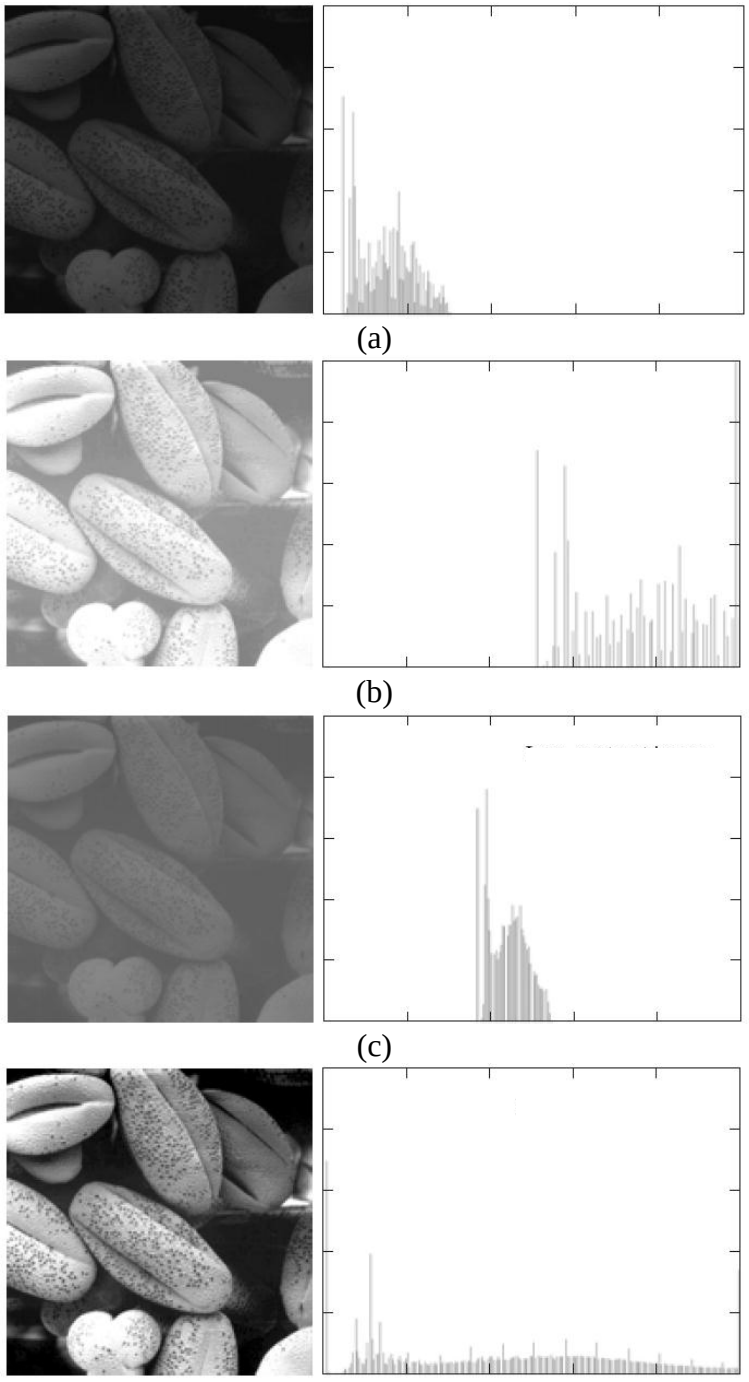
(a) (b)

Gambar 2.9.(a). Citra Digital "lena_gray_256.tif" (b). Histogram Citranya (Citra Diakses dari http://www.imageprocessingplace.com/DIP- 3E/dip3e book images_downloads.htm)

Berdasarkan histogram citra tersebut maka dapat disimpulkan bahwa citra tidak mempunyai derajat keabuan lebih dari 234 dan kurang dari 26, dan derajat keabuan yang paling sering muncul adalah 154 dengan frekuensi kemunculannya adalah 701.

Menurut Gonzalez (2002: 89), berdasarkan tingkat kecerahan dan kekontrasan citra dapat dibagi menjadi 4 tipe yaitu, citra bertipe gelap, citra terang, citra dengan kekontrasan warna rendah, dan citra dengan kekontrasan

warna tinggi. Jenis- jenis citra ini dan histogramnya diilustrasikan dalam Gambar 2.10 di bawah ini.



(a)

(b)

(c)

(d)

Gambar 2.10 (a). Citra Gelap, (b). Citra Terang, (c). Citra dengan Kekontrasan Warna Rendah, (d). Citra dengan Kekotrasan Warna Tinggi (Gonzalez, R. 2002: 90)

B. Teori Himpunan *Fuzzy*

Saat ini dikenal kajian mengenai teori peluang dan teori himpunan *fuzzy*. Kedua kajian ilmu ini mempunyai kemiripan, yaitu sama-sama membahas mengenai ketidakpastian. Meskipun begitu teori himpunan *fuzzy* dan teori peluang tidak dapat disamakan.

Salah satu yang dapat mendeskripsikan perbedaan teori peluang dan teori himpunan *fuzzy* adalah bahwa teori peluang berhubungan dengan ekspektasi tentang sesuatu hal pada masa yang akan datang dengan didasarkan pada pengetahuan pada masa sekarang dan ekspektasi itu akan berakhir (menjadi kepastian) setelah hal tersebut terjadi (Klir, 1997; 10). Sebagai contoh, misalkan kita tertarik dengan peluang bahwa seseorang yang akan masuk kelas adalah orang yang tinggi dengan didasarkan tinggi rata-rata masyarakat secara keseluruhan di Indonesia. Jika kelas yang dimaksudkan adalah kelas yang terdiri dari anak-anak pemain basket maka ekspektasinya orang yang akan masuk adalah orang yang tinggi akan besar. Sebaiknya jika kelas tersebut hanyalah kelas biasa, misalnya kelas matematika maka ekspektasi kita akan menurun dari ekspektasi sebelumya dan setelah orang tersebut masuk kelas, maka ketidakpastian tentang tinggi orang tersebut menjadi berakhir.

Ketidakpastian dalam teori himpunan *fuzzy* adalah ketidakpastian akan suatu hal yang belum terjadi mengenai kata atau istilah yang tidak dapat didefinisikan secara tegas, namun ketidakpastian tersebut tetap ada meskipun hal itu telah terjadi. Contoh ketidakpastian dalam teori himpunan *fuzzy* misalnya ketika seseorang merasa tidak pasti apakah cuaca esok hari dingin atau tidak.

Ketidakpastian cuaca esok hari tersebut akan berubah menjadi kepastian setelah esok hari tiba, namun dinginnya cuaca adalah hal yang sampai kapanpun akan tetap mengalami ketidakpastian.

Berikut akan dijelaskan mengenai perbedaan himpunan klasik dan himpunan *fuzzy*. Secara umum perbedaan kedua himpunan tersebut terletak dalam hal pengambilan keputusan, dalam logika klasik hanya terdapat 2 kriteria pengambilan keputusan yaitu "ya" dan "tidak", sementara dalam logika *fuzzy* pengambilan keputusan dinyatakan dalam derajat keanggotaan pada fungsi keanggotaan himpunan *fuzzy*.

1. Himpunan Klasik

Definisi 2.6 Himpunan klasik (Klir, 1997: 48)

Himpunan klasik atau himpunan tegas adalah himpunan yang keberadaan suatu elemen dalam himpunan tersebut hanya mempunyai dua kemungkinan, yaitu berada pada himpunan tersebut (menjadi anggota dari himpunan tersebut) atau tidak berada pada himpunan tersebut (tidak menjadi anggota dari himpunan tersebut).

Nilai yang menunjukkan besarnya tingkat keanggotaan dari suatu elemen dalam himpunan disebut sebagai derajat keanggotaan. Sehingga dalam logika klasik, derajat keanggotaan untuk himpunan A dinotasikan oleh Sri Kusuma Dewi, dkk (2006) sebagai berikut;

$$\mu_A(x) = \begin{cases} 1 & ,\text{untuk } x \in A \\ 0 & ,\text{untuk } x \notin A \end{cases} \tag{2.7}$$

Contoh 2.9

Variabel warna citra berdasarkan besarnya nilai *pixel* yang merepresentasikan citra dengan semesta pembicaraan $S = [0\ 255]$, dibagi menjadi 3 himpunan berikut;

GELAP : Jika *pixel* kurang dari 70,

SEDANG : jika nilai *pixel* $70 \leq x \leq 160$,

TERANG : jika *pixel* lebih dari 160.

Berdasarkan 3 himpunan di atas, dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Nilai *pixel* 69 menjadi anggota dari himpunan warna citra GELAP,
2. Nilai *pixel* 70 menjadi anggota dari himpunan warna citra SEDANG,

Contoh tersebut menyatakan bahwa variabel warna citra berdasarkan besarnya nilai *pixel* masih kurang bijaksana atau kurang baik sebab, adanya perubahan kecil pada suatu nilai dapat mengakibatkan perubahan dan perbedaan yang besar pada status nilai tersebut pada suatu himpunan.

2. Himpunan *Fuzzy*

Himpunan *fuzzy* adalah suatu himpunan universal atau semesta yang dapat dinyatakan dengan fungsi keanggotaan, atau dengan kata lain himpunan yang jika anggotanya tidak berada di dalam himpunan lain selain himpunan semesta A, maka tetap merupakan anggota himpunan tersebut meskipun dengan nilai keanggotaan 0.

Definisi 2.7 Himpunan *fuzzy* **(**Zimmermann, 1991)

Jika X adalah koleksi dari obyek-obyek yang dinotasikan secara genetik oleh x, maka suatu himpunan *fuzzy* $\tilde{A}$, dalam X adalah suatu himpunan pasangan berurutan:

$$\tilde{A} = \{(x, \mu_A(x)) | \, x \in X\} \tag{2.8}$$

dengan μ_A adalah fungsi keanggotaan dari himpunan *fuzzy* A, yang merupakan suatu pemetaan dari himpunan semesta X ke rentang $[0,1]$.

Contoh 2.10

Misalkan X adalah himpunan semesta warna citra dalam bentuk *grayscale* yaitu dalam rentang $[0, 255]$. Himpunan *fuzzy* warna citra "GELAP", "SEDANG" dan "TERANG" didefinisikan sebagai berikut

- $\mu_{GELAP}(x) = \begin{cases} 1 - \frac{|x|}{128} & , \text{untuk } 0 \leq x \leq 128 \\ 0 & , \text{untuk } x \text{ yang lain} \end{cases}$
- $\mu_{SEDANG}(x) = 1 - \frac{|x-128|}{128}$, untuk $0 \leq x \leq 255$
- $\mu_{TERANG}(x) = \begin{cases} 1 - \frac{|x-255|}{128} & , \text{untuk } 128 \leq x \leq 255 \\ 0 & , \text{untuk } x \text{ yang lain} \end{cases}$

Sri Kusumadewi (2006), menotasikan himpunan fuzzy dalam beberapa cara diantaranya;

a. Himpunan *fuzzy* dinotasikan sebagai pasangan berurutan, dengan elemen pertama menunjukan nama elemen dan urutan kedua menunjukan nilai keanggotaannya.
b. Himpunan *fuzzy* jika semesta pembicaraanya diskrit dinotasikan sebagai:

$$\tilde{A} = \mu_{\tilde{A}}(x_1)/x_1 + \mu_{\tilde{A}}(x_2)/x_2 + ... + \mu_{\tilde{A}}(x_n)/x_n = \sum_{i=1}^{n} \mu_{\tilde{A}}(x_i)/x_i \quad (2.9)$$

dan jika semesta pembicaraannya kontinu dituliskan sebagai

$$\tilde{A} = \int_x \mu_{\tilde{A}}(x)/x \quad (2.10)$$

Contoh 2.11

Jika himpunan semesta tingkatan warna citra berdasarkan *pixel X* diskrit, misal $X = \{0,15,30,45,60,75,90,105,120,135,150,165,180,195,210,$ $225,240,255\}$, maka himpunan *fuzzy* tingkatan warna TERANG pada Contoh 2.11 dapat dituliskan sebagai berikut:

$$\tilde{A} = \frac{0}{0} + \frac{0{,}108}{15} + \frac{0{,}217}{30} + \frac{0{,}326}{45} + \frac{0{,}434}{60} + \frac{0{,}543}{75} + \frac{0{,}652}{90} + \frac{0{,}760}{105} + \frac{0{,}864}{120} + \frac{0{,}978}{135} + \frac{0}{150} + \frac{0}{165} + \frac{0}{180} + \frac{0}{195} + \frac{0}{210} + \frac{0}{225} + \frac{0}{240} + \frac{0}{225}$$

Jika semesta pembicaraan kontinu yaitu $X = [0,255]$, maka:

$$\tilde{A} = \int_0^{128} (1 - \frac{|x|}{128})/x + \int_{128}^{255} 0/x$$

Semesta pembicaraan pada variabel *fuzzy* dan domain pada himpunan *fuzzy* merupakan himpunan bilangan real yang senantiasa naik atau bertambah secara monoton dari kiri ke kanan. Nilai semesta pembicaraan atau domain dapat berupa bilangan positif atau negatif dan ada kalanya tidak dibatasi batas atasnya (Sri Kusumadewi dkk, 2006).

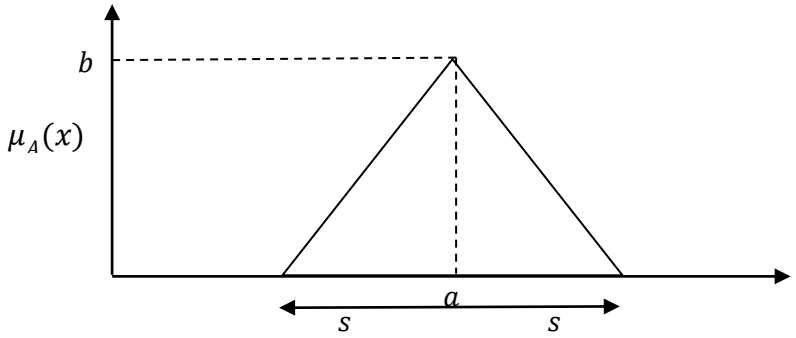
3. Fungsi Keanggotaan Himpunan *Fuzzy*

Fungsi keanggotaan adalah suatu kurva yang menunjukan pemetaan titik-titik input data ke dalam nilai keanggotaannya. Cara yang digunakan untuk mendapatkan nilai keanggotaan adalah dengan mensubstitusikan nilai numerik dari himpunan *fuzzy* yang bersangkutan ke dalam rumus fungsi keanggotaannya.

Menurut Klir (1997: 76), fungsi keanggotaan dapat direpresentasikan dengan berbagai cara diantaranya adalah representasi berupa grafik, representasi berupa tabulasi dan list, representasi secara geometri, dan representasi secara analitik. Representasi yang paling umum dan banyak dipakai dalam sistem yang dibuat berdasarkan logika *fuzzy* adalah representasi secara analitik.

Representasi analitik dapat berupa fungsi keanggotaan segitiga (*triangular membership function*), fungsi keanggotaan trapesium (*trapezoidal membership function*), fungsi keanggotaan Gaussian (*gaussian membership function*), dan fungsi-fungsi keanggotaan lainya. Berikut ini adalah penjelasan dari fungsi keanggotaan tersebut.

a. Fungsi keanggotaan segitiga (*triangular membership function)* direpresentasikan dalam grafik berikut

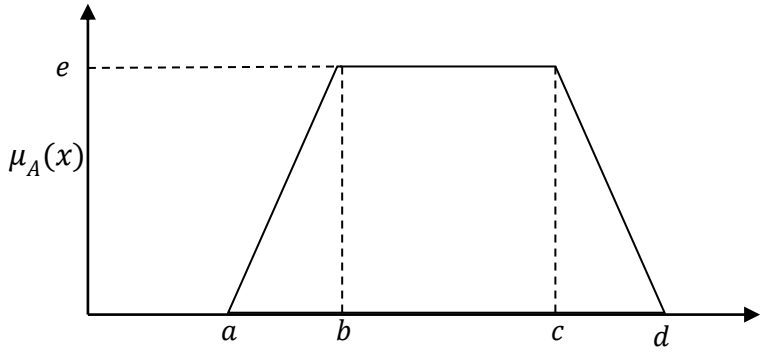


Gambar 2.11 Grafik Fungsi Keanggotaan Segitiga

Fungsi keanggotaan di atas dirumuskan dengan persamaan berikut (Klir,1997:85)

$$\mu_A(x) = \begin{cases} b\left(1 - \left(\frac{|x-a|}{s}\right)\right) & , \text{untuk } a - s \leq x \leq a + s \\ 0 & , \text{untuk } x \text{ yang lain} \end{cases} \quad (2.11)$$

b. Fungsi keanggotaan trapesium (*trapezoidal membership function*) direpresentasikan dalam grafik berikut

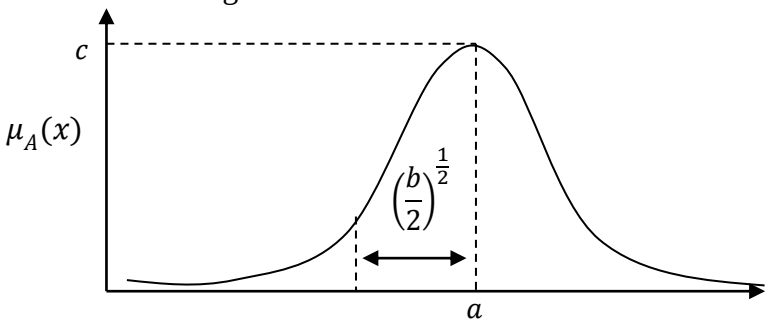


Gambar 2.12 Grafik Fungsi Keanggotaan Trapesium

Fungsi keanggotaan tersebut didefinisikan dengan persamaan berikut (Klir, 1997:86)

$$\mu_A(x) = \begin{cases} \frac{(a-x)e}{a-b} & , \text{untuk } a \leq x \leq b \\ e & , \text{untuk } b \leq x \leq c \\ \frac{(d-x)e}{d-c} & , \text{untuk } c \leq x \leq d \\ 0 & , \text{untuk } x \text{ lain} \end{cases} \quad (2.12)$$

c. Fungsi keanggotaan Gaussian (*Gaussian membership function*) direpresentasikan dalam grafik berikut

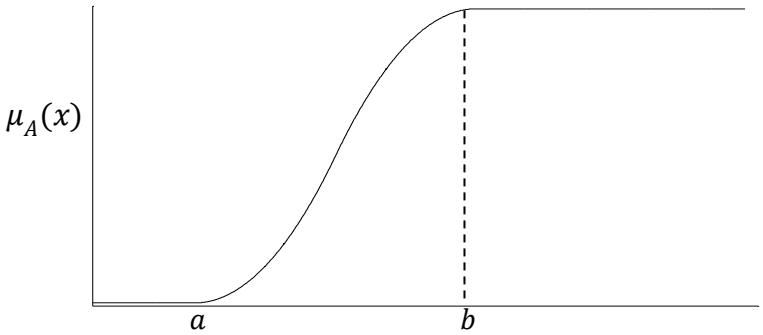


Gambar 2.13 Grafik Fungsi Keanggotaan Gaussian

Fungsi keanggotaan tersebut didefinisikan dengan persamaan (Klir,1997:86)

$$\mu_A(x) = ce^{-\left(\frac{(x-a)^2}{b}\right)} \tag{2.13}$$

d. Fungsi keanggoaan S (*S membership function*) direpresentasikan dalam grafik berikut

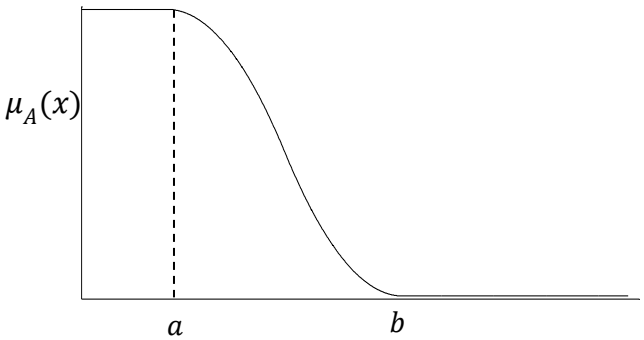


Gambar 2.14 Grafik Fungsi Keanggotaan *S*

Fungsi keanggotaan tersebut didefinisikan dengan persamaan

$$\mu_A(x) = \begin{cases} 0 & ,\forall x < a \\ 2\left(\frac{x-a}{b-a}\right)^2 & ,\forall x \in \left[a, \frac{a+b}{2}\right] \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2 & ,\forall x \in \left[\frac{a+b}{2}, b\right] \\ 1 & ,\forall x > b \end{cases} \tag{2.14}$$

e. Fungsi keanggoaan Z (*Z membership function*) direpresentasikan dalam grafik di bawah ini



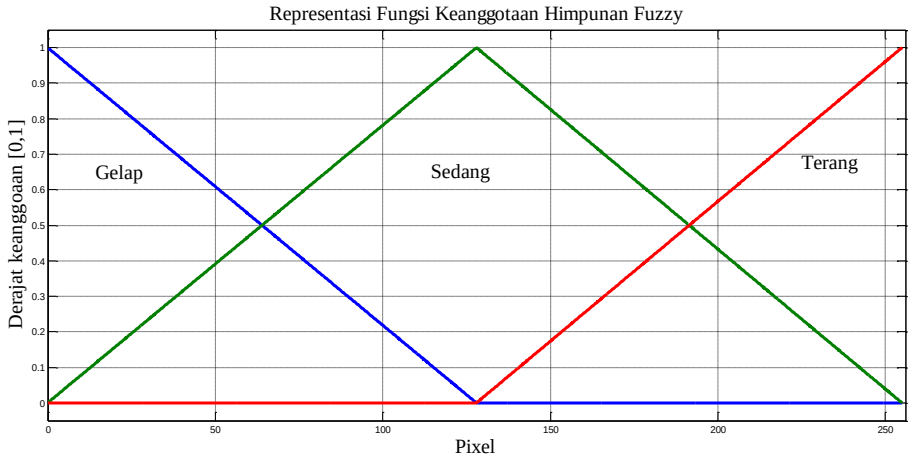
Gambar 2.15 Grafik Fungsi Keanggotaan *Z*

Fungsi keanggotaan tersebut didefinisikan dengan persamaan

$$\mu_A(x) = \begin{cases} 1 & ,\forall x < a \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2 & ,\forall x \in \left[a, \frac{a+b}{2}\right] \\ 2\left(\frac{x-d}{d-c}\right)^2 & ,\forall x \in \left[\frac{a+b}{2}, b\right] \\ 0 & ,\forall x > b \end{cases} \quad (2.15)$$

Contoh 2.12

Fungsi keanggotaan pada contoh 2.11 didefinisikan dalam fungsi keanggotaan segitiga. Representasi dari fungsi keanggotaan tersebut adalah sebagai berikut:



Gambar 2.16 Grafik Fungsi Keanggotaan Segitiga untuk Tingkatan Warna Berdasarkan *Pixel*

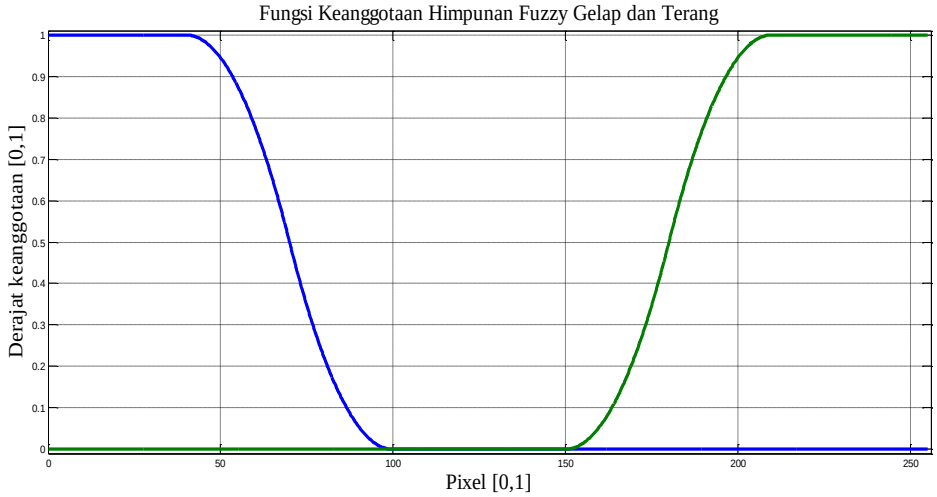
Contoh 2.13

Fungsi keanggotaan himpunan *fuzzy* warna citra "TERANG" dan "GELAP" dapat direpresentasikan dengan menggunakan fungsi keanggotaan $S[40,100]$ dan fungsi Keanggotaan $Z[150,210]$. Fungsi keanggotaan tersebut dirumuskan sebagai berikut

$$\mu_{GELAP} = \begin{cases} 1 & ,\forall x < 40 \\ 1 - 2\left(\frac{x-40}{60}\right)^2 & ,\forall x \in [40,70] \\ 2\left(\frac{x-100}{60}\right)^2 & ,\forall x \in [70,100] \\ 0 & ,\forall x > 100 \end{cases} \quad (2.16)$$

$$\mu_{TERANG} = \begin{cases} 0 & ,\forall x < 150 \\ 2\left(\frac{x-a}{b-a}\right)^2 & ,\forall x \in [150,180] \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2 & ,\forall x \in [180,210] \\ 1 & ,\forall x > 210 \end{cases} \quad (2.17)$$

Representasi dari fungsi keanggotaan pada persamaan (2.18, 2.19) adalah sebagai berikut;



Gambar 2.17 Grafik Fungsi Keanggotaan S dan Z Tingkatan Warna Berdasarkan *Pixel*

4. Operasi pada Himpunan *Fuzzy*

Tiga operasi dasar dalam himpunan *fuzzy* adalah operasi komplemen, irisan dan gabungan. Operasi standar dari ketiga operasi dasar pada himpunan *fuzzy* tersebut didefinisikan sebagai berikut;

Definisi 2.8 Operasi standar komplemen (Klir, 1997: 34)

Operasi standar komplemen untuk himpunan *fuzzy* A disimbolkan dengan $\bar{A}$ dan didefinisikan sebagai

$$\mu_{\bar{A}}(x) = 1 - \mu_A, \forall x \in X \tag{2.18}$$

Definisi 2.9 Operasi standar irisan (Klir, 1997: 34)

Operasi standar irisan dari himpunan *fuzzy* A dan himpunan *fuzzy* B disimbolkan dengan $A \cap B$ dengan definisi sebagai

$$\mu_{A\cap B}(x) = \min[\mu_A, \mu_B], \forall x \in X \tag{2.19}$$

Definisi 2.10 Operasi standar gabungan (Klir, 1997: 34)

Operasi standar gabungan dari himpunan *fuzzy* A dan himpunan *fuzzy* B disimbolkan dengan $A \cup B$ dengan definisi sebagai

$$\mu_{A\cup B}(x) = \max[\mu_A, \mu_B], \forall x \in X \tag{2.20}$$

min dan max adalah operasi minimum dan maksimum. Lebih lanjut pembahasasn tentang operasi pada himpunan *fuzzy* disampaikan di bawah ini.

a. **Operasi Komplemen pada Himpunan *Fuzzy***

Misalkan $c: [0,1] \rightarrow [0,1]$ adalah pemetaan dari fungsi keanggotaan dari himpunan *fuzzy* A ke fungsi keanggotaan dari komplemen dari A, dan didefinisikan sebagai

$$c[\mu_A(x)] = \mu_{\bar{A}}(x) \tag{2.21}$$

Berdasarkan operasi dasar komplemen maka $\mu_{\bar{A}}(x) = 1 - \mu_A$, sehingga $c[\mu_A(x)] = 1 - \mu_A(x)$. Fungsi c disebut sebagai operasi komplemen pada himpunan *fuzzy* jika memenuhi dua aksioma berikut (Klir, 1995:52):

1. Aksioma c_1 yaitu $c(0) = 1$ dan $c(1) = 0$
2. Aksioma c_2 yaitu untuk setiap $a, b \in [0,1]$, jika $a < b$, maka $c(a) \geq c(b)$ dengan a, b adalah fungsi keanggotaan dari himpunan *fuzzy* misalkan $a = \mu_A(x)$ dan $b = \mu_B(x)$.

Aksioma pertama menunjukkan bahwa jika sebuah elemen mempunyai derajat keanggotaan pada himpunan *fuzzy* adalah nol (satu) maka komplemen dari himpunan *fuzzy* tersebut adalah satu (nol). Aksioma kedua menyatakan bahwa naiknya fungsi keanggotaan dari suatu elemen akan menyebabkan menurun atau tetapnya komplemen dari elemen tersebut.

Definisi 2.11 Komplemen *fuzzy* (Wang, 1997: 35)

Sebarang fungsi $c: [0,1] \rightarrow [0,1]$ yang memenuhi aksioma c_1 dan c_2 disebut sebagai komplemen *fuzzy*.

