ANALISIS PERBANDINGAN METODE FILTER GAUSSIAN, MEAN DAN MEDIAN TERHADAP REDUKSI *NOISE SALT AND PEPPERS*

SKRIPSI

Oleh

ASMANIATUL JANNAH
NIM: 04550068

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MALANG
2008

ANALISIS PERBANDINGAN METODE FILTER GAUSSIAN, MEAN DAN MEDIAN TERHADAP REDUKSI *NOISE SALT AND PEPPERS*

SKRIPSI

Diajukan Kepada:
Dekan Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh

ASMANIATUL JANNAH
NIM. 04550068

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MALANG
2008

ANALISIS PERBANDINGAN METODE FILTER GAUSSIAN, MEAN DAN MEDIAN TERHADAP REDUKSI *NOISE SALT AND PEPPERS*

Oleh

ASMANIATUL JANNAH
NIM. 04550068

Telah Disetujui untuk Diuji

Malang, 25 Juli 2008

Dosen Pembimbing I,	Dosen Pembimbing II,
Ririen Kusumawati, M. Kom.	**Ach. Nashichuddin,M.A.**
NIP. 150 368 775	**NIP. 150 302 531**

Mengetahui,

Ketua Jurusan Teknik Informatika

Suhartono, M. Kom
NIP 150 327 241

LEMBAR PENGESAHAN

ANALISIS PERBANDINGAN METODE FILTER GAUSSIAN, MEAN DAN MEDIAN TERHADAP REDUKSI *NOISE SALT AND PEPPERS*

SKRIPSI

Oleh

ASMANIATUL JANNAH
NIM. 04550068

Diajukan Kepada:
Dekan Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Malang
untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 29 Juli 2008

Susunan Dewan Penguji:		**Tanda Tangan**
1. Penguji Utama	**: M. Amin Hariyadi, M. T.**	**( )**
2. Ketua	**: Muhamad Faisal, M. T**	**( )**
3. Sekretaris	**: Ririen Kusumawati, M. Kom**	**( )**
4. Anggota	**: Achmad Nasichuddin, M.A**	**( )**

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik informatika

Suhartono, M. Kom
NIP 150 327 241

DAFTAR ISI

BAB III DESAIN DAN IMPLEMENTASI

BAB IV HASIL DAN PEMBAHASAN

BAB V KESIMPULAN DAN SARAN

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

ABSTRAK

Jannah, Asmaniatul. 2008. **Analisis Perbandingan Metode Filter Gaussian, Mean dan Median Terhadap Reduksi *Noise Salt and Peppers*.** Jurusan Teknik Informatika Fakultas Sains dan Teknologi UIN Malang. Pembimbing:(1) Ririen Kusumawati,M.Kom. (2)Achmad Nashichuddin, M.A.

Kata Kunci : *Filtering* citra, metode filter Gaussian, filter mean, filter median.

Noise merupakan kesalahan yang terjadi dalam proses pengambilan citra yang menyebabkan sebuah nilai intensitas piksel tidak mencerminkan nilai intensitas piksel yang sebenarnya. Perbaikan citra yang mengandung *noise* merupakan salah satu bidang yang penting dalam pengolahan citra digital. Metode yang akan digunakan pada tugas akhir ini untuk memperbaiki citra yang mengandung *noise* adalah Gaussian filtering, mean filtering dan median filtering. Metode tersebut akan diterapkan pada citra yang mengandung *Salt and Peppers noise*.

Filter Gaussian adalah filter linier dengan nilai pembobotan untuk setiap anggotanya dipilih berdasarkan bentuk fungsi Gaussian. Sama halnya dengan filter Gaussian, filter mean adalah filter linear yang bekerja dengan menggantikan intensitas nilai pixel dengan rata-rata dari nilai pixel tersebut dengan nilai pixel-pixel tetangganya. Sedangkan filter median merupakan salah satu filtering non linier yang mengurutkan nilai intensitas sekelompok piksel, kemudian menganti nilai piksel yang diproses dengan nilai mediannya. Kualitas citra di ukur dengan dua besaran yaitu MSE (*Mean Square Error*) dan PSNR (*Peak Signal to Noise Ratio*). Pada penelitian ini untuk kernel 3 x 3 nilai PSNR tertinggi terjadi pada filter median disusul filter mean dan selanjutnya filter Gaussian, sedangkan kernel 5 x 5 nilai PSNR tertinggi terjadi pada filter median disusul filter Gaussian dan selanjutnya filter mean. Dari ke 2 ukuran kernel, peningkatan PSNR terbesar dengan citra *noise salt and peppers* terjadi pada metode *median filtering*. Hal ini dibuktikan dari ke tiga citra uji dengan kernel 3 x 3 dan 5 x 5 dengan berbagai ukuran citra yang menunjukkan peningkatan nilai PSNR terjadi pada filter median. Dengan peningkatan PSNR yang lebih besar dari pada metode Gaussian dan mean filter, maka dapat disimpulkan metode filter median merupakan metode yang paling baik dalam mengurangi *noise* jenis *salt and peppers* yang ada pada citra.

ABSTRACT

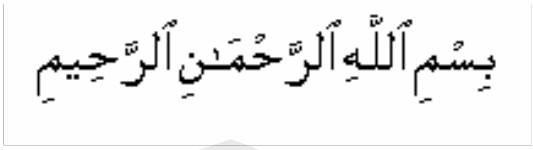
Jannah, Asmaniatul. 2008. **The Comparison Analysis Of Gaussian, Mean, And Median Filter Methods To The Salt And Pepper Noise.** Thesis. Informatic Departement. Science and Technology Facilty. The State Islamic University of Malang. Advisor: (1) Ririen Kusumawati, M.Kom. (2) Achmad Nashichuddin, M.A.

Keywords **:** Filtering citra, gaussian, mean, median filter methods

Noise is the processing mistake of taking citra that not showing the real number pixel itself. The improvement of the citra of noise is one of the important cases in the processing of citra digital. Therefore, on this research, gaussian, mean, and median filtering are methods for repairing the citra of salt pepper noise.

Filter gaussianis a linier filter with the weight for each chosen member based on the form of gaussian function. As a gaussian filter, mean filter is a linier filter that work as a changer of the intensity number of pixel with the average level from pixel number and other numbers of neighbour pixel. Otherwise, median filter is one of filtering non linier which order the intensity number f a pixel group, changes it while processing with the median number. The avality of citra is measured by MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ratio). On this research, for kernel with 3x3, the highest PSNR number is shown on the median, mean, and then gaussian filter. On the other hand, kernel with 5x5, the highest PSNR number is shown on the median, then gaussian, and the last is mean filter. From both kernel measurement, the bigger improvement of PSNR with salt and pepper noise citra is shown in median filtering method. It is proved by the three test of citra with 3x3 and 5x5 by some measurement of citra shows the improvement PSNR number in median filter. Finally, the conclusion of this research from those explanation above is median filter method is the best mehod in minimizing the salt and pepper noise of citra.

MOTTO


بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ


قُلْ هُوَ ٱللَّهُ أَحَدٌ ﴿١﴾ ٱللَّهُ ٱلصَّمَدُ ﴿٢﴾ لَمْ يَلِدْ وَلَمْ يُولَدْ ﴿٣﴾
وَلَمْ يَكُن لَّهُۥ كُفُوًا أَحَدٌۢ ﴿٤﴾

Dengan menyebut nama Allah

Yang Maha Pengasih, Maha Penyayang

Katakanlah (Muhammad), Dialah Allah, Yang Maha Esa (1)

Allah tempat meminta segala sesuatu (2)

(Allah) tidak beranak dan tidak pula diperanakkan (3)

Dan tidak ada sesuatu yang setara dengan Dia (4) (Al - Ikhlash [112])

Teriring Doa Semoga Skripsi ini Bermanfaat dan menjadi Kesuksesan Dunia dan Akhirat

Skripsi ini kupersembahkan kepada:

Kedua Orang Tuaku Tercinta; M. Ali Dlofir Dan Siti Nurbaiyah Yang Selalu Mendidik, Mendoakan Dan Menyayangiku… Semoga Allah Selalu Memberikan Keselamatan, Kebahagiaan Dunia-Akhirat Dan Umur Panjang…Amin

Saudara-Saudaraku; Mas Yasin, Mba' Ulfa, Neng, Mas Rohim, Mas Malik, Mas Darwis, Mba' Ana, Mba' Dan Kakak Iparku Semua, Keponakanku Yang Lucu-Lucu, Trimakasih Atas Dukungan Dan Bantuannya Selama Ini, Semoga Allah Membalas Dengan Yang Lebih Baik... Amin

Dosen2 Teknik Informatika Yang Terhormat; P.Suhartono, Bu Ririen, P.Syahid, P.Fatchur, P.Faisal, P. Yaqin, P.Totok, P.Amin, P.Zainal, Bu Roro . Semoga Allah Membalas Kebaikan Kepada Mereka…Amin

Para Dosen Dan Guru-Guruku Terhormat Yang Telah Menyalurkan Ilmunya Kepadaku Dengan Penuh Sabar Dan Semangat …

Saudara-Saudaraku Seperjuangan, Lia, Ima, Ida, Alfiyah, Mirziah dan Mas Wawan, Makasih Atas Bantuannya...

Saudara-Saudaraku Di LKQS, Mas Rozy, Mas Ilham, Mas Rozak Dan Mas Lubys, Thk's Atas Dukungan Dan Do'anya, Terus Semangat Dan Berjuang Yukk…

Dan Sahabat-Sahabatku Semuanya... Makasih Banyak...

KATA PENGANTAR

Maha Suci Allah, atas kasih sayang dan karunia-Nya sehingga penulis bisa menyelesaikan Tugas Akhir berjudul :

ANALISIS PERBANDINGAN METODE FILTER GAUSSIAN, MEAN DAN MEDIAN TERHADAP REDUKSI *NOISE SALT AND PEPPERS*

Penulis berkeyakinan bahwa pengerjaan Tugas Akhir ini tak lepas dari bantuan berbagai pihak. Pada kesempatan ini penulis menyampaikan terima kasih kepada semua pihak, khususnya pada :

1. Keluarga tercinta, terutama Bapak dan Ibu yang selalu mendukung dengan sabar dan ikhlas dalam penyelesaian skripsi ini.
2. Ibu Ririen Kusumawati, M. Kom, atas segala bimbingan dan petunjuknya hingga skripsi ini dapat terselesaikan tepat pada waktunya. Terima kasih banyak.
3. Segenap staf dan dosen pengajar Teknik Informatika UIN Malang.
4. Teman-teman di Wisma Indah 86 yang selalu ceria dan membuat hati ini damai.
5. Saudara-saudaraku di LKQS yang selalu menjadi motivator penulis. Terimakasih untuk semuanya.

6. Dan kepada seluruh pihak yang mendukung penulisan skripsi yang tidak dapat disebutkan satu persatu penulis ucapkan terimakasih yang sebesar-besarnya.

Penulis menyadari bahwa pengerjaan tugas akhir ini tak luput dari kekurangan. Untuk itu dalam kesempatan ini penulis mohon maaf atas segala ketidaksempurnaan yang ada. Mudah-mudahan Tugas Akhir ini dapat menjadi sebuah referensi yang bermanfaat. Amien.

Malang, Juli 2008

Penulis

BAB I

PENDAHULUAN

1.1. Latar Belakang

Citra merupakan suatu representasi, kemiripan atau imitasi dari suatu obyek atau benda. Citra yang dikenal dalam komputer adalah citra dalam format digital.

Citra digital dapat mengalami penurunan kualitas atau gangguan, dan gangguan pada citra digitaldisebut dengan derau (*noise*), *noise* adalah gambar atau *pixel* yang mengganggu kualitas citra. Derau dapat disebabkan oleh gangguan fisik (optik) pada alat akuisisi maupun secara disengaja akibat proses pengolahan yang tidak sesuai. Contohnya adalah bintik gelap dan terang yang muncul secara acak yang menyebar pada obyek (citra) maupun latar belakangnya. Bintik acak ini disebut dengan derau *salt & pepper*.

Citra berkualitas rendah yang disebabkan *noise* memerluakan langkah-langkah perbaikan untuk meningkatkan kualitas citra tanpa mengurangi lebih banyak kualitas detail citra serta menghasilkan citra dengan informasi yang cukup akurat, salah satu teknik perbaikan citra yaitu metode *filtering* citra. Metode *filtering* citra dapat menghaluskan dan mengurangi derau pada citra, baik secara linear maupun secara non-linear. Ada banyak teknik *filtering* citra, dan pada penelitian ini penulis menggunakan filter Gaussian, filter median dan filter rata-rata untuk mereduksi *noise salt and peppers*.

Pada penelitian terdahulu (Sony, 2006), menjelaskan tentang perbaikan citra dengan menggunakan metode filter median dan filter mean dan dibandingkan

kualitas citra hasil antara kedua filter tersebut. Pada penelitian kali ini penulis mencoba untuk lebih mengembangkan penelitian terdahulu dengan menambahkan filter Gaussian untuk selanjutnya dilakukan perbandingan kualitas citra hasil antara filter Gaussian, mean dan median.

Filter Gaussian dirancang berdasarkan sistem linier dengan nilai pembobotan untuk setiap anggotanya dipilih berdasarkan bentuk fungsi Gaussian (Usman Ahmad, 2005:70). Mean filter juga merupakan salah satu *filtering* linear yang bekerja dengan menggantikan intensitas nilai *pixel* dengan rata-rata dari nilai *pixel* tersebut dengan nilai *pixel-pixel* tetangganya. Sedangkan filter median adalah salah satu filter non-linear yang mengurutkan nilai intensitas sekelompok *pixel*, kemudian mengganti nilai *pixel* yang diproses dengan nilai mediannya. Cara kerja mean *filtering* dan median *filtering* tidak tergantung pada nilai-nilai yang berbeda dengan nilai-nilai yang umum pada dalam lingkungannya sehingga filter ini dapat mempertahankan detail citra asli.

Kualitas 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 *codebook* yang dihasilkan terhadap vektor *input*. Semakin kecil nilai *MSE* menunjukkan semakin sesuai dengan vektor *input*. Parameter PSNR bernilai sebaliknya, semakin besar parameter PSNR semakin bagus *codebook* yang dihasilkan.

Melalui metode-metode di atas kualitas citra masukan dengan *noise salt and peppers* dapat diperbaiki dan diharapkan dapat lebih mudah mendekati bentuk aslinya dan membandingkan diantara filter tersebut mana yang lebih baik

digunakan. Disebabkan karena hal tersebut penulis merasa tertarik untuk mempelajari, menganalisis, dan mengimplementasikannya dalam skripsi ini. Dengan mengimplementasikan metode ini diharapkan penulis mendapat kesimpulan yang sesuai dengan semua teori tentang filter Gaussian, filter median dan filter rata-rata.

Berkaitan dengan pencitraan maka manusia adalah citra terbaik yang telah diciptakan oleh Allah swt. Hal ini sebagaimana dinyatakan oleh Allah swt di dalam Al Qur'anNya pada Surah At-Tiin Ayat 4, sebagaimana berikut:

لَقَدْ خَلَقْنَا الْإِنْسَانَ فِي أَحْسَنِ تَقْوِيمٍ

Artinya: Sesungguhnya Kami telah menciptakan manusia dalam bentuk yang sebaik-baiknya. (QS. At-Tin: 4)

Imam Muslim meriwayatkan hadits yang artinya "*Sesungguhnya Allah Maha Indah dan mencintai keindahan"* (Ahmad Umar, 2004:185). Disebutkan dalam ayat al-Qur'an bahwa manusia diciptakan dari citra Yang Maha Mencipta, yaitu Allah swt. Jika dirujukkan lebih dalam pada hadits seperti disebutkan di atas, Allah Maha indah, tentunya ciptaan-Nya pun adalah kreasi yang sangat indah. Atas keindahan-Nya itulah, Allah juga menyukai keindahan, dengan pengertian lain adalah keindahan di atas bentuk yang sebaik-baiknya.

Gambar semisal foto dan lain sebagainya, bagi sebagian orang adalah kenangan yang membawa beribu makna. Seperti disebutkan dalam sebuah kata bijak, satu kata hanya berarti satu kata tetapi sebuah gambar mewakili beribu makna. Meski saat sekarang, posisi gambar telah digeser oleh gambar bergerak

seperti video, hanya saja penggunaan gambar sebagai media penyimpan sebuah peristiwa tetap berada pada posisi yang strategis. Atas dasar itulah, maka banyak bermuncullan program (*software*) yang berfungsi untuk meningkatkan kualitas gambar dengan memakai berbagai metode perbaikan citra, misalnya perbaikan dari segi warna (*colour*), pencerahan (*brigtness*), pengurangan derau pada citra dan lain sebagainya. Dan dalam penulisan skripsi ini penulis memilih metode pengurangan derau (reduksi *noise*) sebagai metode perbaikan citra.

Oleh karena itu penulis memilih judul **Analisis Perbandingan Metode Filter Gaussian, Mean Dan Median Terhadap Reduksi *Noise Salt And Pepper*.**

1.2. Rumusan Masalah

Rumusan masalah dari skripsi ini adalah:

Bagaimana analisis perbandingan metode filter Gaussian, mean dan median terhadap reduksi *noise Salt And Pepper* ?

1.3. Batasan Masalah

Batasan masalah pada skripsi ini antara lain:

1. Menggunakan perhitungan MSE dan PSNR untuk membandingkan kualitas citra.
2. Menggunakan kernel kernel 3x3 dan 5x5.
3. Citra yang digunakan adalah citra bitmap *grayscale* dengan skala keabuan 8 bit.

1.4 Tujuan

Tujuan pembuatan skripsi ini adalah:

Melakukan analisis perbandingan metode filter Gaussian, mean dan median terhadap reduksi *noise Salt And Pepper*.

1.5 Manfaat

Manfaat yang diharapkan dari penulisan skripsi ini adalah untuk memperkaya pengetahuan tentang perancangan filter Gaussian, mean dan median, dan untuk menunjukkan kualitas dari citra hasil *filtering* atau seberapa banyak *noise salt and peppers* pada citra berkurang.

1.6 Sistematika Penulisan

Untuk memberikan gambaran dan kerangka yang jelas mengenai pokok bahasan dalam setiap bab dalam penelitian ini maka diperlukan sistematika pembahasan. Berikut gambaran sistematika pembahasan pada masing-masing bab:

BAB I: PENDAHULAN

Bab ini berisi latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat penelitian dan sistematika pembahasan.

BAB II: TINJAUAN PUSTAKA

Bab dua menjelaskan tentang teori yang berhubungan dengan permasalahan penelitian.

BAB III: DESAIN DAN IMPLEMENTASI

Pada bab ini akan dibahas tentang langkah dan pembuatan perangkat lunak serta rancangan program.

BAB IV: HASIL DAN PEMBAHASAN

Meliputi hasil yang dicapai dari perancangan sistem dan implementasi program. Sehingga dapat ditarik suatu kesimpulan.

BAB V: KESIMPULAN DAN SARAN

Berisi kesimpulan dan saran berdasarkan hasil yang telah dicapai sehingga dapat digunakan sebagai bahan pertimbangan bagi pihak-pihak yang berkepentingan serta kemungkinan pengembangannya.

1.7 Metode Penelitian

Untuk mencapai tujuan yang telah dirumuskan sebelumnya, maka metodologi pengumpulan data yang dilakukan dalam penulisan skripsi ini adalah *library research* yaitu suatu cara penelitian dan pengumpulan data teoritis dari buku-buku, artikel, jurnal dan berbagai literatur yang mendukung penyusunan skripsi.