Contoh 2.14

Salah satu jenis dari komplemen *fuzzy* adalah Sugeno *Class* (Sugeno [1977]), yang didefinisikan sebagai

$$c_\lambda = \frac{1-a}{1+\lambda a} \tag{2.22}$$

dengan $\lambda \in (-1, \infty)$. Fungsi tersebut merupakan salah satu komplemen *fuzzy* sebab memenuhi aksioma c_1 dan c_2. Hal ini ditunjukkan sebagai berikut:

1. $c_\lambda(0) = \frac{1-0}{1+\lambda 0} = \frac{1}{1} = 1$ dan $c_\lambda(1) = \frac{1-1}{1+\lambda} = \frac{0}{1+\lambda} = 0$,
2. Ambil sebarang $a, b \in [0,1]$ dengan $a < b$ maka $c_\lambda(a) = \frac{1-a}{1+\lambda a} \geq \frac{1-b}{1+\lambda b} = c_\lambda(b)$.

b. **Operasi Gabungan pada *Fuzzy* (*s-norm*)**

Misalkan $s: [0,1] \times [0,1] \rightarrow [0,1]$ adalah pemetaan dari fungsi keanggotaan himpunan *fuzzy* A dan B ke fungsi keanggotaan himpunan *fuzzy* gabungan A dan B yaitu

$$s[\mu_A(x), \mu_B(x)] = \mu_{A \cup B}(x) \tag{2.23}$$

Maka berdasarkan operasi dasar irisan dua himpunan *fuzzy* $\mu_{A \cup B}(x) = \max[\mu_A(x), \mu_B(x)]$, maka $[\mu_A(x), \mu_B(x)] = \max[\mu_A(x), \mu_B(x)]$.

Fungsi s memenuhi kriteria gabungan pada himpunan *fuzzy* jika memenuhi aksioma-aksioma berikut (Wang, 1997: 37):

1. Aksioma s_1 yaitu $s(1,1) = 1$ dan $s(0, a) = s(a, 0) = a$ (sifat ketertutupan),
2. Aksioma s_2 yaitu $s(a, b) = s(b, a)$ (sifat komutatif),
3. Aksioma s_3 yaitu Jika $a \leq a'$ dan $b \leq b'$, maka $s(a, b) \leq s(a', b')$ (sifat tidak turun),
4. Aksioma s_4 yaitu $s(s(a, b)c) = s\big(a, s(b, c)\big)$ (sifat asosiatif)

Aksioma pertama menunjukan bahwa fungsi gabungan pada *fuzzy* adalah bersifat tertutup. Aksioma kedua menenjukkan bahwa urutan tidak

berpengaruh pada hasil dari fungsi $s-norm$. Aksioma ketiga menunjukkan bahwa semakin besar nilai keanggotaan dari dua himpunan *fuzzy* akan meningkatkan nilai keanggotaan dari gabungan dua himpunan *fuzzy* tersebut. Aksioma ke empat menunjukkan bahwa operasi gabungan dapat diterapkan pada lebih dari dua himpuan *fuzzy*.

Definisi 2.12 $s-norm$ (Wang, 1997:37).

Sebarang fungsi s: $[0,1]\times[0,1]\rightarrow[0,1]$ yang memenuhi aksioma s_1-s_4 disebut sebagai $s-norm$.

Contoh 2.15

Operasi standar gabungan yaitu max adalah salah satu contoh dari $s-norm$ sebab memenuhi aksioma s_1-s_4, hal ini ditunjukkan sebagai berikut;

1. $\max(1,1)=1$ dan $\max(0,a)=\max(a,0)=a$,
2. $\max(a,b)=\max(b,a)$,
3. Jika $a\leq a'$ dan $b\leq b'$ maka $\max(a,b)\leq\max(a',b')$,
4. $\max(\max(a,b),c)=\max(a,\max(b,c))$.

Operasi penjumlahan biasa juga memenuhi aksioma s_1-s_4, operasi penjumlahan dan max adalah dua bentuk $s-norm$ yang paling umum dipakai, beberapa contoh dari $s-norm$ yang lain adalah (Wang, 1997:37):

1. Dombi Class (Dombi[1982])

$$s_\lambda(a,b)=\frac{1}{1+\left[\left(\frac{1}{a}-1\right)^{-\lambda}+\left(\frac{1}{b}-1\right)^{-\lambda}\right]^{-\left(\frac{1}{\lambda}\right)}} \tag{2.24}$$

Dengan $\lambda\in(0,\infty)$.

2. Dubois-Prade class (Dubois dan Prade [1980])

$$s_\alpha(a,b) = \frac{a+b-ab-\min(a,b,1-\alpha)}{\max(1-a,1-b,\alpha)} \quad (2.25)$$

Dengan parameter $\alpha \in (0,\infty)$.

3. Yager Class (Yager [1980])

$$s_\omega(a,b) = \min\left[1,\left(a^\omega + b^\omega\right)^{\frac{1}{\omega}}\right] \quad (2.26)$$

dengan parameter ù $\in (0,\infty)$.

Banyaknya fungsi $s-norm$ yang dibentuk dikarenakan fungsi $s-norm$ yang berbeda akan menghasilkan *output* sistem *fuzzy* yang berbeda pula. Aplikasi-aplikasi tertentu terkadang membutuhkan fungsi $s-norm$ yang khusus sehingga hasil dari aplikasi dapat maksimal.

c. Operasi Irisan pada *Fuzzy* (*t-norm*)

Misalkan $t: [0,1] \times [0,1] \rightarrow [0,1]$ adalah pemetaan dari fungsi keanggotaan himpunan *fuzzy A* dan fungsi keanggotaan himpunan *fuzzy B* ke fungsi keanggotaan irisan himpunan *fuzzy A* dan B, dengan definisi

$$t[\mu_A(x), \mu_B(x)] = \mu_{A\cap B}(x) \quad (2.27)$$

Maka berdasarkan operasi dasar irisan dua himpunan *fuzzy* $\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)]$, $t[\mu_A(x), \mu_B(x)] = \min[\mu_A(x), \mu_B(x)]$.

Fungsi t memenuhi kriteria irisan pada himpunan *fuzzy* jika memenuhi aksioma-aksioma berikut (Wang, 1997:41):

1. Aksioma t_1 yaitu $t(1,1) = 1$ dan $t(1,a) = t(a,1) = a$ (sifat ketertutupan),

2. Aksioma t_2 yaitu $t(a,b) = t(b,a)$ (sifat komutatif),
3. Aksioma t_3 yaitu Jika $a \leq a'$ dan $b \leq b'$, maka $t(a,b) \leq t(a',b')$ (sifat tidak turun),di bawah
4. Aksioma t_4 yaitu $(t(a,b)c) = t\big(a,t(b,c)\big)$ (sifat asosiatif).

Penjelasan mengenai aksioma–aksioma yang menjadi syarat fungsi $t-norm$ sama dengan penjelasan aksioma yang yang menjadi syarat fungsi $s-norm$.

Definisi 2.13 $t-norm$ (Wang, 1997:41)

Sebarang fungsi $t: [0,1] \times [0,1] \rightarrow [0,1]$ yang memenuhi aksioma $t_1 - t_4$ disebut sebagai $t-norm$.

Contoh 2.16

Operasi standar irisan min adalah salah satu contoh dari $t-norm$ sebab memenuhi akisoma $t_1 - t_4$, hal ini ditunjukkan sebgai berikut;

1. $\min(1,1) = 1$ dan $\min(1,a) = \min(a,1) = a$,

2. $\min(a,b) = \min(b,a)$,

3. Jika $a \leq a'$ dan $b \leq b'$ maka $\min(a,b) \leq \min(a',b')$,

4. $\min(\min(a,b),c) = \min(a,\min(b,c))$.

Hal yang sama juga berlaku untuk operasi perkalian biasa. Fungsi min dan perkalian biasa adalah salah satu $t-norm$ yang paling umum dipakai, beberapa contoh dari $t-norm$ yang lain adalah (Wang, 1997:41):

1. Dombi Class (Dombi[1982])

$$t_\lambda(a,b) = \frac{1}{1+\left[\left(\frac{1}{a}-1\right)^{-\lambda}+\left(\frac{1}{b}-1\right)^{-\lambda}\right]^{-\left(\frac{1}{\lambda}\right)}} \qquad (2.28)$$

Dengan $\lambda \in (0, \infty)$.

2. Dubois-Prade class (Dubois dan Prade [1980])

$$t_\alpha(a, b) = \frac{ab}{\max(a,b,\alpha)} \tag{2.29}$$

Dengan parameter $\alpha \in (0, \infty)$.

3. Yager Class (Yager [1980])

$$t_\omega(a, b) = 1 - \min\left[1, \left((1-a)^\omega + (1-b)^\omega\right)^{\frac{1}{\omega}}\right] \tag{2.30}$$

dengan parameter $\omega \in (0, \infty)$.

$t - norm$ memiliki variasi bentuk yang bermacam-macam, hal ini dikarenakan masing-masing bentuk $t - norm$ akan memberikan efek yang berbeda pada aplikasi *fuzzy* yang berbeda.

5. Logika *Fuzzy*

Logika adalah ilmu yang mempelajari secara sistematis aturan-aturan penalaran yang absah (*valid*) (Frans Susilo, 2006). Logika yang biasa dipakai dalam kehidupan sehari-hari maupun dalam penalaran ilmiah adalah logika dwi nilai, yaitu logika yang setiap pernyataan mempunyai dua kemungkinan nilai, yaitu benar atau salah. Asumsi dasar dalam logika dwi nilai, yakni bahwa setiap proporsi hanya mempunyai dua nilai kebenaran tersebut. Filosof Yunani kuno Aristoteles, mempermasalahkan pernyataan-pernyataan yang menyangkut masa depan, misalkan pernyataan: "minggu depan ia akan datang." Pernyataan semacam ini tidak memiliki nilai benar, dan tidak pernah salah, karena peristiwa yang diungkapkan oleh pernyataan semacam itu tidak tentu, sampai yang diungkapkannya tersebut terjadi (atau tidak terjadi).

Peryataan semacam ini kemudian diatasi oleh penemuan logikawan Polandia Jan Lukasiewicsz pada tahun 1920-an yang mengembangkan logika tri nilai dengan memasukan nilai nilai kebenaran ketiga, yaitu nilai tak tertentu. Logika ini bukanlah sistem logika yang baru, melainkan merupakan semacam pengembangan dari logika dwi nilai, dalam arti bahwa semua kata perangkai dalam logika tri nilai itu didefinisikan seperti dalam logika dwi nilai sejauh menyangkut nilai kebenaran. Tentu saja salah satu akibatnya tidak semua aturan logika yang berlaku dalam logika dwi nilai berlaku dalam logika Lukasiewicsz itu.

Logika tri nilai secara umum menghasilkan logika n-nilai yang juga dipelopori oleh Lukasiewicsz pada tahun 1930-an. Nilai logika dalam logika ini dinyatakan dengan suatu bilangan rasional dalam selang [0,1] yang diperoleh dengan membagi sama besar selang tersebut menjadi $n-1$ bagian. Maka himpunan T_n nilai-nilai kebenaran dalam logika n-nilai adalah himpunan n buah bilangan rasional sebagai berikut:

$$T_n = \{0 = \frac{0}{n-1}, \frac{1}{n-1}, \frac{2}{n-1}, \dots, \frac{n-1}{n-1} = 1\} \qquad (2.31)$$

Nilai kebenaran tersebut juga dapat dipandang sebagai derajat kebenaran suatu pernyataan, dapat dikatakan bahwa logika dwi nilai merupakan kejadian khusus dari logika n-nilai, yaitu untuk $n = 2$

Berikut adalah penjelasan mengenai variabel linguistik dan variabel numerik,

1. Variabel Numerik

Variabel numerik adalah variabel yang nilainya berupa nilai atau angka. Sebagai contoh "kecepatan kendaraan 45 km/jam", dengan "kecepatan" adalah variabel numerik dari pernyataan tersebut.

2. Variabel Linguistik

Variabel linguistik didefinisikan oleh (Zadeh, 1973 & 1975) dalam (Wang, 1997: 60). Sebagai berikut;

Definisi 2.14 Variabel linguistik (Sri Kusumadewi, dkk. 2006)

Variabel linguistik direpresentasikan dengan karakteristik (*x*,*T*(*x*),*U*,*M*) dengan:

- *x* adalah nama variabel, contohnya x adalah derajat keabuan dari suatu citra.
- *T*(*x*) merupakan himpunan istilah dari *x*, yakni himpunan semua nama dari nilai linguistik *x* yang mana setiap nilainya didefinisikan sebagai variabel *fuzzy* di *U*. Contohnya $T(x) = \{Terang,\ Sedang,\ Gelap\}$.
- *U* adalah domain dimana variabel linguistik didefinisikan, contohnya $U = [0{,}255]$, dengan 255 adalah nilai maksimum dari derajat keabuan.
- *M* merupakan aturan semantik untuk menghubungkan setiap nilai *x* dengan artinya. Contohnya M berkorelasi dengan "Terang", "Sedang" dan "Gelap", dengan fungsi keanggotaan pada Gambar 2.16.

6. Proposisi *Fuzzy*

Terdapat dua jenis proposisi *fuzzy* yaitu proposisi *fuzzy atomic*, dan proposisi *fuzzy compound*. Proposisi *fuzzy atomic* adalah pernyataan tunggal yaitu

$x\ adalah\ A$, dengan x adalah variabel linguistik dan A adalah nilai linguistik dari x. Proposisi *fuzzy compound* adalah komposisi dari proposisi *fuzzy atomic* dengan menggunakan kata hubung "dan", "$atau$" atau "$tidak$" yang merepresentasikan operasi irisan, operasi gabungan dan operasi komplemen pada himpunan *fuzzy*. Sebagai contoh misalkan x merepresentasikan warna pada citra dalam contoh 2.1, maka pernyataan-pernyataan berikut adalah proposisi *fuzzy*, yaitu

$$x\ adalah\ T \tag{2.32}$$

$$x\ adalah\ S \tag{2.33}$$

$$x\ adalah\ G \tag{2.34}$$

$$x\ adalah\ T\ atau\ x\ tidak\ S \tag{2.35}$$

$$x\ tidak\ T\ dan\ x\ tidak\ G \tag{2.36}$$

$$(x\ adalah\ T\ dan\ x\ tidak\ G)\ atau\ x\ adalah\ S \tag{2.37}$$

T, S, dan G secara berurutan adalah himpunan *fuzzy* TERANG, SEDANG dan GELAP. Tiga proposisi *fuzzy* yang pertama adalah proposi *fuzzy atomic* sementara sisanya adalah proposisi *fuzzy compound.*

Proposisi *fuzzy compound* diartikan sebagai relasi *fuzzy*. Cara untuk menentukan fungsi keanggotaan dari relasi *fuzzy* yang demikian adalah sebagai berikut (Wang, 1997:63)

1. Operasi irisan pada himpunan *fuzzy* digunakan untuk kata sambung "*dan*" . Sebagai contoh misalkan x dan y adalah variabel linguistik dari domain U dan V, dan A dan B secara berturut-turut adalah himpunan *fuzzy* pada U dan V, maka proposisi *fuzzy compound* $x\ adalah\ A\ dan\ y\ adalah\ B$

diintrepretasikan sebagai relasi *fuzzy* pada $A \cap B$ pada $U \times V$ dengan fungsi keanggotaan

$$\mu_{A\cap B}(x,y) = t[\mu_A(x), \mu_B(y)] \tag{2.38}$$

dengan t: $[0,1] \times [0,1] \rightarrow [0,1]$ adalah $t - norm$.

2. Operasi gabungan pada himpunan *fuzzy* digunakan untuk kata sambung "*atau*". Proposisi *fuzzy compound* $x\ adalah\ A\ atau\ y\ adalah\ B$ diintrepretasikan sebagai relasi *fuzzy* pada $A \cup B$ pada $U \times V$ dengan fungsi keanggotaan

$$\mu_{A\cup B}(x,y) = s[\mu_A(x), \mu_B(y)] \tag{2.39}$$

dengan s: $[0,1] \times [0,1] \rightarrow [0,1]$ adalah $s - norm$.

3. Operasi komplemen pada himpunan *fuzzy* digunakan untuk kata sambung "*tidak*". Proposisi *fuzzy compound* $FP = (x\ adalah\ T\ dan\ x\ tidak\ G)\ atau\ x\ adalah\ S$ diintrepretasikan dengan fungsi keanggotaan

$$\mu_{FP}(x_1, x_2, x_3) = s\left\{t\left[\mu_T(x_1), c\left(\mu_G(x_2)\right)\right], \mu_S(x_3)\right\} \tag{2.40}$$

dengan s, t dan c secara berurutan adalah $s - norm$, $t - norm$ dan komplemen *fuzzy*, kemudian himpunan *fuzzy* $S = SEDANG, G = GELAP$ dan $T = TERANG$ sesuai dengan contoh 3.3, dan $x_1 = x_2 = x_3 = x$.

7. Aturan *Fuzzy* IF- THEN

Banyak sistem *fuzzy* yang dibentuk berdasarkan aturan *fuzzy* IF THEN. Aturan *fuzzy* IF-THEN adalah pernyataan yang berbentuk

$$IF < proposisi\ fuzzy >, THEN < proposisi\ fuzzy > \quad (2.41)$$

Bagian paling penting dalam aturan fuzzy *IF-THEN* adalah interpretasi dari aturan tersebut. Penjelasan tentang hal ini akan diilustrasikan dengan contoh, hal ini ditujukan untuk mempermudah penjelasan tentang interpretasi aturan *fuzzy IF-THEN* . Sebagai contoh, misalkan terdapat aturan *fuzzy* yaitu "*IF* warna citra TERANG, *THEN pixel-pixel* pada citra itu KECIL" maka aturan *fuzzy IF-THEN* dedefinisikan sebagai berikut:

$$IF < PF_1 > THEN < PF_2 > \quad (2.42)$$

Salah satu jenis interpretasi aturan *fuzzy IF-THEN* adalah implikasi Mamdani. Implikasi Mamdani dari aturan *fuzzy IF-THEN* (2.2) diintepretasikan sebagai relasi *fuzzy* $\mu_{Q_{MM}}$ atau $\mu_{Q_{MP}}$ dalam $U \times V$ dengan fungsi keanggotaan (Wang, 1997: 67)

$$\mu_{Q_{MM}}(x, y) = \min[\mu_{PF_1}(x), \mu_{PF_2}(y)] \quad (2.43)$$

atau

$$\mu_{Q_{MP}}(x, y) = \mu_{PF_1}(x)\mu_{PF_2}(y) \quad (2.44)$$

C. *Mean Square Error* (MSE) dan *Peak Signal to Noise Ratio* (PSNR)

Setiap sistem dibentuk dengan tujuan untuk menyelesaikan suatu permasalahan, tentunya sistem yang dibentuk diharapkan memiliki akurasi atau tigkat ketepatan yang tinggi dalam menyelesaikan suatu permasalahan. Oleh karena itu diperlukan sesuatu yang dapat mengukur tingat ketepatan suatu sistem. Reduksi *impulse noise* pada tugas akhir ini dianalisis tingkat keakuatannya dengan

menggunakan dua metode yaitu MSE (*Mean Square Error*) dan PSNR (*Peak Signal to Noise Ratio*)**.**

1. *Mean Square Error* (MSE)

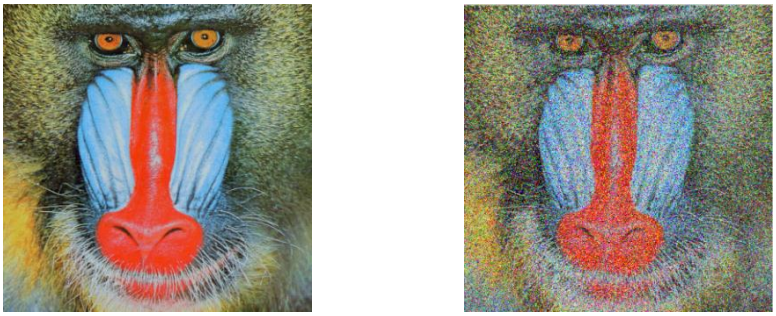
Misalkan Img adalah citra yang dihasilkan oleh filter dan Org adalah citra asli dengan ukuran *pixel* yang sama yaitu NM, maka MSE didefinisikan sebagai (Schulte, 2006: 9)

$$MSE(Img, Org) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}[Org(i,j)-Img(i,j)]^2}{NM} \quad (2.45)$$

Berdasarkan rumus yang disampaikan di atas, maka nilai MSE akan semakin kecil jika nilai dari $Org(i,j)$ semakin dekat dengan nilai dari $Img(i,j)$, hal ini mengindikasikan bahwa semakin kecil nilai MSE maka semakin mirip citra Org dan Img.

Contoh 2.17

Berikut adalah dua citra bertipe RGB, citra pertama adalah citra asli dan yang kedua adalah citra yang terkorupsi oleh *impulse noise* sebesar 30%.



(a) (b)

Gambar 2.18 (a). Citra 'mandril_color.tif' (b). Citra 'mandril_color.tif' yang Terkorupsi *Impulse Noise* Sebesar 30% (diakses dari http://www.image-processingplace.com/DIP-3E/dip3e book images_downloads.htm)

Dengan menggunakan *script m-file MSECOL*.m (lampiran I) diperoleh MSE dari dua gambar di bawah ini yaitu sebesar $5.8794e + 003$ atau 5879,4.

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

Misalkan Img adalah citra yang dihasilkan oleh filter, Org adalah citra asli dan S adalah nilai maksimal dari intensitas suatu *pixel*, yaitu $S = 2^m - 1$, dengan m adalah bit pada citra, maka PSNR didefinisikan sebagai:

$$PSNR(Img, Org) = 10 \log_{10} \frac{S^2}{MSE(Img,Org)} \quad (2.46)$$

satuan dari PSNR adalah *decibel* (db) (Schulte, 2006:10).

Contoh 2.18

PSNR dari citra pada Gambar 2.18 dapat dicari dengan menggunakan bantuan MATLAB sebagai berikut;

```
>>PSNR=((log(255^2/MSECOL(A,B))/log(10)*10))
PSNR =
10.4584
```

Jadi PSNR dari kedua citra tersebut adalah 10.4584 dB.

Berdasarkan rumus PSNR yang dituliskan di atas, terlihat bahwa nilai PSNR ditentukan oleh nilai dari MSE, semakin kecil nilai MSE maka akan menyebabkan semakin besarnya nilai dari PSNR. Hal ini mengindikasikan bahwa semakin besar nilai dari PSNR maka citra asli akan semakin mirip dengan citra yang dihasilkan oleh proses filter.

BAB III

PEMBAHASAN

A. Metode *Fuzzy Two - Step Filter*

Fuzzy Two-Step Filter adalah salah satu metode filter non linear. Sebagaimana metode filter non linear yang lain metode ini terdiri dari dua tahap. Tahap pertama adalah tahap deteksi *noise* sedangkan tahap kedua adalah tahap *filtering*.

1. Tahap deteksi *noise* dengan dasar-dasar teori *fuzzy*

   Tahap deteksi *noise* dilakukan terhadap masing-masing komponen warna secara terpisah yaitu komponen merah, hijau dan biru. Tahap ini terdiri dari empat bagian yaitu

   a. Mencari nilai gradien *fuzzy*

      Masing-masing komponen warna dicari selisih dari *pixel* pusat dengan *pixel-pixel* yang terdapat pada persekitaran *pixel* pusat tersebut, nilai ini disebut nilai gradien. Selanjutnya akan ada tiga nilai gradien yaitu nilai gradien pertama, kedua dan ketiga yang kemudian dihubungkan dengan aturan *fuzzy* pertama. *Output* dari aturan *fuzzy* pertama adalah nilai gradien *fuzzy*.

   b. Deteksi lokal

      Hasil yang diperoleh pada bagian pertama digunakan sebagai *input* untuk bagian kedua dengan menerapkan aturan *fuzzy* kedua, sehingga diperoleh *pixel-pixel* yang terkena *impulse noise*.

c. Deteksi Global

Bagian ini ditujukan untuk mengembangkan metode sehingga dapat lebih efektif dan dapat diterapkan untuk *noise* yang lebih beragam. Bagian ini akan menerapkan histogram citra.

d. Menentukan *pixel-pixel* yang terdapat pada himpunan *fuzzy impulse noise*

Himpunan *fuzzy impulse noise* adalah himpunan yang menyatakan suatu *pixel* terkorupsi oleh *impulse noise* atau tidak. Sehingga jika sebuah *pixel* mempunyai derajat keanggotaan tidak nol pada himpunan ini maka *pixel* tersebut digolongkan sebagai *pixel* yang bernoise. Bagian ini adalah *output* terakhir yang dihasilkan pada tahap deteksi. Dengan demikian bagian ini diakhiri dengan diperolehnya himpunan *pixel-pixel* yang mempunyai derajat keanggotaan tidak nol pada himpunan *fuzzy impulse noise* untuk masing-masing komponen yang kemudian dinyatakan dengan $\mu_{Impulse}^{R}$, $\mu_{Impulse}^{G}$ dan $\mu_{Impulse}^{B}$.

2. Tahap filter

Tahap filter akan terfokus pada *pixel-pixel* yang terkorupsi *impulse noise*. Tahap ini dibagi menjadi tiga bagian, diantaranya adalah

a. Melakukan Iterasi pertama

Iterasi pertama dilakukan dengan menerapkan jendela ketetangaan dengan ukruan (3×3) atau dalam hal ini $(2N + 1 \times 2N + 1)$ dengan $N = 1$. *Input* yang digunakan adalah *pixel-pixel* yang mempunyai derajat keanggotaan tak nol pada himpunan *fuzzy impulse noise*.

b. Melakukan Iterasi selanjutnya

Iterasi berikutnya dilakukan dengan mengubah ukuran dari jendela ketetanggan yaitu dengan menggunakan $N = 2,3,4,$ dst dan mengubah parameter himpunan *fuzzy impulse noise*. *Input* yang digunakan adalah *output* yang diperoleh dari iterasi pertama.

c. Menentukan kriteria berhenti

Kriteria berhenti ditujukan untuk menghentikan proses iterasi, dimana kriteria ini ditentukan sedemikian rupa sehingga iterasi akan berhenti jika hasil yang diperoleh sudah maksimal.

Tahapan-tahapan yang disampaikan di atas adalah garis besar dari metode *fuzzy to step filter* yang diterapkan untuk mereduksi *impulse noise* pada citra berwarna atau biasa disingkat FTSFC. Berikut ini adalah penjelasan lebih lanjut mengenai tahap-tahap yang sudah disampaikan di atas.

1. Deteksi *Impulse Noise* dengan Dasar-dasar Teori *Fuzzy*

Deteksi *impulse noise* dengan *fuzzy* dilakukan dengan menggunakan nilai gradien *fuzzy*. Tujuan dari tahap ini adalah menentukan *pixel* yang terkorupsi oleh *impulse noise*. *Impulse noise* dapat terjadi pada masing-masing vektor pada ruang warna RGB, sehingga nilai gradien *fuzzy* akan dicari pada masing-masing vektor (komponen warna). Nilai gradien *fuzzy* dalam skripsi ini akan dicari pada vektor warna merah saja, untuk vektor warna biru dan hijau penjelasannya identik. Nilai gradien *fuzzy* inilah yang kemudian menjadi *output* dalam aturan *fuzzy* pertama dengan inputnya adalah nilai gradien.

a. Nilai Gradien *Fuzzy*

Setiap *pixel* (i, j) pada komponen merah dalam citra yang bukan *pixel* tepi, kita terapkan persekitaran (3×3) pada setiap *pixel* tersebut, seperti ilustrasi pada Gambar 3.1. Setiap persekitaran dari (i, j) berkorespondensi dengan salah satu dari arah berikut { Utara=U, Timur=T, Selatan=S, Barat=B, Barat Laut=BL, Timur Laut=TL, Barat Daya= BD, Tenggara= TG}.

	-2	-1	0	1	2
-2					
-1		$\boldsymbol{BL}$	$\boldsymbol{U}$	$\boldsymbol{TL}$	
0		$\boldsymbol{B}$	$\boldsymbol{(i,j)}$	$\boldsymbol{T}$	
1		$\boldsymbol{BD}$	$\boldsymbol{S}$	$\boldsymbol{TG}$	
2					

Posisi	Arah
$(i-1, j-1)$	BL
$(i-1, j)$	U
$(i-1, j+1)$	TL
$(I, j-1)$	B
$(I, j+1)$	T
$(i+1, j-1)$	BD
$(i+1, j)$	S
$(i+1, j+1)$	TG

Gambar 3.1 Persekitaran dari Sebuah *Pixel* Pusat, Posisi dan Arahnya

Jika A_R menandakan komponen warna merah sebagai *input* citra. Selanjutnya akan dicari nilai gradien, yaitu selisih dari *pixel* pusat dengan *pixel-pixel* yang berada di sekitar *pixel* pusat tersebut. Secara matematis nilai gradien yang dinyatakan dengan $\nabla_{(k,l)} A_R(i, j)$, didefinisikan sebagai;

$$\nabla_{(k,l)} A_R(i, j) = A_R(i + k, j + l) - A_R(i, j), \text{dengan } k, l \in \{-1, 0, 1\} \qquad (3.1)$$

setiap pasangan (k, l) berkorespondensi dengan salah satu dari delapan arah, dan (i, j) disebut sebagai pusat (*center*) dari gradien. Dalam pembahasan selanjutnya nilai gradien dengan definisi di atas disebut sebagai nilai gradien pertama.