BAB II

TINJAUAN PUSTAKA

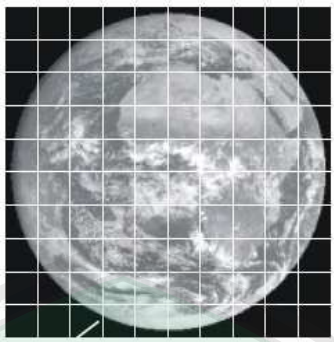
Dalam skripsi ini diperlukan beberapa landasan teori untuk memahami proses pengolahan cita digital. Kajian pustaka ini akan menjelaskan tentang :

2.1. Citra Digital

Citra adalah gambar dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinu menjadi gambar diskrit melalui proses sampling. Gambar analog dibagi menjadi N baris dan M kolom sehingga menjadi gambar diskrit. Dimana setiap pasangan indeks baris dan kolom menyatakan suatu titik pada citra. Nilai matriksnya menyatakan nilai kecerahan titik tersebut. Titik-titik tersebut dinamakan sebagai elemen citra, atau *pixel* (*picture elemen*). Dalam kamus komputer, gambar atau foto diistilahkan sebagai citra digital yang mempunyai representasi matematis berupa matriks $Cm \times n = (cij)$.

Gonzales and Woods (1992:2) mendefinisikan citra digital sebagai fungsi intensitas cahaya dua-dimensi f(x,y) dimana x dan y menunjukkan koordinat spasial, dan nilai f pada suatu titik (x,y) sebanding dengan *brightness* (*gray level*) dari citra di titik tersebut.

Citra digital dapat dideskripsikan seperti berikut:



Gambar 2.1 Representasi citra digital

2.1.1. Matriks *bitmap*

Citra *bitmap* adalah susunan bit-bit warna untuk tiap *pixel* yang membentuk pola tertentu. Pola-pola warna ini menyajikan informasi yang dapat dipahami sesuai dengan persepsi indera penglihatan manusia. Format file ini merupakan format grafis yang fleksibel untuk *platform Windows* sehingga dapat dibaca oleh program grafis manapun. Format ini mampu menyimpan informasi dengan kualitas tingkat 1 bit samapi 24 bit. (http://slametriyanto.web.id)

Citra bitmap didefinisikan sebagai fungsi f – (x,y) dengan x dan y adalah koordinat bidang. Besaran f untuk tiap koordinat (x,y) disebut intensitas atau derajat keabuan citra pada titik tersebut.

Pada gambar 2.1 ditunjukkan gambar *bitmap* beserta nilai matrisnya.

(a) *bitmap* 15×10 *pixel* (b) Matriks *bitmap*

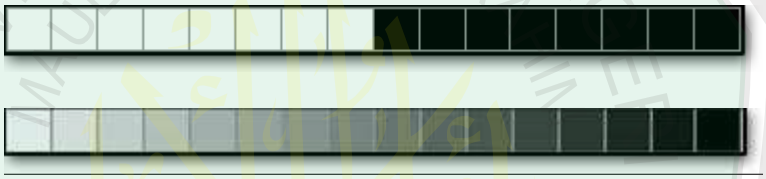
Gambar 2.2. *Bitmap* dengan nilai matriksnya

Dari definisi di atas yang diperjelas oleh gambar 1.1, *bitmap* dimodelkan dalam bentuk matriks. Nilai *pixel* atau entri-entri dari matriks ini mewakili warna yang ditampilkan dimana ordo matriks merupakan dimensi panjang dan lebar dari *bitmap*.

Nilai-nilai warna ditentukan berdasarkan intensitas cahaya yang masuk. Dalam komputer, derajat intensitas cahaya diwakili oleh bilangan cacah. Nilai 0 menerangkan tidak adanya cahaya sedangkan nilai yang lain menerangkan adanya cahaya dengan intensitas tertentu. Nilai-nilai ini bisa didapatkan melalui fungsi-fungsi yang disediakan oleh bahasa pemrograman berdasarkan input berupa lokasi entri-entri matriks yang hendak dicari.

2.1.2. *Pixel*

Pixel (*Picture Elements*) adalah nilai tiap-tiap entri matriks pada *bitmap*. Rentang nilai-nilai *pixel* ini dipengaruhi oleh banyaknya warna yang dapat ditampilkan. Jika suatu *bitmap* dapat menampilkan 256 warna maka nilai-nilai *pixel*nya dibatasi dari 0 hingga 255. Suatu *bitmap* dianggap mempunyai ketepatan yang tinggi jika dapat menampilkan lebih banyak warna. Prinsip ini dapat dilihat dari contoh pada gambar 2.3 yang memberikan contoh dua buah *bitmap* dapat memiliki perbedaan dalam menangani transisi warna putih ke warna hitam.

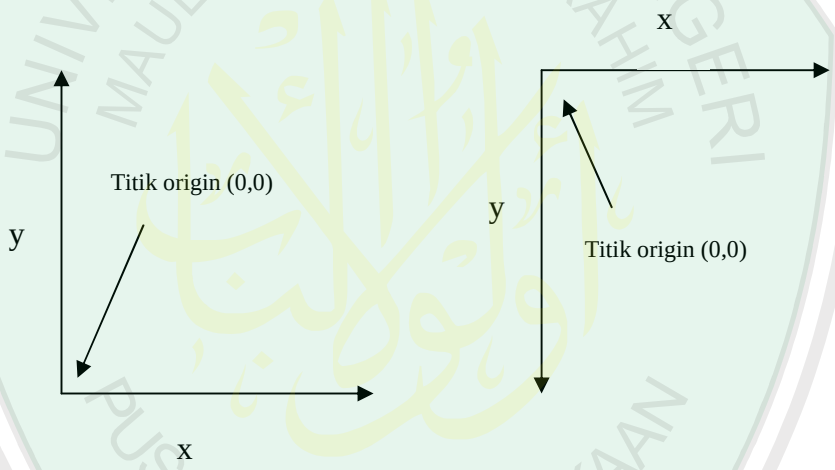


Gambar 2.3. Perbedaan ketepatan warna *bitmap*

Perbedaan ketepatan warna *bitmap* pada gambar 2 menjelaskan bahwa *bitmap* sebelah atas memberikan nilai untuk warna lebih sedikit daripada *bitmap* dibawahnya. Untuk *bitmap* dengan pola yang lebih kompleks dan dimensi yang lebih besar, perbedaan keakuratan dalam memberikan nilai warna akan terlihat lebih jelas.

Menurut Usman Ahmad (2005:14) sebuah *pixel* adalah sampel dari pemandangan yang mengandung intensitas citra yang dinyatakan dalam bilangan bulat. Sebuah citra adalah kumpulan *pixel-pixel* yang disusun dalm larik dua

dimensi. Indeks baris dan kolom (x,y) dari sebuah *pixel* dinyatakan dalam bilangan bulat. *Pixel* (0,0) terletak pada sudut kiri atas pada citra, indeks x begerak ke kanan dan indeks y bergerak ke bawah. Konvensi ini dipakai merujuk pada cara penulisan larik yang digunakan dalam pemrograman komputer. Letak titik origin pada koordinat grafik citra dan koordinat pada grafik matematika terdapat perbedaan. Hal yang berlawanan untuk arah vertikal berlaku pada kenyataan dan juga pada sistem grafik dalam matematika yang sudah lebih dulu dikenal. Gambar berikut memperlihatkan perbedaan kedua sistem ini.



(a) koordinat pada grafik matematika (b) koordinat pada citra

Gambar 2.4 Perbedaan letak titik origin pada koordinat grafik dan pada citra

2.1.3. Dimensi dan Resolusi

Dimensi *bitmap* adalah ukuran *bitmap* yang dinotasikan dengan menulis lebar × tinggi *bitmap.* Satuan ukur dimensi *bitmap* dapat berupa satuan ukur metris maupun *pixel*. Dimensi yang digunakan oleh *bitmap* mewakili ordo matriks citra itu sendiri. Contoh pada gambar 2.1 menunjukkan sebuah *bitmap* berdimensi

15×10 *pixel* yang diwakili oleh matriks C10×15. Model matriks untuk *bitmap* dipengaruhi oleh kerapatan *pixel* atau resolusi. Kerapatan *pixel* ini digunakan *bitmap* dalam mendekati kekontinyuan. Semakin besar resolusi suatu *bitmap*, obyek yang ditampilkan citra tersebut semakin akurat.

Kerapatan titik-titik pada citra dinamakan resolusi, yang menunjukkan seberapa tajam gambar ini ditampilkan yang ditunjukkan dengan jumlah baris dan kolom. Resolusi merupakan ukuran kuantitas bukan kualitas. *Pixel* merupakan satuan ukuran terhadap jumlah area *photo-receptor* pada sensor gambar kamera, yang menentukan seberapa banyak data yang dapat ditangkap.

Resolusi digunakan untuk pendataan (*sampling*) citra dari sensor. Sensor mengubah citra dari fungsi kontinu ke fungsi diskrit sehingga semakin besar resolusi citra maka informasi yang dihasilkan akan semakin baik, sebab data yang diperoleh menjadi lebih banyak.

2.1.4. Citra *Grayscale*

Grayscale (skala keabuan) merupakan suatu istilah untuk menyebutkan satu citra yang memiliki warna putih, abu-abu dan hitam. Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara hitam sebagai warna minimal dan warna putih sebagai warna maksimalnya, sehingga warna antaranya adalah abu-abu.

Pada citra digital banyaknya kemungkinan nilai dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Misalnya pada citra skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$ dan nilai maksimumnya adalah $2^4-1 = 15$. Sedangkan untuk skala keabuan 8 bit, maka jumlah

kemungkinan nilainya adalah $2^8 = 256$, dan nilai maksimumnya adalah $2^8 - 1 = 255$. Sehingga Makin besar angka grayscale, citra yang terbentuk makin mendekati kenyataan. (Balza dan Kartika, 2005:9)

2.2. Pemrosesan Citra Digital

Pemrosesan citra digital melibatkan suatu pengolah yang disebut filter. Citra hasil didapatkan dari konvolusi antara citra asal dengan sebuah kernel filter.

2.2.1. Filter

Filter adalah alat untuk memproses data yang mempunyai ciri mengambil data asli untuk memproduksi data hasil sebagaimana yang diinginkan. Dalam pengolahan citra, respon perambatan filter memberikan gambaran bagaimana *pixel-pixel* pada citra diproses. (Gonzales dan Woods, 1992:119)

2.2.2. Konvolusi

Konvolasi adalah operasi yang mendasar dalam pengolahan citra. Konvolusi didefinisikan secara sederhana sebagai operasi penjumlahan dari perkalian dengan notasi operasi (*), yang mengalikan sebuah citra dengan sebuah *maks* atau *kernel*. Konvolusi 2 buah fungsi f (x) dan g (x) didefinisikan sebagai berikut:

$$h(x) = f(x) * g(x) = \sum_{x=-n}^{n} \sum_{y=-N}^{N} f(u,v) g(x+u, y+u) \qquad (2.1)$$

dimana :

f (x,y)	:	Citra asli
h (x,y)	:	*Linier-position invariant operator*
g (x, y)	:	Citra hasil konvolusi
x, y, u dan v	:	Posisi titik dalam citra

Misalkan citra $f(x,y)$ yang berukuran 5 x 5 dan sebuah kernel atau maks ukuran 3 x 3 masing-masing prosesnya adalah sebagai berikut:

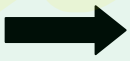
$$f(x,y) = \begin{bmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{bmatrix} \qquad g(x,y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \mathbf{4} & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Posisi (0,0) dari kernel

Berikut operasi konvolusi antara citra $f(x,y)$ dengan kernel $g(x,y)$, dapat diilustrasikan sebagai berikut :

(1) Tempatkan kernel pada sudut kiri atas, kemudian hitung nilai *pixsel* pada posisi (0,0) dari kernel, berikut prosesnya:

4	**4**	**3**	5	4
6	**6**	**5**	5	2
5	**6**	**6**	6	2
6	7	5	5	3
3	5	2	4	4

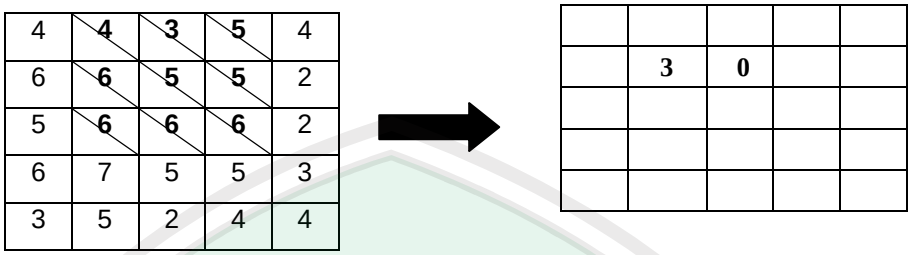


	3			

Hasil konvolusi = 3. nilai ini dihitung dengan cara berikut:

(0 x 4)+(-1 x 4)+(0 x 3)+(-1 x 6)+(4 x 6)+(-1 x 5)+(0 x 5)+(-1 x 6)+(0 x 6) = 3

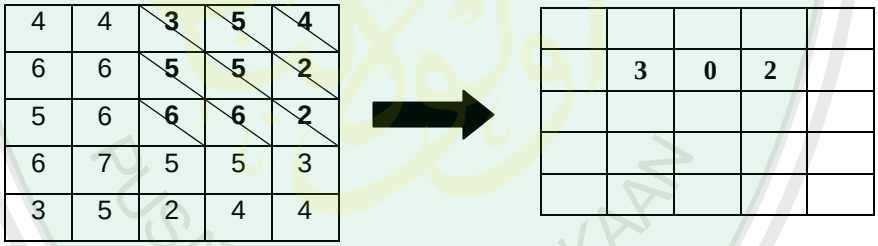
(2) Geser kernel satu *pixsel* ke kanan,kemudian hitung nilai pixsel pada posisi (0,0) dari kernel.



Hasil konvolusi = 0. nilai ini dihitung dengan cara sebagai berikut:

(0 x 4) +(-1 x 3)+(0 x 5) +(-1 x 6)+(4 x 5) +(-1 x 5)+(0 x 6)+(-1 x 6) +(0 x 6) = 0

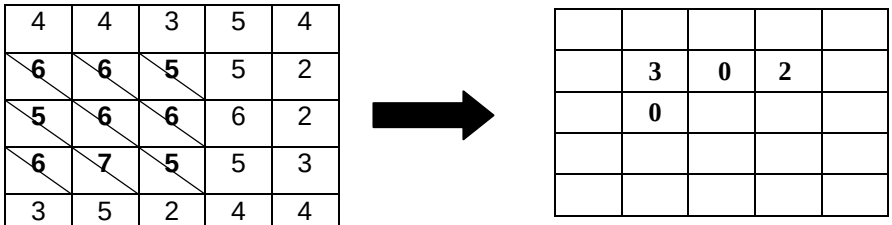
(3) Geser kernel satu *pixel* ke kanan, kemudian hitung nilai *piksel* pada posisi (0,0) dari kernel.



Hasil konvolusi = 2. Nilai ini dihitung dengan cara sebagai berikut :

(0 x 3)+(-1 x 5)+(0 x 4)+(-1 x 5)+(4 x 5)+ (-1 x 2) (0 x 6)+(-1 x 6)+(0 x 2) = 2

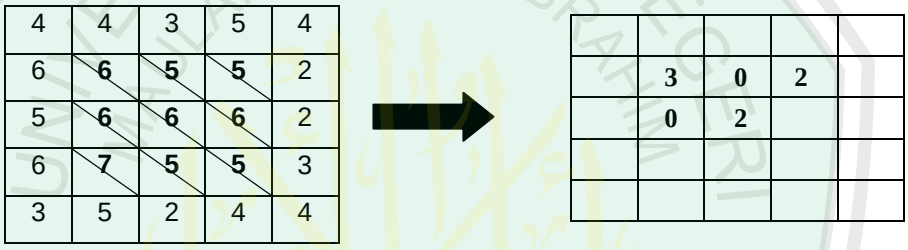
(4) Selanjutnya, geser kernel satu *pixel* ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Setiap kali konvolusi, geser kernel satu *pixel* ke kanan.



Hasil konvolusi = 0. Nilai ini dihitung dengan cara sebagai berikut :

(0 x 6)+(-1 x 6)+(0 x 5)+(-1 x 5)+(4 x 6)+(-1 x 6)+(0 x 6)+(-1 x 7)+(0 x 5) = 0

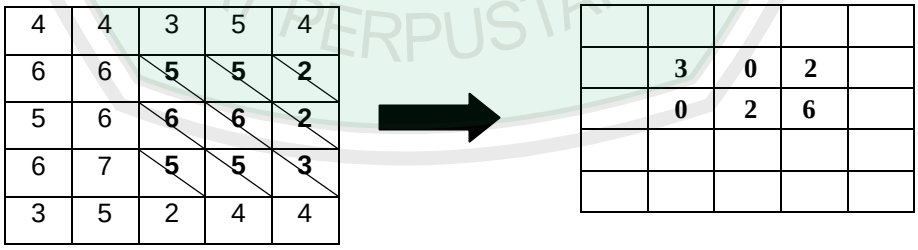
(5) Proses berikut sama dengan sebelumnya :



Hasil konvolusi = 2. Nilai ini dihitung dengan cara sebagai berikut :

(0 x 6)+(-1 x 5)+(0 x 5) +(-1 x 6) +(4 x 6) +(-1 x 6) +(0 x 7) +(-1 x 5) +(0 x 5) = 2

(6)



Hasil konvolusi = 6. Nilai ini dihitung dengan cara sebagai berikut :

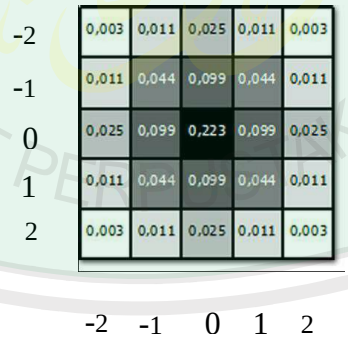
(0 x 5)+(-1 x 5)+(0 x 2)+(-1 x 6)+(4 x 6)+(-1 x 2)+(0 x 5)+(-1 x 5)+(0 x 3) = 6

Dengan cara yang sama seperti di atas, maka *pixel-pixel* pada baris ketiga dikonvolusikan sehingga menghasilkan citra hasil sebagai berikut:

	3	**0**	**2**	
	0	**2**	**6**	
	6	**0**	**2**	

2.2.3. Kernel Filter

Kernel atau *mask* memberikan petunjuk tentang apa yang harus dilakukan filter terhadap data. Pada umumnya kernel mempunyai panjang dan lebar ganjil. Pola bilangan ganjil n bertujuan agar matriks kernel mempunyai jari-jari r sehingga n=2r-1. Contoh cara penentuan lokasi entri-entri matriks dapat dilihat pada contoh gambar 4 dengan (*i, j)* yang berjalan dari -2 hingga 2 dan (*x, y*) yang berjalan dari 0 sampai 4.



Gambar 2.5 Penentuan lokasi entri pada kernel filter

2.2.4. Filter Gaussian

Filter Gaussian adalah salah satu filter linier dengan nilai pembobotan untuk setiap anggotanya dipilih berdasarkan bentuk fungsi Gaussian. Filter Gaussian dipilih sebagai filter penghalusan berdasarkan pertimbangan bahwa filter ini mempunyai pusat kernel.