Jika salah satu nilai gradien pertama pada arah tertentu besar, maka dapat diprediksi bahwa terjadi *impulse noise* pada pusat dari gradien tersebut yaitu posisi (i, j). Meskipun demikian perkiraan ini dapat salah dalam dua kasus;

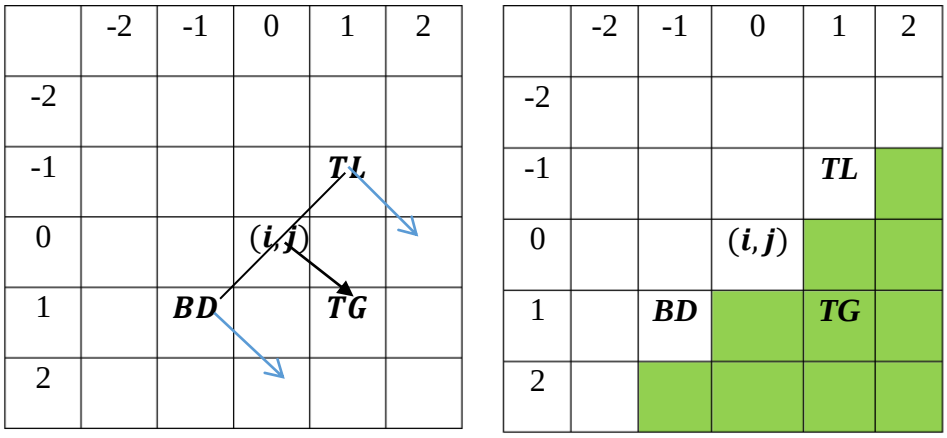
1) Pusat persekitaran terletak pada tepi dari citra sehingga nilai gradiennya besar,
2) Salah satu dari persekitaran tersebut yang terkorupsi *impulse noise*, sementara pusatnya tidak terkorupsi *impulse noise*,

Pada dasarnya kedua permasalahan di atas dapat diselesaikan dengan menggunakan nilai gradien *fuzzy*. Hanya saja untuk persoalan kedua harus ditemukan terlebih dahulu kedelapan nilai gradien *fuzzy* dan kemudian ditentukan apakah *pixel* pusatnya terkorupsi *impulse noise*, lebih lanjut tentang hal ini akan dijelaskan pada tahap deteksi lokal. Nilai gradien *fuzzy* adalah nilai yang diperoleh dari aturan *fuzzy* pertama. Aturan *fuzzy* pertama menghubungkan nilai gradien pertama, nilai gradien kedua dan ketiga.

Nilai gradien kedua dan nilai gradien ketiga diperoleh dengan mencari selisih dua *pixel* yang terletak pada persekitaran dari *pixel* pusat yang arahnya membentuk sudut siku-siku dengan nilai gradien pertamanya. Hal ini diilustrasikan dalam Gambar 3.2.a, sebagai contoh untuk nilai gradien pertama dengan arah Tenggara (TG) yaitu untuk $(k, l) = (1,1)$, nilai gradien pertamanya adalah $\nabla_{(1,1)} A_R(i, j) = A_R(i + 1, j + 1) - A_R(i, j)$ sedangkan nilai gradien kedua dan ketiga pada arah tenggara (TG) adalah $\nabla_{(1,1)} A_R(i +$

$1, j-1) = A_R(i+2, j) - A_R(i+1, j-1)$ dan $\nabla_{(1,1)} A_R(i-1, j+1) = A_R(i, j+2) - A_R(i-1, j+1)$.

Nilai gradien kedua dan ketiga akan digunakan untuk membentuk aturan *fuzzy* yang dapat menentukan apakah suatu *pixel* merupakan *pixel* tepi atau *pixel* yang terkorupsi oleh *impulse noise*. Hal ini diilustrasikan oleh contoh yang terdapat pada Gambar 3.2.b. Pada gambar tersebut *pixel* pusat adalah *pixel* tepi, dan berada pada arah tenggara (TG), dengan demikian maka nilai dari $\nabla_{(1,1)} A_R(i, j)$ akan besar begitu pula dengan kedua nilai gradien pada arah yang sama yaitu $\nabla_{(1,1)} A_R(i+1, j-1)$ dan $\nabla_{(1,1)} A_R(i-1, j+1)$ yang nilai gradiennya juga besar dalam hal ini ketiga nilai gradien besar dan *pixel* pusat bukan merupakan *pixel* yang terkorupsi *impulse noise*. Hal ini menjadi acuan untuk pembuatan aturan *fuzzy* pertama yang ditujukan untuk membedakan *pixel* yang terkorupsi *noise* dan *pixel* tepi.



Gambar 3.2. (a) Persekitaran dari *Pixel* Pusat pada Arah Tenggara dan Dua Persekitaran Lain yang Berhubungan dengan Arah Tenggara. (b) Contoh dari *Pixel* yang Berada Pada Tepi.

Tabel di bawah ini (3.1) adalah tabel yang menunjukkan nilai gradien untuk masing-masing arah. Kolom ketiga adalah nilai gradien pertama untuk masing-masing arah dan kolom keempat adalah nilai dua gradien yang berhubungan dengan nilai gradien pertama yaitu nilai gradien kedua dan nilai gradien ketiga

Tabel 3.1 Arah ,Nilai Gradien Pertama, Kedua dan Ketiga

No	Arah	Nilai gradien pertama	Nilai gradien kedua dan gradien ketiga
1	BL	$\nabla_{BL}A_R(i,j)$	$\nabla_{BL}A_R(i+1,j-1), \nabla_{BL}A_R(i-1,j+1)$
2	U	$\nabla_{U}A_R(i,j)$	$\nabla_{U}A_R(i,j-1), \nabla_{U}A_R(i,j+1)$
3	TL	$\nabla_{TL}A_R(i,j)$	$\nabla_{TL}A_R(i-1,j-1), \nabla_{TL}A_R(i+1,j+1)$
4	T	$\nabla_{T}A_R(i,j)$	$\nabla_{T}A_R(i-1,j), \nabla_{T}A_R(i+1,j)$
5	TG	$\nabla_{TG}A_R(i,j)$	$\nabla_{TG}A_R(i-1,j+1), \nabla_{TG}A_R(i+1,j-1)$
6	S	$\nabla_{S}A_R(i,j)$	$\nabla_{S}A_R(i,j+1), \nabla_{S}A_R(i,j-1)$
7	BD	$\nabla_{BD}A_R(i,j)$	$\nabla_{BD}A_R(i+1,j+1), \nabla_{BD}A_R(i-1,j-1)$
8	B	$\nabla_{B}A_R(i,j)$	$\nabla_{B}A_R(i+1,j), \nabla_{B}A_R(i-1,j)$

Berikut adalah penjabaran dari tabel tersebut

1. Arah barat laut (BL)

   $\nabla_{BL}A_R(i,j) = A_R(i-1,j-1) - A_R(i,j)$, $\nabla'_{BL}A_R(i,j) = \nabla_{BL}A_R(i+1,j-1) = A_R(i,j-2) - A_R(i+1,j-1)$, dan $\nabla''_{BL}A_R(i,j) = \nabla_{BL}A_R(i-1,j+1) = A_R(i-2,j) - A_R(i-1,j+1)$.

2. Arah utara (U)

   $\nabla_{U}A_R(i,j) = A_R(i-1,j) - A_R(i,j)$, $\nabla'_{U}A_R(i,j) = \nabla_{U}A_R(i,j-1) = A_R(i-1,j-1) - A_R(i,j-1)$ dan $\nabla''_{U}A_R(i,j) = \nabla_{U}A_R(i,j+1) = A_R(i-1,j+1) - A_R(i,j+1)$.

3. Arah timur laut (TL)

   $\nabla_{TL}A_R(i,j) = A_R(i-1,j+1) - A_R(i,j)$,$\nabla'_{TL}A_R(i,j) = \nabla_{TL}A_R(i-1,j-1) = A_R(i-2,j) - A_R(i-1,j-1)$ dan $\nabla''_{TL}A_R(i,j) = \nabla_{TL}A_R(i+1,j+1) = A_R(i,j+2) - A_R(i+1,j+1)$.

4. Arah timur (T)

   $\nabla_T A_R(i,j) = A_R(i,j+1) - A_R(i,j)$, $\nabla'_T A_R(i,j) = \nabla_T A_R(i-1,j) = A_R(i-1,j+1) - A_R(i-1,j)$ dan $\nabla''_T A_R(i,j) = \nabla_T A_R(i+1,j) = A_R(i+1,j+1) - A_R(i+1,j)$.

5. Arah tenggara (TG)

   $\nabla_{TG}A_R(i,j) = A_R(i+1,j+1) - A_R(i,j)$,$\nabla'_{TG}A_R(i,j) = \nabla_{TG}A_R(i-1,j+1) = A_R(i,j+2) - A_R(i-1,j+1)$ dan $\nabla''_{TG}A_R(i,j) = \nabla_{TG}A_R(i+1,j-1) = A_R(i+2,j) - A_R(i+1,j-1)$.

6. Arah selatan (S)

   $\nabla_S A_R(i,j) = A_R(i+1,j) - A_R(i,j)$, $\nabla'_S A_R(i,j) = \nabla_S A_R(i,j+1) = A_R(i+1,j+1) - A_R(i,j+1)$ dan $\nabla''_S A_R(i,j) = \nabla_S A_R(i,j-1) = A_R(i+1,j-1) - A_R(i,j-1)$.

7. Arah barat daya (BD)

   $\nabla_{BD}A_R(i,j) = A_R(i+1,j-1) - A_R(i,j)$, $\nabla'_{BD}A_R(i,j) = \nabla_{BD}A_R(i+1,j+1) = A_R(i+2,j) - A_R(i+1,j+1)$ dan $\nabla''_{BD}A_R(i,j) = \nabla_{BD}A_R(i-1,j-1) = A_R(i,j-2) - A_R(i-1,j-1)$.

8. Arah barat (B)

$\nabla_B A_R(i,j) = A_R(i,j-1) - A_R(i,j)$, $\nabla'_B A_R(i,j) = \nabla_B A_R(i+1,j) = A_R(i+1,j-1) - A_R(i+1,j)$ dan $\nabla''_B A_R(i,j) = \nabla_B A_R(i-1,j) = A_R(i-1,j-1) - A_R(i-1,j)$.

Himpunan delapan arah pada persekitaran dilambangkan dengan $D = \{BL, U, TL, T, TG, S, BD, B\}$, nilai gradien pertama dilambangkan dengan $\nabla_D A_R(i,j)$, nilai gradien kedua dilambangkan dengan $\nabla'_D A_R(i,j)$, dan nilai gradien ketiga dilambangkan dengan $\nabla''_D A_R(i,j)$ serta nilai gradien *fuzzy* dilambangkan dengan $\nabla^F_D A_R(i,j)$, selanjutnya didefinisikan aturan *fuzzy* pertama sebagai berikut:

IF $|\nabla_D A_R(i,j)|$ besar AND $|\nabla'_D A_R(i,j)|$ kecil, OR $|\nabla_D A_R(i,j)|$ besar AND $|\nabla''_D A_R(i,j)|$ kecil, OR $|\nabla_D A_R(i,j)|$ big positif AND $|\left(\nabla'_D A_R(i,j) AND \nabla''_D A_R(i,j)\right)$ big negatif, OR $|\nabla_D A_R(i,j)|$ big negatif AND $\left(\nabla'_D A_R(i,j) AND \nabla''_D A_R(i,j)\right)$ big positif THEN $\nabla^F_D A_R(i,j)$ besar.

Kata "besar" direpresentasikan sebagai himpunan *fuzzy* besar dan dilambangkan dengan μ_B, "kecil" direpresentasikan sebagai himpunan *fuzzy* kecil dan dilambangkan dengan μ_K, "big positif" direpresentasikan sebagai himpunan *fuzzy* big positif dan dilambangkan dengan μ_{BP} dan "big negatif" direpresentasikan sebagai himpunan *fuzzy* big negatif dan dilambangkan dengan μ_{BN}. Fungsi keanggotaan dari himpunan tersebut dirumuskan dengan menggunakan fungsi keanggotaan trapesium. Hasil pada skripsi ini diperoleh

dengan menggunakan parameter $a = 70$ dan $b = 125$ untuk himpunan *fuzzy* besar dan kecil serta $c = 15$ dan $d = 25$. Sehingga persamaannya adalah sebagai berikut;

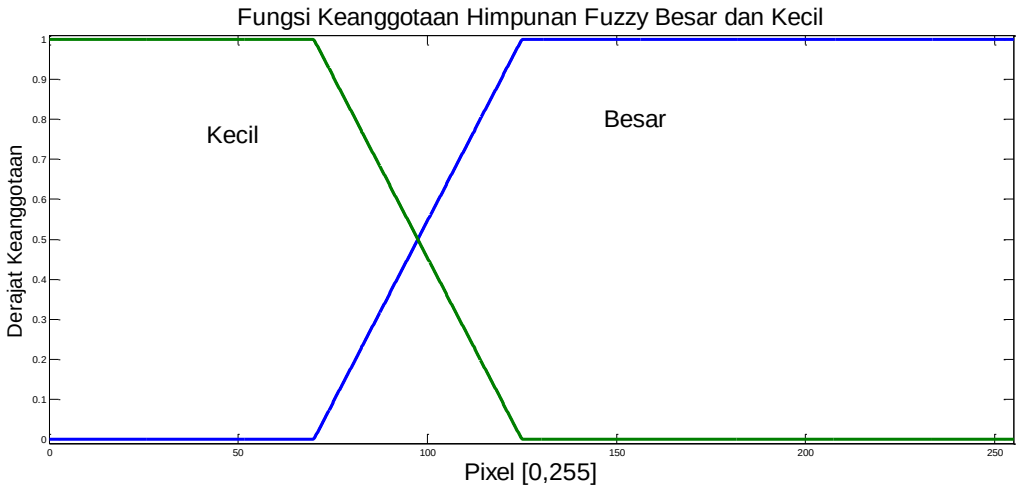
$$\mu_B(x) = \begin{cases} 1 & ,\text{untuk } x > 125 \\ \frac{(x-70)}{45} & ,\text{untuk } 70 \leq x \leq 125 \\ 0 & ,\text{untuk } x < 70 \end{cases} \quad (3.2)$$

$$\mu_K(x) = \begin{cases} 0 & ,\text{untuk } x > 125 \\ \frac{(125-x)}{45} & ,\text{untuk } 70 \leq x \leq 125 \\ 1 & ,\text{untuk } x < 70 \end{cases} \quad (3.3)$$

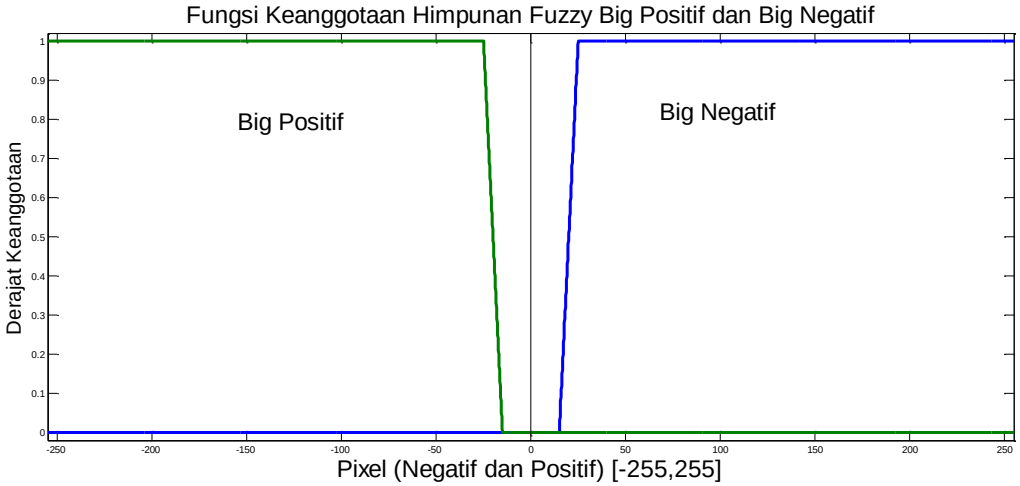
$$\mu_{BP}(x) = \begin{cases} 1 & ,\text{untuk } x > 25 \\ \frac{(x-15)}{10} & ,\text{untuk } 15 \leq x \leq 25 \\ 0 & ,\text{untuk } x < 15 \end{cases} \quad (3.4)$$

$$\mu_{BN}(x) = \begin{cases} 0 & ,\text{untuk } x > -15 \\ \frac{(-15-x)}{10} & ,\text{untuk } -25 \leq x \leq -15 \\ 1 & ,\text{untuk } x < -25 \end{cases} \quad (3.5)$$

persamaan (3.2) sampai (3.5) direpresentasikan dalam gambar berikut



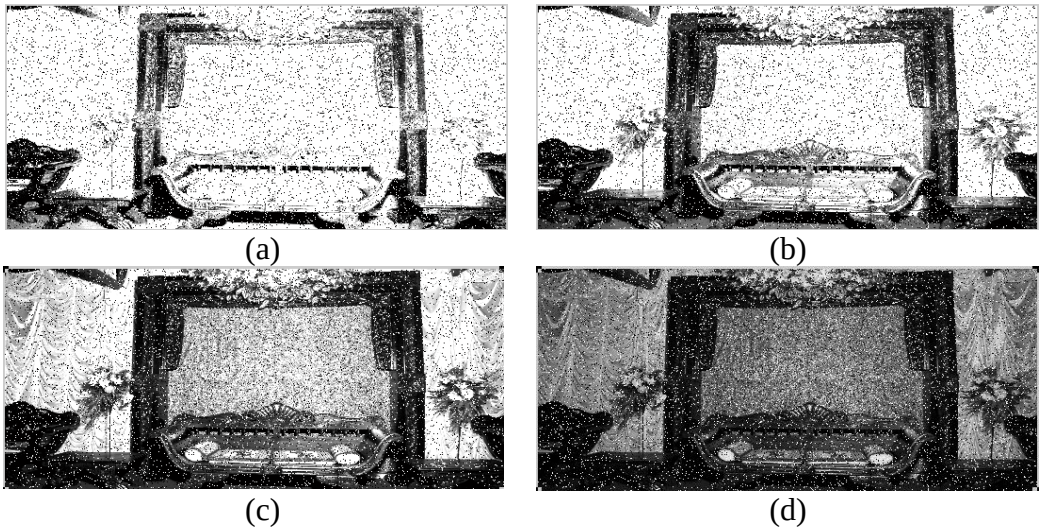
(a)



(b)

Gambar 3.3.(a) Fungsi Keanggotaan Himpunan *Fuzzy* Kecil dan Besar, (b) Fungsi Keanggotaan Himpunan *Fuzzy* Big Positif dan Big Negatif.

Kedua fungsi keanggotaan yang direpresentasikan oleh persamaan (3.2) dan (3.3) bergantung pada nilai dari parameter a dan b. Alasan dipilihnya nilai $a = 70$ dan $b = 125$ didasarkan pada hasil observasi pada citra di bawah ini



(a) (b)

(c) (d)

Gambar 3.4 (a) semua nilai $\left|\nabla_{BL} A_R(i,j)\right| \in [0,30]$, (b) semua nilai $\left|\nabla_{BL} A_R(i,j)\right| \in [30,70]$, (c) semua nilai $\left|\nabla_{BL} A_R(i,j)\right| \in [70,140]$, (d) semua nilai $\left|\nabla_{BL} A_R(i,j)\right| \in [140,255]$

Penjelasan mengenai citra di atas adalah sebagai berikut;

1. Nilai gradien untuk arah tertentu (D) yang terletak pada interval $[0,30]$ (setiap pikselnya berada pada selang $[0,30]$) cenderung bukan *pixel-pixel* yang bernoise dan bukan pula *pixel* tepi. Dapat dikatakan bahwa *pixel-pixel* pada interval ini hanya membentuk sebuah daerah yang homogen. Seperti ditunjukkan oleh Gambar 3.4 (a)
2. *Pixel-pixel* tepi atau *pixel-pixel* yang mungkin terkorupsi *noise* biasanya mempunyai nilai gradien yang berada pada interval $[30,70]$. Hal ini diilustrasikan oleh Gambar 3.4 (b), meskipun begitu *pixel-pixel* ini kita beri derajat keanggotaan yang kecil pada himpunan *pixel* ber*noise* karena *pixel-pixel* ini hampir tidak bernoise.
3. Nilai gradien untuk arah tertentu (D) yang terletak pada selang $[70,150]$ adalah *pixel* tepi atau *pixel* yang bernoise. Hal ini diilustrasikan oleh Gambar 3.4 (c) , dalam interval ini ada beberapa macam klasifikasi dari *pixel* baik yang bernoise maupun yang merupakan *pixel* tepi. Hal ini ditunjukkan oleh derajat keanggoataan dalam himpunan *fuzzy impulse noise* antara nol (untuk *pixel* yang bebas *impulse noise*) sampai 1 (*pixel* yang pasti terkorupsi *noise*).
4. Nilai gradien untuk arah tertentu yang lebih dari 140 dianggap merupakan *pixel* yang pasti ber*noise* atau memiliki kontur *pixel* yang besar, seperti yang diunjukkan oleh Gambar 3.4 (d). Karena *pixel-pixel* yang berada pada posisi ini kita anggap pasti merupakan *pixel* yang terkorupsi *noise*, maka kita beri derajat keanggotaan 1 untuk himpunan *fuzzy impulse noise*.

Karena *pixel-pixel* yang dicari hanyalah *pixel* yang beroise saja maka kita pilih $a \in [50\ 80]$ dan $b \in [100\ 150]$.

Berdasarkan hasil observasi aturan *fuzzy* pertama, maka kita dapat memperoleh subrule berikut: Jika nilai gradien pertama untuk suatu arah D adalah "big positif" ("big negatif") dan dua nilai gradien yang berhubungan dengan gradien dasar dengan arah yang sama yaitu gradien kedua dan ketiga adalah "big negatif" ("big positif"), maka nilai gradien *fuzzy* untuk arah tersebut akan "besar". Keberhasilan aturan *fuzzy* pertama dalam menentukan adanya *impulse noise* atau tidak pada citra sangat dipengaruhi oleh nilai parameter c dan d, dengan demikian nilai c dan d harus dipilih dengan baik.

Pixel yang terletak pada tepi dapat menyebabkan besarnya nilai gradien pertama, kedua dan ketiga, meskipun demikian nilai gradien yang dihasilkan sering kali berbeda tanda. Hal inilah yang melandasi dibutuhkan pendefinisian himpunan *fuzzy* "big positif" dan "big negatif". Kemudian *noise* yang berada pada bagain tepi dapat meyebabkan nilai gradien yang tidak terlalu besar. Dengan demikian himpunan *fuzzy* "big positif" dan "big negatif" di definisikan dengan menggunakan parameter yang mendekati nilai nol. Sehingga dipilih nilai $c \in [15\ 20]$ dan $d \in [20\ 30]$.

Aturan *fuzzy* pertama mengandung konjungsi dan disjungsi. Dalam logika *fuzzy* representasi dari konjungsi dan disjungsi adalah dengan menggunakan $t - norm$ dan $s - norm$ (Cornelis, 2002). Jika $s - norm$ yang digunakan adalah operasi penjumlahan biasa dan $t - norm$ yang digunakan adalah perkalian biasa dan fungsi keanggotaan himpunan *fuzzy* $|\nabla_D A_R(i,j)|$ besar

dilambangkankan dengan μ_{RB}, $|\nabla_D A_R(i,j)|$ kecil dilambangkankan dengan μ_{RK}, $|\nabla_D A_R(i,j)|$ big positif dilambangkankan dengan μ_{RBP}, $|\nabla_D A_R(i,j)|$ big negatif dilambangkankan dengan μ_{RBN}, $|\nabla'_D A_R(i,j)|$ kecil dilambangkan dengan μ'_{RK}, $|\nabla'_D A_R(i,j)|$ besar dilambangkankan dengan μ'_{RB}, $|\nabla'_D A_R(i,j)|$ big positif dilambangkankan dengan μ'_{RBP}, $|\nabla'_D A_R(i,j)|$ big negatif dilambangkankan dengan μ'_{RBN}, $|\nabla''_D A_R(i,j)|$ kecil dilambangkan dengan μ''_{RK}, $|\nabla''_D A_R(i,j)|$ besar dilambangkankan dengan μ''_{RB}, $|\nabla''_D A_R(i,j)|$ big positif dilambangkankan dengan μ''_{RBP}, $|\nabla''_D A_R(i,j)|$ big negatif dilambangkankan dengan μ''_{RBN} dan $\nabla^F_D A_R(i,j)$ besar dilambangkan dengan μ_{RBH} maka interpretasi dari aturan *fuzzy* pertama dapat dituliskan sebagai berikut

$$\mu_{RBH} = \mu_{RB}\mu'_{RK} + \mu_{RB}\mu''_{RK} + \mu_{RBP}\left(\mu'_{RBN}\mu''_{RBN}\right) + \mu_{RBN}\left(\mu'_{RBP}\mu''_{RBP}\right) \quad (3.6)$$

Nilai dari μ_{RBH} disebut dengan *activation degree* dari aturan *fuzzy* pertama. Nilai ini adalah nilai yang menyebabkan $\nabla^F_D A_R(i,j)$ dapat digolongkan sebagai himpunan *fuzzy* besar. Dengan demikian maka nilai ini adalah derajat keanggotaan dari himpunan *fuzzy* $\nabla^F_D A_R(i,j)$.

b. Deteksi Pertama (Deteksi Lokal)

Bagian deteksi diharapkan akan menghasilkan keputusan bahwa sebuah *pixel* pusat (i,j) terkorupsi *impulse noise* atau tidak. Bagian ini didasarkan pada aturan *fuzzy* kedua yang didefinisikan sebagai berikut:

IF lebih dari empat $\nabla_D^F A_R(i,j)$ besar THEN *pixel* pusat yaitu $A_R(i,j)$ adalah *pixel* yang terkorupsi *impulse noise*.

Berdasarkan aturan *fuzzy* kedua tersebut, dapat disimpulkan bahwa jika sebagain besar dari kedelapan $\nabla_D^F A_R(i,j)$ besar, maka *pixel* pusatnya bernoise.

Parameter yang menunjukkan bahwa $\nabla_D^F A_R(i,j)$ besar, adalah jika nilainya berada pada himpunan *fuzzy* besar (3.2) dan derajat keanggotaannya lebih besar dari 0.01. selanjutnya *activation degree* dari aturan *fuzzy* kedua untuk *pixel* tertentu $A_R(i,j)$ digunakan untuk menentukan derajat keanggotaan dari himpunan *fuzzy* komponen merah yang terkorupsi *impulse noise*, yang dilambangkan dengan $\mu^R{}_{impulse}(.)$.

Metode deteksi ini disebut dengan metode deteksi lokal. Bagian berikutnya yaitu metode deteksi kedua akan membahas metode deteksi secara global. Metode ini akan fokus untuk mengembangkan metode agar lebih akurat dalam mendeteksi adanya *impulse noise*, dan memungkinkan metode untuk dapat menentukan apakah *pixel-pixel* terkorupsi *impulse noise* saja tanpa tercampur dengan jenis *noise* yang lain. Hal ini dikarenakan citra mungkin saja terkorupsi *noise* yang beragam jenisnya.

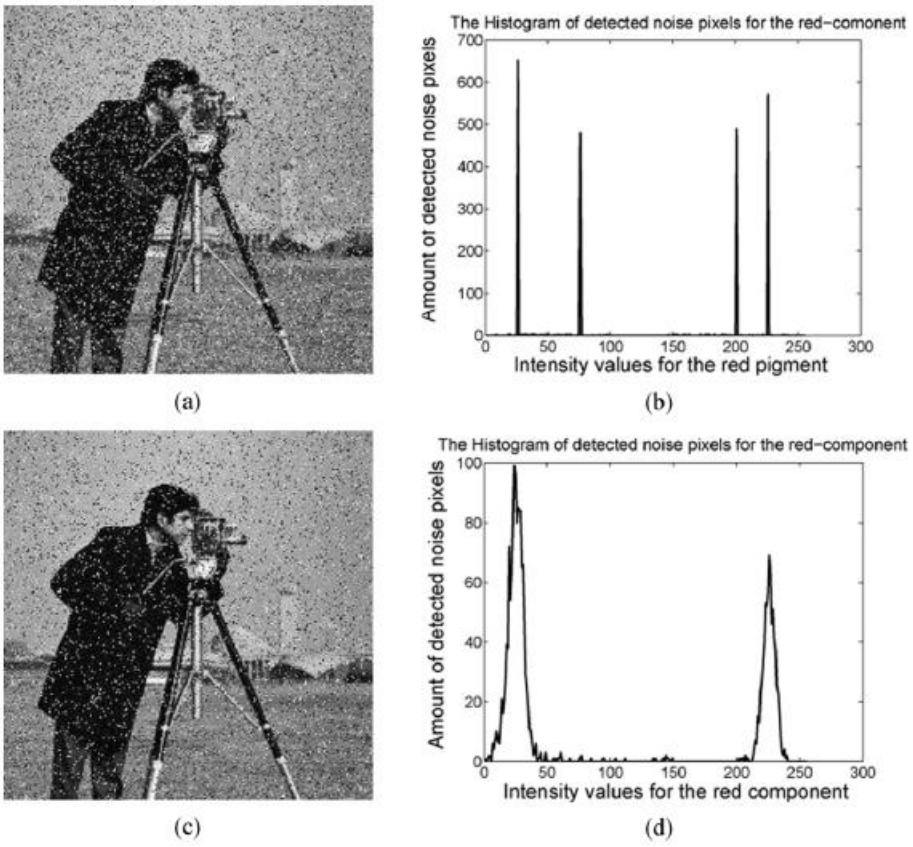
c. Deteksi Kedua (Deteksi Global)

Jenis *noise* yang terdapat pada citra mungkin saja berupa campuran antara jenis *noise* satu dengan jenis *noise* yang lain. Oleh karenanya diperlukan metode deteksi kedua yang akan mungkin dapat mereduksi *noise* dengan tipe campuran (*mixed noise*) ini dan sekaligus memastikan suatu *noise* hanyalah murni *impulse noise*.

Semua *pixel* pada komponen warna merah yang mempunyai *activation degree* yang tidak nol pada aturan *fuzzy* kedua direpresentasikan dalam histogram Gambar 3.5(b) dan 3.5(d). Dengan demikian akan diperoleh tiga histogram, yaitu dari komponen warna merah, komponen warna hijau dan komponen warna biru. Sumbu vertikal dalam histogram ini menandakan banyaknya *pixel* yang terkorupsi *noise* dengan nilai *pixel* pada sumbu horizontal. Gambar histogram *noise* 3.5.(a) menunjukkan bahwa citra hanya terkorupsi oleh *impulse noise* saja, tanpa tercampur oleh *noise* yang lain. Berdasarkan gambar tersebut terlihat bahwa histogramnya mengandung banyak puncak-puncak yang curam.

Selanjutnya Gambar 3.5.(c) adalah citra yang terkorupsi oleh campuran dari *impulse noise* dan *Gaussian noise*. Berdasarkan gambar tersebut histogramnya tidak hanya mempunyai puncak-puncak curam, namun juga mengandung beberapa nilai yang berada disamping nilai-nilai puncak curam tersebut.

Berdasarkan observasi tersebut maka jika histogram *noise* tidak mengandung nilai-nilai yang curam, maka hal ini mengindikasikan bahwa citra tidak hanya terkorupsi oleh *impulse noise*. Namun jika terdapat beberapa nilai puncak yang curam maka citra hanya terkorupsi oleh *impulse noise* saja tanpa tercampur jenis *noise* yang lain. Informasi ini akan digunakan untuk mengembangkan metode deteksi lokal.



Gambar 3.5 (a) Citra Terkorupsi *Impulse Noise* sebesar 20%, (b). Histogram dari Citra a, (c). Citra yang Terkorupsi *Impulse Noise* sebesar 10% dan Gaussian *noise* dengan $\alpha = 0.5$, (d). Histogram dari Citra c. (Schulte, 2006:6)

Jika nilai pada sumbu vertikal yang paling besar disimbolkan dengan $\max_{deteksi}$ dan $HISTR(q)$ menandakan banyaknya *pixel* ber*noise* dengan nilai *pixel* q pada komponen warna merah maka histogram mempunyai nilai yang curam jika $\max_{deteksi} / \sum_{q=0}^{q=2^m-1} HISTR(q) > 0.08$. Hasil inilah yang menjadi syarat untuk dilakukannya proses deteksi global, dengan kata lain jika syarat ini tidak dipenuhi maka deteksi global tidak akan dilakukan.

Selanjutnya jika kondisi di atas sudah dipenuhi maka akan ditentukan nilai-nilai *pixel* lain yang memenuhi kondisi $\max_{deteksi} / \sum_{q=0}^{q=2^m-1} HISTR(q) > 0.08$. Nilai-nilai *pixel* tersebut

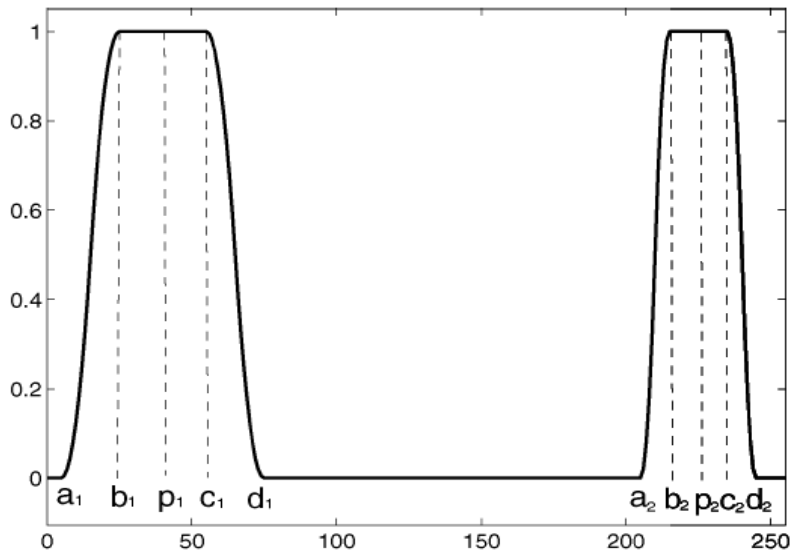
dilambangkan dengan p_k, dengan $k = \{1,2,3,\dots,n\}$, dan n adalah banyaknya nilai yang memenuhi kriteria tersebut. Nilai 0.08 yang digunakan sebagai parameter adalah hasil yang diperoleh dari penelitian dan observasi yang dilakukan oleh Sculte (2006: 7).

d. Himpunan *Fuzzy Impulse Noise*

Setiap *pixel* yang terkorupsi oleh *impulse noise* dapat direpresentasikan dalam himpunan *fuzzy*. Himpunan *fuzzy* tersebut disimbolkan dalam $\mu_{noise\_p_k}$ dan dirumuskan dalam persamaan berikut:

$$\mu_{noise\_p_k}(x) = \begin{cases} 0 & ,\forall x \le a_k, \text{atau } \forall x \ge d_k \\ 2\left(\frac{x-a_k}{b_k-a_k}\right)^2 & ,\forall x \in \left]a_k, \frac{(a_k+b_k)}{2}\right] \\ 1-2\left(\frac{x-a_k}{b_k-a_k}\right)^2 & ,\forall x \in \left]\frac{(a_k+b_k)}{2}, b_k\right] \\ 1 & ,\forall x \in [b_k, c_k] \\ 1-2\left(\frac{x-c_k}{d_k-c_k}\right)^2 & ,\forall x \in \left]c_k, \frac{(c_k+d_k)}{2}\right] \\ 2\left(\frac{x-d_k}{d_k-c_k}\right)^2 & ,\forall x \in \left]\frac{(c_k+d_k)}{2}, d_k\right] \end{cases} \quad (3.7)$$

Representasi dari himpunan *fuzzy* di atas adalah sebagai berikut



Gambar 3.6 Himpunan Keanggotaan *Impulse Noise*

Kemudian fungsi keanggotaan *impulse noise* untuk komponen merah yaitu $\mu_{impulse}^{R}$ didefinisikan dengan

$$\mu_{impulse}^{R}\big(A_R(i,j)\big) = \max_{k\in\{1,2,\dots n\}}\left(\mu_{noise_{p_k}}\big(A_R(i,j)\big)\right) \tag{3.8}$$

Parameter (a_k, b_k, c_k, d_k) yang digunakan dalam persamaan (3.7), digunakan untuk mendefinisikan kemiringan dari masing-masing intensitas nilai p_k. Berdasarkan gambar 3.5, terdapat dua jenis histogram, yaitu histogram yang direpresentasikan oleh Gambar 3.5.(b) yang merupakan histogram dari gambar yang terkorupsi *impulse noise* saja yang mempuyai kemiringan sangat kecil dan histogram pada Gambar 3.5.(d) yang merupakan histogram yang terkorupsi *noise* campuran *Gaussian noise* dan *impulse noise* yang mempunyai kemiringan lebih besar. Berikut adalah observasi yang dilakukan untuk membedakan kedua jenis histogram tersebut; (1). Estimasi standar deviasi σ, yang berkorespondensi/berhubungan dengan histogram yang curam (*fixed impulse noise*) dan (2). Estimasi standar deviasi σ, yang berkorespondensi / berhubungan dengan *mixed noise*. Hasil yang diperoleh dari observasi terhadap 10 gambar berbeda (Zlokolica, 2004 :3) menyatakan bahwa σ dapat dirumuskan dengan persamaan berikut;

$$\sigma = 0{,}2661 p_k - 0{,}7827 \tag{3.9}$$

Misalkan $THRa = THRd = \min(25, \lfloor\sigma\rceil)$ maka pareameter pada persamaan (3.7) dapat diperoleh dengan perhitungan sebagai berikut:

$$a_k = p_k - THRa \tag{3.10}$$

$$b_k = p_k - THRb\ , \qquad THRb = \frac{2}{3} THR\ a \tag{3.11}$$

$$c_k = p_k + THRc\ , \qquad THRc = \frac{2}{3} THRd \tag{3.12}$$

$$d_k = p_k + THRd \tag{3.13}$$

Tujuan utama pemilihan parameter ini adalah untuk mengembangkan metode agar lebih baik dalam menentukan *pixel-pixel* yang terkorupsi *noise* disamping juga memungkinkan untuk mereduksi citra yang terkorupsi oleh *mixed impulse noise*. Ketika nilai $\lfloor \sigma \rceil > 20$, maka $THRa$ dan $THRd$ nilainya akan lebih dari 20. Besarnya nilai dari $THRa$ dan $THRd$ dapat menyebabkan luasnya cakupan fungsi keanggotaan himpunan *fuzzy* yang dapat mengakibatkan efek *blurring* pada citra.

Akhir dari tahap ini adalah diperolehnya *pixel-pixel* tertentu yang mempunyai derajat keanggotaan tidak nol pada himpunan *fuzzy impulse noise* untuk ketiga komponen warna yaitu merah, hijau dan biru dan disimbolkan secara berturut-turut sebagai $\mu^R_{Impulse}$, $\mu^G_{Impulse}$ dan $\mu^B_{Impulse}$.

2. Proses Filter

Hasil yang diperoleh pada fase deteksi adalah diperolehnya tiga himpunan *fuzzy* yang menyatakan *pixel-pixel* yang terkorupsi oleh *impulse noise* pada masing-masing komponen warna yang dinyatakan dengan $\mu^R_{Impulse}$, $\mu^G_{Impulse}$ dan $\mu^B_{Impulse}$. Hasil ini akan digunakan sebagai sebagai input dalam proses filter.

Berbeda dengan metode reduksi *noise* pada citra jenis RGB yang biasanya, bagian filter pada metode reduksi citra jenis RGB yang akan digunakan pada skripsi ini adalah dengan mencari selisih nilai intensitas dari setiap komponen warna yang berbeda. Perbedaaan nilai intensitas antara dua komponen warna yang berbeda dihitung dengan persamaan berbentuk matriks berikut;

$$RG(i,j) = A_R(i,j) - A_G(i,j),\ GR(i,j) = A_G(i,j) - A_R(i,j) = -RG(i,j) \quad (3.14)$$

$$BG(i,j) = A_B(i,j) - A_G(i,j),\ GB(i,j) = A_G(i,j) - A_B(i,j) = -GB(i,j) \quad (3.15)$$

$$BR(i,j) = A_B(i,j) - A_R(i,j),\ RB(i,j) = A_R(i,j) - A_B(i,j) = -RB(i,j) \quad (3.16)$$

a. Iterasi Pertama

Masing-masing nilai yang terdapat pada persamaan 3.14-3.16, dibuat himpunan *fuzzy* dengan derajat keanggotaannya didasarkan pada aturan *fuzzy* berikut;

Aturan *fuzzy* ketiga (kombinasi komponen warna merah dan hijau)

IF $A_R(i,j)$ terkorupsi *impulse noise* OR $A_G(i,j)$ terkorupsi *impulse noise* THEN ($RG(i,j)$ terkorupsi *impulse noise*) AND ($GR(i,j)$ terkorupsi *impulse noise*).

Aturan *fuzzy* keempat (kombinasi komponen warna merah dan biru)

IF $A_R(i,j)$ terkorupsi *impulse noise* OR $A_B(i,j)$ terkorupsi *impulse noise* THEN ($RB(i,j)$ terkorupsi *impulse noise*) AND ($BR(i,j)$ terkorupsi *impulse noise*).

Aturan *fuzzy* kelima (kombinasi komponen warna hijau dan biru)

IF $A_G(i,j)$ terkorupsi *impulse noise* OR $A_B(i,j)$ terkorupsi *impulse noise* THEN ($GB(i,j)$ terkorupsi *impulse noise*) AND ($BG(i,j)$ terkorupsi *impulse noise*).

Hasil yang diperoleh pada fase deteksi adalah himpunan *pixel-pixel* pada setiap komponen warna yang terkorupsi oleh *impulse noise*. Himpunan ini digunakan sebagai *input* untuk aturan-aturan *fuzzy* yang didefinisikan di atas. Himpunan tersebut beserta pensimbolannya antara lain adalah "$A_R(i,j)$ adalah *impulse noise*" yang disimbolkan dengan $\mu^R_{impulse}\big(A_R(i,j)\big)$, "$A_G(i,j)$ adalah *impulse noise*" yang disimbolkan dengan $\mu^G_{impulse}\big(A_G(i,j)\big)$, dan "$A_B(i,j)$ adalah *impulse noise*" disimbolkan dengan $\mu^B_{impulse}\big(A_B(i,j)\big)$. Derajat keanggotaan himpunan *fuzzy impulse noise* untuk matriks yang didefinisikan pada persamaan 3.14-3.16 disimbolkan dengan $\mu_{RG}(i,j)$ untuk matrik RG pada posisi (i,j), $\mu_{RB}(i,j)$ untuk matrik RB pada posisi (i,j), simbol yang lain menyesuaikan. Aturan *fuzzy* ketiga, keempat dan kelima menggunakan disjungsi, yang dalam hal ini direpresentasikan dengan menggunakan $s-norm$. Jika $s-norm$ yang digunakan adalah operasi maksimum, maka akan diperoleh

$$\mu_{RG}(i,j) = \max\Big(\mu^R_{Impulse}\big(A_R(i,j)\big), \mu^G_{Impulse}\big(A_G(i,j)\big)\Big) \tag{3.17}$$

$$\mu_{GR}(i,j) = \max\Big(\mu^G_{Impulse}\big(A_G(i,j)\big), \mu^R_{Impulse}\big(A_R(i,j)\big)\Big) \tag{3.18}$$

$$\mu_{RB}(i,j) = \max\Big(\mu^R_{Impulse}\big(A_R(i,j)\big), \mu^B_{Impulse}\big(A_B(i,j)\big)\Big) \tag{3.19}$$

$$\mu_{BR}(i,j) = \max\left(\mu^{B}_{Impulse}\left(A_B(i,j)\right), \mu^{R}_{Impulse}\left(A_R(i,j)\right)\right) \quad (3.20)$$

$$\mu_{BG}(i,j) = \max\left(\mu^{B}_{Impulse}\left(A_B(i,j)\right), \mu^{G}_{Impulse}\left(A_G(i,j)\right)\right) \quad (3.21)$$

$$\mu_{GB}(i,j) = \max\left(\mu^{G}_{Impulse}\left(A_G(i,j)\right), \mu^{B}_{Impulse}\left(A_B(i,j)\right)\right) \quad (3.22)$$

Berdasarkan aturan *fuzzy* ketiga, keempat dan kelima, dapat diperoleh kesimpulan bahwa jika *pixel* pada komponen merah terkorupsi oleh *impulse noise* yaitu derajat keanggotaan pada himpunan *fuzzy* $\mu^{R}_{Impulse}\left(A_R(i,j)\right)$ besar dan *pixel* pada komponen warna hijau dan biru tidak terkorupsi oleh *noise* (derajat keanggotaan dari $\mu^{G}_{Impulse}\left(A_G(i,j)\right)$ kecil dan derajat keanggotaan dari $\mu^{B}_{Impulse}\left(A_B(i,j)\right)$ kecil) maka perbedaan warna antara merah-hijau dan merah-biru akan terkorupsi *impulse noise* yaitu $\mu^{GR}_{Impulse}\left(A_{GR}(i,j)\right), \mu^{RG}_{Impulse}\left(A_{RG}(i,j)\right), \mu^{RB}_{Impulse}\left(A_{RB}(i,j)\right)$ dan $\mu^{BR}_{Impulse}\left(A_{BR}(i,j)\right)$ mempunyai derajat keanggotaan yang besar.

Seperti yang sudah disampaikan sebelumnya bahwa proses filter hanya akan dilakukan terhadap *pixel-pixel* yang terkorupsi oleh *impulse noise* yaitu *pixel-pixel* yang mempunyai derajat keanggotaan lebih besar dari nol pada himpunan *fuzzy impulse noise*. *Pixel-pixel* yang lain yang tidak terkorupsi oleh *impulse noise* dibiarkan seperti semula. Berikut adalah contoh jika suatu *pixel* pada komponen yang berwarna merah terkorupsi *impulse noise* ($\mu^{R}_{Impulse}\left(A_R(i,j)\right) > 0$) maka terdapat beberapa kasus;

Kasus 1

Jika komponen warna hijau dan biru tidak terkorupsi oleh *impulse noise* yaitu $\mu^G\left(A_G(i,j)\right) = \mu^B\left(A_B(i,j)\right) = 0$, maka estimasi *pixel* pada komponen warna merah setelah proses filter yang disimbolkan dengan $F_R(i,j)$ adalah

$$F_R(i,j) = \frac{1}{2}\left(A_G(i,j) + \Delta_{RG}(i,j)\right) + \frac{1}{2}\left(A_B(i,j) + \Delta_{RB}(i,j)\right) \quad (3.23)$$

Kasus 2

Jika komponen warna hijau terkorupsi *impulse noise* dan komponen warna biru tidak terkorupsi oleh *impulse noise*, yaitu $\left(\mu^G\left(A_G(i,j)\right) > 0\right)$ dan $\mu^B\left(A_B(i,j)\right) = 0$, maka estimasi *pixel* komponen warna merah setelah proses filter adalah

$$F_R(i,j) = A_B(i,j) + \Delta_{RB}(i,j) \quad (3.24)$$

Kasus 3

Jika komponen warna biru terkorupsi *impulse noise* dan komponen warna hijau tidak terkorupsi oleh *impulse noise*, yaitu $\left(\mu^G\left(A_G(i,j)\right) = 0\right)$ dan $\mu^B\left(A_B(i,j)\right) > 0$, maka estimasi *pixel* komponen warna merah setelah proses filter adalah

$$F_R(i,j) = A_G(i,j) + \Delta_{RG}(i,j) \quad (3.25)$$

Kasus 4

Jika komponen warna biru dan hijau terkorupsi oleh *impulse noise*, yaitu $\left(\mu^G\left(A_G(i,j)\right) > 0\right)$ dan $\mu^B\left(A_B(i,j)\right) > 0$, maka estimasi *pixel* komponen warna merah setelah proses filter adalah

$$F_R(i,j) = \frac{\sum_{k=i-N}^{i+N}\sum_{l=j-N}^{j+N} A_R(k,l)\times\left(1-\mu_R\left(R(k,l)\right)\right)}{\sum_{k=i-N}^{i+N}\sum_{l=j-N}^{j+N}\left(1-\mu_R\left(R(k,l)\right)\right)} \quad (3.26)$$

$\Delta_{RB}(i,j)$ dan $\Delta_{RG}(i,j)$ adalah estimasi intensitas perbedaan komponen warna, yang dicari dengan persamaan berikut;

$$\Delta_{RB}(i,j) = \frac{\sum_{k=i-N}^{i+N}\sum_{l=j-N}^{j+N} RB(k,l)\times\left(1-\mu_{RB}\left(RB(k,l)\right)\right)}{\sum_{k=i-N}^{i+N}\sum_{l=j-N}^{j+N}\left(1-\mu_{RB}\left(RB(k,l)\right)\right)} \quad (3.27)$$

$$\Delta_{RG}(i,j) = \frac{\sum_{k=i-N}^{i+N}\sum_{l=j-N}^{j+N} A_{RG}(k,l)\times\left(1-\mu_{RG}\left(RG(k,l)\right)\right)}{\sum_{k=i-N}^{i+N}\sum_{l=j-N}^{j+N}\left(1-\mu_{RG}\left(RG(k,l)\right)\right)} \quad (3.28)$$

Iterasi yang pertama menggunakan jendela persekitaran dengan ukuran 3×3 $(N = 1)$. Proses iterasi selanjutnya dilakukan dengan mengubah ukuran dari jendela persekitarannya. Sebagai contoh untuk iterasi kedua maka ukuran jendela persekitaran yang digunakan adalah 5×5 $(N = 2)$, dan seterusnya.

b. Iterasi Selanjutnya

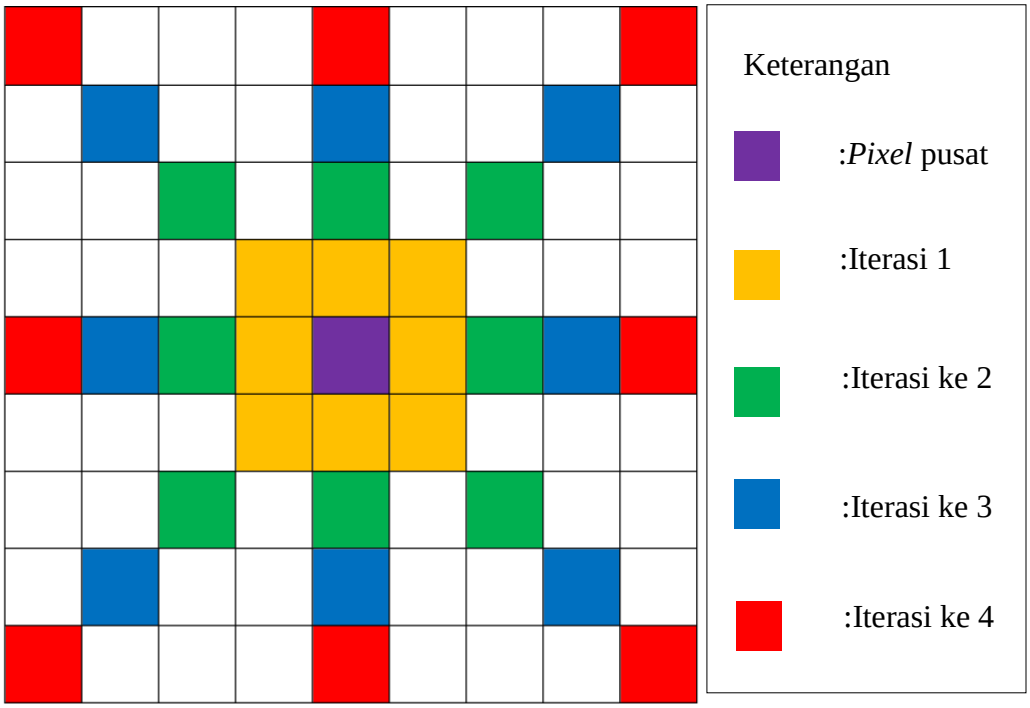
Hasil yang diperoleh dari iterasi pertama masih memungkinkan adanya *pixel* tertentu yang terkorupsi oleh *impulse noise*. Sehingga dibutuhkan iterasi selanjutnya untuk mereduksi *impulse noise* yang terjadi pada *pixel* tersebut.

Iterasi berikutnya didasarkan pada iterasi pertama, perbedaanya terletak pada ukuran persekitaran yang digunakan. Dalam hal ini ukuran persekitaran yang digunakan dalam proses iterasi berbetuk $(2N + 1 \times 2N + 1)$, maka iterasi pertama menggunakan $N = 1$ sebab iterasi pertama menggunakan ukuran persekitaran (3×3). Iterasi berikutnya akan menggunakan persekitaran dengan $N = 2,3,4,$ dst. yaitu berukuran $5 \times 5, 7 \times 7, 9 \times 9,$ dan seterusnya. Ilustrasi dari persekitaran yang dibentuk untuk proses iterasi berikutnya dapat dilihat pada Gambar 3.7 Disamping itu, parameter a_k, b_k, c_k

dan d_k yang digunakan untuk menentukan himpunan *fuzzy* pada Persamaan 3.10-3.13 diganti dengan parameter berikut;

$$a_k^e = \frac{1}{2}(a_k^{e-1} + p_k), \quad b_k^e = \frac{1}{2}(b_k^{e-1} + p_k) \tag{3.29}$$

$$c_k^e = \frac{1}{2}(c_k^{e-1} + p_k) \quad d_k^e = \frac{1}{2}(d_k^{e-1} + p_k) \tag{3.30}$$



Gambar 3.7 Persekitaran yang Digunakan untuk Iterasi Pertama, Kedua, Ketiga dan Keempat.

c. Kriteria Berhenti (*Stopping Criteria*)

Proses iterasi harus dihentikan sebab jika tidak akan menyebabkan proses filter berjalan lama, oleh karenanya dibutuhkan kriteria berhenti (*stopping criteria*). Berikut ini adalah kriteria-kriteria yang dapat menyebabkan berhentinya proses iterasi. Misalkan $\#_e^R$ menyatakan banyaknya nilai *pixel* yang terdapat pada himpunan *fuzzy impulse noise*

pada iterasi ke $e, (e \geq 2)$ untuk komponen warna merah, maka iterasi akan berhenti jika dipenuhi salah satu dari kriteria berikut;

a. Jika $\#_e^R = 0$ untuk sebarang $e \geq 2$

   Artinya tidak ada *pixel* yang terdapat pada himpunan *fuzzy impulse noise*, hal ini mengindikasikan bahwa sudah tidak terdapat *impulse noise* pada citra.

b. Jika $\#_{e-1}^R = \#_e^R$,

   Hal ini mengindikasikan bahwa proses iterasi sudah mencapai batas maksimal, karena sudah tidak dapat mengurangi jumlah *pixel* yang terkena *impulse noise* (terdapat pada himpunan *fuzzy impulse noise*).

c. Jika $\#_1^R \geq \#_2^R \geq \#_3^R \geq \cdots \geq \#_{e-1}^R \geq \#_e^R$, maka dicari nilai $\Delta_e = \#_{e-1}^R - \#_e^R$, yang cukup kecil. Sehingga dapat diasumsikan bahwa hasil terakhir sudah sesuai, dalam hal ini dapat diilih nilai Δ_e tertentu sehingga jika sudah dipenuhi nilai itu maka iterasi akan dihentikan.

B. Aplikasi Metode untuk Citra Fotografi

Metode reduksi *noise* yang sudah dipaparkan di atas akan diaplikasikan pada beberapa citra fotografi dengan menggunakan tingkat *noise* yang beragam. Tujuannya adalah untuk mengetahui tingkat efektifitas metode ini jika diterapkan untuk mereduksi *impulse noise*. Adapun parameter yang digunakan untuk mengukur tingkat efektifitas dari metode ini adalah MSE dan PSNR, seperti yang sudah dijabarkan pada bab II, semakin kecil nilai MSE dan semakin besar nilai PSNR maka citra akan semakin mirip dengan aslinya.

Citra yang akan digunakan sebagai sampel adalah citra yang biasa muncul di majalah *National Geographic* yang merupakan hasil dari fotografer profesional. Berikut adalah citra yang akan digunakan untuk menguji metode tersebut;

(a) (b)

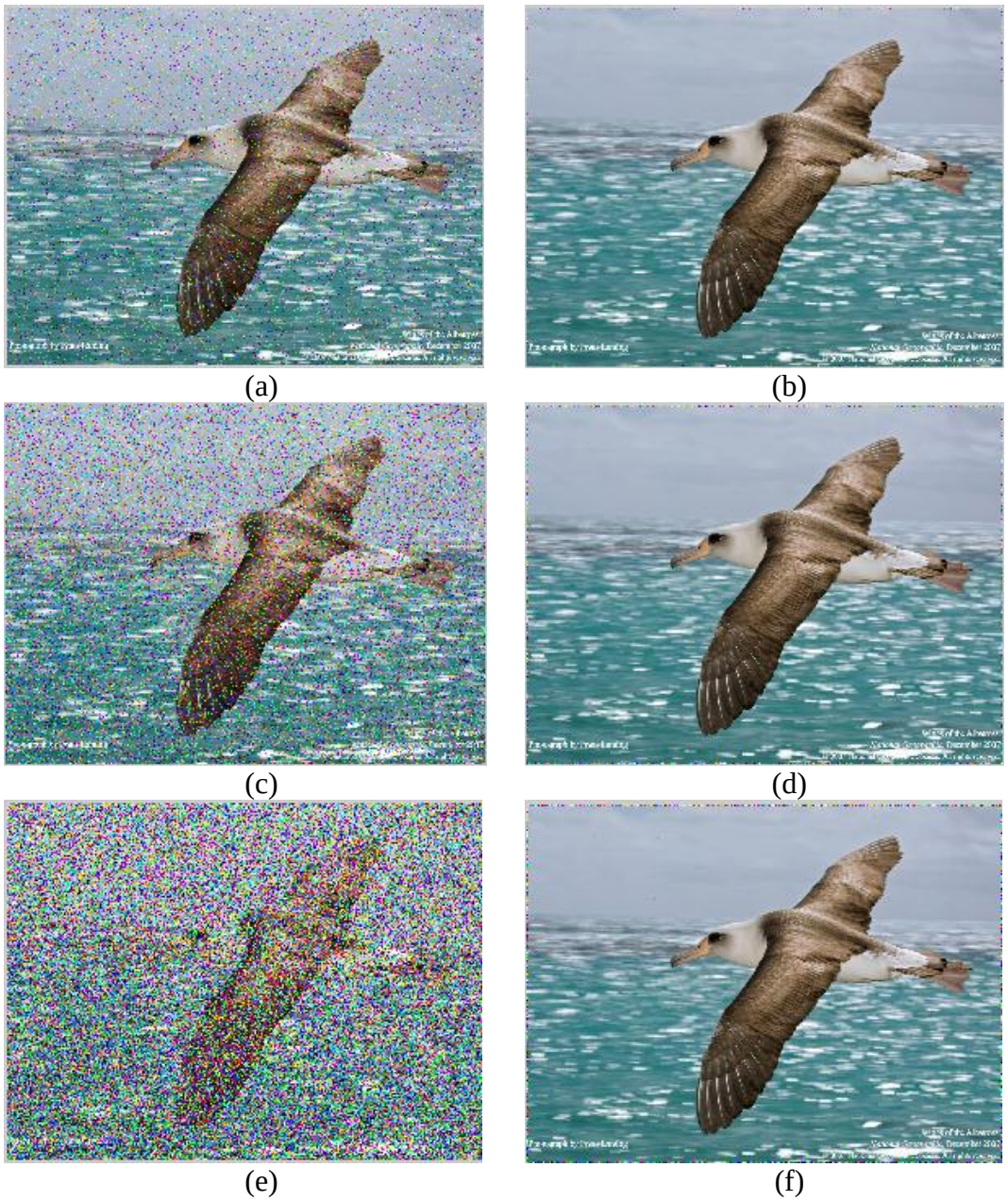
Gambar 3.8 Citra yang Akan Digunakan (a). Citra "*Wings of The Albatross*.jpg", (b). Citra "*African Wildlife Haven*.jpg" (diakses dari http://national-geographic-wallpapers.en.lo4d.com/details).