Untuk menghitung atau menentukan nilai-nilai setiap elemen dalam filter penghalus Gaussian yang akan dibentuk berlaku persamaan (2.2):

$$\frac{h(x,y)}{c} = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (2.2)$$

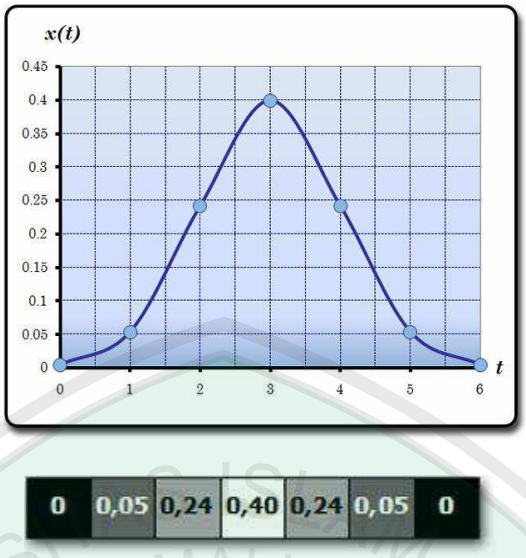
σ = Lebar dari fungsi Gaussian

c = Konstanta normalisasi

g (x,y) = Citra hasil konvolusi

Menurut Usman Ahmad (2005:70), filter Gaussian sangat baik untuk menghilangkan *noise* yang bersifat sebaran nomal, yang banyak di jumpai pada sebaran citra hasil proses digitasi menggunakan kamera karena merupakan fenomena alamiah akibat sifat pantulan cahaya dan kepekaan sensor cahaya pada kamera itu sendiri.

Gambar 2.7 menunjukkan respon perambatan dari kernel Gaussian dengan skala σ = 3.



0	0,05	0,24	0,40	0,24	0,05	0

Gambar 2.6 Respon perambatan kernel Gaussian

Sebagai contoh penerapan, kernel Gaussian dari gambar 2.8 akan dikonvolusikan dengan data yang terdiri dari 8 titik beserta hasilnya.

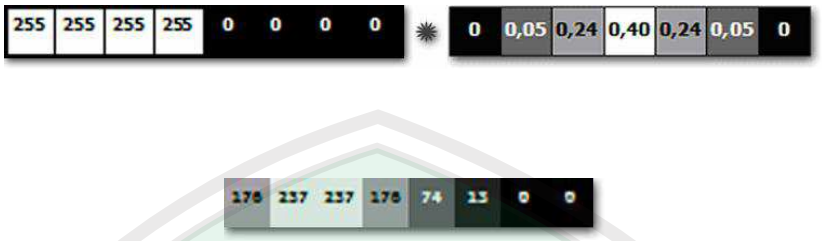
255	255	255	255	0	0	0	0

0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
12,8	0,0	0,0	0,0	0,0	0,0	0,0	0,0	12,8
61,2	12,8	0,0	0,0	0,0	0,0	0,0	0,0	74,0
102,0	61,2	12,8	0,0	0,0	0,0	0,0	0,0	176,0
61,2	102,0	61,2	12,8	0,0	0,0	0,0	0,0	237,2
12,8	61,2	102,0	61,2	0,0	0,0	0,0	0,0	237,2
0,0	12,8	61,2	102,0	0,0	0,0	0,0	0,0	176,0
0,0	0,0	12,8	61,2	0,0	0,0	0,0	0,0	74,0
0,0	0,0	0,0	12,8	0,0	0,0	0,0	0,0	12,8
0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0

Gambar 2.9 Konvolusi *bitmap* 8 *pixel* dengan kernel Gaussian

Dari gambar 2.9 dijelaskan bahwa hasil konvolusi dari *bitmap* 8 *pixel* adalah *bitmap* sepanjang 14 *pixel*. Kelebihan data akibat konvolusi dapat

dikurangi dengan mengurangi sama panjang bagian terluar sehingga hasil yang diperoleh bisa dilihat pada Gambar 2.10.



Gambar 2.7*Bitmap* hasil pengurangan *pixel*

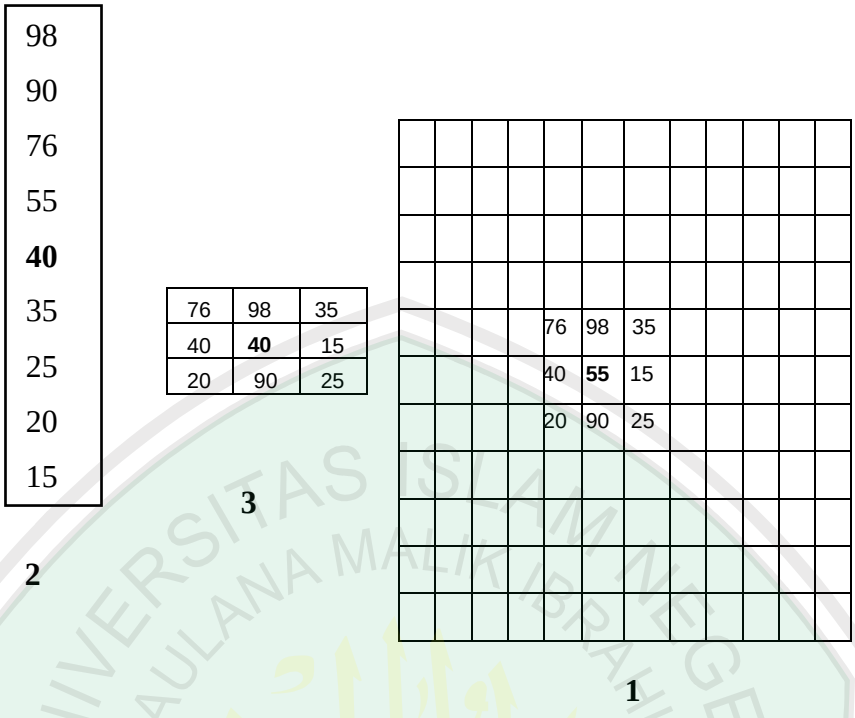
Diperlihatkan bahwa kernel filter Gaussian menyebabkan nilai-nilai tiap *pixel* saling mendekati satu sama lain dengan nilai *pixel* sebelahnya. Nilai-nilai ini pada saat diterjemahkan menjadi warna akan menghasilkan efek penghalusan.

2.2.5. Filter Median

Cara kerja filter median dalam jendela tertentu mirip dengan filter linier namun prosesnya bukan lagi dengan pembobotan.

Rinaldi Munir (2004:126) menjelaskan filter median sebagai suatu jendela yang memuat sejumlah *pixel* ganjil. Jendela digeser titik demi titik pada seluruh daerah citra. Pada setiap pergeseran dibuat jendela baru. Titik tengah dari jendela ini diubah dengan nilai median dari jendela tersebut.

Berikut disajikan ilustrasi penggunaan filter median berukuran 3 x 3 *pixel* terhadap bitmap 2 dimensi.



Gambar 2.8 Ilustrasi penerapan filter median berukuran 3x3 *pixel*

Cara mencari nilai median di atas adalah:

1. Baca nilai *pixel* yang akan di proses beserta *pixel-pixel* tetangganya
2. Urutkan nilai-nilai *pixel* dari yang paling kecil hingga yang paling besar
3. Pilih nilai pada bagian tengah untuk nilai yang baru bagi *pixel* (x,y)

Median pada kelompok tersebut adalah 40 (cetak tebal). Titik tengah dari jendela (55) diganti dengan nilai median (40). Jadi, filter median menghilangkan nilai *pixel* yang sangat berbeda dengan *pixel* tetangganya.

Filter median sebenarnya merupakan pendekatan alternatif yang digunakan untuk mengatasi kelemahan *window averaging*. Teknik ini bekerja dengan cara mengisi nilai dari setiap *pixel* dengan nilai median tetangganya. Proses pemilihan

median ini diawali dengan terlebih dahulu mengurutkan nilai-nilai *pixel* tetangga, baru kemudian dipilih nilai tengahnya (median). (Desi Alex Lestari, 2003:30)

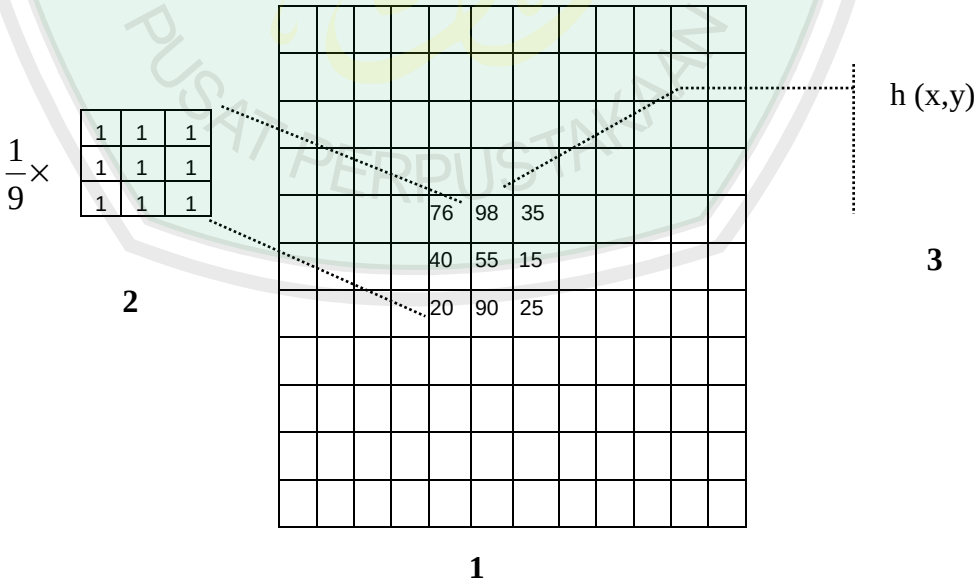
2.2.6. Filter Rata-Rata

Menurut Usman Ahmad (2005:61) salah satu filter linier adalah filter rata-rata dari intensitas pada beberapa *pixel* lokal dimana setiap *pixel* akan digantikan nilainya dengan rata-rata dari nilai intensitas *pixel* tersebut dengan *pixel-pixel* tetangganya, dan jumlah *pixel* tetangga yang dilibatkan tergantung pada filter yang dirancang. Secara matematis, hal ini dapat dinyatakan dengan persamaan :

$$h(x,y) = \frac{1}{M} \sum_{(k,l) \in N} f(k,l) \qquad (2.3)$$

M = jumlah *pixel* pada jendela sebesar N*N

Operasi konvolusi pada persamaan (2.1) berubah bentuk menjadi perhitungan rata-rata intensitas *pixel-pixel* lokal seperti diperlihatkan pada persamaan (2.3) dan diperlihatkan pada gambar 2.12



Gambar 2.9 Ilustrasi penggunaan filter rata-rata berukuran 3x3 *pixel*

Cara mencari nilai mean di atas adalah:

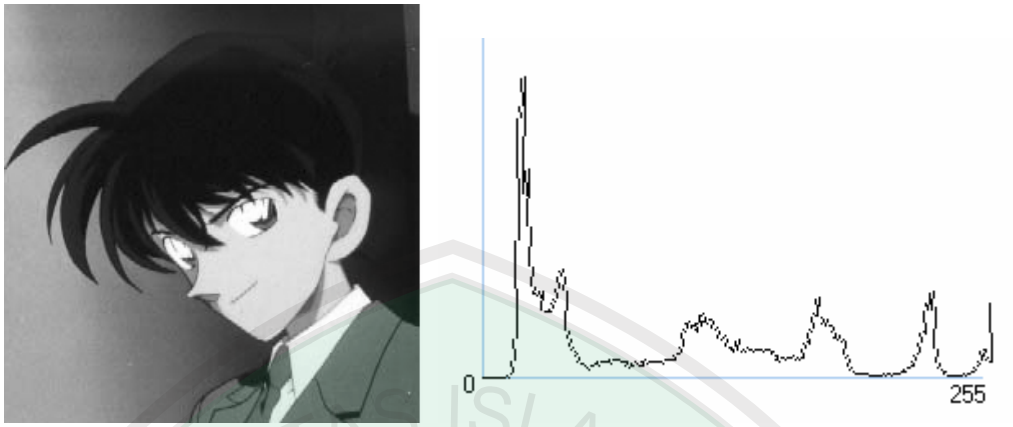
1. Baca nilai *pixel* yang akan di proses beserta *pixel-pixel* tetangganya
2. Gantikan nilai *pixel* denga bobot rata-rata (pada kernel ukuran 3 x 3 bobot maks adalah 1/9 dan 1/25 untuk kernel 5 x 5)
3. Pilih nilai pada bagian tengah untuk nilai yang baru bagi *pixel* (x,y)

Jumlah pembobotan pada filter rata-rata adalah satu, dengan demikian karakteristik global dari citra dapat tetap terpelihara.

2.2.7. Histogram

Informasi penting mengenai isi citra digital dapat diketahui dengan membuat histogram citra. Rinaldi Munir (2004:95) menyatakan bahwa histogram citra adalah grafik yang menggambarkan penyebaran nilai-nilai intensitas *pixel* dari suatu citra atau bagian tertentu di dalam citra. Dari sebuah histogram dapat diketahui frekuensi kemunculan nisbi (*relative*) dari intensitas pada citra tersebut. Histogram juga dapat menunjukkan banyak hal tentang kecerahan (*brightnees*) dan kontras (*contrast*) dari sebuah gambar. Karena itu, histogram adalah alat bantu yang berharga dalam pekerjaan pengolahan citra baik secara kualitatif maupun kuantitatif.

Berikut contoh gambar Kotaro beserta histogramnya



(a) Kotaro 256 x 256 (b) Histogram citra Kotaro

Gambar 2.10 Citra Kotaro dan histogramnya

2.3. *Noise Salt & Pepper*

Noise adalah variasi intensitas yang bersifat acak yang berarti gangguan pada citra. Salah satu jenis *noise* yang sering dijumpai adalah *noise salt and pepper*. *Noise salt and pepper* mengandung *pixel-pixel* intensitas gelap dan terang yang bersifat acak sehingga akan mengotori obyek maupun latar belakang citra. (Usman Ahmad, 2005). Berikut contoh citra dengan *noise salt and pepper*.

Gambar 2.11 Citra dengan *noise salt and pepper*

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

Dalam citra digital terdapat suatu standar pengukuran error (galat) kualitas citra, yaitu besaran PSNR dan MSE.

Tingkat keberhasilan dan performa dari suatu metode *filtering* pada citra dihitung dengan menggunakan *Peak Sinyal to Noise Ratio* atau biasa disingkat dengan PSNR. Meskipun performa metode *filtering* juga dapat diukur dengan teknik visual (hanya melihat pada citra hasil dan membandingkannya dengan citra yang terdapat *noise*). Namun hasil pengukuran teknik visual setiap orang berbeda – beda. Sehingga MSE dan PSNR merupakan solusi pengukuran performa yang baik.

PSNR adalah ukuran rasio antara kekuatan maksimum sinyal yang mungkin dan kekuatan sinyal yang telah rusak. Dikarenakan beberapa sinyal mempunyai pola data yang berubah-ubah, PSNR biasanya dinyatakan dalam skala *dicibel* dalam bentuk logaritma.

PSNR secara umum digunakan untuk mengukur kualitas pada penyusunan ulang citra. Hal ini lebih mudah didefinisikan dengan *mean square error* (MSE), misal I (x, y) adalah citra masukan dan I’ (x, y) adalah citra keluaran, keduanya memiliki M baris dan N kolom, maka didefinisikan sebagai berikut :

$$MSE = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x,y) - I'(x,y)]^2 \qquad (2.4)$$

Rumus untuk menghitung PSNR adalah :

PSNR = 20 * log10 (255 / sqrt (MSE)) (2.5)

Dimana : x = ukuran baris dari citra

y = ukuran kolom dari citra

I = matriks citra awal

I' = matriks citra hasil

Satuan PSNR adalah dB (*decibel*). Semakin besar nilai PSNR maka citra dapat dikatakan telah berkurang *noise*nya. (Desy Intan Permatasari, 2007:10)

2.6. Manusia dan Citra Kesempurnaan

Sebagaimana yang telah disebutkan dalam bab 1 tentang penciptaan manusia dalam bentuk yang sebaik-baiknya (QS. Al-Tiin ayat 4) sekiranya perlu ditambahkan disini pada bab 2 penjelasan yang lebih dalam sebagai justifikasi dan afirmasi terhadap penggunaan dalil tersebut, khususnya kata *khalaqna* dan *taqwim.* Allah berfirman dalam ayat tersebut sebagaimana kalimat berikut:

لَقَدْ خَلَقْنَا الإِنْسَانَ فِي أَحْسَنِ تَقْوِيمٍ

Artinya: Sesungguhnya Kami telah menciptakan manusia dalam bentuk yang sebaik-baiknya. (QS. At-Tin: 4)

Sebagaimana firman Allah swt di atas, Allah telah menciptakan manusia dalam bentuk yang sebaik-baiknya, dengan perawakan yang sempurna serta beranggotakan badan yang normal. (Abdullah bin Muhammad, 2006:500)

M.Qurash Shihab (2005:377) menjelaskan, kata *Khalaqna* (Kami telah menciptakan) terdiri atas kata *khalaqa* dan *na* yang berfungi sebagai kata ganti nama. Kata *na* (kami) yang menjadi kata ganti nama itu menunjuk kepada jamak (banyak), tetapi bisa juga digunakan untuk menunjuk satu pelaku saja dengan maksud menggabungkan pelaku tersebut. Para raja biasa menunjuk dirinya dengan menggunakan kata "kami". Allah juga sering menggunakan kata tersebut untuk menunjuk diri-Nya. Dari sisi lain, penggunaan kata ganti bentuk jamak itu (Kami) yang menunjuk kepada Allah mengisyaratkan adanya keterlibatan selain-Nya dalam perbuatan yang ditunjuk oleh kata yang dirangkaikan dengan kata ganti tersebut. Jadi, kata *khalaqna* mengisyaratkan keterlibatan selain Allah alam penciptaan manusia. Dalam hal ini adalah ibu bapak manusia. Di tempat lain Allah menegaskan bahwa Dia adalah *Ahsan al-Khaiqin* / sebaik-baik Pencipta (QS. Al-Mu'minun [23]:14). Ini menunjukkan bahwa ada pencipta lain, namun tidak sebaik Allah. Peranan yang lain itu sebagai "pencipta" sama sekali tidak seperti Allah, melainkan hanya sebagai alat atau perantara. Ibu bapak mempunyai peranan yang cukup berarti dalam penciptaan anak-anaknya, termasuk dalam penyempurnaan keadaan fisik dan psikisnya. Para ilmuan mengakui bahwa keturunan dan pendidikan merupakan dua faktor yang sangat dominan dalam pembentukan fisik dan kepribadian anak.

Kata *taqwim* diartikan sebagai *menjadikan sesuatu memiliki*, *qiwam* yakni bentuk fisik yang pas dengan fungsinya. Jadi, kalimat *ahsan taqwim* berarti bentuk fisik dan psikis yang sebaik-baiknya, yang menyebabkan manusia dapat melaksanakan fungsinya sebaik mungkin. Jika demikian, tidaklah tepat

memahami ungkapan *sebaik-baik bentuk* terbatas dalam pengertian fisik semata. Ayat ini dikemukakan dalam konteks penggambaran anugerah Allah kepada manusia dan tentu tidak mungkin anugerah tersebut terbatas pada bentuk fisik.

Sayyid Quthub dalam tafsir yang berjudul Dibawah Naungan Al Qur'an menjelaskan bahwasannya Allah telah menampakkan kenyataan atas karuniaNya yaitu dengan menciptakan manusia dalam keadaan yang sebaik-baiknya. Dikhususkan manusia dalam surah At-Tiin dan di beberapa surah dalam Al Qur'an, bahwa Ia paling baik susunannya, strukturnya dan paling baik kejadiannya. Surah At-Tiin memfokuskan perhatian terhadap makhluk manusia, karena karuniaNya itu.

BAB III

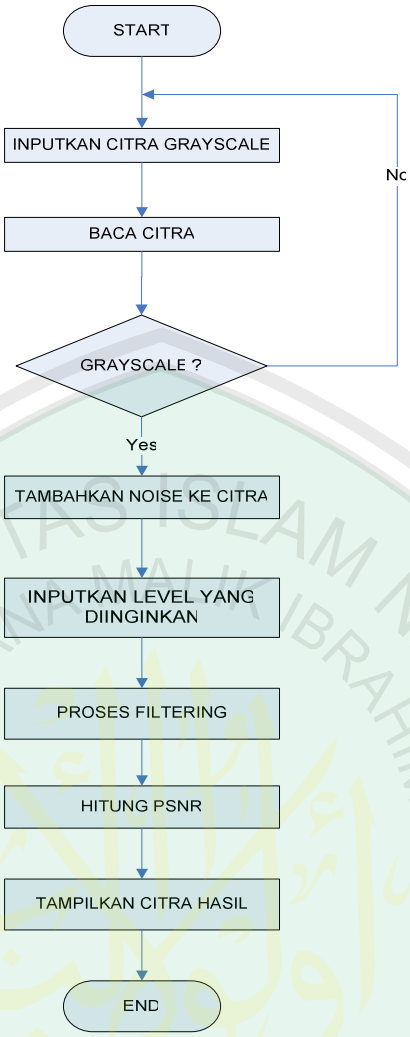
DESAIN DAN IMPLEMENTASI

Bab ini membahas tentang desain, implementasi desain dan metode *output* dalam pengurangan *noise* pada citra *grayscale*. Desain dan implementasi ini meliputi deskripsi sistem, desain data, desain proses dan implementai desain.

3.1. Deskripsi Sistem

Subbab ini akan membahas mengenai deskripsi sistem yang dikerjakan pada skripsi ini. Tujuan pembuatan sistem ini adalah untuk memperbaiki citra ber-*noise* dengan menggunakan tiga metode *output* yaitu filter Gaussian, mean dan median. Pada awalnya pengguna memasukkan *input* data berupa citra. Citra masukan adalah citra *grayscale* karena sistem hanya dibatasi untuk memproses citra *grayscale*. Kemudian pengguna diminta untuk memasukkan parameter untuk menambahkan *noise* pada citra. Jika parameter telah dimasukkan, maka sistem siap melakukan proses pengurangan *noise* citra.

Flowchart berikut merupakan implementasi dari metode penghilangan *noise* dengan menggunakan ketiga metode *output* di atas.



Gambar 3.1 Flowchart sistem secara keseluruhan

Sesuai dengan gambar 3.1, jika pengguna tidak memasukkan citra *grayscale* maka proses tidak dapat dilanjutkan dan sistem akan meminta untuk memasukkan citra dalam bentuk citra *grayscale*. Kemudian tambahkan *noise* pada citra *input* dan selanjutnya dilakukan proses pengurangan *noise* pada citra. Pada saat proses pengurangan *noise* maka PSNR dan MSE (tingkat *error*) citra juga

dihitung kemudian ditampilkan ke pengguna. Sehingga pengguna mengetahui apakah *output* yang dihasilkan telah berkurang *noise*nya atau tidak.

3.2. Desain Sistem

Pada subbab ini akan dijelaskan mengenai desain aplikasi sistem untuk implementasi metode *output.* Desain aplikasi ini meliputi desain data, algoritma yang digunakan dalam sistem yang digambarkan dengan flowchart dan desain proses. Desain data berisikan penjelasan data yang diperlukan untuk dapat menerapkan metode perbaikan citra ber*noise* ini. Desain data meliputi data masukan, data selama proses dan data keluaran. Desain proses antara lain menjelaskan tentang proses penambahan *noise* dan memasukkan tingkat intensitas *noise* yang akan diberikan pada citra, proses reduksi *noise* (*filtering citra*) dan proses pembentukan kembali sinyal menjadi citra.

3.2.1. Desain Data

Data yang digunakan untuk implementasi perangkat lunak ini dibagi menjadi tiga bagian utama, yaitu data masukan, data yang digunakan selama proses perbaikan citra ber-*noise* dan data keluaran.

3.2.1.1. Data Masukan

Data masukan yang pertama dari pengguna adalah arsip citra yang dipilih oleh pengguna. Pada sistem ini citra yang dimasukkan berupa arsip citra *grayscale* dengan format .bmp. Data masukan kedua adalah tingkat banyaknya *noise* yang akan ditambahkan pada citra masukan.

Daftar data masukan (daftar citra yang digunakan) pada sistem ini dapat dilihat pada tabel 3.1.

No	Nama Data	Jenis Data	Keterangan
1.	Fella	(bmp, *grayscale*)	Citra digunakan sebagai citra asli
2.	Bumi	(bmp, *grayscale*)	Citra digunakan sebagai citra asli
3.	Kotaro	(bmp, *grayscale*)	Citra digunakan sebagai citra asli

3.2.1.2. Data Selama Proses

Pada tahap proses *filtering* untuk memperbaiki citra ber-*noise* terdapat beberapa tahap yaitu penambahan *noise*, pemilihan matriks *input*, konvolusi matriks *input* dengan koefisien *filter* dan pembentukan matriks *output*.

Pada proses penambahan *noise*, dihasilkan satu data citra ber-*noise.* Data ini akan digunakan sebagai data *input* untuk proses selanjutnya, yaitu pemilihan matriks *input*. Pada proses ini matriks citra *noise* yang awalnya berukuran 2 dimensi (memiliki baris dan kolom) akan dipecah dan diambil per kolom. Sehingga proses ini akan menghasilkan data berupa matriks yang berukuran 1 kolom dan n baris. Data ini akan digunakan sebagai *input* untuk proses selanjutnya yaitu konvolusi dengan koefisien *filter*. Data yang dihasilkan adalah matriks, dan hasil ini akan digunakan untuk data pada proses berikutnya. Hal ini akan terus-menerus dilakukan hingga proses pembentukan sinyal *output*. Data pada setiap proses akan berukuran 1 dimensi, sehingga setiap kolom disimpan pada suatu matriks *temporary* yang berguna untuk menggabungkan seluruh matriks kolom.

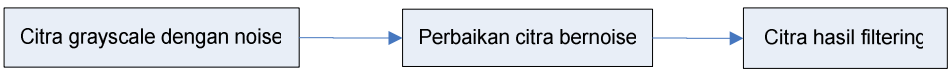
3.2.1.3. Data Keluaran

Data keluaran yang dihasilkan sistem adalah citra hasil *filtering* dengan metode *Gaussian filtering, mean filtering dan median filtering.* Data lain yang akan ditampilkan pada pengguna adalah nilai PSNR (*peak sinyal-to-noise ratio*) dan nilai MSE (*mean square error*) dari setiap citra hasil. Nilai PSNR dan MSE adalah ukuran yang digunakan untuk mengukur kualitas dalam pemrosesan citra, khususnya penghilangan *noise.*

3.3. Desain Proses

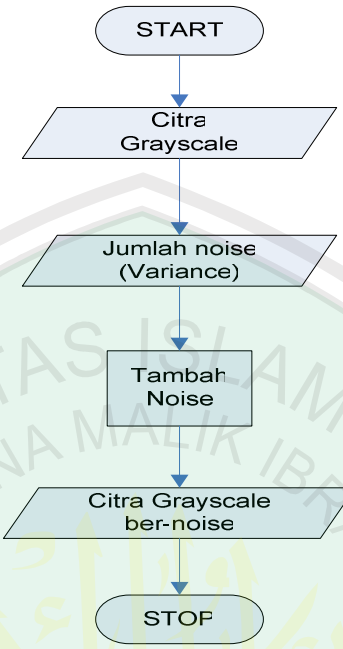
Desain proses digunakan untuk mengetahui proses apa saja yang berlangsung pada sistem. Desain proses untuk aplikasi ini menggunakan diagram alir flowchart. *Tools* yang akan digunakan untuk pembuatan diagram alir adalah Microsoft Visio 2003.

Diagram alir menunjukkan hubungan antar proses, data masukan, data selama proses dan data keluaran yang terlibat dalam sistem. Secara garis besar, jalannya sistem ini adalah pengguna memasukkan citra yang tanpa *noise*, kemudian sistem memberikan *noise* pada citra, sistem melakukan proses *filtering* pada citra yang telah ber*noise*, kemudian hasil *output* yang akan ditampilkan ke pengguna adalah citra hasil *filtering*. Jika diilustrasikan pada diagram alir dapat dilihat sebagai berikut :



Gambar 3.2 Diagram alir untuk sistem

3.3.1. Proses Penambahan *noise*



Gambar 3.3 Diagram alir proses penambahan *noise*

Pada proses ini dilakukan penambahan *noise* terhadap data citra yang telah dipilih pengguna. Pengguna memasukkan tingkat intensitas *noise* yang akan diberikan pada citra. Parameter yang diperlukan pada proses penambahan *noise* ini adalah nilai *variance*, semakin besar nilai *variance* maka citra akan semakin mengandung banyak *noise*. Jenis *noise* yang dikenakan pada citra adalah *noise salt and peppers*. Kisaran nilai variance yang diberikan pada sistem ini adalah 0.05, 0.1 dan 0.15. Jika proses penambahan *noise* telah dilakukan, maka akan dilanjutkan dengan proses berikutnya yaitu proses pemilihan matriks *input*.

3.3.2. Proses Pemilihan Matriks *Input*

Gambar 3.4 Diagram alir proses pemilihan matriks *input*

Proses ini bertujuan untuk mengubah matrik masukan 2 dimensi menjadi matrik 1 dimensi karena sistem ini akan diimplementasikan pada matrik 1 dimensi. Pada proses ini dilakukan pengambilan nilai matriks tiap kolom dari data matriks citra 2D hasil dari proses penambahan *noise*. Sehingga data yang dihasilkan berupa matriks yang berukuran 1 kolom dan n baris. Kemudian data

keluaran ini akan digunakan untuk *input* data proses berikutnya yaitu proses konvolusi matriks. Sehingga pada saat dilakukan proses *filtering*, data yang diolah berukuran 1 dimensi bukan 2 dimensi lagi. Urutan prosesnya dapat didefinisikan sebagai berikut :

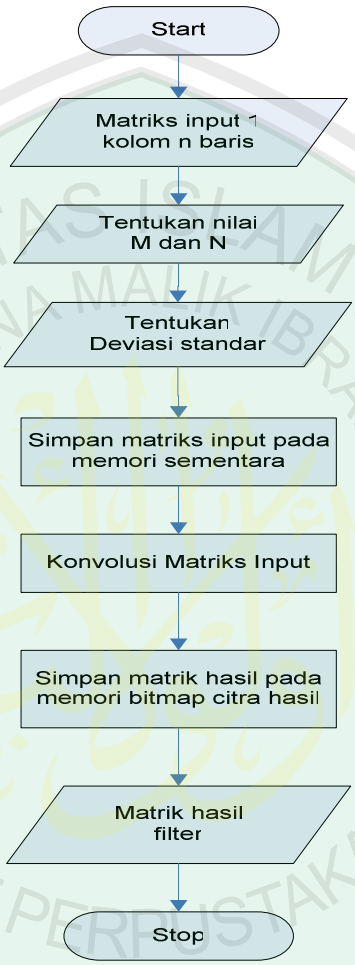
1. Hitung jumlah kolom matrik masukan
2. Ambil nilai tiap kolom dari matrik masukan dan simpan pada matrik sementara
3. Hasil keluaran berupa matrik 1 dimensi

3.3.3. Proses Konvolusi Matrik *Input* Dengan Filter Gaussian

Pada proses ini dilakukan konvolusi antara matriks *input* dengan koefisien filter Gaussian. Proses konvolusi yang dilakukan adalah konvolusi 1 dimensi (penjelasan mengenai konvolusi telah dijelaskan pada bab 2). Proses konvolusi ini akan dilakukan pada setiap kolom dari matriks citra 2D. Proses konvolusi merupakan salah satu proses yang penting karena proses ini juga merupakan proses yang mendasari perhitungan koefisien filter yang menjadi kunci utama dalam perbaikan citra ber-*noise*. Data yang dihasilkan dari proses konvolusi ini adalah matriks 1 dimensi (1 kolom, n baris). Urutan prosesnya dapat didefinisikan sebagai berikut :

1. Deklarasikan matrik masukan 1 dimensi dari proses sebelumnya.
2. Tentukan konstanta M dan N untuk menentukan batas konvolusi (persamaan 2.1)
3. Tentukan deviasi standar dengan menggunakan persamaan 2.2
4. Simpan matrik *input* pada memori sementara

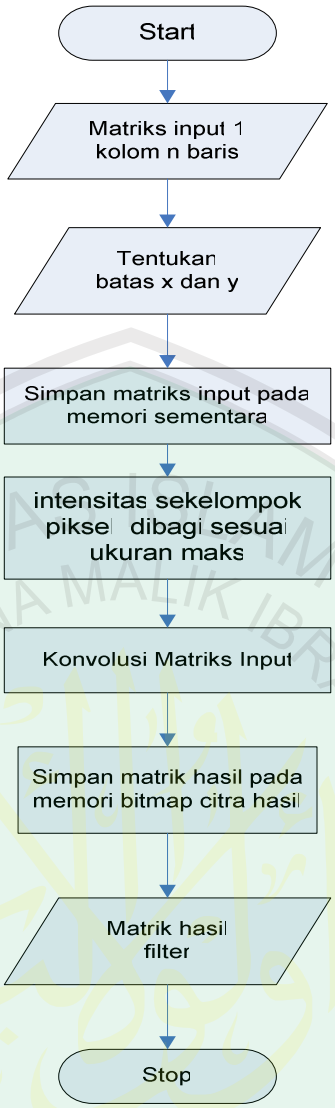
5. Lakukan proses konvolusi matrik masukan
6. Simpan matrik hasil pada memori bitmap citra hasil
7. Dihasilkan matrik hasil filter Gaussian



Gambar 3.5 Diagram alir proses filter Gaussian

3.3.4. Proses Konvolusi Matrik *Input* Dengan Filter Mean

Pada proses ini tidak jauh berbeda dengan proses konfolusi pada filter Gaussian. Berikut diagram alir dan urutan prosesnya:



Gambar 3.6 Diagram alir proses filter mean

Urutan prosesnya dapat didefinisikan sebagai berikut :

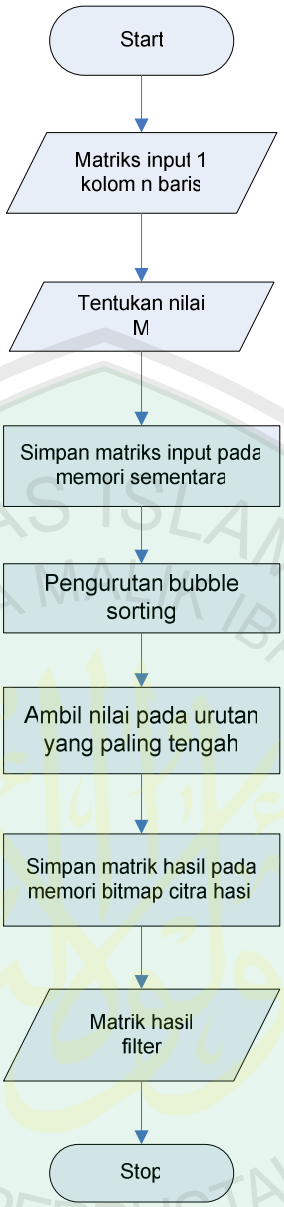
1. Deklarasikan matrik masukan 1 dimensi dari proses sebelumnya
2. Tentukan batas x dan y untuk menentukan batas konvolusi
3. Simpan matrik *input* pada memori sementara
4. Bagi intensitas sekelompok *pixel* sesuai ukuran maks
5. Lakukan proses konvolusi matrik masukan

6. Simpan matrik hasil pada memori bitmap citra hasil
7. Dihasilkan matrik hasil filter mean

3.3.5. Proses Pengolahan Matrik *Input* Dengan Filter Median

Pada proses filter median tidak terjadi proses konvolusi, karena pada operasi filter median tidak ada bobot yang dipakai. Nilai-nilai keabuan dari titik-titik di dalam jendela diurutkan dari nilai terkecil sampai nilai terbesar kemudian dihitung mediannya, seperti yang telah dijelaskan pada bab 2.

Berikut diagram alir dari proses filter median:



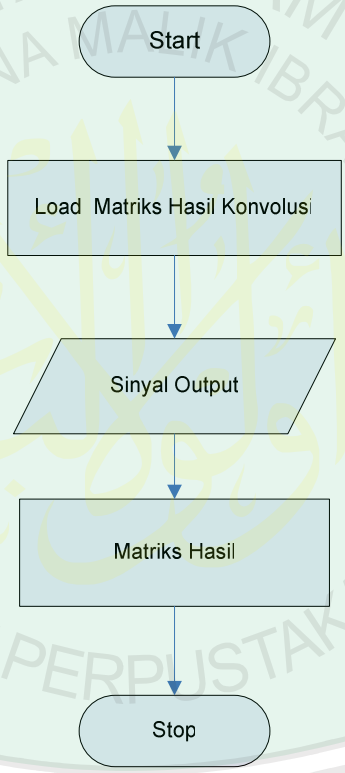
Gambar 3.7 Diagram alir proses filter median

Urutan prosesnya dapat didefinisikan sebagai berikut:

1. Deklarasikan matrik masukan 1 dimensi dari proses sebelumnya.
2. Tentukan nilai M untuk menentukan jarak tetangga yang diikutkan dalam maks median

3. Simpan matrik *input* pada memori sementara
4. Urutkan nilai matrik *input* dengan *buble shorting*
5. Ambil nilai pada urutan yang paling tengah
6. Simpan matrik hasil pada memori bitmap citra hasil
7. Dihasilkan matrik hasil filter median

3.3.6. Proses Pembentukan Sinyal *Output*



Gambar 3.8 Diagram alir proses pembentukan sinyal

Pada proses ini akan dilakukan pembentukan sinyal dengan cara memasukkan matriks hasil konvolusi, selanjutnya data sinyal *output* dimasukkan dan akan didapatkan matriks hasil.

3.4. Implementasi Desain

Subab ini membahas proses pembuatan sistem yang telah dirancang pada bab desain sistem. Pembuatan perangkat lunak dibagi ke dalam 3 bagian utama, yaitu:

1. Penambahan *noise*
2. Pemilihan matriks *input*
3. Filter Gaussian, mean dan median

3.4.1. Penambahan *noise*

Proses penambahan *noise* digunakan untuk menambahkan *noise* pada citra asli yang telah dipilih oleh pengguna. Proses penambahan *noise* yang digunakan pada sistem ini menggunakan variabel Derau yang digunakan untuk menentukan *variance* (porsi banyaknya titik yang mengandung derau). Jenis *noise* yang digunakan pada sistem ini adalah *salt and peppers noise*. Potongan kode program untuk menambahkan *noise* pada citra sebagai berikut :

```
Randomize;
Derau :=  StrToFloat(EditDerau.Text);
if (Image1.Picture.Bitmap.PixelFormat = pf8bit)
  then
    begin
      temp := Random;
                if (temp < Derau/2) then
                  Ko[x, y] := 0
                else if (temp > 1-(Derau/2)) then
                  Ko[x, y] := 255
                else
                  Ko[x, y] := Ki[x, y];
              end;
     end;
end;
```

Pertama periksa apakah format citra pada *Picture* adalah keabuan (pf8bit) kemudian tentukan porsi banyaknya derau dan jalankan fungsi Randomize agar hasil perandoman untuk membangkitkan derau selalu berbeda setiap kali *procedure* dijalankan. Kemudian, ambil nilai deviasi standar atau porsi derau yang terdistribusi ke dalam tiga daerah yaitu daerah berderau *Pepper* (nilai 0), daerah tidak berderau dan daerah berderau *Salt* (nilai 255).