Berikut adalah hasil yang ditampilkan dalam bentuk tabel yang diperoleh dengan mengeksekusi *script* m-file yang terdapat pada Lampiran II yang diterapkan pada citra pertama yaitu "*Wings of The Albatross*.jpg". *Script* m-file tersebut dibuat berdasarkan fungsi-fungsi yang dibuat oleh Stefan Sculte, P. hD., penulis jurnal yang dibahas dalam skripsi ini.

Tabel 3.2 Hasil MSE, PSNR dengan metode FTSFC dari Citra "*Wings of The Albatross*.jpg" dengan Beragam Prosentase *Impulse Noise*

	Prosentase *Impulse Noise*							
	3%	5%	10%	15%	20%	30%	40%	50%
MSE	0.26	0.39	0.82	1.16	1.86	3.50	7.00	13.01
PSNR	53.92	52.20	49.02	47.48	45.44	42.69	39.68	36.99

Di bawah ini adalah sampel gambar-gambar citra yang terkena *impulse noise* dan yang sudah direduksi dengan menggunakan metode *fuzzy two step filter*, hasil selengkapnya mengenai output dari *script* m-file terdapat pada Lampiran III.



(a) (b) (c) (d) (e) (f)

Gambar 3.9. Citra Ber*noise* dan Citra Hasil Filter dengan FTSFC untuk Berbagai Tingkatan *Impulse Noise* (a) *Impulse Nosie* 5%, (b).Hasil Filter *Impulse Noise* 5 %, (c). *Impulse Noise* 20%, (d).Hasil Filter *Impulse Noise* 20%, (e). *Impulse Noise* 50% dan (f). Hasil Filter *Impulse Noise* 50%.

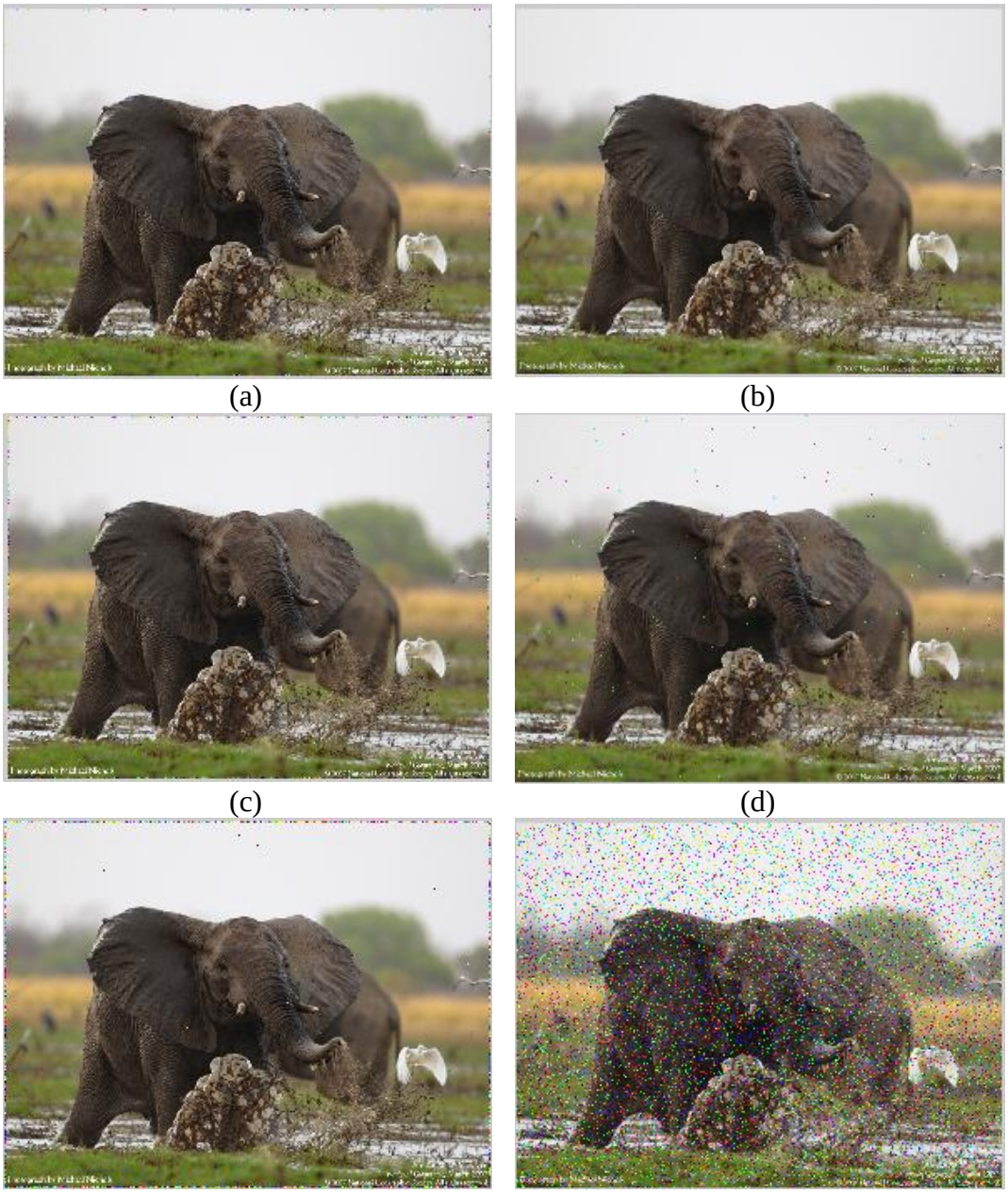
Selanjutnya akan dibandingkan hasil yang diperoleh dengan menggunakan metode FTSFC dan dengan salah satu filter linear. Secara teori seharusnya hasil yang diperoleh dengan menggunakan filter non linear akan lebih baik dibandiangkan dengan filter linear. Filter linear yang akan digunakan sebagai pembanding adalah Median filter. Berikut adalah hasil yang diperoleh dengan menggunakan *script* m-file pada Lampiran II dan *script* m-file pada Lampiran IV dengan sampel gambar yang digunakan adalah "*African Wildlife Haven*.jpg".

Tabel 3.3. MSE dan PSNR dari Citra yang Direduksi dengan Metode FTSFC dan Median Filter dari Citra "*African Wildlife Haven*.jpg" dengan Beragam Prosentase *Impulse Noise.*

Prosentase *Impulse Noise*							
		3%	5%	10%	20%	35%	50%
Median Fiter	MSE	40.8627	42.3831	49.4730	100.69298	570.5652	2247.5962
	PSNR	32.0175	31.8588	31.1871	28.1008	20.5677	14.6136
FTSFC	MSE	0.3756	0.4991	0.8413	2.1266	6.2554	19.6878
	PSNR	52.3827	51.1493	48.8809	44.8539	40.1682	35.1888

Hasil yang diperoleh di atas menunjukkan bahwa filter non linear lebih baik dalam mereduksi *impulse noise*, terbukti dengan lebih kecilnya MSE dari FTSFC dibandingkan dengan MSE dari Median Filter dan lebih besarnya PSNR dari FTSFC dibandingkan dengan PSNR dari Median Filter.

Berikut ini adalah gambar dari citra yang sudah direduksi *impulse noise*-nya dengan menggunakan metode FTSFC dan Median Filter.



(a) (b)

(c) (d)

(e) (f)

Gambar 3.10 Citra Hasil Filter dengan Metode Median Filter dan FTSFC untuk Berbagai Tingkatan *Impulse Noise* (a). FTSFC dengan *Impulse Noise* 5 %, (b). Median Filter dengan *Impulse Nosie* 5%, (c). FTSFC dengan *Impulse Noise* 20%, (d). Median Filter dengan *Impulse Noise* 20%, (e). FTSFC dengan *Impulse Noise* 50%, dan (f). Median filter dengan *Impulse Noise* 50%.

Berdasarkan gambar di atas terlihat bahwa hasil reduksi dengan menggunakan FTSFC lebih baik dalam dua hal yang pertama adalah lebih banyak

mereduksi *pixel* yang terkena *noise* dan yang kedua tidak menyebabkan efek *blurring* pada gambar. Meskipun demikian metode FTSFC juga masih memiliki kelemahan, jika diperhatikan citra yang dihasilkan setelah reduksi dengan menggunakan metode FTSFC bagian tepinya masih terdapat *impulse noise*. Hal ini mungkin disebabkan karena *pixel* yang berada pada bagian tepi citra tidak diikutkan dalam proses deteksi *noise* ataupun proses filter.

BAB IV

PENUTUP

A. KESIMPULAN

Berdasarkan pembahasan yang sudah dilakukan pada bab III diperoleh beberapa kesimpulan, diantaranya;

1. Prosedur atau tahapan pengunaan metode *fuzzy two step filter* untuk mereduksi *impulse noise* pada citra beritpe RGB adalah sebagai berikut;
   a. Menentukan *pixel-pixel* yang terkorupsi *impulse noise* dengan cara sebagai berikut;
      1) Mencari nilai gradien *fuzzy* dari setiap *pixel* dalam citra dengan menggunakan aturan *fuzzy* pertama yang didasarkan pada nilai gradien pertama, kedua dan ketiga.
      2) Mencari *pixel* yang terkena *fixed impulse noise* dengan menerapkan aturan *fuzzy* kedua, bagian ini disebut sebagai deteksi lokal. Pengembangan dari deteksi lokal adalah deteksi global. Deteksi global ditujukan untuk mengembangkan metode agar lebih akurat dalam menentukan *pixel* yang terkorupsi oleh *impulse noise* dan juga memungkinkan untuk mengembangkan metode agar dapat mendeteksi *impulse noise* yang bercampur dengan noise tipe lainya (*mixed impulse noise*) atau *random impulse noise*

3) Menentukan himpunan *fuzzy impulse noise* untuk ketiga komponen warna kemudian menentukan *pixel-pixel* yang mempunyai derajat keanggotaan tidak nol pada himpunan ini.

b. Menerapkan proses filter terhadap *pixel* yang terdeteksi terkorupsi *impulse noise*, yaitu pixel yang mempunyai derajat keanggotaan tidak nol pada himpunan *fuzzy impulse noise*. Kemudian menghentikan proses filter (iterasi) jika sudah diperoleh hasil yang diinginkan.

2. Berdasarkan hasil yang diperoleh pada bagian aplikasi, metode *fuzzy two - step filter* (FTSFC) mempunyai tingkat keakuratan yang baik dalam mereduksi *impulse noise* pada citra hasil fotografi dengan berbagai tingkat prosentase *impulse noise* yang diberikan. Hal ini dibuktikan dengan kecilnya nilai MSE dan besarnya nilai PSNR yang dihasilkan. Metode ini juga lebih baik jika dibandingkan dengan metode filter linear, misalnya metode Median filter. Hal ini dibuktikan dengan lebih kecilnya MSE dan lebih besarnya PSNR yang dihasilkan oleh metode FTSFC dibandingkan dengan metode Median filter yang merupakan salah satu filter linear.
3. Metode ini masih mempunyai kelemahan, yaitu *pixel* yang berada pada bagian tepi citra yang terkorupsi oleh *impulse noise* tidak dapat tereduksi. Hal ini mungkin disebabkan karena *pixel-pixel* yang berada pada bagian tepi citra tidak diikutkan dalam tahap deteksi maupun tahap filter.

B. SARAN

Berdasarkan hasil-hasil yang sudah diperoleh pada skripsi ini tentang metode *fuzzy two - step filter*, penulis menyarankan;

1. Coba terapkan metode ini pada citra yang terkena *random impulse noise* atau *mixed impulse noise* dan analisis hasil yang diperoleh.
2. Cari metode filter non linear yang lain, baik yang sudah menggunakan *fuzzy* atau yang belum kemudian bandingkan hasil yang diperoleh dengan metode ini. Dengan demikian maka dapat ditentukan metode yang lebih banyak mereduksi *noise*. Analisis kelebihan dan kekurangannya dan coba untuk mengabungkan kedua metode tersebut.
3. Mengembangkan metode filter non linear *fuzzy two-step filter*, dengan mengubah fungsi keanggotaan dan operasi pada himpunan *fuzzy* yaitu $t-norm$ dan $s-norm$, dengan demikian maka dimungkinkan akan diperoleh hasil yang lebih baik yaitu diperolehnya PSNR yang lebih besar.
4. Metode ini mungkin juga dapat dikembangkan dengan cara tidak mengabaikan *pixel* yang ada pada bagian tepi namun dengan menambahkan *pixel* nol pada bagian tepi citra. Hal ini memungkinkan *pixel* yang berada pada tepi citra ikut dalam proses *filtering* sehingga *impulse noise* yang terletak pada bagian tepi citra dapat tereduksi.

DAFTAR PUSTAKA

Arakawa, K. Median filter based on fuzzy rules and its application to image restoration, *Fuzzy Sets Syst.*, vol. 77, Hlm. 3–13.

Cornelis, C., Deschrijver, G., and Kerre, E. 2002. Classification of intuitionistic fuzzy implicators : An Algebraic Approach, in Proc. 6th Joint Conf. Information sciences. Hlm. 105-108.

Gonzalez, Rafael C., and Woods, Richard E. 2002. *Digital Image Processing: 2nd Edition*. New Jersey: Prentice Hall

Haris, Papasaika-Hanusch. *Digital Image Processing Using Matlab*. Swiss: Institute of Geodesy and Photogrammetry, ETH Zurich.

Hussain, A., Bhatti, S. M., and Jaffar M. A. 2012. Fuzzy based Impulse Noise Reduction Method. *Springer*. No 60. Hlm. 551-571.

Kalaykov, I. dkk, Eds., Real-time image noise cancellation based on fuzzy similarity, in *Fuzzy Filters for Image Processing*, 1st ed. Heidelberg, Germany: Physica Verlag, vol. 122, Hlm. 54–71.

Kaur, J., Kaur, P., and Kaur, P. 2012. Review of impulse noise reduction technique using fuzzy logic for image processing. *International Journal of Engineering Research & Technology (IJERT)*. Vol. 1 Issue 5. July – 2012.

Klir, George J., Clair, Ute St, and Yuan, Bo. 1995. *Fuzzy Set and Fuzzy Logic: theory and applications*. USA :Prentice Hall International

_____, 1997. *Fuzzy Set Theory: Foundation and applications*. USA :Prentice Hall International.

Koli, Manohar Annappa. 2012. Review of Impulse Noise Reduction Techniques. *International Journal on Computer Science and Engineering (IJCSE)*. Vol. 4 No. 02. Hlm. 184-196. Februari 2012.

Krishnan, M. H., and Viswanathan, R. 2013. A New Concept of Reduction of Gaussian Noise in Image Based on Fuzzy Logic. *Applied Mathematical Science*. Vol.7, no. 12, Hlm. 595-602.

Kusumadewi, S., Hartati, S., Harjoko, A., dan Wardoyo, R. (2006), *Fuzzy Multi-Attribute Decision Making (Fuzzy MADM),*Yogyakarta: Graha Ilmu.

Lee, C. S. dan Kuo, Y. H. 2000. Adaptive fuzzy filter and its application to image enhancement, in *Fuzzy Techniques in Image Processing*, 1st ed. Heidelberg, Germany:Physica Verlag, vol. 52, Hlm. 172–193.

Lotfi, Ehsan. 2013. An Adaptive fuzzy filter for Gaussian Noise Reduction using Image Histrogram Estimation. *Advanced in Digital Multimedia (ADMM)*. Vol 1, no 4. Hlm. 190-193.

McAndraw, Alasdair. 2004. *An Introduction to Digital Image Processing with MATLAB.* School of Computer Science and Mathematics: Victoria University Technology.

Munir, Rinaldi. 2004. *Pengolahan Citra Digital*. Jakarta.

Russo, F. dan Ramponi, G. 1996. Removal of impulse noise using a FIRE filter. Prosiding ke 3, *IEEE international Conference Image Processing*, vol 1. Hlm. 975-978.

Schulte, Stefan, dkk. 2006. Fuzzy Two-Step Filter for Impulse Noise Reduction From Color Image. *IEEE Transaction on Image Processing*. vol 15. no 11. Hlm. 3567-3578

_____, 2006. A Fuzzy Impulse Noise Detection and Reduction Method. *IEEE Transaction on Image Processing*. vol. 15. no. 5. Hlm. 1153–1162. May 2006.

Susilo, Frans. 2006. *Himpunan & Logika Kabur serta aplikasinya.* Yogyakarta: Graha Ilmu.

Wang, Li-Xin. 1997. *A Course in Fuzzy System and Control*. USA: Prentice Hall International.

Wati, Dwi Ana Ratna. 2011. *Sistem Kendali Cerdas*. Yogyakarta: Graha Ilmu.

Xu, H., Zhu, G., Peng, H., dan Wang, D. 2004. Adaptive fuzzy switching filter for images corrupted by impulse noise, *Pattern Recognit. Lett.*, vol. 25, Hlm. 1657–1663.

Zimmermann. 1991. *Fuzzy Sets Theory and Its Applications*. Edisi 2. Kluwer Academic Publishers. Massachussets

Zlokolica, V., and Philips, W. 2004. Motion and detail adaptive denoising of video. *IS&T/SPIE symp. Electronic Imaging*. Hlm. 1417-1421.

http://national-geographic-wallpapers.en.lo4d.com/details, diakses pada tanggal 26 maret 2013 pukul 23.35

http://www.imageprocessingplace.com/DIP-_3E/dip3e_book_images_downloads.htm, diakses pada tanggal 4 maret 2013 pukul 09.15

LAMPIRAN I

Script m-file untuk menghitung MSE (*Mean Square Error*) dari dua citra bertipe RGB.

```
%++++++++++++++++++++++++++++++++++++++++++++++++++++++
% Fungsi untuk menghitung MSE dari citra bertipe RGB
%++++++++++++++++++++++++++++++++++++++++++++++++++++++
function hasil = MSECOL(A,B);
[mA nA,d] = size(A); % ukuran citra A
[mB nB,d] = size(B); % ukuran citra B
A = double(A); % ubah tipe A menjadi double
B = double(B); % ubah tipe B menjadi double
if (mA ~= mB | nA ~= nB) % jika ukuran A dan B tidak sama
error('Arrays A dan B harus mempunyai ukuran yang sama');
end;
if (d ~= 3) % jika citra bukan jenis RGB
error('Hanya untuk citra tipe RGB, cari citra lain');
end;
hasil1 = 0;
hasil2 = 0;
hasil3 = 0;
for i = 1:mA
for j = 1:nA
hasil1 = hasil1 + (A(i,j,1) - B(i,j,1))^2;
hasil2 = hasil2 + (A(i,j,2) - B(i,j,2))^2;
hasil3 = hasil3 + (A(i,j,3) - B(i,j,3))^2;
end;
end;
hasil1 = hasil1 / ((mA)*(nA));
hasil2 = hasil2 / ((mA)*(nA));
hasil3 = hasil3 / ((mA)*(nA));
hasil = (hasil1+hasil2+hasil3)/3.0; % Hasil akhir
```

LAMPIRAN II

Script m-file `Hasil_akhir.m`, digunakan untuk menghitung hasil dari metode *fuzzy two-step filter*, beserta fungsi-fungsi yang membangunnya. Hasil yang diperoleh pada lampiran ini didasarkan pada fungsi-fungsi yang dibuat oleh Stefan Schulte, Ph.D., Valérie De Witte,Ph.D , Mike Nachtegael, Ph.D., Dietrich Van Der Weken, Ph.D., and Prof Dr. Etienne E. Kerre, M.Sc. penulis jurnal "*Fuzzy Two-Step Filter for Impulse Noise Reduction From Color Images*".

1. *Script* m-file `Hasil_akhir.m`

```
disp('============================================================')
disp('Fuzzy Two Step Filter to Reduction Impulse Noise of Color
Image')
disp('============================================================')
A=input('Masukan Citra yang akan direduksi: ');
a=input('Masukan Prosentase impulse noise: ');
B=imnoise(A, 'salt & pepper', a);
Hasil_fixed=FIDRMC(B,A);
Hasil_fixed=uint8(Hasil_fixed);
A=uint8(A);
figure, subplot(1,3,1), imshow(A),title('Citra Asli'),
subplot(1,3,2),imshow(B),title('Citra bernoise'),subplot(1,3,3),
imshow(Hasil_fixed),title('Citra hasil filter fixed'),
clear all
```

2. *Script* m-file `FIDRMC.m`

```
%*****************************************************************
% Fuzzy Impulse Noise Detection and Reduction Method for Colour
Images
%  The paper of the FIDRMC is proposed in:
%  Stefan Schulte, Valérie De Witte, , Mike Nachtegael
%  Dietrich Van Der Weken and  Etienne E. Kerre:
%  A Fuzzy Two-step Filter for Impulse Noise Reduction From Colour
Images,
%  IEEE Transactions on Image Processing 15(11), 2006, 3567 - 3578
% Stefan Schulte (stefan.schulte@Ugent.be):
% Last modified: 15/01/06
% Inputs:  A = the noisy input image
%          O = the original noise-free image
% Outputs:  Fs = the filtered image
%*****************************************************************
function Fs = FIDRMC(A,O)
```

```
[M,N,DIM] = size(A);
A = double(A);
A1 = A(:,:,1);
A2 = A(:,:,2);
A3 = A(:,:,3);

F1 = double(A1);
F2 = double(A2);
F3 = double(A3);

flag = 0; loop = 0;
while (flag == 0)
    fixed1 = Detection(A1);
    if ((fixed1(1,1) ~= -1) || (loop > 4))
        flag = 1;
    else
        loop = loop + 1;
    end
end
flag = 0; loop = 0;
while (flag == 0)
    fixed2 = Detection(A2);
    if ((fixed2(1,1) ~= -1) || (loop > 4))
        flag = 1;
    else
        loop = loop + 1;
    end
end
flag = 0; loop = 0;
while (flag == 0)
    fixed3 = Detection(A3);
    if ((fixed3(1,1) ~= -1) || (loop > 4))
        flag = 1;
    else
        loop = loop + 1;
    end
end

Windowsize = 1;
[F1,F2,F3] =
DenoiseFIDRMC(A1,A2,A3,fixed1,fixed2,fixed3,Windowsize);

Fs(:,:,1) = F1;
Fs(:,:,2) = F2;
Fs(:,:,3) = F3;
if ((fixed1(1,1)== -1)&&(fixed2(1,1)== -1)&&(fixed3(1,1)==-1))
   disp(sprintf('======================'));
   Fs = FIDRMC(A,O);
   if (MSEC(Fs,O,2) > 800)
       Fs = FIDRMCRANDOM(Fs,O);
   end
   disp(sprintf(' Random-Valued Impulse Noise'));
   disp(sprintf('  => Output MSE:  %10.5f',MSEC(Fs,O,2)));
   disp(sprintf('  => Output PSNR:
   %10.5f',(log(255^2/MSEC(Fs,O,2))/log(10)*10)));
```

```
      disp(sprintf('======================'));
   else
      disp(sprintf('======================'));
      disp(sprintf(' Fixed-Valued Impulse Noise'));
      disp(sprintf('  => Output MSE:  %10.5f',MSEC(Fs,O,2)));
      disp(sprintf('  => Output PSNR:
      %10.5f',(log(255^2/MSEC(Fs,O,2))/log(10)*10)));
      disp(sprintf('======================'));
   end
```

3. Fungsi `detection.mexw32`, dibuat dengan bahasa `C`.

```
/*****************************************************************
% Fuzzy Fixed-valued Impulse Noise Detection Method for Colour
Images
%  The paper of the FIDRMC is proposed in:
%  Stefan Schulte, Valérie De Witte, , Mike Nachtegael
%  Dietrich Van Der Weken and  Etienne E. Kerre:
%  A Fuzzy Two-step Filter for Impulse Noise Reduction From Colour
Images,
%  IEEE Transactions on Image Processing 15(11), 2006, 3567 - 3578
% Stefan Schulte (stefan.schulte@Ugent.be):
% Last modified: 15/01/06
%*****************************************************************/
#include "mex.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

double LARGE (double x, double p1, double p2) {
   double res = 0.0;
   if ((x > p1) && (x < p2))   res = (x-p1)/(p2-p1);
   else if (x > p2)            res = 1.0;
   else                        res = 0.0;
   return res;
}

double absol(double a) {
   double b;
   if(a<0)   b=-a;
   else   b=a;
   return b;
}

double minimum(double a, double b) {
   double x;
   if(a<=b)   x = a;
   else   x=b;
   return x;
}

double maximum(double a, double b) {
   double x;
   if(a<=b)   x = b;
   else   x=a;
```

```
   return x;
}

/*****************************************************************
*  The main program of the FIDRMC for colour images
*****************************************************************/
void callMem(double **A1,double *spv,int M, int N) {
   int i,j,k, loop;
   double *med, *histo,*sorted,*maxi;
   double som, tel, hlp,res, tmp, slope1, diff1, diff2, diff3;
   double l1, l2, l3, minslop1, minslop2, slope2;
   double a = 80.0, b = 160.0;
   int **xy, **adj;

   int rand1a = 0;
   int rand1b = 0;
   int rand2a = 0;
   int rand2b = 0;

   xy = malloc(9*sizeof(int));
   adj = malloc(9*sizeof(int));
   for(i=0;i<9;i++){
      xy[i] = malloc(2*sizeof(int));
      adj[i] = malloc(2*sizeof(int));
   }
   med = malloc(8*sizeof(double));
   histo = malloc(256*sizeof(double));
   sorted = malloc(256*sizeof(double));
   maxi = malloc(10*sizeof(double));

   xy[0][0] = -1; xy[0][1] = -1; xy[1][0] = -1; xy[1][1] = 0;
   xy[2][0] = -1; xy[2][1] =  1; xy[3][0] =  0; xy[3][1] = -1;
   xy[4][0] =  0; xy[4][1] =  0; xy[5][0] =  0; xy[5][1] = 1;
   xy[6][0] =  1; xy[6][1] = -1; xy[7][0] =  1; xy[7][1] = 0;
   xy[8][0] =  1; xy[8][1] =  1;

   adj[0][0] = 1; adj[0][1] = -1; adj[1][0] =  0; adj[1][1] = -1;
   adj[2][0] = 1; adj[2][1] =  1; adj[3][0] =  1; adj[3][1] =  0;
   adj[4][0] = 0; adj[4][1] =  0; adj[5][0] =  1; adj[5][1] =  0;
   adj[6][0] = 1; adj[6][1] =  1; adj[7][0] =  0; adj[7][1] = -1;
   adj[8][0] = 1; adj[8][1] = -1;


   for(i=2; i<M-2; i++){
      for(j=2; j<N-2; j++){
         /*-------------------------------------------
                Gradient values method
         -------------------------------------------*/
         tel = 0;
         for (k = 0; k<9; k++){
            if (k==4) continue;
            diff1 = A1[i+xy[k][1]][j+xy[k][0]] - A1[i][j];
            diff2 = A1[i+xy[k][1]+adj[k][1]][j+xy[k][0]+adj[k][0]]
                   - A1[i+adj[k][1]][j+adj[k][0]];
```

```
            diff3 = A1[i+xy[k][1]-adj[k][1]][j+xy[k][0]-adj[k][0]]
                   - A1[i-adj[k][1]][j-adj[k][0]];

            l1 = LARGE(absol(diff1),a,b);
            l2 = LARGE(absol(diff2),a,b);
            l3 = LARGE(absol(diff3),a,b);


            res = 1.0-  maximum( maximum(minimum(1-l1,1-l2),
                  minimum(1-l1,1-l3)),
                  maximum(minimum(l1,l2),minimum(l1,l3)));
            res = l1*res;

            for (loop = 0; loop<tel; loop++){
                if (res < med[loop]){
                   hlp = med[loop];
                   med[loop] = res;
                   res = hlp;
                }
            }
            med[(int)tel] = res;
            tel++;
         }
         if (minimum( minimum(med[6],med[7]),med[5]) >= 0.5)
            histo[(int)A1[i][j]]++;

      }
   }

   for (i=0; i<256; i++) sorted[i] =  histo[i];

   for (i=0; i<256; i++)
      for (j=i+1; j<256; j++)
         if(sorted[j]>sorted[i]){
            tmp = sorted[j];
            sorted[j] = sorted[i];
            sorted[i] = tmp;
         }

   som = 0.0;
   for (j = 0; j<10; j++)
      for (i = 0; i<256; i++){
         if (j==0) som +=  histo[i];
         if( histo[i]==sorted[j]) maxi[j] = i;
      }

   if ((sorted[0]/som) < 0.09){
    // printf("\t NO FIXED VALUE IMPULSE NOISE DETECTED USE
FIDRMCRANDOM %f \n",(sorted[0]/som) );
     spv[0] = -1;
   }
   else{
      minslop1 = 20;
      minslop2 = 20;
      for (i = 0; i<10; i++){
```

```
            slope1 = 0;
            for (j = maxi[i]+1; j< minimum(256,maxi[i]+31);j++){
               if( histo[j] <=  histo[(int)maxi[i]]*0.015) break;
               slope1++;
            }
            slope1 = minimum(slope1, minslop1);
            minslop1 = minimum(slope1,minslop1);

            slope2 = 0;
            for (j = maxi[i]-1; j >= maximum(0,maxi[i]-30);j--){
               if( histo[j] <=  histo[(int)maxi[i]]*0.015) break;
               slope2++;
            }
            slope2 = minimum(slope2,minslop2);
            minslop2 = slope2;
            spv[i+0*11] = maxi[i];
            spv[i+1*11] = maxi[i] - minimum(slope2,30)-
                          (minimum(slope2,30)*(1.0/3.0));

            spv[i+2*11] = maxi[i] - minimum(slope2,30);
            spv[i+3*11] = maxi[i] + minimum(slope1,30);
            spv[i+4*11] = maxi[i] +
                         minimum(slope1,30)+(minimum(slope1,30)*(1.0
                         /3.0));
         }

            spv[10+0*5] = 0;
         for (i =0; i<10; i++){
            if ((sorted[i]/som) < 0.005) break;
            spv[10+0*5] = spv[10+0*5] + 1;
         }
      }
      for(i=0;i<9;i++)  free(xy[i]);
      free(xy);
      for(i=0;i<9;i++)  free(adj[i]);
      free(adj);
      free(med);
      free(histo);
      free(sorted);
      free(maxi);

   }  /* End of callFuzzyShrink */

   #define Im1      prhs[0]
   #define OUT plhs[0]

   /***  The interaction with Matlab (mex):
   *        nlhs = amount of output arguments (= 1)
   *        nrhs = amount of input arguments (= 1)
   *     *plhs[] = link to the output
   *     *prhs[] = link to the input
   **/
   void mexFunction( int nlhs, mxArray  *plhs[], int nrhs, const
   mxArray  *prhs[] ) {
       int row, col, i, M, N;
```

```
    double **spv, **A1;

    if (nlhs!=1)
    mexErrMsgTxt("It requires one output arguments only [M1].");
    if (nrhs!=1)
    mexErrMsgTxt("this method requires one input argument [Im1]");

    /* Get input values */
    M = mxGetM(Im1);
    N = mxGetN(Im1);

    /**
    * Allocate memory for return matrices
    **/
    OUT = mxCreateDoubleMatrix(11, 5, mxREAL);
    spv = mxGetPr(OUT);

    /**
    * Dynamic allocation of memory for the input array
    **/
    A1 = malloc(M*sizeof(int));
    for(i=0;i<M;i++)
      A1[i] = malloc(N*sizeof(double));

     /**
     * Convert ARRAY_IN and INPUT_MASK to 2x2 C arrays (MATLAB
stores a two-dimensional matrix
     * in memory as a one-dimensional array)
     ***/
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A1[row][col] = (mxGetPr(Im1))[row+col*M];
          }

    /* Call the main FIDRMC function */
    callMem(A1,spv,M,N);

    for(i=0;i<M;i++)  free(A1[i]);
    free(A1);
}
/* end mexFcn*/
```