3.4.2. Pemilihan Matriks *Input*

Proses pemilihan matriks *input* dilakukan dengan cara menghitung ukuran matriks data masukan, berapa baris dan berapa kolom. Setelah diketahui ukuran kolom, maka iterasi akan dilakukan sebanyak jumlah kolom. Kemudian ambil nilai matriks pada tiap kolom, simpan pada matriks sementara dan akan dikenai proses *filtering*. Algoritma dari proses pemilihan matrik *input* telah dijelaskan secara detail pada subbab 3.3.2 Potongan kode program untuk proses pemilihan matriks *input* adalah sebagai berikut :

```
Begin
      SetLength (Ki, w, h);
      SetLength(Ko, w, h);
       for y := 0 to h-1 do
        begin
          PC :=  Image3.Picture.Bitmap.ScanLine[y];
          PH :=  Image2.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            begin
              Ki[x, y] := PC[x];
              Ko[x, y] := PH[x];
            end;
       end;
end;
```

3.4.3. Proses Filter Gaussian

Pada subbab ini akan dibahas mengenai proses *filtering* dengan metode Gaussian. Berikut potongan kode program filter Gaussian:

```
for x := M to w-1-M do
for y := N to h-1-N do
            begin
              jumlah := 0;
              for u := -M to M do
                for v := -N to N do
                 jumlah := jumlah+Mask[u,v]*Ki[x-u,y-v];
              Ko[x,y] := Round(jumlah);
            end;
```

Dengan menentukan maks yang digunakan yaitu konstanta M dan N, operasi filter Gaussian dilakukan dengan perhitungan *sum of producst* sesuai dengan persamaan 2.1.

3.4.4. Proses Filter Mean

Pada subbab ini akan dibahas tentang kode-kode program pada filter mean. Potongan kode programnya (maks 3x3) adalah sebagai berikut:

```
 for x := 1 to w-2 do
 for y := 1 to h-2 do
begin;
   Ko[x,y] := Round((Ki[x-1,y-1]+Ki[x,y-1]
     +Ki[x+1,y-1]+Ki[x-1,y]+Ki[x,y]+Ki[x+1,y]
     +Ki[x-1,y+1]+Ki[x,y+1]+Ki[x+1,y+1])/9);
end;
```

Pada operasi ini memerlukan nilai keabuan titik-titik tetangganya, maka titik yang berada pada tepi citra (titik-titik paling atas, kiri, kanan atau bawah)

tidak memiliki tetangga. Oleh karena itu, konvolusi hanya dilakukan pada titik kedua dari tepi (lihat perintah for x dan for y). Karena maks yang digunakan adalah 3x3 maka perhitungan nilai intensitas sekelompok *pixel* dibagi 9. Lebih detailnya operasi filter mean dijelaskan pada bab 2.

3.4.5. Proses Filter Median

Jalannya prose filter median adalah sebagai berikut :

```
for x := M to w-1-M do
  for y := M to h-1-M do
    begin
      for u := -M to M do
       for v := -M to M do
        Urutan[(2*M+1)*u+v+((2*M+1)*(2*M+1)+1) div 2]
          := Ki[x-u,y-v];
```

Potongan program di atas menjelaskan pengambilan nilai keabuan dari titik-titik yang masuk dalam jangkauan maks median, yaitu dari –M sampai M dan selanjutnya disimpan pada variabel Urutan untuk selanjutnya diurutkan.

```
      for i := 2 to (2*M+1)*(2*M+1) do
        for j := (2*M+1)*(2*M+1) downto i do
          if (Urutan[j] < Urutan[j-1]) then
            begin
              temp      := Urutan[j];
              Urutan[j]   := Urutan[j-1];
              Urutan[j-1] := temp;
            end;
      Ko[x,y] := Urutan[((2*M+1)*(2*M+1)+1) div 2];
```

Kode program di atas menjelaskan teknik pengurutan yang digunakan adalah *bubble sorting* karena teknik ini merupakan teknik pengurutan data (nilai intensitas sekelompok *pixel*) yang paling sederhana. Nilai (2*M+1)*(2*M+1) adalah total banyaknya titik dalam maks median.

Setelah diurutkan, ambil nilai pada urutan yang paling tengah, yaitu ((2*M+1)*(2*M+1)+1) div 2. nilai inilah yang merupakan nilai median dan kemudian diambil sebagai keabuan hasil. Berikut kode progamnya:

```
Ko[x,y] := Urutan[((2*M+1)*(2*M+1)+1) div 2];
```

3.4.6. Proses Pembentukan Sinyal *Output*

Untuk selanjutnya matriks Ko di-*Copy*-kan ke dalam memori data bitmap citra hasil. Kode programnya adalah sebagai berikut:

```
for y := 0 to h-1 do
  begin
    PH :=  Image6.Picture.Bitmap.ScanLine[y];
    for x := 0 to w-1 do
     PH[x] := Ko[x, y];
  end;
Ki := nil;
Ko := nil;
```

Setelah Ki dan Ko tidak diperlukan lagi, bebaskn memori yang telah di pesan tadi dengan memberi nilai Nil pada kedua variabel tersebut.

BAB IV

HASIL DAN PEMBAHASAN

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui apakah sistem dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba.

Sebelumnya perlu diketahui lingkungan uji coba yang digunakan dalam melakukan uji coba dalam skripsi ini.

4.1. Lingkungan Uji Coba

Pada subbab ini dijelaskan mengenai lingkungan uji coba yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam uji coba antara lain adalah :

Tabel 4.1 Lingkungan uji coba

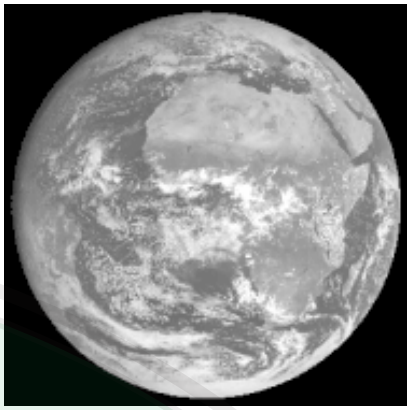
Perangkat Keras	Prosesor : Intel Celeron 2.66 GHz Memori : 256 MB
Perangkat Lunak	Sistem Operasi : Microsoft Windows XP Professional 2002 SP 2 Perangkat Pengembang : Delphi 0.7

4.2. Data Uji Coba

Citra yang digunakan dalam perbaikan citra ber*noise* ini berfomat bmp dan terbatas hanya pada citra *grayscale*. Uji coba yang digunakan menggunakan citra Fella.bmp (180*218), bumi.bmp (187*187) dan kotaro.bmp (256*256). Untuk masing-masing citra akan memberikan hasil MSE dan PSNR yang berbeda. Hasil uji coba akan memberikan nilai berupa MSE (*mean-square-error*) dan PSNR (*peak sinyal-to-noise-ratio*). MSE digunakan untuk menghitung beda (kesalahan) antara citra masukan dan citra keluaran, sedangkan PSNR digunakan untuk menghitung rasio citra keluaran terhadap *noise.*

Hasil uji coba pada bab ini akan ditampilkan berupa hasil uji coba citra keluaran serta tabel nilai PSNR dan MSE. Hal ini dilakukan supaya pengamatan terhadap citra keluaran lebih leluasa.

(a)



(b)

(c)

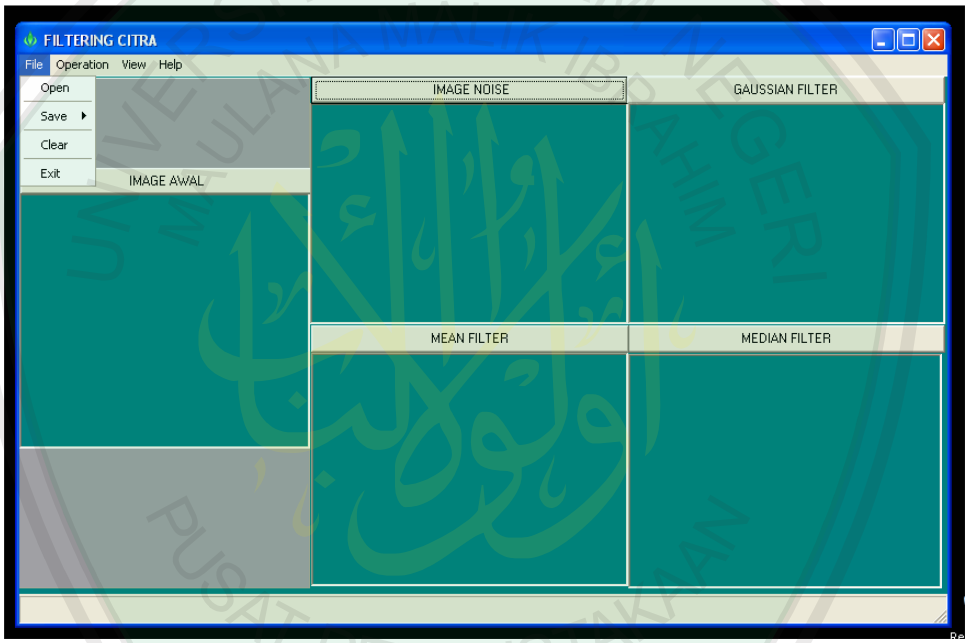
Gambar 4.1 Citra yang digunakan dalam ujicoba;

(a) Fella.bmp; (b) bumi.bmp; (c) kotaro.bmp

4.3. Hasil Uji Coba

Berikut ini adalah seluruh hasil uji coba yang menunjukkan kinerja dari algoritma *filtering citra* untuk perbaikan citra ber-*noise*. Hasil uji coba akan menunjukkan hasil MSE dan PSNR yang berbeda-beda untuk gambar yang berbeda-beda. Hasil dari uji coba ini akan digunakan untuk menarik kesimpulan pada skripsi ini.

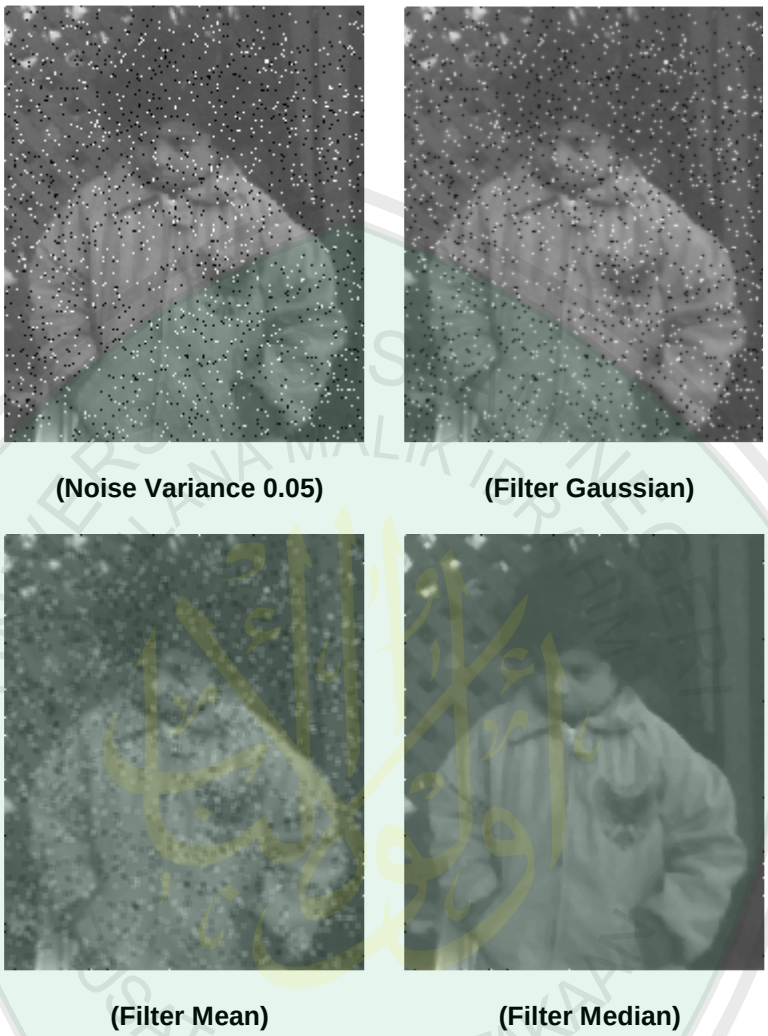
4.3.1. Tampilan Awal Program



Gambar 4.2 Tampilan awal program

4.3.2. Uji coba citra input dengan *noise variance* 0.05 dengan kernel 3 x 3

4.3.2.1. Citra Fella dengan ukuran 180*218

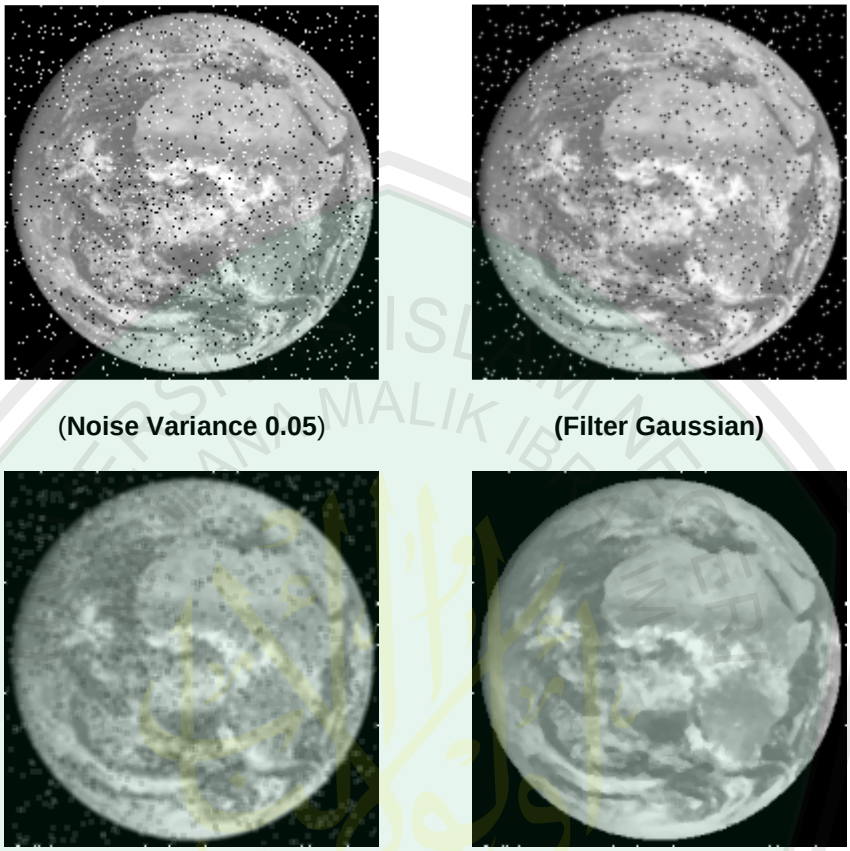


Gambar 4.3 Citra Fella dengan ukuran 180*218 (kernel 3 x 3)

Tabel 4.2 Hasil Uji coba citra Fella *Noise variance* 0.05

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	116.320514780822	106.611544342492	8.03366462793039
PSNR	27.474240453621	28.8527612641392	39.0816666308133

4.3.2.2. Citra bumi dengan ukuran 187*187



(Noise Variance 0.05) (Filter Gaussian)

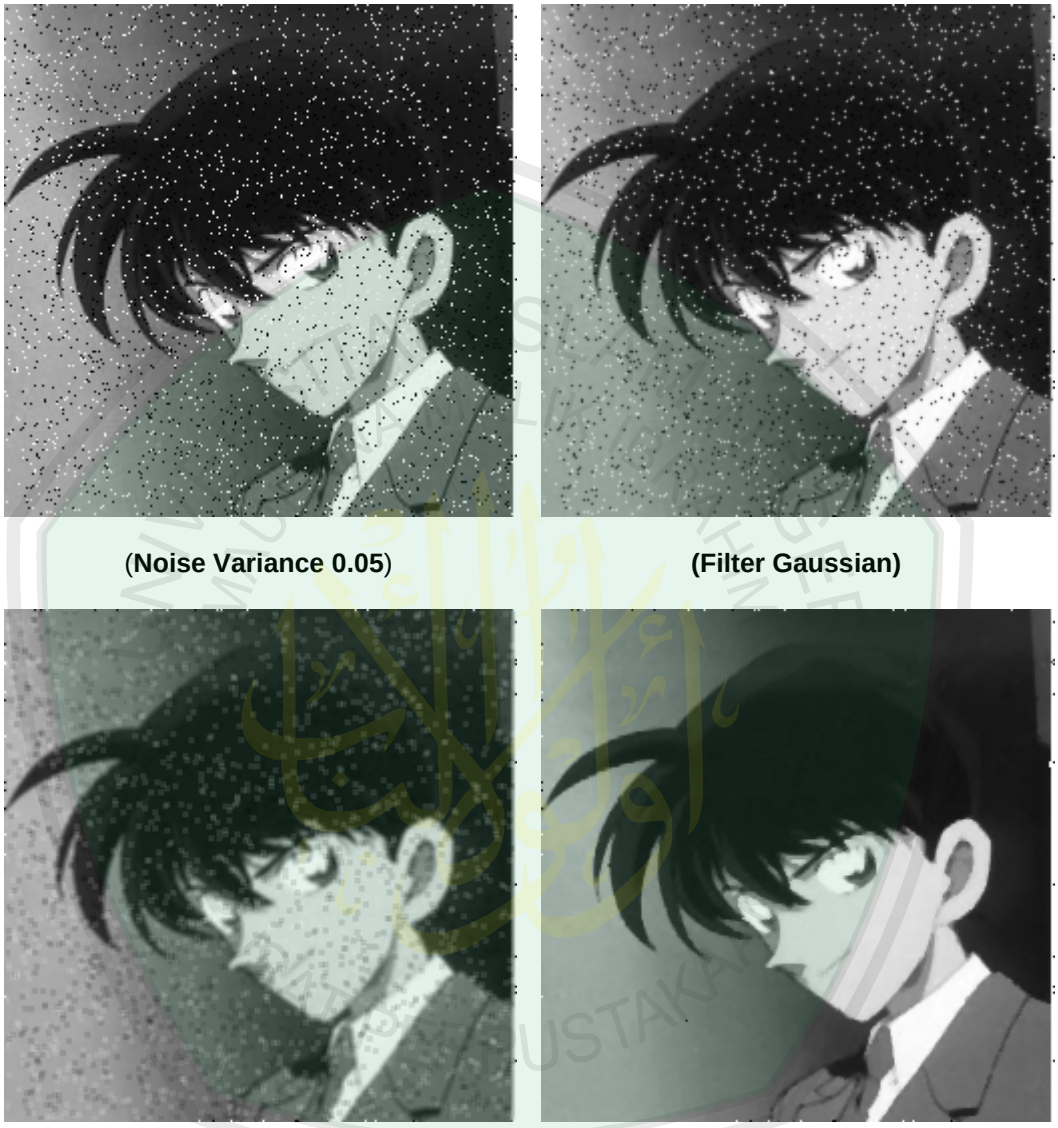
(Filter Mean) (Filter Median)