4. Fungsi `DenoiseFIDRMC.mexw32` dibuat denga bahasa C

```
/*****************************************************************
% Fuzzy Fixed-valued Impulse Noise Reduction Method for Colour
Images
%  The paper of the FIDRMC is proposed in:
%  Stefan Schulte, Valérie De Witte, , Mike Nachtegael
%  Dietrich Van Der Weken and  Etienne E. Kerre:
%  A Fuzzy Two-step Filter for Impulse Noise Reduction From Colour
Images,
%  IEEE Transactions on Image Processing 15(11), 2006, 3567 - 3578
% Stefan Schulte (stefan.schulte@Ugent.be):
```

```
% Last modified: 15/01/06
%*************************************************************/
#include "mex.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

double LARGE (double x, double p1, double p2) {
   double res = 0.0;
   if ((x > p1) && (x < p2))   res = (x-p1)/(p2-p1);
   else if (x > p2)            res = 1.0;
   else                        res = 0.0;
   return res;
}

double absol(double a) {
   double b;
   if(a<0)   b=-a;
   else   b=a;
   return b;
}

double minimum(double a, double b) {
   double x;
   if(a<=b)   x = a;
   else   x=b;
   return x;
}

double maximum(double a, double b) {
   double x;
   if(a<=b)   x = b;
   else   x=a;
   return x;
}

double NOISE(double x, double ** noise, int amount) {
   double ret, tmp;
   int i;
   ret = 0.0; tmp = 0.0;

   for (i = 0; i<amount; i++){
      if((x == noise[i][0]) || ((x < noise[i][0]) && (x >=
        noise[i][1])) || ((x > noise[i][0]) && (x<=noise[i][3])))
          tmp = 1.0;
      else if((x<noise[i][1]) && (x>noise[i][2]))
             tmp =((noise[i][1]-x)/(noise[i][1]-noise[i][2]));
      else if ((x>noise[i][3])&&(x<noise[i][4]))
             tmp = ((noise[i][4]-x)/(noise[i][4]-noise[i][3]));
      else tmp = 0.0;
      ret = maximum(ret,tmp);
   }
   return ret;
}
```

```
double Median(double * x, int len) {
   double tmp;
   int i,j;
   for (i = 0; i<len; i++){
       for (j = i+1; j<len; j++){
           if (x[j] < x[i]){
               tmp = x[i];
               x[i] = x[j];
               x[j] = tmp;
           }
       }
   }
   return x[(int)(len/2)];
}

double EQUALI(double x, double cen, double a, double b) {
   double res, diff;
   diff = absol(x-cen);
   if (diff <= a) res = 1.0;
   else if ((diff > a) && (diff <= b)) res = (b-diff)/(b-a);
   else res = 0;
   return res;
}

/*************************************************************
*  The main progam of the FIDRMC noise reduction method
*************************************************************/
void callDenoise(double **A1,double **A2,double **A3,double
**fixed1,double **fixed2,double **fixed3,double *filt1,double
*filt2,double *filt3,int M, int N,int W) {
   int i,j,k,l, amount1, amount2, amount3,it, loop;
   double som, wei;
   int *over_i, *over_j, OVER, OVER2, ok,teller;
   double noise1, noise2, noise3,cen1, cen2, cen3,glob;
   double noise1a, noise2a, noise3a, nR, nG, nB;
   double *RG, *RB, *GB, *NRG, *NRB, *NGB, *cor;
   double coR, coG, coB, soR, soG, soB, resR, resG, resB, resR2,
resG2, resB2;
   double centRG, centRB, centGB, cenRG, cenRB, cenGB, memRG,
memRB, memGB, mem ;
   double corr1, corr2;

   int rand1a = 0;
   int rand1b = 0;
   int rand2a = 0;
   int rand2b = 0;

   over_i = malloc(M*N*sizeof(double));
   over_j = malloc(M*N*sizeof(double));
   RG = malloc((2*W+1)*(2*W+1)*sizeof(double));
   RB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   GB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   NRG = malloc((2*W+1)*(2*W+1)*sizeof(double));
   NRB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   NGB = malloc((2*W+1)*(2*W+1)*sizeof(double));
```

```
cor = malloc((2*W+1)*(2*W+1)*sizeof(double));

for (i=0; i<(2*W+1)*(2*W+1); i++) {
    RG[i] = 0;  GB[i] = 0; RB[i] = 0; cor[i] = 0;
    NRG[i] = 0;  NGB[i] = 0; NRB[i] = 0;
}

for (i=0; i < M; i++)
      for (j=0; j < N; j++) {
          filt1[i+j*M] = A1[i][j];
          filt2[i+j*M] = A2[i][j];
          filt3[i+j*M] = A3[i][j];
       }
amount1 = fixed1[10][0];
amount2 = fixed2[10][0];
amount3 = fixed3[10][0];
it = 1;
OVER = 0;
OVER2 = 0;
while((it == 1)||(OVER != 0)) {
   // First iteration
   if(it==1){
       for(i=2; i<M-2; i++){
           for(j=2; j<N-2; j++){
                ok = 0;
               /* step 1. Determine the local window*/
               if(i < W) {
                  rand1a = i;
                  rand1b = W;
               }
               else {
                  if (i>M-W-1){
                      rand1a = W;
                      rand1b = M-i-1;
                  }
                  else{
                      rand1a = W;
                      rand1b = W;
                  }
               }
               if(j < W) {
                  rand2a = j;
                  rand2b = W;
               }
               else {
                  if (j > N-W-1){
                     rand2a = W;
                     rand2b = N-j-1;
                  }
                  else{
                     rand2a = W;
                     rand2b = W;
                  }
              }
              /* end step 1. */
```

```
cen1 = A1[i][j];
cen2 = A2[i][j];
cen3 = A3[i][j];
noise1a = NOISE(cen1, fixed1, amount1);
noise2a = NOISE(cen2, fixed2, amount2);
noise3a = NOISE(cen3, fixed3, amount3);


glob = minimum(minimum(noise1a,noise2a),noise3a);
/***********************************************
*     THE RED COMPONENT
***********************************************/
noise1 = NOISE(cen1, fixed1, amount1);
if(noise1 > 0){
   teller =0;
   for (k=i-rand1a; k<=i+rand1b; k++){
      for (l=j-rand2a; l<=j+rand2b; l++){
         nR = NOISE(A1[k][l], fixed1, amount1);
         nG = NOISE(A2[k][l], fixed2, amount2);
         nB = NOISE(A3[k][l], fixed3, amount3);

         RG[teller] = A1[k][l] - A2[k][l];
         RB[teller] = A1[k][l] - A3[k][l];

         NRG[teller] = 1.0 - maximum(nR,nG);
         NRB[teller] = 1.0 - maximum(nR,nB);
         cor[teller] = A1[k][l];
         teller++;
      }
   }
   // one of the three components is noise-free
   if (glob == 0) {
       if ((noise2a==0) && (noise3a==0)) {
           coG = 0; coB = 0; soG = 0; soB = 0;
           for (k=0; k<teller; k++){
               coG +=  NRG[k]*RG[k];
               coB +=  NRB[k]*RB[k];
               soG +=  NRG[k];
               soB +=  NRB[k];
           }
           if (soB == 0) {
               if (soG == 0) resR2 = cen1;
               else resR2 = minimum
                (maximum(cen2+(coG/soG),0),255);
           }
           else if (soG == 0) resR2 = minimum
           (maximum(cen3+(coB/soB),0),255);
           else  resR2 = minimum
           (maximum((((cen3+(coB/soB))+(cen2+(coG
           /soG)))/2.),0),255);
       }
       else if (noise2a==0) {
           coG = 0; soG = 0;
           for (k=0; k<teller; k++){
```

```
            coG +=  NRG[k]*RG[k];
            soG +=  NRG[k];
        }
        if (soG == 0) resR2 = cen1;
        else  resR2 = minimum
         (maximum(cen2+(coG/soG),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NRB[k]*RB[k];
            soB +=  NRB[k];
        }
        if (soB == 0) resR2 = cen1;
        else  resR2 = minimum
         (maximum(cen3+(coB/soB),0),255);
    }
}
else resR2 = -300;

cenRG = Median(RG,teller); cenRG = RG[2];
cenRB = Median(RB,teller); cenRB = RB[2];

centRG = absol(cen1-cen2);
centRB = absol(cen1-cen3);

memRG = EQUALI(centRG,cenRG,10.0,20.0);
memRB = EQUALI(centRB,cenRB,10.0,20.0);

corr1 = EQUALI(absol(cen1-
cor[2]),absol(cor[5]-cor[3]),10.0,20.0);
corr2 = EQUALI(absol(cen1-
cor[6]),absol(cor[5]-cor[3]),10.0,20.0);
mem = minimum(minimum(memRG,memRB),
maximum(maximum(corr1,corr2),1.0-glob));

if (resR2 == -300){
   som = 0.0; wei = 0.0;
   for (k=i-rand1a; k<=i+rand1b; k++){
      for (l=j-rand2a; l<=j+rand2b; l++){
      wei += 1.0 - NOISE(A1[k][l], fixed1,
                amount1);
      som += A1[k][l]*(1.0 -
            NOISE(A1[k][l],fixed1, amount1));
      }
    }

    if (wei == 0) resR = Median(cor,teller);
    else resR = (som/wei);
}
else resR = resR2;

noise1 = NOISE(resR, fixed1, amount1);
if ((noise1 > 0) && (ok==0)){
   over_i[OVER2] = i;
```

```
      over_j[OVER2] = j;
      OVER2++;

      ok = 1;
   }
}
else resR = cen1;

/*********************************************
*     THE GREEN COMPONENT
*********************************************/
noise2 = NOISE(cen2, fixed2, amount2);
if(noise2 > 0){
   teller =0;
   for (k=i-rand1a; k<=i+rand1b; k++){
      for (l=j-rand2a; l<=j+rand2b; l++){
         nR = NOISE(A1[k][l], fixed1, amount1);
         nG = NOISE(A2[k][l], fixed2, amount2);
         nB = NOISE(A3[k][l], fixed3, amount3);

         RG[teller] = A2[k][l] - A1[k][l];
         GB[teller] = A2[k][l] - A3[k][l];

         NRG[teller] = 1.0 - maximum(nR,nG);
         NGB[teller] = 1.0 - maximum(nG,nB);
         cor[teller] = A2[k][l];
         teller++;
      }
   }

   // one of the three components is noise-free
   if (glob == 0) {
       if ((noise1a==0) && (noise3a==0)) {
           coR = 0; soR = 0; coB = 0; soB = 0;
           for (k=0; k<teller; k++){
               coR +=  NRG[k]*RG[k];
               coB +=  NGB[k]*GB[k];
               soR +=  NRG[k];
               soB +=  NGB[k];
           }
           if (soB == 0) {
               if (soR == 0) resG2 = cen2;
               else resG2 = minimum
                (maximum(cen1+(coR/soR),0),255);
           }
           else if (soR == 0) resG2 = minimum
           (maximum(cen3+(coB/soB),0),255);
           else  resG2 = minimum (
           maximum(((cen3+(coB/soB))+(cen1+(coR/s
           oR)))/2.0 ,0) ,255);
       }
       else if (noise1a==0) {
           coR = 0; soR = 0;
           for (k=0; k<teller; k++){
               coR +=  NRG[k]*RG[k];
```

```
            soR +=  NRG[k];
        }
        if (soR == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen1+(coR/soR),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NGB[k]*GB[k];
            soB +=  NGB[k];
        }
        if (soB == 0) resG2 = cen2;
         else  resG2 = minimum
         (maximum(cen3+(coB/soB),0),255);
    }
}
else resG2 = -300;

cenRG = Median(RG,teller); cenRG = RG[2];
cenGB = Median(GB,teller); cenGB = GB[2];

centRG = absol(cen2-cen1);
centGB = absol(cen2-cen3);

memRG = EQUALI(centRG,cenRG,10.0,20.0);
memGB = EQUALI(centGB,cenGB,10.0,20.0);

corr1 = EQUALI(absol(cen2-
cor[2]),absol(cor[5]-cor[3]),10.0,20.0);
corr2 = EQUALI(absol(cen2-
cor[6]),absol(cor[5]-cor[3]),10.0,20.0);
mem = minimum(minimum(memRG,memGB),
maximum(maximum(corr1,corr2),1.0-glob));

if (resG2 == -300){
   som = 0.0; wei = 0.0;
   for (k=i-rand1a; k<=i+rand1b; k++){
      for (l=j-rand2a; l<=j+rand2b; l++){
      wei += 1.0 - NOISE(A2[k][l], fixed2,
      amount2);
      som +=  A2[k][l]*(1.0 - NOISE(A2[k][l],
      fixed2, amount2));
      }
    }
    if (wei == 0) resG = Median(cor,teller);
    else resG = (som/wei);
}
else resG = resG2;

noise2 = NOISE(resG, fixed2, amount2);
if ((noise2 > 0) && (ok==0)){
   over_i[OVER2] = i;
   over_j[OVER2] = j;
   OVER2++;
```

```
      ok = 1;
   }
}
else resG = cen2;

/*********************************************
*     THE BLUE COMPONENT
*********************************************/
noise3 = NOISE(cen3, fixed3, amount3);
if(noise3 > 0){
   teller =0;
   for (k=i-rand1a; k<=i+rand1b; k++){
      for (l=j-rand2a; l<=j+rand2b; l++){
         nR = NOISE(A1[k][l], fixed1, amount1);
         nG = NOISE(A2[k][l], fixed2, amount2);
         nB = NOISE(A3[k][l], fixed3, amount3);

         RB[teller] = A3[k][l] - A1[k][l];
         GB[teller] = A3[k][l] - A2[k][l];

         NRB[teller] = 1.0 - maximum(nR,nB);
         NGB[teller] = 1.0 - maximum(nG,nB);
         cor[teller] = A3[k][l];
         teller++;
      }
   }

   // one of the three components is noise-free
   if (glob == 0) {
       if ((noise1a==0) && (noise2a==0)) {
           coR = 0; soR = 0; coG = 0; soG = 0;
           for (k=0; k<teller; k++){
               coG +=  NGB[k]*GB[k];
               coR +=  NRB[k]*RB[k];
               soG +=  NGB[k];
               soR +=  NRB[k];
           }
           if (soR == 0) {
               if (soG == 0) resB2 = cen3;
               else resB2 = minimum
                (maximum(cen2+(coG/soG),0),255);
           }
           else if (soG == 0) resB2 = minimum
            (maximum(cen1+(coR/soR),0),255);
           else  resB2 = minimum (
            maximum(((cen1+(coR/soR))+(cen2+(coG/
            soG)))/2.0 ,0) ,255);
       }
       else if (noise2a==0) {
           coG = 0; soG = 0;
           for (k=0; k<teller; k++){
               coG +=  NGB[k]*GB[k];
               soG +=  NGB[k];
           }
           if (soG == 0) resB2 = cen3;
```

```
            else  resB2 = minimum
            (maximum(cen2+(coG/soG),0),255);
        }
        else {
            coR = 0; soR = 0;
            for (k=0; k<teller; k++){
                coR +=  NRB[k]*RB[k];
                soR +=  NRB[k];
            }
            if (soR == 0) resB2 = cen3;
            else  resB2 = minimum
            (maximum(cen1+(coR/soR),0),255);
        }
    }
    else resB2 = -300;

    cenGB = Median(GB,teller); cenGB = GB[2];
    cenRB = Median(RB,teller); cenRB = RB[2];

    centGB = absol(cen3-cen2);
    centRB = absol(cen3-cen1);

    memGB = EQUALI(centGB,cenGB,10.0,20.0);
    memRB = EQUALI(centRB,cenRB,10.0,20.0);

    corr1 = EQUALI(absol(cen3-
    cor[2]),absol(cor[5]-cor[3]),10.0,20.0);
    corr2 = EQUALI(absol(cen3-
    cor[6]),absol(cor[5]-cor[3]),10.0,20.0);
    mem = minimum(minimum(memGB,memRB),
    maximum(maximum(corr1,corr2),1.0-glob));

    if (resB2 == -300){
       som = 0.0; wei = 0.0;
       for (k=i-rand1a; k<=i+rand1b; k++){
          for (l=j-rand2a; l<=j+rand2b; l++){
          wei += 1.0 - NOISE(A3[k][l], fixed3,
          amount3);
          som +=  A3[k][l]*(1.0 - NOISE(A3[k][l],
          fixed3, amount3));
          }
       }
       if (wei == 0) resB = Median(cor,teller);
       else resB = (som/wei);
    }
    else resB = resB2;

    noise3 = NOISE(resB, fixed3, amount3);
    if ((noise3 > 0) && (ok==0)){
       over_i[OVER2] = i;
       over_j[OVER2] = j;
       OVER2++;
       ok = 1;
    }
}
```

```
            else resB = cen3;
            filt1[i+j*M] = resR;
            filt2[i+j*M] = resG;
            filt3[i+j*M] = resB;
          }
      }
}

 // Secound iteration
else {
   OVER2 = 0;
   for (loop = 0; loop<OVER;loop++){
       ok = 0;

      i = over_i[loop];
      j = over_j[loop];

      /* step 1. Determine the local window*/
      if(i < W) {
         rand1a = i;
         rand1b = W;
      }
      else {
         if (i>M-W-1){
            rand1a = W;
            rand1b = M-i-1;
         }
         else{
            rand1a = W;
            rand1b = W;
         }
      }

      if(j < W) {
         rand2a = j;
         rand2b = W;
      }
      else {
          if (j > N-W-1){
             rand2a = W;
             rand2b = N-j-1;
          }
          else{
             rand2a = W;
             rand2b = W;
          }
      }
      /* end step 1. */

      cen1 = A1[i][j];
      cen2 = A2[i][j];
      cen3 = A3[i][j];
      noise1a = NOISE(cen1, fixed1, amount1);
      noise2a = NOISE(cen2, fixed2, amount2);
      noise3a = NOISE(cen3, fixed3, amount3);
```

```
glob = minimum(minimum(noise1a,noise2a),noise3a);


/*********************************************
*     THE RED COMPONENT
*********************************************/
noise1 = NOISE(cen1, fixed1, amount1);
if(noise1 > 0){
   teller =0;
    for (k=i-rand1a; k<=i+rand1b; k+=W){
        for (l=j-rand2a; l<=j+rand2b; l+=W){
        nR = NOISE(filt1[k+l*M], fixed1, amount1);
        nG = NOISE(filt2[k+l*M], fixed2, amount2);
       nB = NOISE(filt3[k+l*M], fixed3, amount3)

      RG[teller] = filt1[k+l*M] - filt2[k+l*M];
      RB[teller] = filt1[k+l*M] - filt3[k+l*M];

      NRG[teller] = 1.0 - maximum(nR,nG);
      NRB[teller] = 1.0 - maximum(nR,nB);
      cor[teller] = filt1[k+l*M];
      teller++;
        }
    }

    // one of the three components is noise-free
    if (glob == 0) {
        if ((noise2a==0) && (noise3a==0)) {
            coG = 0; soG = 0;  coB = 0; soB = 0;
            for (k=0; k<teller; k++){
                coG +=  NRG[k]*RG[k];
                coB +=  NRB[k]*RB[k];
                soG +=  NRG[k];
                soB +=  NRB[k];
            }
            if (soB == 0) {
                if (soG == 0) resR2 = cen1;
                else resR2 = minimum
            (maximum(cen2+(coG/soG),0),255);
            }
            else if (soG == 0) resR2 = minimum
            (maximum(cen3+(coB/soB),0),255);
            else  resR2 = minimum (
            maximum(((cen3+(coB/soB))+(cen2+(coG/soG))
            )/2.0 ,0) ,255);
        }
        else if (noise2a==0) {
            coG = 0; soG = 0;
            for (k=0; k<teller; k++){
               coG +=  NRG[k]*RG[k];
               soG +=  NRG[k];
            }
            if (soG == 0) resR2 = cen1;
```

```
            else  resR2 = minimum
            (maximum(cen2+(coG/soG),0),255);
        }
        else {
            coB = 0; soB = 0;
            for (k=0; k<teller; k++){
                coB +=  NRB[k]*RB[k];
                soB +=  NRB[k];
            }
            if (soB == 0) resR2 = cen1;
            else  resR2 = minimum
            (maximum(cen3+(coB/soB),0),255);
        }
    }
    else resR2 = -300;

    centRG = absol(cen1-cen2);
    centRB = absol(cen1-cen3);

    cenRG = Median(RG,teller); cenRG = RG[2];
    cenRB = Median(RB,teller); cenRB = RB[2];
    memRG = EQUALI(centRG,cenRG,10.0,20.0);
    memRB = EQUALI(centRB,cenRB,10.0,20.0);
    mem = minimum(memRG,memRB);
    if (resR2 == -300){
        som = 0.0; wei = 0.0;
        for (k=i-rand1a; k<=i+rand1b; k+=W){
          for (l=j-rand2a; l<=j+rand2b; l+=W){
          wei += 1.0 - NOISE(filt1[k+l*M], fixed1,
               amount1);
          som +=  filt1[k+l*M]*(1.0 -
               NOISE(filt1[k+l*M], fixed1, amount1));
          }
       }
       if (wei == 0) resR = Median(cor,teller);
       else resR = mem*cen1 + (1.0 - mem)*(som/wei);
    }
    else resR = resR2;
    noise1 = NOISE(resR, fixed1, amount1);
    if ((noise1 > 0) &&  (ok==0)){
       over_i[OVER2] = i;
       over_j[OVER2] = j;
       OVER2++;
       ok = 1;
    }
}
else resR = cen1;

/*********************************************
*     THE GREEN COMPONENT
*********************************************/
noise2 = NOISE(cen2, fixed2, amount2);
if(noise2 > 0){
   teller =0;
    for (k=i-rand1a; k<=i+rand1b; k+=W){
```

```
    for (l=j-rand2a; l<=j+rand2b; l+=W){
    nR = NOISE(filt1[k+l*M], fixed1, amount1);
    nG = NOISE(filt2[k+l*M], fixed2, amount2);
    nB = NOISE(filt3[k+l*M], fixed3, amount3);

         RG[teller] = filt2[k+l*M] - filt1[k+l*M];
         GB[teller] = filt2[k+l*M] - filt3[k+l*M];

         NRG[teller] = 1.0 - maximum(nR,nG);
         NGB[teller] = 1.0 - maximum(nG,nB);
         cor[teller] = filt2[k+l*M];
         teller++;
    }
}

// one of the three components is noise-free
if (glob == 0) {
    if ((noise1a==0) && (noise3a==0)) {
        coR = 0; soR = 0;  coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coR +=  NRG[k]*RG[k];
            coB +=  NGB[k]*GB[k];
            soR +=  NRG[k];
            soB +=  NGB[k];
        }
        if (soB == 0) {
            if (soR == 0) resG2 = cen2;
            else resG2 = minimum
             (maximum(cen1+(coR/soR),0),255);
        }
        else if (soR == 0) resG2 = minimum
        (maximum(cen3+(coB/soB),0),255);
        else  resG2 = minimum (
        maximum(((cen3+(coB/soB))+(cen1+(coR/soR))
        )/2.0 ,0) ,255);
    }
    else if (noise1a==0) {
        coR = 0; soR = 0;
        for (k=0; k<teller; k++){
           coR +=  NRG[k]*RG[k];
           soR +=  NRG[k];
        }
        if (soR == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen1+(coR/soR),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NGB[k]*GB[k];
            soB +=  NGB[k];
        }
        if (soB == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen3+(coB/soB),0),255);
```

```
        }
      }
      else resG2 = -300;

      centRG = absol(cen2-cen1);
      centGB = absol(cen2-cen3);

      cenRG = Median(RG,teller); cenRG = RG[2];
      cenGB = Median(GB,teller); cenGB = GB[2];
      memRG = EQUALI(centRG,cenRG,10.0,20.0);
      memGB = EQUALI(centGB,cenGB,10.0,20.0);
      mem = minimum(memRG,memGB);
      if (resG2 == -300){
          som = 0.0; wei = 0.0;
          for (k=i-rand1a; k<=i+rand1b; k+=W){
            for (l=j-rand2a; l<=j+rand2b; l+=W){
            wei += 1.0 - NOISE(filt2[k+l*M], fixed2,
                 amount2);
            som +=  filt2[k+l*M]*(1.0 -
                 NOISE(filt2[k+l*M], fixed2, amount2));
            }
         }
         if (wei == 0) resG = Median(cor,teller);
         else resG = mem*cen2 + (1.0 - mem)*(som/wei);
      }
      else resG = resG2;
      noise2 = NOISE(resG, fixed2, amount2);
      if ((noise2 > 0) && (ok==0)){
         over_i[OVER2] = i;
         over_j[OVER2] = j;
         OVER2++;
         ok = 1;
      }
  }
  else resG = cen2;

  /*********************************************
  *     THE BLUE COMPONENT
  *********************************************/
 noise3 = NOISE(cen3, fixed3, amount3);
  if(noise3 > 0){
     teller =0;
      for (k=i-rand1a; k<=i+rand1b; k+=W){
          for (l=j-rand2a; l<=j+rand2b; l+=W){
              nR = NOISE(filt1[k+l*M], fixed1, amount1);
              nG = NOISE(filt2[k+l*M], fixed2, amount2);
              nB = NOISE(filt3[k+l*M], fixed3, amount3);

               RB[teller] = filt3[k+l*M] - filt1[k+l*M];
               GB[teller] = filt3[k+l*M] - filt2[k+l*M];

                  NRB[teller] = 1.0 - maximum(nR,nB);
                  NGB[teller] = 1.0 - maximum(nG,nB);
                  cor[teller] = filt3[k+l*M];
                  teller++;
```

```
    }
}

// one of the three components is noise-free
if (glob == 0) {
    if ((noise2a==0) && (noise1a==0)) {
        coG = 0; soG = 0;  coR = 0; soR = 0;
        for (k=0; k<teller; k++){
            coG +=  NGB[k]*GB[k];
            coR +=  NRB[k]*RB[k];
            soG +=  NGB[k];
            soR +=  NRB[k];
        }
        if (soR == 0) {
            if (soG == 0) resB2 = cen3;
            else resB2 = minimum
            (maximum(cen2+(coG/soG),0),255);
        }
        else if (soG == 0) resB2 = minimum
            (maximum(cen1+(coR/soR),0),255);
        else  resB2 = minimum (
            maximum(((cen1+(coR/soR))+(cen2+(coG/s
            oG)))/2.0 ,0) ,255);
    }
    else if (noise2a==0) {
        coG = 0; soG = 0;
        for (k=0; k<teller; k++){
           coG +=  NGB[k]*GB[k];
           soG +=  NGB[k];
        }
        if (soG == 0) resB2 = cen3;
        else  resB2 = minimum
        (maximum(cen2+(coG/soG),0),255);
    }
    else {
        coR = 0; soR = 0;
        for (k=0; k<teller; k++){
            coR +=  NRB[k]*RB[k];
            soR +=  NRB[k];
        }
        if (soR == 0) resB2 = cen3;
        else  resB2 = minimum
        (maximum(cen1+(coR/soR),0),255);
    }
}
else resB2 = -300;

centGB = absol(cen3-cen2);
centRB = absol(cen3-cen1);

cenGB = Median(GB,teller); cenGB = GB[2];
cenRB = Median(RB,teller); cenRB = RB[2];
memGB = EQUALI(centGB,cenGB,10.0,20.0);
memRB = EQUALI(centRB,cenRB,10.0,20.0);
mem = minimum(memGB,memRB);
```

```
                if (resB2 == -300){
                    som = 0.0; wei = 0.0;
                    for (k=i-rand1a; k<=i+rand1b; k+=W){
                      for (l=j-rand2a; l<=j+rand2b; l+=W){
                          if ((k>0) && (l>0) && (k<M) && (l<N)){
                          wei += 1.0 - NOISE(filt3[k+l*M], fixed3,
                              amount3);
                          som +=  filt3[k+l*M]*(1.0 -
                              NOISE(filt3[k+l*M],fixed3,
                              amount3));
                          }
                      }
                   }
                   if (wei == 0) resB = Median(cor,teller);
                   else resB = mem*cen3 + (1.0 - mem)*(som/wei);
                }
                else resB = resB2;
                noise3 = NOISE(resB, fixed3, amount3);
                if ((noise3 > 0) && (ok==0)){
                   over_i[OVER2] = i;
                   over_j[OVER2] = j;
                   OVER2++;
                   ok = 1;
                }
             }
             else resB = cen3;

             filt1[i+j*M] = resR;
             filt2[i+j*M] = resG;
             filt3[i+j*M] = resB;

          }// end for
       }//end else

       it++;
       W++;
       if ((OVER == OVER2)||(it==15)) break;
       OVER = OVER2;
    }
    free(RG); free(RB); free(GB);
    free(NRG); free(NRB); free(NGB);
    free(cor);
    free(over_i);
    free(over_j);
}


#define Im1      prhs[0]
#define Im2      prhs[1]
#define Im3      prhs[2]
#define FIXED1   prhs[3]
#define FIXED2   prhs[4]
#define FIXED3   prhs[5]
#define WSIZE    prhs[6]
```

```
#define OUT1 plhs[0]
#define OUT2 plhs[1]
#define OUT3 plhs[2]

/**
*  The interaction with Matlab (mex):
*        nlhs = amount of output arguments (= 3)
*        nrhs = amount of input arguments (= 7)
*     *plhs[] = link to the output
*     *prhs[] = link to the input
*
**/
void mexFunction( int nlhs, mxArray  *plhs[], int nrhs, const
mxArray  *prhs[] ) {
    int row, col, i, M, N, M2, N2,W;
    double **fixed1,**fixed2,**fixed3, **A1, **A2, **A3,
**filt1,**filt2,**filt3;

    if (nlhs!=3)
        mexErrMsgTxt("It requires three output arguments only
        [OUT1 OUT2 OUT3].");
    if (nrhs!=7)
       mexErrMsgTxt("this method requires 7 input argument [R,G,B,
        FIXED1,FIXED2,FIXED3, WSIE]");

    /* Get input values */
    M = mxGetM(Im1);
    N = mxGetN(Im1);

    M2 = mxGetM(FIXED1);
    N2 = mxGetN(FIXED1);

    W = mxGetScalar(WSIZE);

    /**
    * Allocate memory for return matrices
    **/
    OUT1 = mxCreateDoubleMatrix(M, N, mxREAL);
    filt1 = mxGetPr(OUT1);

    OUT2 = mxCreateDoubleMatrix(M, N, mxREAL);
    filt2 = mxGetPr(OUT2);

    OUT3 = mxCreateDoubleMatrix(M, N, mxREAL);
    filt3 = mxGetPr(OUT3);

    /**
    * Dynamic allocation of memory for the input array
    **/
    A1 = malloc(M*sizeof(int));
    for(i=0;i<M;i++)
      A1[i] = malloc(N*sizeof(double));

    A2 = malloc(M*sizeof(int));
```

```
    for(i=0;i<M;i++)
      A2[i] = malloc(N*sizeof(double));

    A3 = malloc(M*sizeof(int));
    for(i=0;i<M;i++)
      A3[i] = malloc(N*sizeof(double));

    fixed1 = malloc(M2*sizeof(int));
    for(i=0;i<M2;i++)
      fixed1[i] = malloc(N2*sizeof(double));

    fixed2 = malloc(M2*sizeof(int));
    for(i=0;i<M2;i++)
      fixed2[i] = malloc(N2*sizeof(double));

    fixed3 = malloc(M2*sizeof(int));
    for(i=0;i<M2;i++)
      fixed3[i] = malloc(N2*sizeof(double));

     /**
     * Convert ARRAY_IN and INPUT_MASK to 2x2 C arrays (MATLAB
stores a two-dimensional matrix
     * in memory as a one-dimensional array)
     ***/
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A1[row][col] = (mxGetPr(Im1))[row+col*M];
          }
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A2[row][col] = (mxGetPr(Im2))[row+col*M];
          }
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A3[row][col] = (mxGetPr(Im3))[row+col*M];
          }
     for (col=0; col < N2; col++)
         for (row=0; row < M2; row++) {
             fixed1[row][col] = (mxGetPr(FIXED1))[row+col*M2];
          }
     for (col=0; col < N2; col++)
         for (row=0; row < M2; row++) {
             fixed2[row][col] = (mxGetPr(FIXED2))[row+col*M2];
          }
      for (col=0; col < N2; col++)
         for (row=0; row < M2; row++) {
             fixed3[row][col] = (mxGetPr(FIXED3))[row+col*M2];
          }

    /* Call callFuzzyShrink function */

callDenoise(A1,A2,A3,fixed1,fixed2,fixed3,filt1,filt2,filt3,M,N,W)
;
    for(i=0;i<M;i++)  free(A1[i]);
    free(A1);
```

```
    for(i=0;i<M;i++)  free(A2[i]);
    free(A2);
    for(i=0;i<M;i++)  free(A3[i]);
    free(A3);
    for(i=0;i<M2;i++)  free(fixed1[i]);
    free(fixed1);
    for(i=0;i<M2;i++)  free(fixed2[i]);
    free(fixed2);
    for(i=0;i<M2;i++)  free(fixed3[i]);
    free(fixed3);
}
/* end mexFcn*/
```