Gambar 4.4 Citra Bumi dengan ukuran 187*187 (kernel 3 x 3)

Tabel 4.3 Hasil Uji coba citra bumi *Noise variance* 0.05

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	249.1062655449491	262.037804912927	82.5589522148242
PSNR	24.1669570976918	23.9471640817165	28.9631618833632

4.3.2.3. Citra Kotaro dengan ukuran 256*256



(Noise Variance 0.05) (Filter Gaussian)

(Filter Mean) (Filter Median)

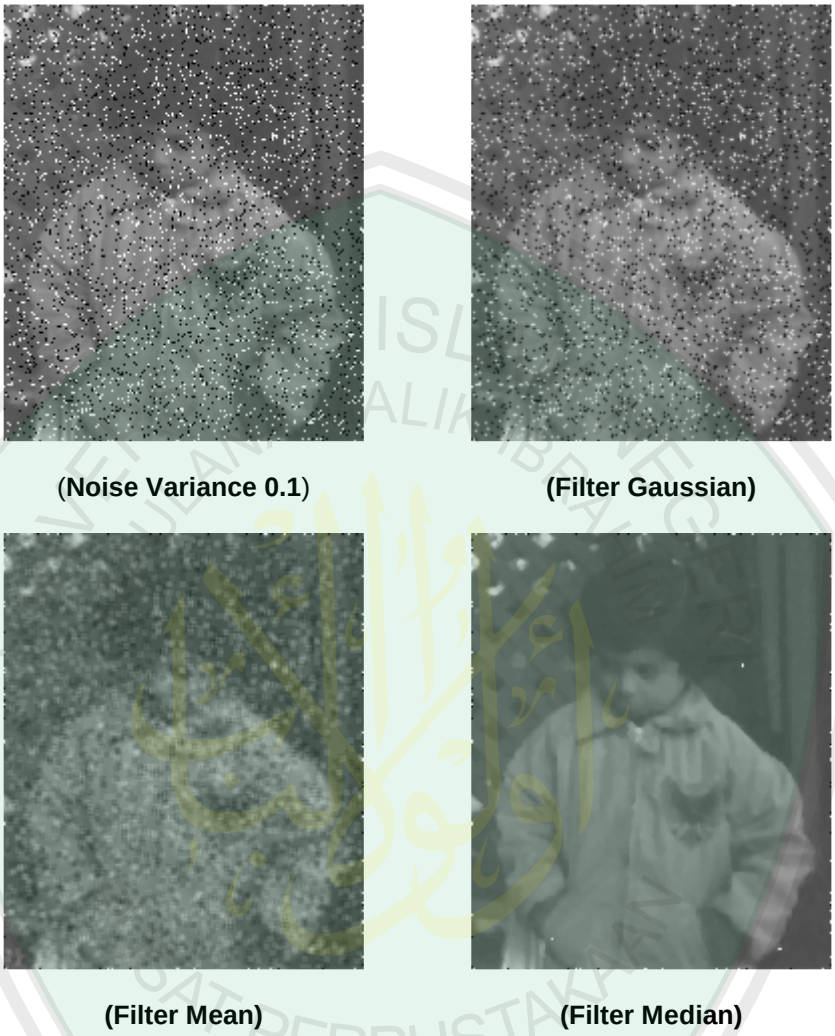
Gambar 4.5 Citra Kotaro dengan ukuran 256*256 (kernel 3 x 3)

Tabel 4.4 Hasil Uji coba citra Kotaro *Noise variance* 0.05

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	204.360015869141	212.52207946773	25.3812713623047
PSNR	25.0268443313347	24.8567630416211	34.08566988504

4.3.3. Uji coba citra input dengan *noise variance* 0.1

4.3.3.1. Citra Fella dengan ukuran 180*218

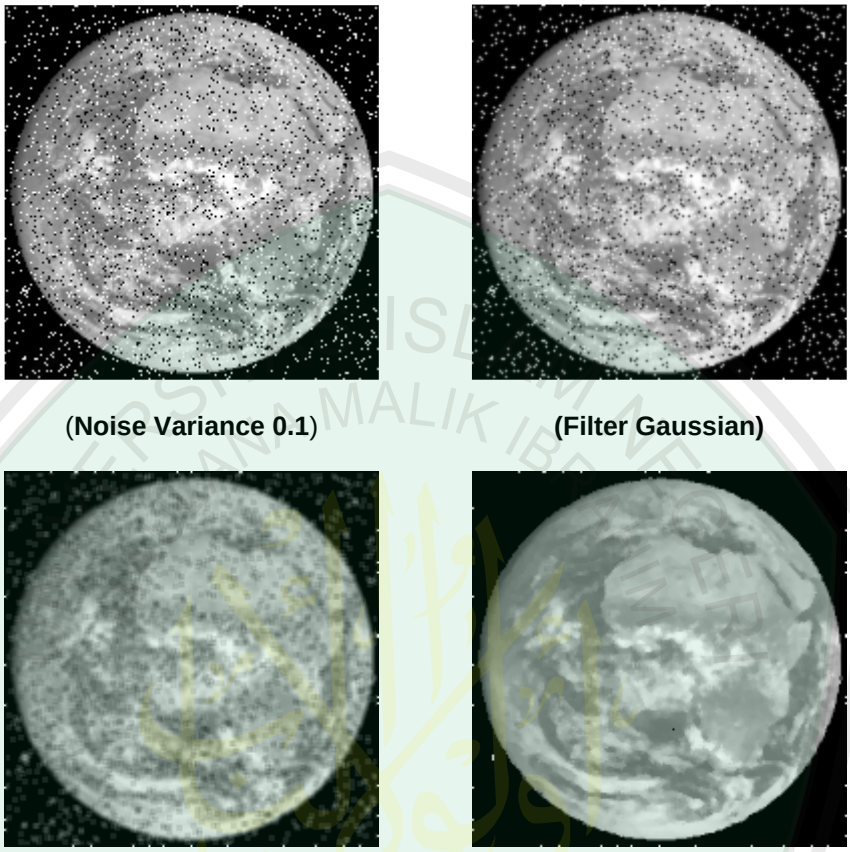


Gambar 4.6 Citra Fella dengan ukuran 180*218 (kernel 3 x 3)

Tabel 4.6 Hasil Uji coba citra Fella *Noise variance* 0.1

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	221.878567787966	199.996177370021	9.34910805300653
PSNR	24.669650069953	25.1205866601879	38.423101816676

4.3.3.2. Citra bumi dengan ukuran 187*187



(Noise Variance 0.1) (Filter Gaussian)

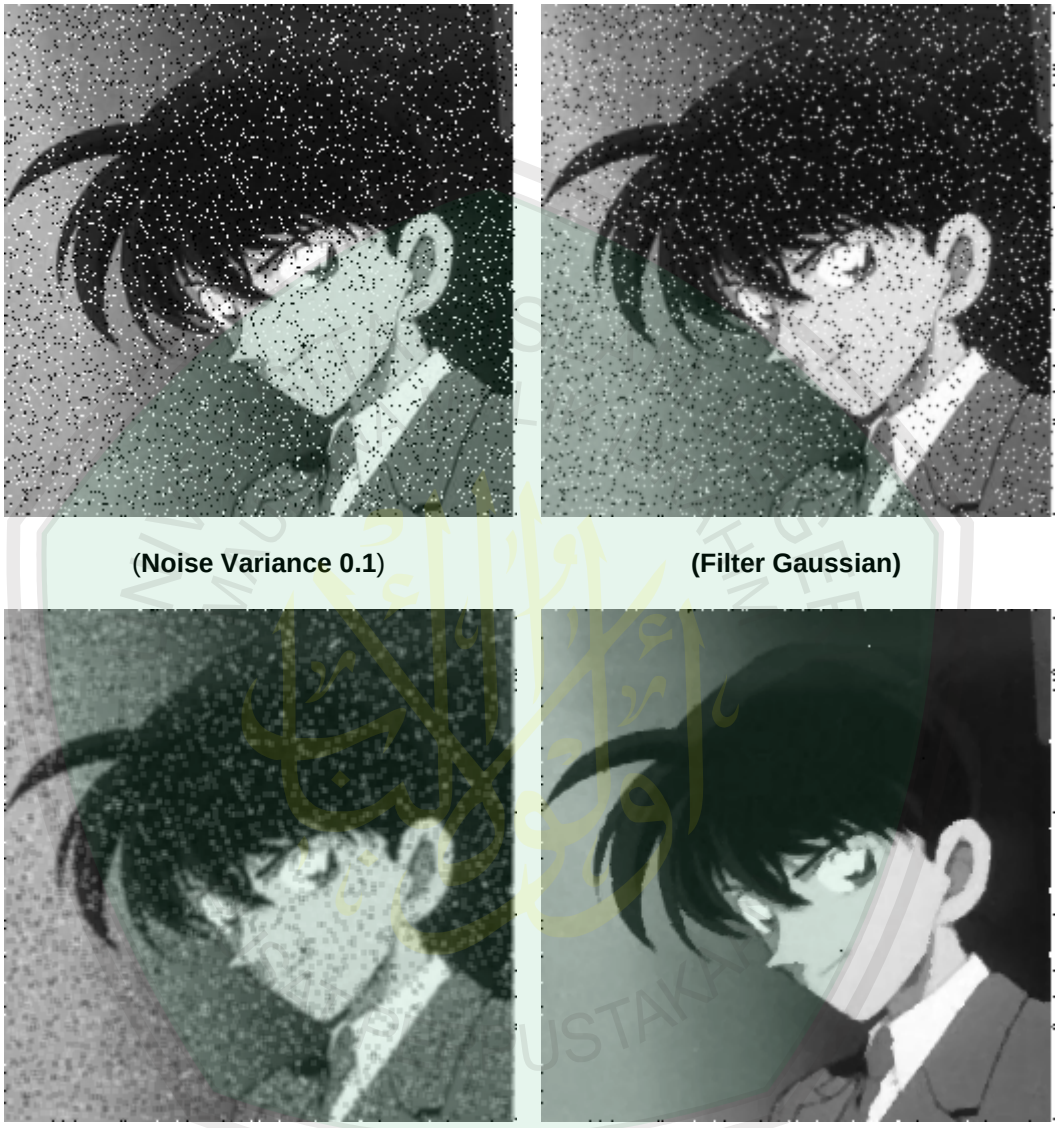
(Filter Mean) (Filter Median)

Gambar 4.7 Citra Bumi dengan ukuran 187*187 (kernel 3 x 3)

Tabel 4.7 Hasil Uji coba citra bumi *Noise variance* 0.1

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	426.720037747757	424.594469387197	97.0169578769881
PSNR	21.8403186199123	21.8510602796272	28.2623270836751

4.3.3.3. Citra Kotaro dengan ukuran 256*256



(Noise Variance 0.1)

(Filter Gaussian)

(Filter Mean)

(Filter Median)

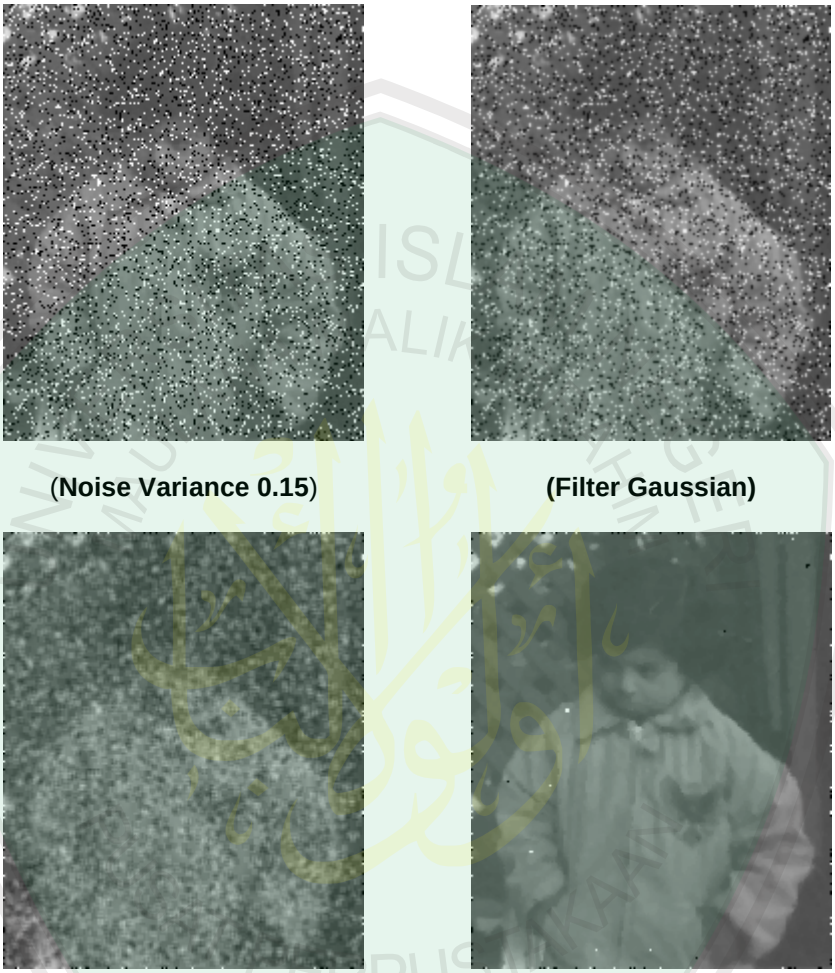
Gambar 4.8 Citra Kotaro dengan ukuran 256*256 (kernel 3 x 3)

Tabel 4.8 Hasil Uji coba citra Kotaro *Noise variance* 0.1

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	383.302734375	376.010208129883	41.9387512207031
PSNR	22.2953844346658	22.3788072531711	31.9046486621288

4.3.4. Uji coba citra input dengan *noise variance* 0.15

4.3.4.1. Citra Fella dengan ukuran 180*218



(Noise Variance 0.15)　　(Filter Gaussian)

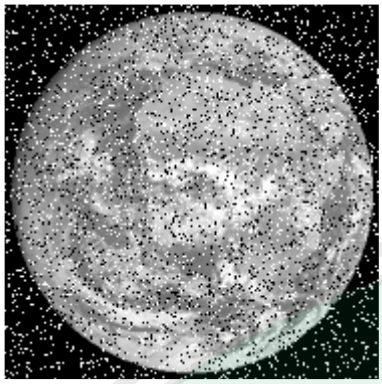
(Filter Mean)　　(Filter Median)

Gambar 4.9 Citra Fella dengan ukuran 180*218 (kernel 3 x 3)

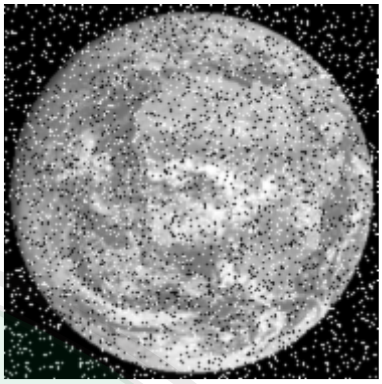
Tabel 4.9 Hasil Uji coba citra Fella *Noise variance* 0.15

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	333.651325178373	301.638379204888	17.6879459734941
PSNR	22.8978750677025	23.335937623431	35.6540295781519

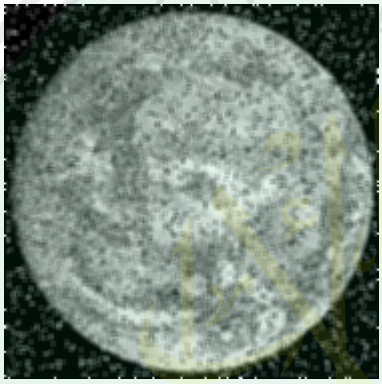
4.3.4.2. Citra bumi dengan ukuran 187*187



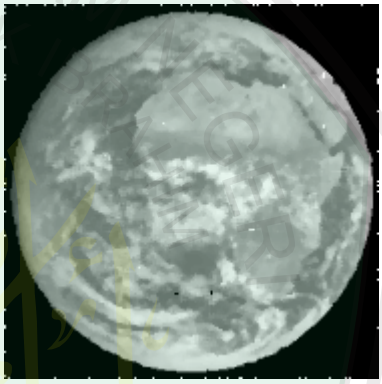
(**Noise Variance 0.15**)



(Filter Gaussian)



(Filter Mean)



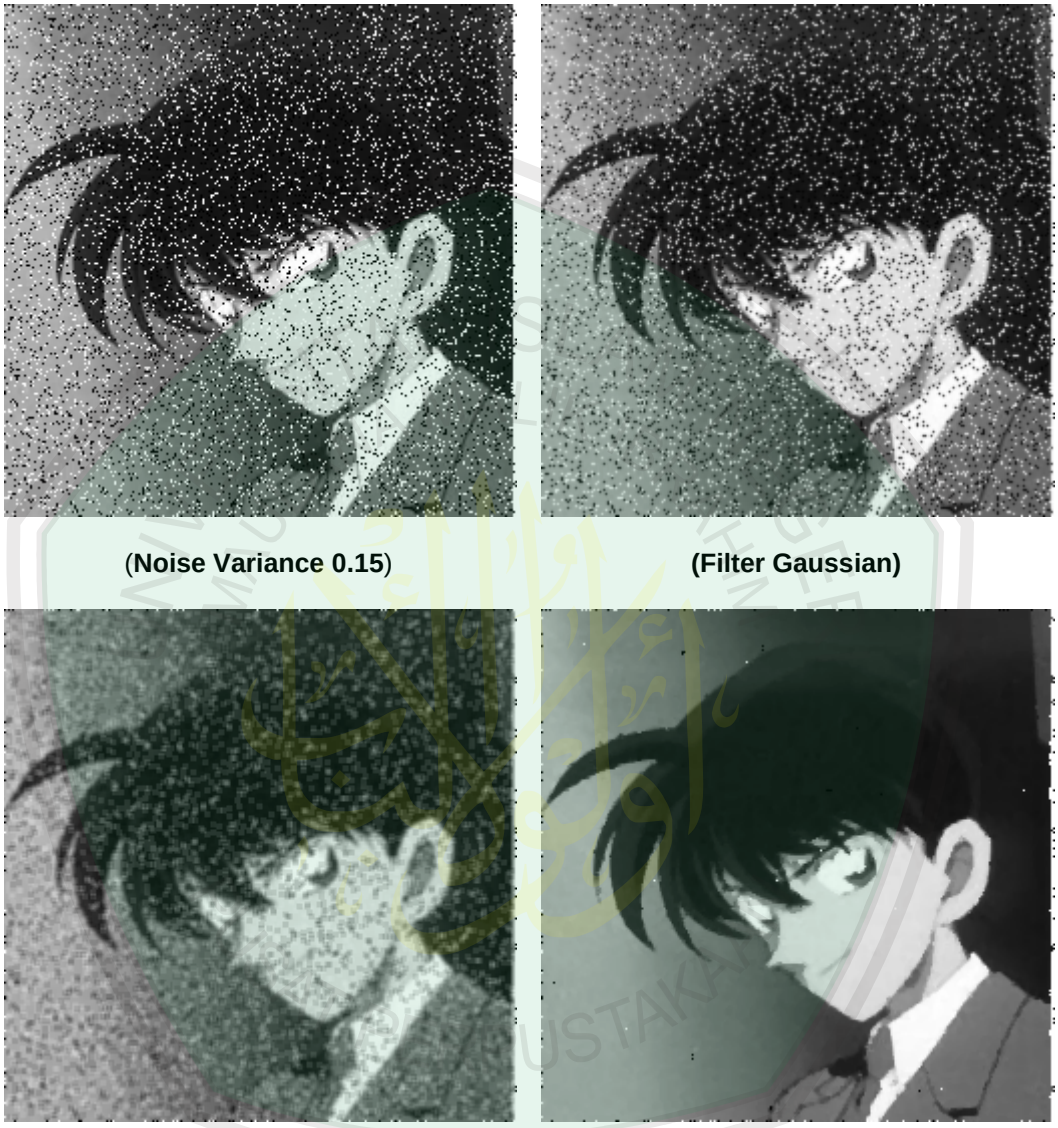
(Filter Median)