5. Fungsi `FIDRMCRANDOM.m`

```
%*****************************************************************
% Fuzzy Impulse Noise Detection and Reduction Method for Colour
Images
%  The paper of the FIDRMC is proposed in:
%  Stefan Schulte, Valérie De Witte, , Mike Nachtegael
%  Dietrich Van Der Weken and  Etienne E. Kerre:
%  A Fuzzy Two-step Filter for Impulse Noise Reduction From Colour
Images,
%  IEEE Transactions on Image Processing 15(11), 2006, 3567 - 3578
% Stefan Schulte (stefan.schulte@Ugent.be):
% Last modified: 15/01/06
% Inputs:  A = the noisy input image
%          O = the original noise-free image
% Outputs: Fs = the filtered image
%*****************************************************************
function Fs = FIDRMCRANDOM(A,O)
  [M,N,DIM] = size(A);
   A = double(A);
   A1 = A(:,:,1);
   A2 = A(:,:,2);
   A3 = A(:,:,3);

   FR = double(A1);
   FG = double(A2);
   FB = double(A3);

   a = 100.0; b = 190.0;
   Windowsize = 1;
   F = double(A);
   F2 = double(A);

   begin = 0;
   loop = 1;
   while (begin == 0)
      for k = 0:4
         F = double(F2);
         FR = double(F(:,:,1));
         FG = double(F(:,:,2));
         FB = double(F(:,:,3));
```

```
         mem(M,N) = 0;

         memR= Detectionb(FR,a/(2^k),b/(2^k));
         memG= Detectionb(FG,a/(2^k),b/(2^k));
         memB= Detectionb(FB,a/(2^k),b/(2^k));
         [FR2,FG2,FB2] =
         DenoiseFIDRMC2(FR,FG,FB,memR,memG,memB,Windowsize);
          F2(:,:,1) = FR2;
          F2(:,:,2) = FG2;
          F2(:,:,3) = FB2;

         if (MSEC(F,O,3) < MSEC(F2,O,3)) | (loop == 40)
            begin = 1;
            break;
         end
      end
      loop = loop +1;
   end
   Fs = F;
```

6. Fungsi `DenoiseFIDRMC2.mexw32`, ditulis dalam bahasa C.

```
/*****************************************************************
*********
% Fuzzy Random Impulse Noise Reduction Method for Colour Images
%
%  The paper of the FIDRMC is proposed in:
%
%  Stefan Schulte, Valérie De Witte, , Mike Nachtegael
%  Dietrich Van Der Weken and  Etienne E. Kerre:
%  A Fuzzy Two-step Filter for Impulse Noise Reduction From Colour
Images,
%  IEEE Transactions on Image Processing 15(11), 2006, 3567 - 3578
%
% Stefan Schulte (stefan.schulte@Ugent.be):
% Last modified: 30/05/06
%*****************************************************************
*********/
#include "mex.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

double LARGE (double x, double p1, double p2) {
   double res = 0.0;
   if ((x > p1) && (x < p2))    res = (x-p1)/(p2-p1);
   else if (x > p2)             res = 1.0;
   else                         res = 0.0;
   return res;
}

double absol(double a) {
   double b;
   if(a<0)   b=-a;
   else   b=a;
```

```
   return b;
}

double minimum(double a, double b) {
   double x;
   if(a<=b)   x = a;
   else   x=b;
   return x;
}

double maximum(double a, double b) {
   double x;
   if(a<=b)   x = b;
   else   x=a;
   return x;
}

double NOISE(double x, double ** noise, int amount) {
   double ret, tmp;
   int i;
   ret = 0.0; tmp = 0.0;

   for (i = 0; i<amount; i++){
      if((x == noise[i][0]) || ((x < noise[i][0]) && (x >=
noise[i][1])) || ((x > noise[i][0]) && (x<=noise[i][3])))
          tmp = 1.0;
      else if((x<noise[i][1]) && (x>noise[i][2]))  tmp =
((noise[i][1]-x)/(noise[i][1]-noise[i][2]));
      else if ((x>noise[i][3])&&(x<noise[i][4])) tmp =
((noise[i][4]-x)/(noise[i][4]-noise[i][3]));
      else tmp = 0.0;
      ret = maximum(ret,tmp);
   }
   return ret;
}

double Median(double * x, int len) {
   double tmp;
   int i,j;
   for (i = 0; i<len; i++){
       for (j = i+1; j<len; j++){
           if (x[j] < x[i]){
               tmp = x[i];
               x[i] = x[j];
               x[j] = tmp;
           }
       }
   }
   return x[(int)(len/2)];
}

double EQUALI(double x, double cen, double a, double b) {
   double res, diff;
   diff = absol(x-cen);
   if (diff <= a) res = 1.0;
```

```
   else if ((diff > a) && (diff <= b)) res = (b-diff)/(b-a);
   else res = 0;
   return res;
}


/*****************************************************************
*  The main program for FIDRMC Method for Random-valued impulse
noise
*****************************************************************/
void callDenoise(double **A1,double **A2,double **A3,double
**mem1,double **mem2,double **mem3,double *filt1,double
*filt2,double *filt3,int M, int N,int W) {
   int i,j,k,l, it, loop;
   double som, wei;
   int *over_i, *over_j, OVER, OVER2, ok,teller;
   double noise1, noise2, noise3,cen1, cen2, cen3,glob;
   double noise1a, noise2a, noise3a, nR, nG, nB;
   double *RG, *RB, *GB, *NRG, *NRB, *NGB, *cor;
   double coR, coG, coB, soR, soG, soB, resR, resG, resB, resR2,
resG2, resB2;
   double centRG, centRB, centGB, cenRG, cenRB, cenGB, memRG,
memRB, memGB, mem ;
   double corr1, corr2;

   int rand1a = 0;
   int rand1b = 0;
   int rand2a = 0;
   int rand2b = 0;

   over_i = malloc(M*N*sizeof(double));
   over_j = malloc(M*N*sizeof(double));
   RG = malloc((2*W+1)*(2*W+1)*sizeof(double));
   RB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   GB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   NRG = malloc((2*W+1)*(2*W+1)*sizeof(double));
   NRB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   NGB = malloc((2*W+1)*(2*W+1)*sizeof(double));
   cor = malloc((2*W+1)*(2*W+1)*sizeof(double));

   for (i=0; i<(2*W+1)*(2*W+1); i++) {
       RG[i] = 0;  GB[i] = 0; RB[i] = 0; cor[i] = 0;
       NRG[i] = 0;  NGB[i] = 0; NRB[i] = 0;
   }

   for (i=0; i < M; i++)
         for (j=0; j < N; j++) {
             filt1[i+j*M] = A1[i][j];
             filt2[i+j*M] = A2[i][j];
             filt3[i+j*M] = A3[i][j];
          }


   it = 1;
   OVER = 0;
```

```
OVER2 = 0;
while((it == 1)||(OVER != 0)) {
   // First iteration
   if(it==1){
       for(i=2; i<M-2; i++){
           for(j=2; j<N-2; j++){
                ok = 0;
               /* step 1. Determine the local window*/
               if(i < W) {
                  rand1a = i;
                  rand1b = W;
               }
               else {
                  if (i>M-W-1){
                      rand1a = W;
                      rand1b = M-i-1;
                  }
                  else{
                      rand1a = W;
                      rand1b = W;
                  }
               }

               if(j < W) {
                  rand2a = j;
                  rand2b = W;
               }
               else {
                  if (j > N-W-1){
                     rand2a = W;
                     rand2b = N-j-1;
                  }
                  else{
                     rand2a = W;
                     rand2b = W;
                  }
               }
               /* end step 1. */

               cen1 = A1[i][j];
               cen2 = A2[i][j];
               cen3 = A3[i][j];
               noise1a = mem1[i][j];
               noise2a = mem2[i][j];
               noise3a = mem3[i][j];


               glob = minimum(minimum(noise1a,noise2a),noise3a);

               /*********************************************
               *     THE RED COMPONENT
               *********************************************/
               noise1 = mem1[i][j];
               if(noise1 > 0){
                  teller =0;
```

```
for (k=i-rand1a; k<=i+rand1b; k++){
   for (l=j-rand2a; l<=j+rand2b; l++){
      nR = mem1[k][l];
      nG = mem2[k][l];
      nB = mem3[k][l];

      RG[teller] = A1[k][l] - A2[k][l];
      RB[teller] = A1[k][l] - A3[k][l];

      NRG[teller] = 1.0 - maximum(nR,nG);
      NRB[teller] = 1.0 - maximum(nR,nB);
      cor[teller] = A1[k][l];
      teller++;
   }
}

// one of the three components is noise-free
if (glob == 0) {
    if ((noise2a==0) && (noise3a==0)) {
        coG = 0; coB = 0; soG = 0; soB = 0;
        for (k=0; k<teller; k++){
            coG +=  NRG[k]*RG[k];
            coB +=  NRB[k]*RB[k];
            soG +=  NRG[k];
            soB +=  NRB[k];
        }
        if (soB == 0) {
            if (soG == 0) resR2 = cen1;
            else resR2 = minimum
             (maximum(cen2+(coG/soG),0),255);
        }
        else if (soG == 0) resR2 = minimum
        (maximum(cen3+(coB/soB),0),255);
        else  resR2 = minimum
        (maximum((((cen3+(coB/soB))+(cen2+(coG
        /soG)))/2.),0),255);
    }
    else if (noise2a==0) {
        coG = 0; soG = 0;
        for (k=0; k<teller; k++){
            coG +=  NRG[k]*RG[k];
            soG +=  NRG[k];
        }
        if (soG == 0) resR2 = cen1;
        else  resR2 = minimum
         (maximum(cen2+(coG/soG),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NRB[k]*RB[k];
            soB +=  NRB[k];
        }
        if (soB == 0) resR2 = cen1;
```

```
            else  resR2 = minimum
             (maximum(cen3+(coB/soB),0),255);
         }
      }
      else resR2 = -300;

      cenRG = Median(RG,teller); cenRG = RG[2];
      cenRB = Median(RB,teller); cenRB = RB[2];

      centRG = absol(cen1-cen2);
      centRB = absol(cen1-cen3);

      memRG = EQUALI(centRG,cenRG,10.0,20.0);
      memRB = EQUALI(centRB,cenRB,10.0,20.0);

      corr1 = EQUALI(absol(cen1-
      cor[2]),absol(cor[5]-cor[3]),10.0,20.0);
      corr2 = EQUALI(absol(cen1-
      cor[6]),absol(cor[5]-cor[3]),10.0,20.0);
      mem = minimum(minimum(memRG,memRB),
      maximum(maximum(corr1,corr2),1.0-glob));

      if (resR2 == -300){
         som = 0.0; wei = 0.0;
         for (k=i-rand1a; k<=i+rand1b; k++){
            for (l=j-rand2a; l<=j+rand2b; l++){
               wei += 1.0 - mem1[k][l];
               som +=  A1[k][l]*(1.0 - mem1[k][l]);
            }
          }

          if (wei == 0) resR = Median(cor,teller);
          else resR = (som/wei);
      }
      else resR = resR2;

      noise1 = 0;
      if ((noise1 > 0) && (ok==0)){
         over_i[OVER2] = i;
         over_j[OVER2] = j;
         OVER2++;

         ok = 1;
      }
   }
   else resR = cen1;

   /*********************************************
   *     THE GREEN COMPONENT
   *********************************************/
   noise2 = mem2[i][j];
   if(noise2 > 0){
      teller =0;
      for (k=i-rand1a; k<=i+rand1b; k++){
         for (l=j-rand2a; l<=j+rand2b; l++){
```

```
      nR = mem1[k][l];
      nG = mem2[k][l];
      nB = mem3[k][l];

      RG[teller] = A2[k][l] - A1[k][l];
      GB[teller] = A2[k][l] - A3[k][l];

      NRG[teller] = 1.0 - maximum(nR,nG);
      NGB[teller] = 1.0 - maximum(nG,nB);
      cor[teller] = A2[k][l];
      teller++;
   }
}

// one of the three components is noise-free
if (glob == 0) {
    if ((noise1a==0) && (noise3a==0)) {
        coR = 0; soR = 0; coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coR +=  NRG[k]*RG[k];
            coB +=  NGB[k]*GB[k];
            soR +=  NRG[k];
            soB +=  NGB[k];
        }
        if (soB == 0) {
            if (soR == 0) resG2 = cen2;
            else resG2 = minimum
             (maximum(cen1+(coR/soR),0),255);
        }
        else if (soR == 0) resG2 = minimum
        (maximum(cen3+(coB/soB),0),255);
        else  resG2 = minimum (
        maximum(((cen3+(coB/soB))+(cen1+(coR/s
        oR)))/2.0 ,0) ,255);
    }
    else if (noise1a==0) {
        coR = 0; soR = 0;
        for (k=0; k<teller; k++){
            coR +=  NRG[k]*RG[k];
            soR +=  NRG[k];
        }
        if (soR == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen1+(coR/soR),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NGB[k]*GB[k];
            soB +=  NGB[k];
        }
        if (soB == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen3+(coB/soB),0),255);
    }
```

```
   }
   else resG2 = -300;

   cenRG = Median(RG,teller); cenRG = RG[2];
   cenGB = Median(GB,teller); cenGB = GB[2];

   centRG = absol(cen2-cen1);
   centGB = absol(cen2-cen3);

   memRG = EQUALI(centRG,cenRG,10.0,20.0);
   memGB = EQUALI(centGB,cenGB,10.0,20.0);

   corr1 = EQUALI(absol(cen2-
   cor[2]),absol(cor[5]-cor[3]),10.0,20.0);
   corr2 = EQUALI(absol(cen2-
   cor[6]),absol(cor[5]-cor[3]),10.0,20.0);
   mem = minimum(minimum(memRG,memGB),
   maximum(maximum(corr1,corr2),1.0-glob));

   if (resG2 == -300){
      som = 0.0; wei = 0.0;
      for (k=i-rand1a; k<=i+rand1b; k++){
         for (l=j-rand2a; l<=j+rand2b; l++){
            wei += 1.0 - mem2[k][l];
            som +=  A2[k][l]*(1.0 - mem2[k][l]);
         }
       }
       if (wei == 0) resG = Median(cor,teller);
       else resG = (som/wei);
   }
   else resG = resG2;

   noise2 = 0;
   if ((noise2 > 0) && (ok==0)){
      over_i[OVER2] = i;
      over_j[OVER2] = j;
      OVER2++;
      ok = 1;
   }
}
else resG = cen2;

/***********************************************
*     THE BLUE COMPONENT
***********************************************/
noise3 = mem3[i][j];
if(noise3 > 0){
   teller =0;
   for (k=i-rand1a; k<=i+rand1b; k++){
      for (l=j-rand2a; l<=j+rand2b; l++){
         nR = mem1[k][l];
         nG = mem2[k][l];
         nB = mem3[k][l];

         RB[teller] = A3[k][l] - A1[k][l];
```

```
      GB[teller] = A3[k][l] - A2[k][l];

      NRB[teller] = 1.0 - maximum(nR,nB);
      NGB[teller] = 1.0 - maximum(nG,nB);
      cor[teller] = A3[k][l];
      teller++;
   }
}

// one of the three components is noise-free
if (glob == 0) {
    if ((noise1a==0) && (noise2a==0)) {
        coR = 0; soR = 0; coG = 0; soG = 0;
        for (k=0; k<teller; k++){
            coG +=  NGB[k]*GB[k];
            coR +=  NRB[k]*RB[k];
            soG +=  NGB[k];
            soR +=  NRB[k];
        }
        if (soR == 0) {
            if (soG == 0) resB2 = cen3;
            else resB2 =
          minimum(maximum(cen2+(coG/soG),0),25
          5);
        }
        else if (soG == 0) resB2 = minimum
        (maximum(cen1+(coR/soR),0),255);
        else  resB2 = minimum (
        maximum(((cen1+(coR/soR))+(cen2+(coG/s
        oG)))/2.0 ,0) ,255);
    }
    else if (noise2a==0) {
        coG = 0; soG = 0;
        for (k=0; k<teller; k++){
            coG +=  NGB[k]*GB[k];
            soG +=  NGB[k];
        }
        if (soG == 0) resB2 = cen3;
        else  resB2 = minimum
        (maximum(cen2+(coG/soG),0),255);
    }
    else {
        coR = 0; soR = 0;
        for (k=0; k<teller; k++){
            coR +=  NRB[k]*RB[k];
            soR +=  NRB[k];
        }
        if (soR == 0) resB2 = cen3;
        else  resB2 = minimum
        (maximum(cen1+(coR/soR),0),255);
    }
}
else resB2 = -300;

cenGB = Median(GB,teller); cenGB = GB[2];
```

```
                cenRB = Median(RB,teller); cenRB = RB[2];

                centGB = absol(cen3-cen2);
                centRB = absol(cen3-cen1);

                memGB = EQUALI(centGB,cenGB,10.0,20.0);
                memRB = EQUALI(centRB,cenRB,10.0,20.0);

                corr1 = EQUALI(absol(cen3-
                cor[2]),absol(cor[5]-cor[3]),10.0,20.0);
                corr2 = EQUALI(absol(cen3-
                cor[6]),absol(cor[5]-cor[3]),10.0,20.0);
                mem = minimum(minimum(memGB,memRB),
                maximum(maximum(corr1,corr2),1.0-glob));

                if (resB2 == -300){
                   som = 0.0; wei = 0.0;
                   for (k=i-rand1a; k<=i+rand1b; k++){
                      for (l=j-rand2a; l<=j+rand2b; l++){
                         wei += 1.0 - mem3[k][l];
                         som +=  A3[k][l]*(1.0 - mem3[k][l]);
                      }
                    }
                    if (wei == 0) resB = Median(cor,teller);
                    else resB = (som/wei);
                }
                else resB = resB2;

                noise3 = 0;
                if ((noise3 > 0) && (ok==0)){
                   over_i[OVER2] = i;
                   over_j[OVER2] = j;
                   OVER2++;
                   ok = 1;
                }
             }
             else resB = cen3;
             filt1[i+j*M] = resR;
             filt2[i+j*M] = resG;
             filt3[i+j*M] = resB;
          }
       }
    }

 // Secound iteration
else {
   OVER2 = 0;
   for (loop = 0; loop<OVER;loop++){
       ok = 0;

      i = over_i[loop];
      j = over_j[loop];

      /* step 1. Determine the local window*/
      if(i < W) {
```

```
   rand1a = i;
   rand1b = W;
}
else {
   if (i>M-W-1){
      rand1a = W;
      rand1b = M-i-1;
   }
   else{
      rand1a = W;
      rand1b = W;
   }
}

if(j < W) {
   rand2a = j;
   rand2b = W;
}
else {
    if (j > N-W-1){
       rand2a = W;
       rand2b = N-j-1;
    }
    else{
       rand2a = W;
       rand2b = W;
    }
}
/* end step 1. */

cen1 = A1[i][j];
cen2 = A2[i][j];
cen3 = A3[i][j];
noise1a = mem1[i][j];
noise2a = mem2[i][j];
noise3a = mem3[i][j];

glob = minimum(minimum(noise1a,noise2a),noise3a);


/***********************************************
*     THE RED COMPONENT
***********************************************/
noise1 = mem1[i][j];
if(noise1 > 0){
   teller =0;
    for (k=i-rand1a; k<=i+rand1b; k+=W){
        for (l=j-rand2a; l<=j+rand2b; l+=W){
                nR = mem1[k][l];
                nG = mem2[k][l];
                nB = mem3[k][l];

                RG[teller] = filt1[k+l*M] -
                filt2[k+l*M];
```

```
            RB[teller] = filt1[k+l*M] -
            filt3[k+l*M];

            NRG[teller] = 1.0 - maximum(nR,nG);
            NRB[teller] = 1.0 - maximum(nR,nB);
            cor[teller] = filt1[k+l*M];
            teller++;
    }
}

// one of the three components is noise-free
if (glob == 0) {
    if ((noise2a==0) && (noise3a==0)) {
        coG = 0; soG = 0;  coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coG +=  NRG[k]*RG[k];
            coB +=  NRB[k]*RB[k];
            soG +=  NRG[k];
            soB +=  NRB[k];
        }
        if (soB == 0) {
            if (soG == 0) resR2 = cen1;
            else resR2 = minimum
             (maximum(cen2+(coG/soG),0),255);
        }
        else if (soG == 0) resR2 = minimum
        (maximum(cen3+(coB/soB),0),255);
        else  resR2 = minimum (
        maximum(((cen3+(coB/soB))+(cen2+(coG/soG))
        )/2.0 ,0) ,255);
    }
    else if (noise2a==0) {
        coG = 0; soG = 0;
        for (k=0; k<teller; k++){
           coG +=  NRG[k]*RG[k];
           soG +=  NRG[k];
        }
        if (soG == 0) resR2 = cen1;
        else  resR2 = minimum
        (maximum(cen2+(coG/soG),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NRB[k]*RB[k];
            soB +=  NRB[k];
        }
        if (soB == 0) resR2 = cen1;
        else  resR2 = minimum
        (maximum(cen3+(coB/soB),0),255);
    }
}
else resR2 = -300;

centRG = absol(cen1-cen2);
```

```
    centRB = absol(cen1-cen3);

    cenRG = Median(RG,teller); cenRG = RG[2];
    cenRB = Median(RB,teller); cenRB = RB[2];
    memRG = EQUALI(centRG,cenRG,10.0,20.0);
    memRB = EQUALI(centRB,cenRB,10.0,20.0);
    mem = minimum(memRG,memRB);

    if (resR2 == -300){
        som = 0.0; wei = 0.0;
        for (k=i-rand1a; k<=i+rand1b; k+=W){
          for (l=j-rand2a; l<=j+rand2b; l+=W){
                 wei += 1.0 - mem1[k][l];
                 som +=  filt1[k+l*M]*(1.0 -
                 mem1[k][l]);
          }
       }
       if (wei == 0) resR = Median(cor,teller);
       else resR = mem*cen1 + (1.0 - mem)*(som/wei);
    }
    else resR = resR2;
    noise1 = 0;
    if ((noise1 > 0) &&  (ok==0)){
       over_i[OVER2] = i;
       over_j[OVER2] = j;
       OVER2++;
       ok = 1;
    }
}
else resR = cen1;


/*********************************************
*     THE GREEN COMPONENT
*********************************************/
noise2 = mem2[i][j];
if(noise2 > 0){
   teller =0;
    for (k=i-rand1a; k<=i+rand1b; k+=W){
        for (l=j-rand2a; l<=j+rand2b; l+=W){
             nR = mem1[k][l];
             nG = mem2[k][l];
             nB = mem3[k][l];

             RG[teller] = filt2[k+l*M] - filt1[k+l*M];
             GB[teller] = filt2[k+l*M] - filt3[k+l*M];

             NRG[teller] = 1.0 - maximum(nR,nG);
             NGB[teller] = 1.0 - maximum(nG,nB);
             cor[teller] = filt2[k+l*M];
             teller++;
        }
    }

    // one of the three components is noise-free
```

```
if (glob == 0) {
    if ((noise1a==0) && (noise3a==0)) {
        coR = 0; soR = 0;  coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coR +=  NRG[k]*RG[k];
            coB +=  NGB[k]*GB[k];
            soR +=  NRG[k];
            soB +=  NGB[k];
        }
        if (soB == 0) {
            if (soR == 0) resG2 = cen2;
            else resG2 = minimum
        (maximum(cen1+(coR/soR),0),255);
        }
        else if (soR == 0) resG2 = minimum
        (maximum(cen3+(coB/soB),0),255);
        else  resG2 = minimum (
        maximum(((cen3+(coB/soB))+(cen1+(coR/soR))
        )/2.0 ,0) ,255);
    }
    else if (noise1a==0) {
        coR = 0; soR = 0;
        for (k=0; k<teller; k++){
           coR +=  NRG[k]*RG[k];
           soR +=  NRG[k];
        }
        if (soR == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen1+(coR/soR),0),255);
    }
    else {
        coB = 0; soB = 0;
        for (k=0; k<teller; k++){
            coB +=  NGB[k]*GB[k];
            soB +=  NGB[k];
        }
        if (soB == 0) resG2 = cen2;
        else  resG2 = minimum
        (maximum(cen3+(coB/soB),0),255);
    }
}
else resG2 = -300;

centRG = absol(cen2-cen1);
centGB = absol(cen2-cen3);

cenRG = Median(RG,teller); cenRG = RG[2];
cenGB = Median(GB,teller); cenGB = GB[2];
memRG = EQUALI(centRG,cenRG,10.0,20.0);
memGB = EQUALI(centGB,cenGB,10.0,20.0);
mem = minimum(memRG,memGB);

if (resG2 == -300){
    som = 0.0; wei = 0.0;
    for (k=i-rand1a; k<=i+rand1b; k+=W){
```

```
            for (l=j-rand2a; l<=j+rand2b; l+=W){
                wei += 1.0 - mem2[k][l];
                som +=  filt2[k+l*M]*(1.0 - mem2[k][l]);
            }
         }
         if (wei == 0) resG = Median(cor,teller);
         else resG = mem*cen2 + (1.0 - mem)*(som/wei);
      }
      else resG = resG2;
      noise2 = 0;
      if ((noise2 > 0) && (ok==0)){
         over_i[OVER2] = i;
         over_j[OVER2] = j;
         OVER2++;
         ok = 1;
      }
  }
  else resG = cen2;

  /*********************************************
  *     THE BLUE COMPONENT
  *********************************************/
 noise3 = mem3[i][j];
  if(noise3 > 0){
     teller =0;
      for (k=i-rand1a; k<=i+rand1b; k+=W){
          for (l=j-rand2a; l<=j+rand2b; l+=W){
                  nR = mem1[k][l];
                  nG = mem2[k][l];
                  nB = mem3[k][l];

                  RB[teller] = filt3[k+l*M] -
                  filt1[k+l*M];
                  GB[teller] = filt3[k+l*M] -
                  filt2[k+l*M];

                  NRB[teller] = 1.0 - maximum(nR,nB);
                  NGB[teller] = 1.0 - maximum(nG,nB);
                  cor[teller] = filt3[k+l*M];
                  teller++;
          }
      }

     // one of the three components is noise-free
      if (glob == 0) {
          if ((noise2a==0) && (noise1a==0)) {
              coG = 0; soG = 0;  coR = 0; soR = 0;
              for (k=0; k<teller; k++){
                  coG +=  NGB[k]*GB[k];
                  coR +=  NRB[k]*RB[k];
                  soG +=  NGB[k];
                  soR +=  NRB[k];
              }
              if (soR == 0) {
                  if (soG == 0) resB2 = cen3;
```

```
            else resB2 = minimum
           (maximum(cen2+(coG/soG),0),255);
           }
           else if (soG == 0) resB2 = minimum
           (maximum(cen1+(coR/soR),0),255);
           else  resB2 = minimum (
           maximum(((cen1+(coR/soR))+(cen2+(coG/soG))
           )/2.0 ,0) ,255);
       }
       else if (noise2a==0) {
           coG = 0; soG = 0;
           for (k=0; k<teller; k++){
              coG +=  NGB[k]*GB[k];
              soG +=  NGB[k];
           }
           if (soG == 0) resB2 = cen3;
           else  resB2 = minimum
           (maximum(cen2+(coG/soG),0),255);
       }
       else {
           coR = 0; soR = 0;
           for (k=0; k<teller; k++){
               coR +=  NRB[k]*RB[k];
               soR +=  NRB[k];
           }
           if (soR == 0) resB2 = cen3;
           else  resB2 = minimum
           (maximum(cen1+(coR/soR),0),255);
       }
   }
   else resB2 = -300;

   centGB = absol(cen3-cen2);
   centRB = absol(cen3-cen1);

   cenGB = Median(GB,teller); cenGB = GB[2];
   cenRB = Median(RB,teller); cenRB = RB[2];
   memGB = EQUALI(centGB,cenGB,10.0,20.0);
   memRB = EQUALI(centRB,cenRB,10.0,20.0);
   mem = minimum(memGB,memRB);

   if (resB2 == -300){
       som = 0.0; wei = 0.0;
       for (k=i-rand1a; k<=i+rand1b; k+=W){
         for (l=j-rand2a; l<=j+rand2b; l+=W){
             if ((k>0) && (l>0) && (k<M) && (l<N)){
                wei += 1.0 - mem3[k][l];
                som +=  filt3[k+l*M]*(1.0 -
                mem3[k][l]);
             }
         }
      }
      if (wei == 0) resB = Median(cor,teller);
      else resB = mem*cen3 + (1.0 - mem)*(som/wei);
   }
```

```
                else resB = resB2;
                noise3 = 0;
                if ((noise3 > 0) && (ok==0)){
                   over_i[OVER2] = i;
                   over_j[OVER2] = j;
                   OVER2++;
                   ok = 1;
                }
             }
             else resB = cen3;
             filt1[i+j*M] = resR;
             filt2[i+j*M] = resG;
             filt3[i+j*M] = resB;

          }// end for
       }//end else
       it++;
       W++;
       if ((OVER == OVER2)||(it==15)) break;
       OVER = OVER2;
    }
    free(RG); free(RB); free(GB);
    free(NRG); free(NRB); free(NGB);
    free(cor);
    free(over_i);
    free(over_j);
}  /* End of callFuzzyShrink */

#define Im1      prhs[0]
#define Im2      prhs[1]
#define Im3      prhs[2]
#define FIXED1   prhs[3]
#define FIXED2   prhs[4]
#define FIXED3   prhs[5]
#define WSIZE    prhs[6]

#define OUT1 plhs[0]
#define OUT2 plhs[1]
#define OUT3 plhs[2]

/**
*  The interaction with Matlab (mex):
*        nlhs = amount of output arguments (= 3)
*        nrhs = amount of input arguments (= 7)
*     *plhs[] = link to the output
*     *prhs[] = link to the input
*
**/
void mexFunction( int nlhs, mxArray  *plhs[], int nrhs, const
mxArray  *prhs[] ) {
    int row, col, i, M, N, M2, N2,W;
    double **fixed1,**fixed2,**fixed3, **A1, **A2, **A3,
**filt1,**filt2,**filt3;

    if (nlhs!=3)
```

```
    mexErrMsgTxt("It requires three output arguments only
     [OUT1 OUT2 OUT3].");
if (nrhs!=7)
   mexErrMsgTxt("this method requires 7 input argument [R,G,B,
    FIXED1,FIXED2,FIXED3, WSIE]");

/* Get input values */
M = mxGetM(Im1);
N = mxGetN(Im1);

M2 = mxGetM(FIXED1);
N2 = mxGetN(FIXED1);

W = mxGetScalar(WSIZE);

/**
* Allocate memory for return matrices
**/
OUT1 = mxCreateDoubleMatrix(M, N, mxREAL);
filt1 = mxGetPr(OUT1);

OUT2 = mxCreateDoubleMatrix(M, N, mxREAL);
filt2 = mxGetPr(OUT2);

OUT3 = mxCreateDoubleMatrix(M, N, mxREAL);
filt3 = mxGetPr(OUT3);

/**
* Dynamic allocation of memory for the input array
**/
A1 = malloc(M*sizeof(int));
for(i=0;i<M;i++)
  A1[i] = malloc(N*sizeof(double));

A2 = malloc(M*sizeof(int));
for(i=0;i<M;i++)
  A2[i] = malloc(N*sizeof(double));

A3 = malloc(M*sizeof(int));
for(i=0;i<M;i++)
  A3[i] = malloc(N*sizeof(double));

fixed1 = malloc(M2*sizeof(int));
for(i=0;i<M2;i++)
  fixed1[i] = malloc(N2*sizeof(double));

fixed2 = malloc(M2*sizeof(int));
for(i=0;i<M2;i++)
  fixed2[i] = malloc(N2*sizeof(double));

fixed3 = malloc(M2*sizeof(int));
for(i=0;i<M2;i++)
  fixed3[i] = malloc(N2*sizeof(double));

 /**
```

```
     * Convert ARRAY_IN and INPUT_MASK to 2x2 C arrays (MATLAB
stores a two-dimensional matrix
     * in memory as a one-dimensional array)
     ***/
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A1[row][col] = (mxGetPr(Im1))[row+col*M];
          }
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A2[row][col] = (mxGetPr(Im2))[row+col*M];
          }
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A3[row][col] = (mxGetPr(Im3))[row+col*M];
          }
     for (col=0; col < N2; col++)
         for (row=0; row < M2; row++) {
             fixed1[row][col] = (mxGetPr(FIXED1))[row+col*M2];
          }
     for (col=0; col < N2; col++)
         for (row=0; row < M2; row++) {
             fixed2[row][col] = (mxGetPr(FIXED2))[row+col*M2];
          }
     for (col=0; col < N2; col++)
         for (row=0; row < M2; row++) {
             fixed3[row][col] = (mxGetPr(FIXED3))[row+col*M2];
          }
callDenoise(A1,A2,A3,fixed1,fixed2,fixed3,filt1,filt2,filt3,M,N,W)
    for(i=0;i<M;i++)  free(A1[i]);
    free(A1);
    for(i=0;i<M;i++)  free(A2[i]);
    free(A2);
    for(i=0;i<M;i++)  free(A3[i]);
    free(A3);
    for(i=0;i<M2;i++)  free(fixed1[i]);
    free(fixed1);
    for(i=0;i<M2;i++)  free(fixed2[i]);
    free(fixed2);
    for(i=0;i<M2;i++)  free(fixed3[i]);
    free(fixed3);
}
/* end mexFcn*/
```

7. Fungsi `Detctionb.mexw32`, ditulis dalam bahasa c

```
/*****************************************************************
% Fuzzy Random Impulse Noise Detection Method for Colour Images
%  The paper of the FIDRMC is proposed in:
%  Stefan Schulte, Valérie De Witte, , Mike Nachtegael
%  Dietrich Van Der Weken and  Etienne E. Kerre:
%  A Fuzzy Two-step Filter for Impulse Noise Reduction From Colour
Images,
%  IEEE Transactions on Image Processing 15(11), 2006, 3567 - 3578
```

```
% Stefan Schulte (stefan.schulte@Ugent.be):
% Last modified: 30/05/06
%*************************************************************/

#include "mex.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

double LARGE (double x, double p1, double p2) {
   double res = 0.0;
   if ((x > p1) && (x < p2))   res = (x-p1)/(p2-p1);
   else if (x > p2)            res = 1.0;
   else                        res = 0.0;
   return res;
}

double SIM (double x, double a, double b) {
   double res = 0.0, y = 0.0, z = 0.0;
   y = maximum(minimum(1-x,1-a), minimum(1-x,1-b));
   z = maximum(minimum(x,a),minimum(x,b));
   res = maximum(y,z);
   return res;
}

double absol(double a) {
   double b;
   if(a<0)   b=-a;
   else   b=a;
   return b;
}

double minimum(double a, double b) {
   double x;
   if(a<=b)   x = a;
   else   x=b;
   return x;
}

double maximum(double a, double b) {
   double x;
   if(a<=b)   x = b;
   else   x=a;
   return x;
}

/*************************************************************
*********************
*  The main program of the FIDRMC method
*************************************************************
*********************/
void callMemb(double **A1,double *mem,double a, double b,int M,
int N) {
   int i,j,k, loop;
```

```
double *med, *histo,*sorted,*maxi;
double som, tel, hlp,res, tmp, slope1, diff1, diff2, diff3;
double l1, l2, l3, minslop1, minslop2, slope2;
int **xy, **adj;

int rand1a = 0;
int rand1b = 0;
int rand2a = 0;
int rand2b = 0;

xy = malloc(9*sizeof(int));
adj = malloc(9*sizeof(int));
for(i=0;i<9;i++){
   xy[i] = malloc(2*sizeof(int));
   adj[i] = malloc(2*sizeof(int));
}

med = malloc(8*sizeof(double));

xy[0][0] = -1; xy[0][1] = -1; xy[1][0] = -1; xy[1][1] = 0;
xy[2][0] = -1; xy[2][1] =  1; xy[3][0] =  0; xy[3][1] = -1;
xy[4][0] =  0; xy[4][1] =  0; xy[5][0] =  0; xy[5][1] = 1;
xy[6][0] =  1; xy[6][1] = -1; xy[7][0] =  1; xy[7][1] = 0;
xy[8][0] =  1; xy[8][1] =  1;

adj[0][0] = 1; adj[0][1] = -1; adj[1][0] =  0; adj[1][1] = -1;
adj[2][0] = 1; adj[2][1] =  1; adj[3][0] =  1; adj[3][1] =  0;
adj[4][0] = 0; adj[4][1] =  0; adj[5][0] =  1; adj[5][1] =  0;
adj[6][0] = 1; adj[6][1] =  1; adj[7][0] =  0; adj[7][1] = -1;
adj[8][0] = 1; adj[8][1] = -1;


for(i=2; i<M-2; i++){
   for(j=2; j<N-2; j++){
      /*-------------------------------------------
             Gradient values method
      -------------------------------------------*/
      tel = 0;
      for (k = 0; k<9; k++){
         if (k==4) continue;
         diff1 = A1[i+xy[k][1]][j+xy[k][0]] - A1[i][j];
         diff2 = A1[i+xy[k][1]+adj[k][1]][j+xy[k][0]+adj[k][0]]
                - A1[i+adj[k][1]][j+adj[k][0]];
         diff3 = A1[i+xy[k][1]-adj[k][1]][j+xy[k][0]-adj[k][0]]
                - A1[i-adj[k][1]][j-adj[k][0]];

         l1 = LARGE(absol(diff1),a,b);
         l2 = LARGE(absol(diff2),a,b);
         l3 = LARGE(absol(diff3),a,b);

         res = 1.0-  maximum( maximum(minimum(1-l1,1-l2),
               minimum(1-l1,1-l3)),
               maximum(minimum(l1,l2),minimum(l1,l3)));
```

```
            res = l1*res;
            for (loop = 0; loop<tel; loop++){
                if (res < med[loop]){
                   hlp = med[loop];
                   med[loop] = res;
                   res = hlp;
                }
            }
            med[(int)tel] = res;
            tel++;
         }
         mem[i+j*M] = minimum( minimum(med[6],med[7]),med[5]);
      }
   }

   for(i=0;i<9;i++)  free(xy[i]);
   free(xy);
   for(i=0;i<9;i++)  free(adj[i]);
   free(adj);
   free(med);
}


#define Im1      prhs[0]
#define PAR1     prhs[1]
#define PAR2     prhs[2]
#define OUT plhs[0]

/**
*  The interaction with Matlab (mex):
*        nlhs = amount of output arguments (= 1)
*        nrhs = amount of input arguments (= 3)
*     *plhs[] = link to the output
*     *prhs[] = link to the input
*
**/
void mexFunction( int nlhs, mxArray  *plhs[], int nrhs, const
mxArray  *prhs[] ) {
    int row, col, i, M, N;
    double **mem, **A1, p1, p2;

    if (nlhs!=1)
        mexErrMsgTxt("It requires one output arguments only
[M1].");
    if (nrhs!=3)
       mexErrMsgTxt("this method requires three input argument
[Im1]");

    /* Get input values */
    M = mxGetM(Im1);
    N = mxGetN(Im1);
    p1 = mxGetScalar(PAR2);
    p2 = mxGetScalar(PAR1);
```

```
    /**
    * Allocate memory for return matrices
    **/
    OUT = mxCreateDoubleMatrix(M, N, mxREAL);
    mem = mxGetPr(OUT);

    /**
    * Dynamic allocation of memory for the input array
    **/
    A1 = malloc(M*sizeof(int));
    for(i=0;i<M;i++)
      A1[i] = malloc(N*sizeof(double));

     /**
     * Convert ARRAY_IN and INPUT_MASK to 2x2 C arrays (MATLAB
stores a two-dimensional matrix
     * in memory as a one-dimensional array)
     ***/
     for (col=0; col < N; col++)
         for (row=0; row < M; row++) {
             A1[row][col] = (mxGetPr(Im1))[row+col*M];
          }

    /* Call the main program */
    callMemb(A1,mem,p1,p2,M,N);

    for(i=0;i<M;i++)  free(A1[i]);
    free(A1);
}
/* end mexFcn*/
```

LAMPIRAN III

Output dari *script* `Hasil_akhir.m` untuk mengeksekusi citra "`Wings of The Albatross.jpg`" dengan berbagai tingkatan *impulse noise*.

```
Hasil_akhir
================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.03
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     0.26358
  => Output PSNR:   53.92172
=======================
```


```
Hasil_akhir
================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.05
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     0.39229
  => Output PSNR:   52.19469
=======================
```


```
Hasil_akhir
================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
```

```
Masukan Prosentase impulse noise: 0.1
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     0.81543
  => Output PSNR:   49.01696
=======================
```


```
Hasil_akhir
=====================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
=====================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.15
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     1.16257
  => Output PSNR:   47.47660
=======================
```

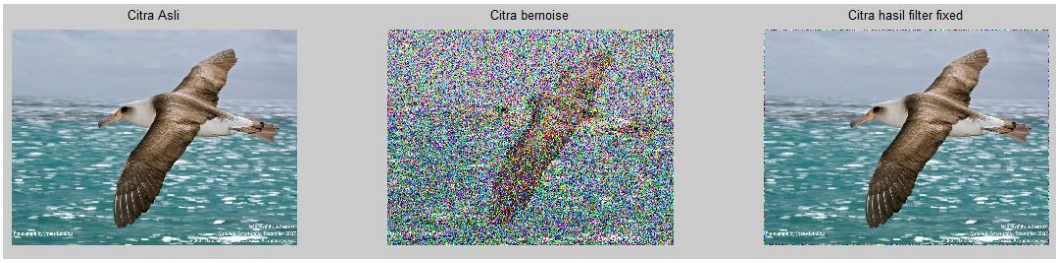

```
Hasil_akhir
=====================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
=====================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.2
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     1.85705
  => Output PSNR:   45.44258
=======================
```


```
Hasil_akhir
=====================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
=====================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.3
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     3.49712
  => Output PSNR:   42.69370
=======================
```


```
Hasil_akhir
=====================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
=====================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.4
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     7.00130
  => Output PSNR:   39.67901
=======================
```


```
Hasil_akhir
=====================================================================
```

```
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
Masukan Citra yang akan direduksi: imread('Wings of The
Albatross.jpg')
Masukan Prosentase impulse noise: 0.5
======================
 Fixed-Valued Impulse Noise
  => Output MSE:    13.01125
  => Output PSNR:   36.98761
======================
```



LAMPIRAN IV

script m-file `medianfilt.m`

```
disp('============================================================')
disp('++++++++++++Reduksi Noise dengan Median Filter++++++++++++')
disp('============================================================')
A=input('Masukan Citra yang akan direduksi: ');
a=input('Masukan Prosentase impulse noise: ');
B=imnoise(A, 'salt & pepper', a);
B=double(B);
trm=medfilt2(B(:,:,1));
tgm=medfilt2(B(:,:,2));
tbm=medfilt2(B(:,:,3));
tm=cat(3,trm,tgm,tbm);
disp(sprintf('  => Output MSE:  %10.5f',MSEC(A,tm,2)));
disp(sprintf('  => Output PSNR:
%10.5f',(log(255^2/MSEC(A,tm,2))/log(10)*10)));
B=uint8(B);
tm=uint8(tm);
figure, subplot(1,3,1), imshow(A),title('Citra Asli'),
subplot(1,3,2)
,imshow(B),title('Citra bernoise'),subplot(1,3,3),
imshow(tm),title('Citra hasil filter fixed'),
clear all
```

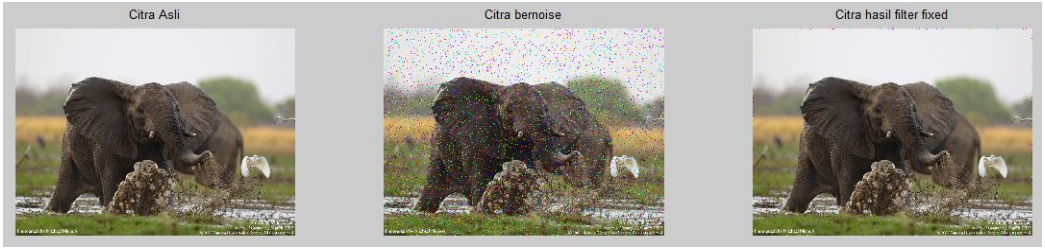
LAMPIRAN V

Output dari *script* mfile `Hasil_akhir.m` dan `medfilt.m` untuk citra "*African Wildlife Haven*.jpg".

1. Output dari script m-file Hasil_akhir.m

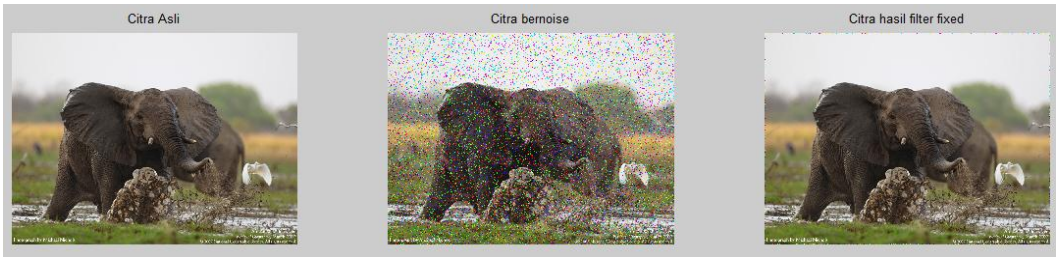
```
Hasil_akhir
================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.03
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     0.37567
  => Output PSNR:   52.38271
=======================
```


```
Hasil_akhir
================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.05
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     0.49906
  => Output PSNR:   51.14928
=======================
```

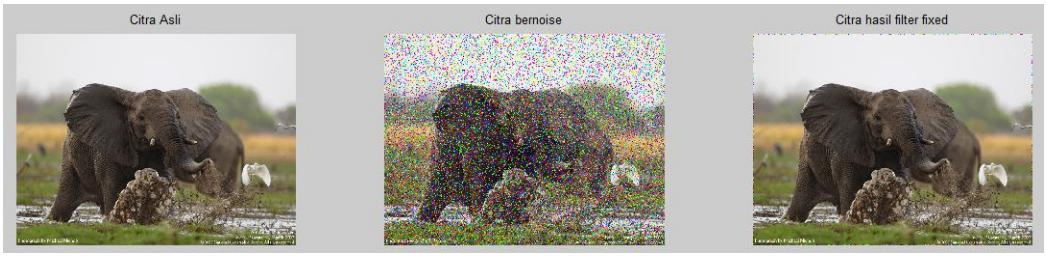



```
Hasil_akhir
================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
================================================================
```

```
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.1
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     0.84137
  => Output PSNR:   48.88092
=======================
```

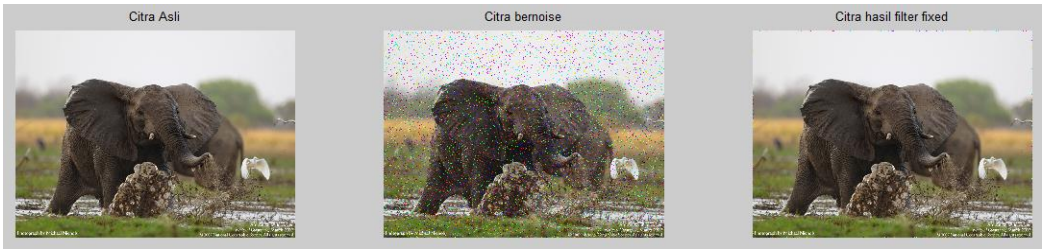



```
Hasil_akhir
=================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
=================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.2
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     2.12664
  => Output PSNR:   44.85387
=======================
```




```
Hasil_akhir
=================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
=================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.35
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:     6.25540
  => Output PSNR:   40.16825
=======================
```


```
Hasil_akhir
==================================================================
Fuzzy Two Step Filter to Reduction Impulse Noise of Color Image
==================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.5
=======================
 Fixed-Valued Impulse Noise
  => Output MSE:    19.68779
  => Output PSNR:   35.18883
=======================
```

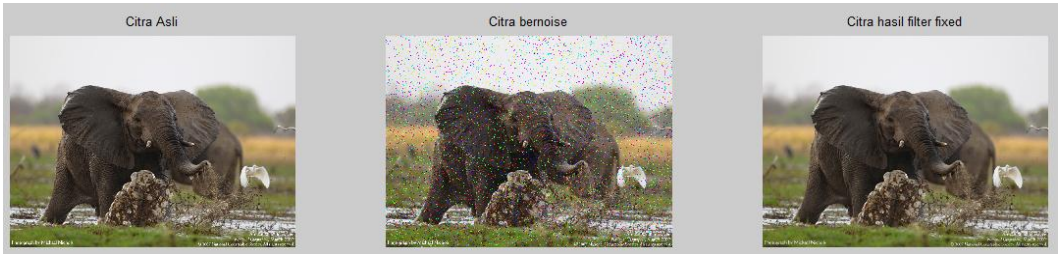


2. Output dari `medfilt.m`

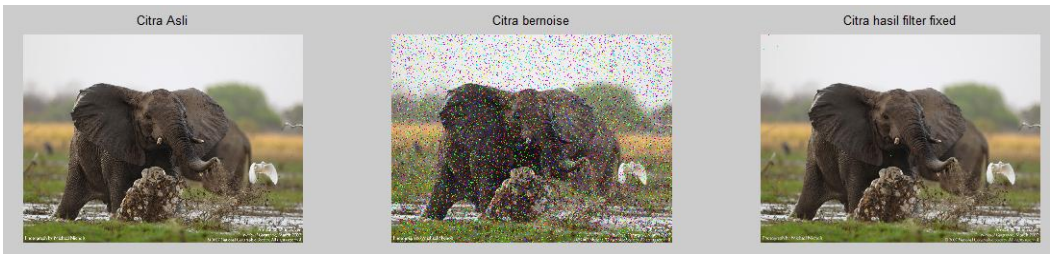
```
medianfilt
==================================================================
++++++++++++Reduksi Noise dengan Median Filter+++++++++++++++
==================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.03
  => Output MSE:    40.86276
  => Output PSNR:   32.01753
```


```
medianfilt
==================================================================
++++++++++++Reduksi Noise dengan Median Filter+++++++++++++++
```

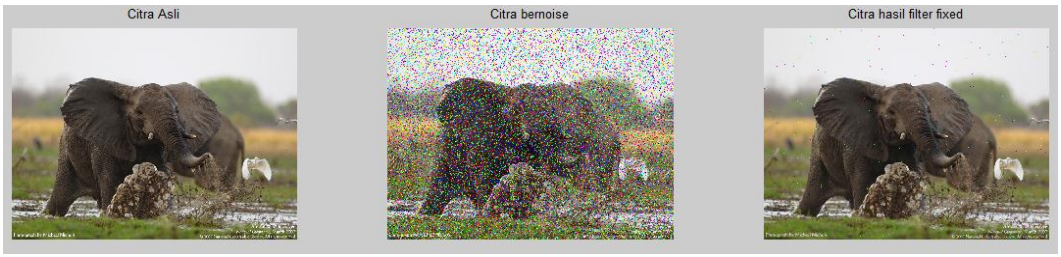
```
==================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.05
  => Output MSE:    42.38318
  => Output PSNR:   31.85887
```




```
medianfilt
==================================================================
++++++++++++Reduksi Noise dengan Median Filter+++++++++++++++++
==================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.1
  => Output MSE:    49.47306
  => Output PSNR:   31.18712
```




```
medianfilt
==================================================================
++++++++++++Reduksi Noise dengan Median Filter+++++++++++++++++
==================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.2
  => Output MSE:   100.69298
  => Output PSNR:   28.10081
```




```
medianfilt
```

```
================================================================
++++++++++++Reduksi Noise dengan Median Filter+++++++++++++++++
================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.35
  => Output MSE:   570.56529
  => Output PSNR:   20.56775
```


```
medianfilt
================================================================
++++++++++++Reduksi Noise dengan Median Filter+++++++++++++++++
================================================================
Masukan Citra yang akan direduksi: imread('African Wildlife
Haven.jpg')
Masukan Prosentase impulse noise: 0.5
  => Output MSE:  2247.59627
  => Output PSNR:   14.61362
```