Gambar 4.10 Citra Bumi dengan ukuran 187*187 (kernel 3 x 3)

Tabel 4.10 Hasil Uji coba citra bumi *Noise variance* 0.15

	Filter Gaussian [3x3]	**Filter Mean [3x3]**	**Filter Median [3x3]**
MSE	632.595813434788	613.810946838662	118.180788698577
PSNR	20.1195404747124	20.2504573140027	27.4053347688024

4.3.4.3. Citra Kotaro dengan ukuran 256*256



(Noise Variance 0.15) (Filter Gaussian)

(Filter Mean) (Filter Median)

Gambar 4.11 Citra Kotaro dengan ukuran 256*256 (kernel 3 x 3)

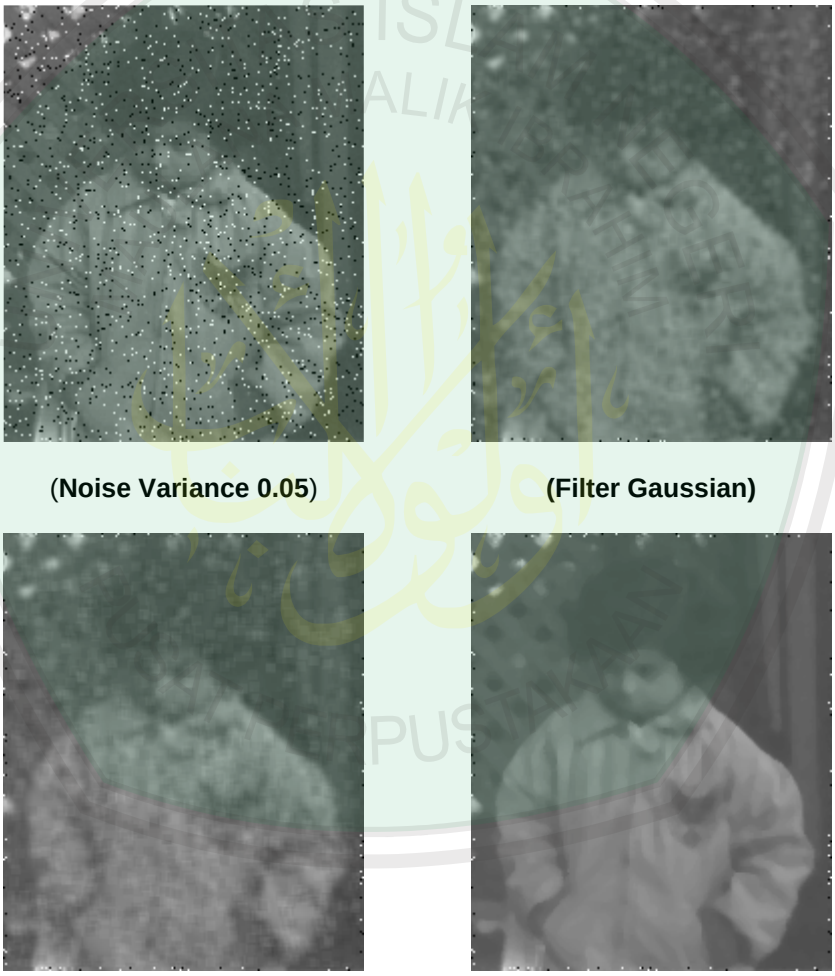
Tabel 4.11 Hasil Uji coba citra Kotaro *Noise variance* 0.15

	Filter Gaussian [3x3]	Filter Mean [3x3]	Filter Median [3x3]
MSE	586.556442260742	564.322708129883	74.4926910400391
PSNR	20.4477051177703	20.6155283440244	29.4096669747126

Dari hasil uji coba citra Fella, bumi dan Kotaro dengan *noise variance* 0.05, 0.1 dan 0.015 pada kernel 3 x 3, tampak bahwa nilai PSNR terbesar terjadi pada filter median disusul filter mean dan PSNR terkecil terjadi pada filter Gaussian (lihat tabel 4.2 – 4.11).

4.3.5. Uji coba citra input dengan *noise variance* 0.05 dengan kernel 5x 5

4.3.5.1. Citra Fella dengan ukuran 180*218

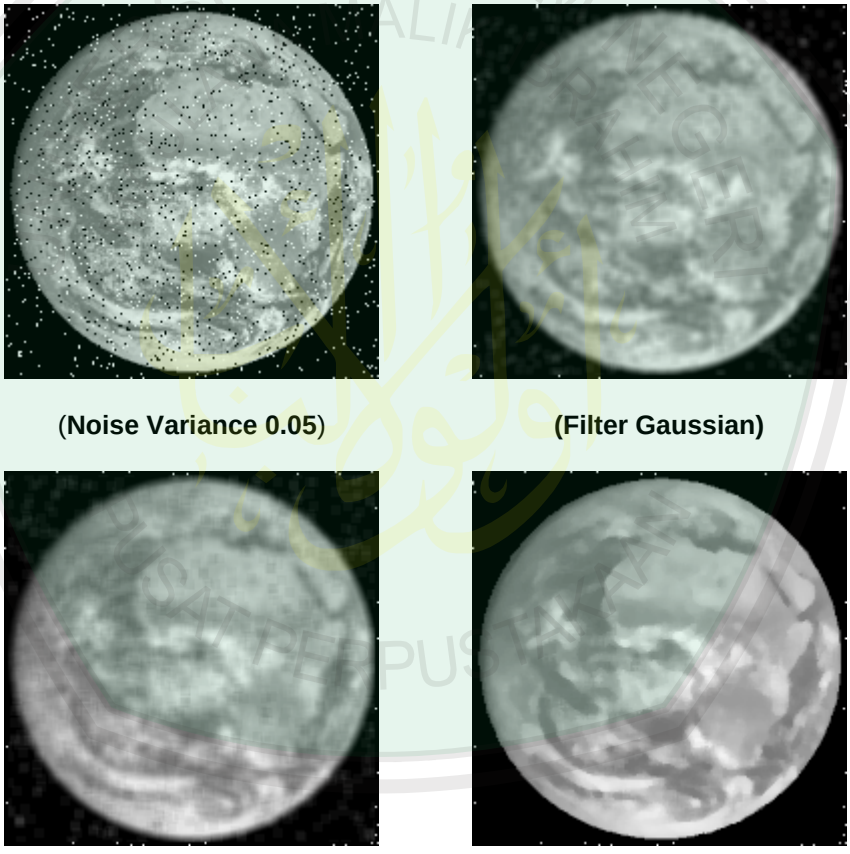


Gambar 4.12 Citra Fella dengan ukuran 180*218 (kernel 5 x 5)

Tabel 4.12 Hasil Uji citra Fella *Noise variance* 0.05

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	58.0910040774612	59.6184760448418	19.0629204892932
PSNR	30.4897147773766	30.3769949037517	35.3289092441754

4.3.5.2. Citra bumi dengan ukuran 187*187



(**Noise Variance 0.05**) **(Filter Gaussian)**

(Filter Mean) **(Filter Median)**

Gambar 4.13 Citra bumi dengan ukuran 187*187 (kernel 5 x 5)

Tabel 4.13 Hasil Uji coba citra bumi *Noise variance* 0.05

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	261.727415711067	295.318939632263	135.543452772473
PSNR	23.9523114388942	23.4278906048591	26.8100181640223

4.3.5.3. Citra Kotaro dengan ukuran 256*256

(Noise Variance 0.05) (Filter Gaussian)

(Filter Mean) (Filter Median)

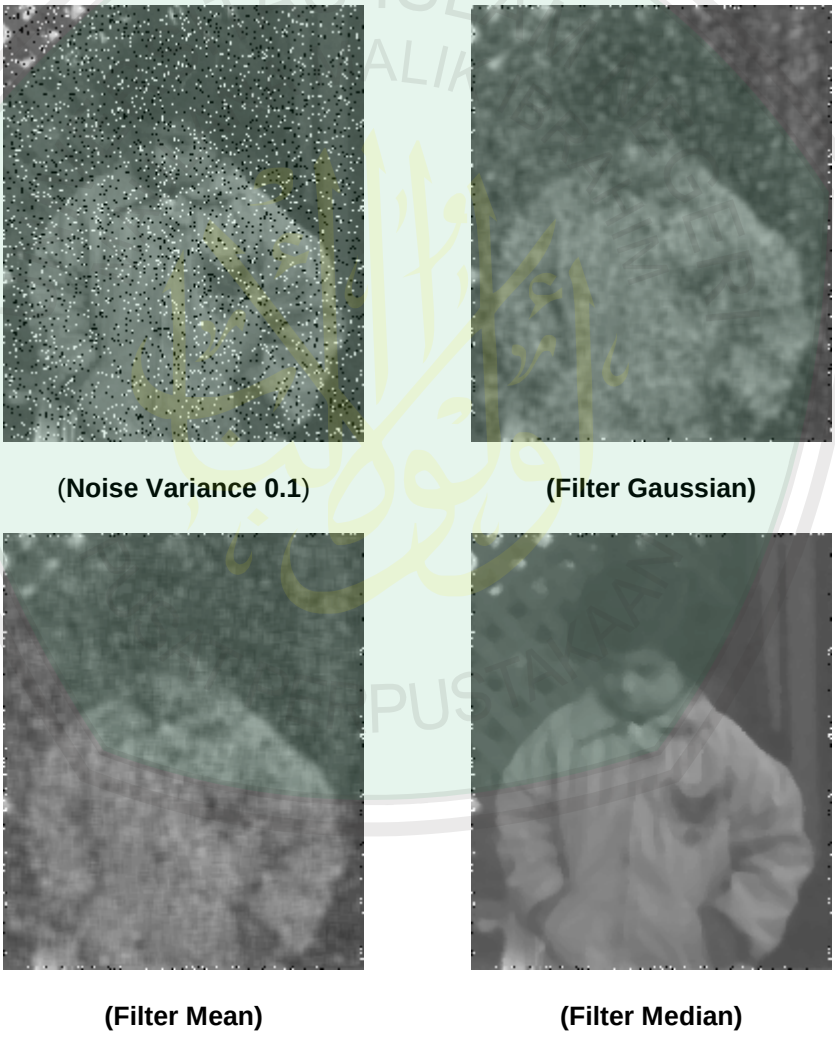
Gambar 4.14 Citra Kotaro dengan ukuran 256*256 (kernel 5 x 5)

Tabel 4.14 Hasil Uji coba citra Kotaro *Noise variance* 0.05

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	179.050231933594	200.910263061523	62.1911163330078
PSNR	25.6010547295414	25.1007823856893	30.1935200838122

4.3.6. Uji coba citra input dengan *noise variance* 0.1

4.3.6.1. Citra Fella dengan ukuran 180*218



(**Noise Variance 0.1**) **(Filter Gaussian)**

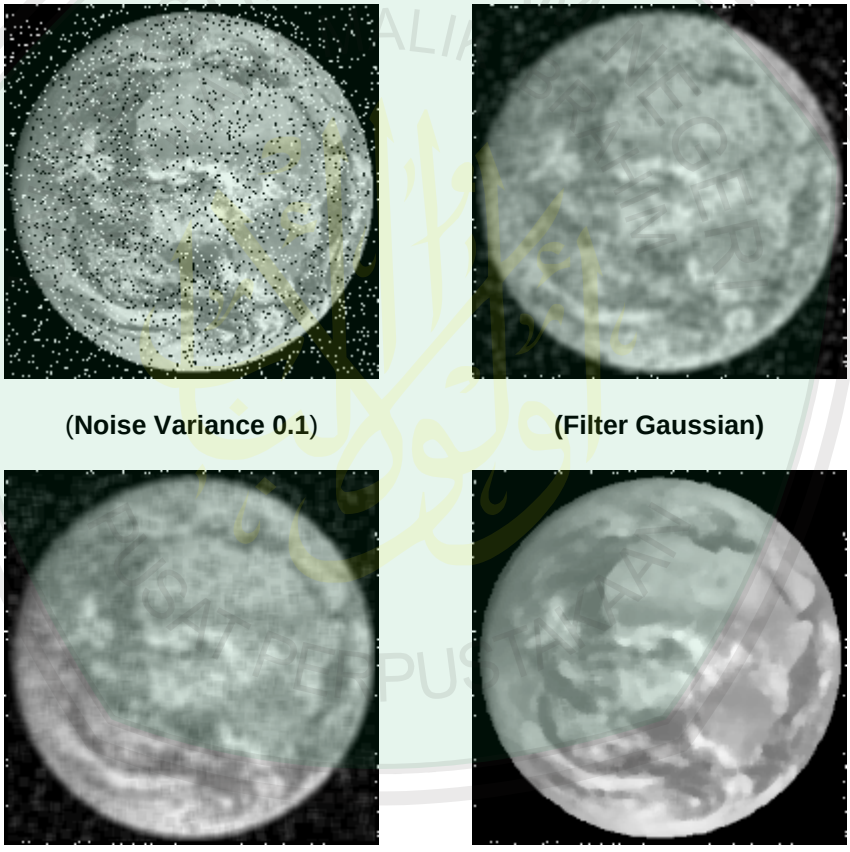
(Filter Mean) **(Filter Median)**

Gambar 4.15 Citra Fella dengan ukuran 180*218 (kernel 5 x 5)

Tabel 4.15 Hasil Uji coba citra Fella *Noise variance* 0.1

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	98.3192915392345	97.7077726809256	19.2676605504554
PSNR	28.2044162038038	28.2315124752615	35.2825137437575

4.3.6.2. Citra bumi dengan ukuran 187*187



(**Noise Variance 0.1**) **(Filter Gaussian)**

(Filter Mean) **(Filter Median)**

Gambar 4.16 Citra Bumi dengan ukuran 187*187 (kernel 5 x 5)

Tabel 4.16 Hasil Uji coba citra bumi *Noise variance* 0.1

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	364.526580685755	395.398267036542	147.211987760606
PSNR	22.5135115901156	22.1604559932505	26.4513718397581

4.3.6.3. Citra Kotaro dengan ukuran 256*256

(**Noise Variance 0.1**) (**Filter Gaussian**)

(**Filter Mean**) (**Filter Median**)

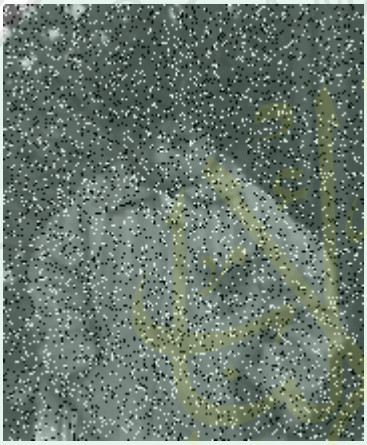
Gambar 4.17 Citra Kotaro dengan ukuran 256*256 (kernel 5 x 5)

Tabel 4.17 Hasil Uji coba citra Kotaro *Noise variance* 0.1

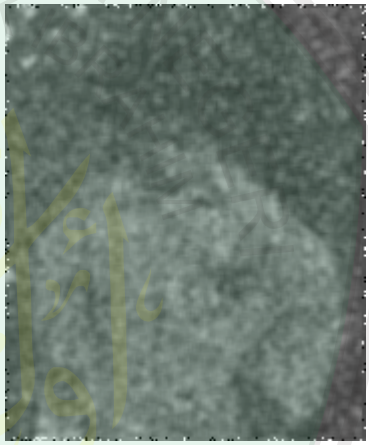
	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	268.87675476760742	287.145690917969	66.7833862304688
PSNR	23.8352710314261	23.5497805748767	29.8841192482697

4.3.7. Uji coba citra input dengan *noise variance* 0.15

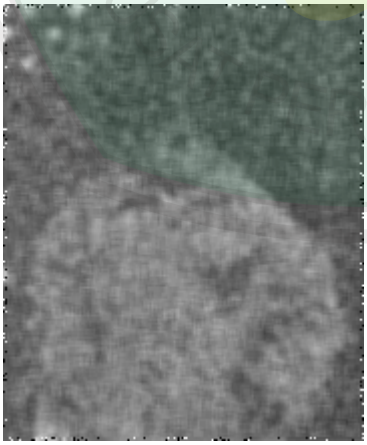
4.3.7.1. Citra Fella dengan ukuran 180*218



(**Noise Variance 0.15**)



(Filter Gaussian)



(Filter Mean)

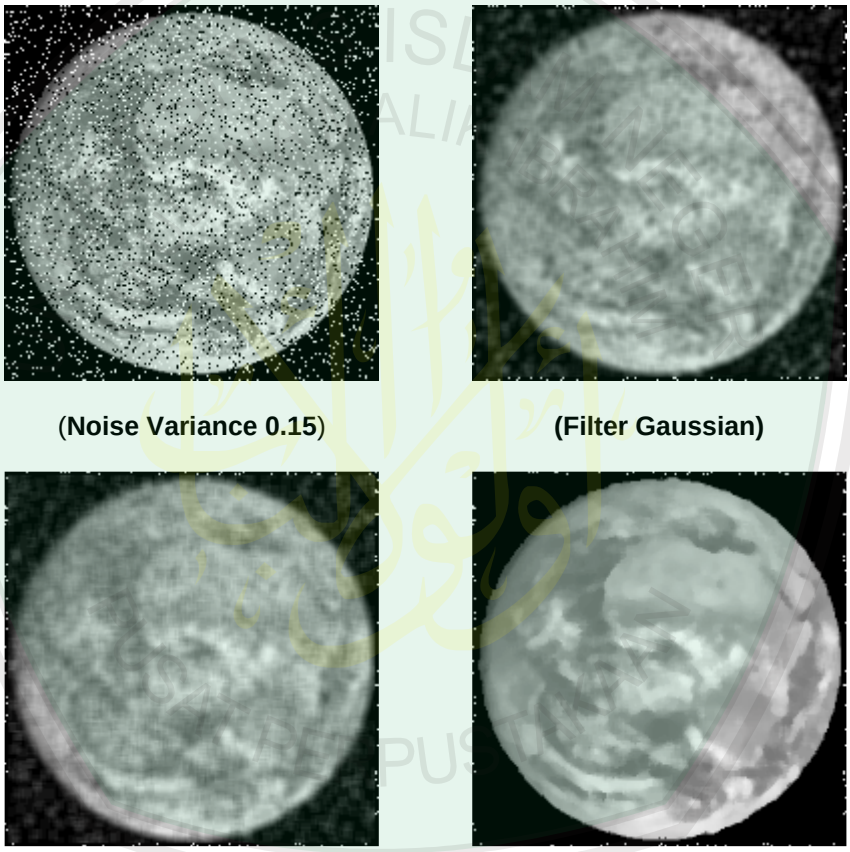
(Filter Median)

Gambar 4.18 Citra Fella dengan ukuran 180*218 (kernel 5 x 5)

Tabel 4.18 Hasil Uji coba citra Fella *Noise variance* 0.15

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	143.95474006115	140.281651376135	20.7802497451545
PSNR	26.5485439121262	26.6607949117397	34.9542959809475

4.3.7.2. Citra bumi dengan ukuran 187*187

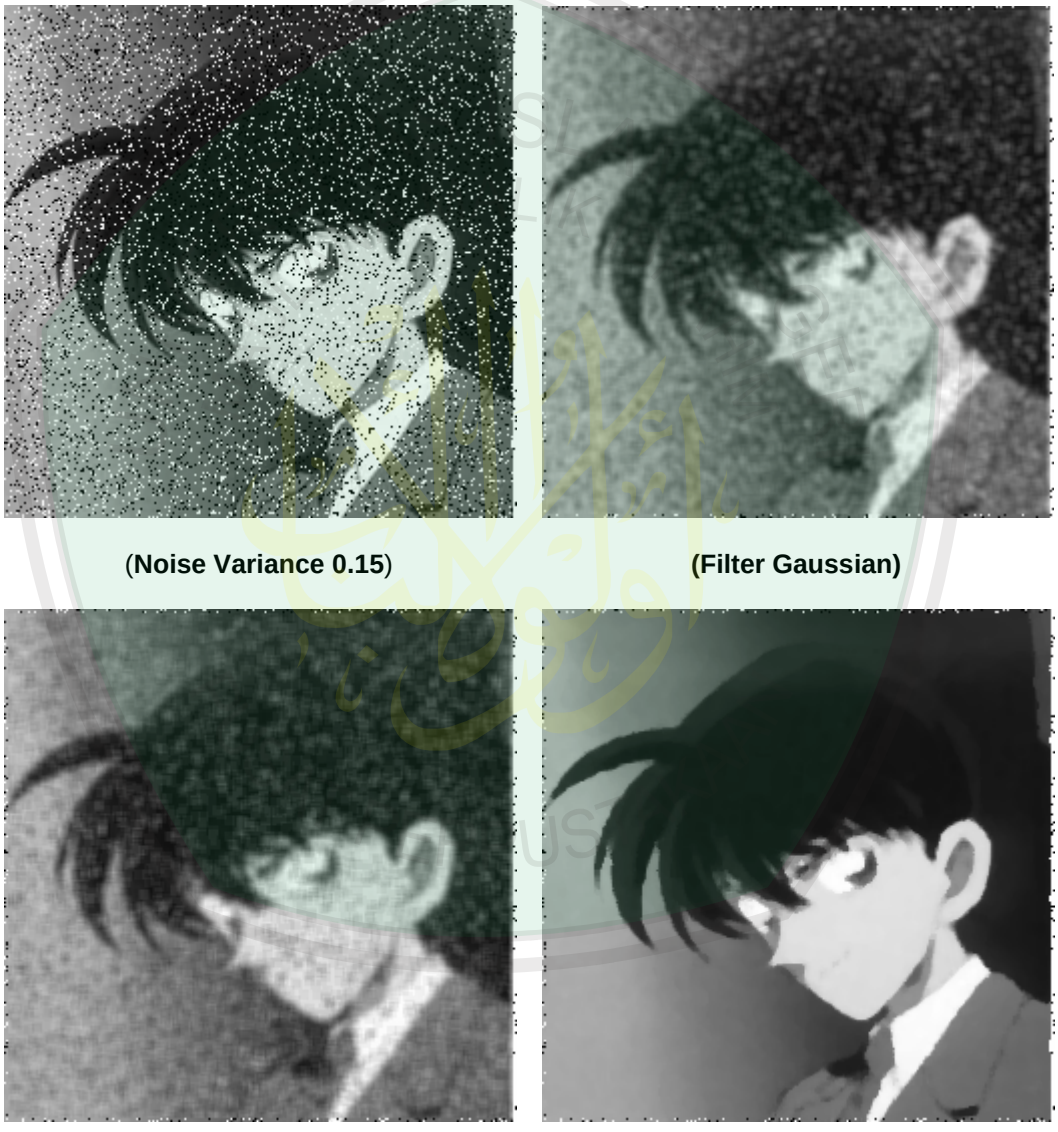


Gambar 4.19 Citra Bumi dengan ukuran 187*187 (kernel 5 x 5)

Tabel 4.19 Hasil Uji coba bumi *Noise variance* 0.15

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	485.933655523478	513.010294832594	163.238868712303
PSNR	21.2650338174325	21.029542804654	26.0025678452964

4.3.7.3. Citra Kotaro dengan ukuran 256*256



(**Noise Variance 0.15**) (**Filter Gaussian**)

(**Filter Mean**) (**Filter Median**)

Gambar 4.20 Citra Kotaro dengan ukuran 256*256 (kernel 5 x 5)

Tabel 4.20 Hasil Uji coba citra Kotaro *Noise variance* 0.15

	Filter Gaussian [5x5]	Filter Mean [5x5]	Filter Median [5x5]
MSE	389.201156616211	404.083770751953	77.379638671875
PSNR	22.2290623864649	22.066089526782	29.2445366343793

Dari hasil uji coba citra Fella, bumi dan Kotaro dengan *noise variance* 0.05, 0.1 dan 0.015 pada kernel 5 x 5, seperti halnya pada kernel 3 x 3 nilai PSNR terbesar terjadi pada filter median, tapi bedanya PSNR terkecil terjadi pada filter mean (lihat tabel 4.12 – 4.20).

4.4. Integrasi Program dan Al-Quran

Skripsi yang dibuat oleh penulis ini, tidak saja berhenti hanya pada tataran penyelesaian tugas akhir, tetapi juga upaya kreasi untuk memperindah momen-momen tertentu bagi setiap orang. Keindahan adalah keindahan, setiap orang memiliki impresi yang bervariasi terhadap keindahan tersebut. Allah sudah menciptakan manusia dalam bentuk yang sangat indah dan dalam bentuk yang terbaik. Maka sudah menjadi kewajiban bagi manusia untuk tetap melestarikan dalam berbagai tindakan, salah satunya mencitrakannya dalam bentuk gambar. Progaram yang penulis rancang adalah salah satu tindakan untuk melestariakan ciptaan Allah swt, misalnya dengan memperbaki gambar/foto manusia sebagai ciptaan Allah swt. Hal ini bukan bermaksud menyaingi keindahan ciptaan Tuhan, tetapi sebagai upaya untuk mensyukuri keindahan itu sendiri.

BAB V

KESIMPULAN DAN SARAN

Pada bab terakhir ini dijelaskan mengenai kesimpulan yang didapat dari pengerjaan skripsi ini, beserta saran-saran yang perlu diperhatikan untuk pengembangan selanjutnya.

5.1 Kesimpulan

Berdasarkan aplikasi yang telah dibuat beserta ujicoba yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut :

- Pemilihan ukuran kernel filter merupakan kunci utama pada perbaikan citra ber-*noise.*
- Semakin besar nilai *variance* yang diberikan pada citra maka citra akan menjadi semakin ber-*noise.*
- Berdasarkan hasil uji coba pada citra Fella, bumi dan Kotaro dengan kernel filter 3 x 3 dengan *variance noise* 0.05, 0.1 dan 0.15 tampak bahwa filter median menghasilkan nilai PSNR tertinggi, disusul filter mean dan PSNR paling rendah terjadi pada filter Gaussian (lihat tabel 4.2 – 4.11).
- Dan berdasarkan hasil uji coba pada citra Fella, bumi dan Kotaro dengan kernel filter 5 x 5 dengan variance *noise* 0.05, 0.1 dan 0.15 tampak bahwa filter median juga menghasilkan nilai PSNR tertinggi tapi bedanya PSNR terkecil terjadi pada filter mean (lihat tabel 4.12 – 4.20).
- Filter median merupakan filter yang paling baik digunakan untuk perbaikan kualitas citra dengan jenis *noise salt and peppers.*

- Citra hasil filtering dengan ukuran kernel 3 x 3 tidak banyak mengurangi atau menghaluskan *noise*, tetapi informasi detail citra masih tetap terjaga.
- Citra hasil filtering dengan ukuran kernel 5 x 5 banyak mengurangi atau menghaluskan *noise*, tetapi banyak menghilangkan informasi detail citra.
- Semakin besar jendela konvolusi (ukuran kernel) maka semakin besar derajat pemilteran, sehingga dapat menghilangkan informasi detail citra *input*, misalnya ketajaman citra *input*.

5.2 Saran

Saran yang hendak disampaikan terkait dengan pengerjaan skripsi ini adalah perbaikan citra ber-*noise* dengan metode filter Gaussian, mean dan median dapat dilakukan juga pada gambar *true colour* (citra warna). Sehingga dapat dibandingkan bagaimana kualitas citra hasil antara citra *grayscale* dan citra *true colour*.

DAFTAR PUSTAKA

Ahmad, Usman. 2005. *Pengolahan Citra Digital dan Teknik Pemrogramannya.* Edisi Pertama. Yogyakarta: Graha ilmu.

http://mathworld.wolfram.com/Convolution.html

Lestari, Desi. 2003. *Implementasi Teknik Watermarking Digital Pada Domain Dct Untuk Citra Berwarna.* Yogyakarta.

Munir, Rinaldi. 2004. *Pengolahan Citra Digital Dengan Pendekatan Algoritmik.* Bandung: Informatika.

R.C. Gonzalez, R.E. Woods, 1992, *Digital Image Processing*, USA : Addison-Wesley Publishing Company.

Syarifuddin, Sony Nuryadin. 2006. *Analisis Filtering Citra Dengan Metode Mean Filter Dan Median Filter*

Weisstein, E.W. *Convolution*. 2006.

Intan Permatasari, Desy. 2007. *Perbaikan Citra Dengan Menggunakan Metode Transformasi Dual-Tree Complex Wavelet.*

Shihab, M. Quraish. 2005. Tafsir al-Misbah : *Pesan, Kesan dan Keserasian Al-Qur'an Volume 15*. Jakarta: Lentara Hati.

Muhammad, Abdullah. 2006. *Tafsir Ibnu Katsir Jilid 8*. Jakarta: Pustaka Asy-Syafi'i.

Quthub, Sayyid. *Tafsir Di Bawah Naungan Al Qu'an.* Alih Bahasa: H. Bey Arifin dan Jamaluddin Kafie.

Hasyim Umar, Ahmad. 2004. *Kisah-Kisah Hadis Nabawi*. Yogyakarta: Mitra Pustaka.

Lampiran 1

Kode Program Penampil Citra

```
procedure TForm1.load_picture;
var
  fc: string;

begin
if (OpenPictureDialog.Execute) then
    begin
      if (Image1= nil) then

Image1.Picture.LoadFromFile(OpenPictureDialog.FileName);
      Image1.ClientHeight:=Image1.Picture.Height;
      Image1.ClientWidth :=Image1.Picture.Width;
      Image1.ClientHeight:=Image1.Picture.Height;
       case (Image1.Picture.Bitmap.PixelFormat) of pf8bit :
     fc := 'keabuan';
      end;
   end;
end;
```

Lampiran 2

Kernel 3 x 3 dan 5 x 5 Pada Filter Gaussian

```
var
 Form1: TForm1;
 {Kernel 3 x 3}

  const
    M = 1;
    N = 1;
    Mask: array [-M..M,-N..N] of real =
    ((0.075113608,    0.123841403,    0.075113608),
     (0.123841403,    0.204179956,    0.123841403),
     (0.075113608,    0.123841403,    0.075113608));

{Kernel 5 x 5}

  const
    Ma = 2;
    Na = 2;
    Mask2: array [-Ma..Ma,-Na..Na] of real =

   ((0.023246844,0.033823958,0.038327566,0.033823958,
     0.023246844),
    (0.033823958,     0.049213568,     0.055766279,
     0.049213568,     0.033823958),
    (0.038327566,     0.055766279,     0.063191473,
     0.055766279,     0.038327566),
    (0.033823958,     0.049213568,     0.055766279,
     0.049213568,     0.033823958),
    (0.023246844,     0.033823958,     0.038327566,
     0.033823958,     0.023246844));
```

Lampiran 3

Kode Program Pembangkit Noise

```
procedure TForm1.Derau1;
var
  x, y,   w, h: integer;
  PC, PH: PByteArray;
  Ki, Ko: array of array of byte;
  Derau, temp: real;
begin
  Randomize;
  Derau :=  StrToFloat(EditDerau.Text);
  w     :=  Image1.Picture.Width;
  h     :=  Image1.Picture.Height;

  if (Image1.Picture.Bitmap.PixelFormat = pf8bit)
  then
    begin
      SetLength(Ki, w, h);
      SetLength(Ko, w, h);
      for y := 0 to h-1 do
        begin
          PC := Image1.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            Ki[x, y] := PC[x];
        end;
      for x := 0 to w-1 do
        for y := 0 to h-1 do
            begin
              temp := Random;
              if (temp < Derau/2) then
                Ko[x, y] := 0
              else if (temp > 1-(Derau/2)) then
                Ko[x, y] := 255
              else
                Ko[x, y] := Ki[x, y];
            end;
      for y := 0 to h-1 do
        begin
          PH := Image3.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            PH[x] := Ko[x, y];
        end;
      Ki := nil;
      Ko := nil;
    end;
end;
```

Lampiran 4

Kode Progaram Filter Gaussian

```
procedure TForm1.Gaussian;
var
  x, y, w, h, u, v: integer;
  PC, PH, PA: PByteArray;
  Ki, Ko, Ka: array of array of byte;
  jumlah,mse,psnr: real;
begin
  w :=  Image3.Picture.Width;
  h :=  Image3.Picture.Height;
  if (  Image3.Picture.Bitmap.PixelFormat = pf8bit)
  then
    begin
      SetLength(Ki, w, h);
      SetLength(Ko, w, h);
      SetLength(Ka, w, h);
      for y := 0 to h-1 do
        begin
          PC :=  Image3.Picture.Bitmap.ScanLine[y];
          PH :=  Image5.Picture.Bitmap.ScanLine[y];
          PA :=  Image1.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            begin
              Ki[x, y] := PC[x];
              Ko[x, y] := PH[x];
              Ka[x, y] := PA[x];
            end;
        end;
      mse:=0;
      for x := M to w-1-M do
        for y := N to h-1-N do
          begin
            jumlah := 0;
            for u := -M to M do
              for v := -N to N do
                jumlah := jumlah+Mask[u,v]*Ki[x-u,y-v];
            Ko[x,y] := Round(jumlah);
            {MSE}

            mse:=mse + (sqr(Ko[x,y]-Ka[x,y])/(w*h));

          end;
          {PSNR}

        psnr := 20 * log10(255/sqrt(mse));
```

```
        frmMeasure.StringGrid1.Cells[1,1]:=floattostr(mse);
        frmMeasure.StringGrid1.Cells[1,2]:=floattostr(psnr);
        frmMeasure.StringGrid1.Cells[1,0]:='        Gaussian
Filter [3x3] ';
      for y := 0 to h-1 do
        begin
          PH :=  Image5.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            PH[x] := Ko[x, y];
        end;
      Ki := nil;
      Ko := nil;
      end;
end;
```

Lampiran 5

Kode Program Filter Mean

```
procedure TForm1.Mean;
var

  x, y, w, h: integer;
  PC, PH, PA: PByteArray;
  Ki,Ko,Ka: array of array of byte;
  mse,psnr : real;
begin
  w :=  Image3.Picture.Width;
  h :=  Image3.Picture.Height;
  if (Image3.Picture.Bitmap.PixelFormat = pf8bit)
  then
    begin
      SetLength(Ki, w, h);
      SetLength(Ko, w, h);
      SetLength(Ka, w, h);
      for y := 0 to h-1 do
        begin
          PC :=  Image3.Picture.Bitmap.ScanLine[y];
          PH :=  Image4.Picture.Bitmap.ScanLine[y];
          PA :=  Image1.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            begin
              Ki[x, y] := PC[x];
              Ko[x, y] := PH[x];
              Ka[x, y] := PA[x];
            end;
        end;

          mse:=0;

          for x := 1 to w-2 do
          for y := 1 to h-2 do
          begin
              Ko[x,y] := Round((Ki[x-1,y-1]+Ki[x,y-1]
              +Ki[x+1,y-1]+Ki[x-1,y]+Ki[x,y]+Ki[x+1,y]
              +Ki[x-1,y+1]+Ki[x,y+1]+Ki[x+1,y+1])/9);

            {Rumus MSE}
            mse:=mse + (sqr(Ko[x,y]-Ka[x,y])/(w*h));

          end;
          {Rumus PSNR}
          psnr := 20 * log10(255/sqrt(mse));
```

```
frmMeasure.StringGrid1.Cells[2,1]:=floattostr(mse);
frmMeasure.StringGrid1.Cells[2,2]:=floattostr(psnr);
frmMeasure.StringGrid1.Cells[2,0]:='Mean Filter [3x3]  ';

        for y := 0 to h-1 do
        begin
          PH :=  Image4.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            PH[x] := Ko[x, y];
        end;
        Ki := nil;
        Ko := nil;
    end;
end;
```

Lampiran 6

Kode program Filter Median

```
procedure TForm1.Median ;
var
  x, y, w, h, u, v, i, j, M, temp: integer;
  PC, PH, PA: PByteArray;
  Ki, Ko, Ka: array of array of byte;
  mse,psnr : real;
  Urutan: array [1..121] of byte;

begin
  w :=  Image3.Picture.Width;
  h :=  Image3.Picture.Height;
  if (Image3.Picture.Bitmap.PixelFormat = pf8bit)
  then
    begin
      SetLength(Ki, w, h);
      SetLength(Ko, w, h);
      SetLength(Ka, w, h);
      for y := 0 to h-1 do
        begin
          PC := Image3.Picture.Bitmap.ScanLine[y];
          PH := Image6.Picture.Bitmap.ScanLine[y];
          PA := Image1.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            begin
              Ki[x, y] := PC[x];
              Ko[x, y] := PH[x];
              Ka[x, y] := PA[x];
            end;
            end;

           M := 1;
          mse:=0;

      for x := M to w-1-M do
        for y := M to h-1-M do
          begin
            for u := -M to M do
              for v := -M to M do
                Urutan[(2*M+1)*u+v+((2*M+1)*(2*M+1)+1)   div
2]
                  := Ki[x-u,y-v];
            for i := 2 to (2*M+1)*(2*M+1) do
              for j := (2*M+1)*(2*M+1) downto i do
                if (Urutan[j] < Urutan[j-1]) then
                  begin
                    temp := Urutan[j];
```

```
                    Urutan[j]   := Urutan[j-1];
                    Urutan[j-1] := temp;

                   Ko[x,y]:=Urutan[((2*M+1)*(2*M+1)+1)div2];
                  end;
                    mse:=mse+(sqr(Ko[x,y] - Ka[x,y])/(w*h));


           end;
          psnr := 20 * log10(255/sqrt(mse));


frmMeasure.StringGrid1.Cells[3,1]:=floattostr(mse);

frmMeasure.StringGrid1.Cells[3,2]:=floattostr(psnr);
frmMeasure.StringGrid1.Cells[3,0]:='Median Filter [3x3]   ';
      for y := 0 to h-1 do
        begin
          PH :=  Image6.Picture.Bitmap.ScanLine[y];
          for x := 0 to w-1 do
            PH[x] := Ko[x, y];

        end;
      Ki := nil;
      Ko := nil;
    end;
end;
```