BAB II

LANDASAN TEORI

2.1. Metode *Adaptive Median Filter*

Adaptive Median Filter dirancang untuk menghilangkan masalah yang dihadapi dengan standar *median filter*. Perbedaan mendasar antara dua *filter* ini adalah bahwa pada *adaptive median filter* besarnya *window* (jendela/kernel) sekitarnya setiap piksel adalah variabel. Variasi ini tergantung pada median dari piksel dalam jendela sekarang atau saat ini. Jika nilai rata-rata adalah impuls, maka ukuran jendela akan diperluas. Jika tidak, proses lebih lanjut dilakukan pada citra dalam spesifikasi jendela saat ini. Pada dasarnya pada "pengolahan" citra diperlukan : piksel pusat dari jendela (*window*) dievaluasi untuk memverifikasi apakah itu suatu impuls atau bukan. Jika itu adalah suatu impuls, maka nilai piksel baru pada gambar yang telah di*filter* akan menjadi nilai median dari piksel dalam jendela itu. Jika piksel pusat bukan suatu impuls, maka nilai dari pusat piksel akan dipertahankan dalam citra yang di*filter*. Piksel (terkecuali) yang dipertimbangkan sebagai sebuah impuls, nilai *grayscale* dalam piksel pada gambar yang di*filter* adalah sama dengan citra masukan. *Adaptive median filter* memiliki tujuan ganda yaitu menghapus impuls *noise* pada gambar dan mengurangi distorsi pada gambar. *Adaptive Median Filter* dapat menangani operasi *filter* pada gambar rusak dengan impuls *noise*. *Filter* ini juga memperhalus *noise*. Dengan demikian, *filter* ini memberikan *output* citra jauh lebih baik dari standar *median filter*.

Filter ini melakukan pengolahan spasial untuk menentukan nilai mana dalam citra yang terkena *noise* dengan membandingkan setiap pikselnya terhadap

tetangganya. Ukuran *window* dapat disesuaikan dengan batasan maksimum *window*. Piksel yang berbeda dengan tetangganya maka dianggap sebagai *noise* untuk kemudian digantikan dengan nilai median piksel yang ada dalam satu *window*.

Misalnya x_{ij}, untuk (i,j) $\in A \equiv \{1, \dots, M\} \times \{1, \dots, N\}$, adalah derajat keabuan dari citra *x* dengan ukuran *M*×*N* pada lokasi (i,j), dan [S_{min} , S_{max}] adalah jangkauan dinamik dari x dengan kata lain $S_{min} \leq X_{ij} \leq S_{max}$ untuk semua $(i,j) \in A$. Kemudian *y* didefinisikan sebagai citra yang terkena *noise*.

Disini akan dijelaskan tentang algoritma *Adaptive Median Filter*. Dimisalkan S_{ij}^{w} adalah sebuah *window* dengan ukuran $w \times w$ dan memiliki pusat di (i,j) sehingga

$$s_{ij}^{w} = \{(k,l): |k-j| \leq w \; and \; |j-l| \leq w\} \quad \text{.....(2.1)}$$

dan $W_{max} \times W_{max}$ adalah ukuran maksimal *window*. Tujuan dari algoritma *Adaptive Median Filter* ini adalah mengidentifikasi kandidat *noise* y_{ij} kemudian mengganti setiap y_{ij} dengan nilai median dari piksel yang ada pada *window* s_{ij}^{w}. Untuk lebih jelasnya dapat dilihat pada penjelasan di bawah ini :

Untuk setiap piksel pada lokasi (i,j) , lakukan :

1. Inisialisasi ukuran pertama *window*, $w = w + 3$, karakteristik matriks *X*.
2. Hitung nilai $S_{ij}^{min,w}$, $S_{ij}^{med,w}$, dan $S_{ij}^{max,w}$ yang merupakan nilai minimum, median, dan maksimum dari piksel-piksel yang ada dalam *window* s_{ij}^{w}.
3. Jika $S_{ij}^{min,w} \leq S_{ij}^{med,w} \leq S_{ij}^{max,w}$, maju ke langkah 5. Jika tidak, atur ukuran $w = w + 2$.

4. Jika $w \leq w_{max}$, maka ulangi dari langkah 2. Selain itu ganti piksel y_{ij} dengan $S_{ij}^{med,w}$ kemudian set $x_{ij} = 0$.
5. Jika $S_{ij}^{min,w} \leq y_{ij} \leq S_{ij}^{max,w}$ maka y_{ij} bukan *noise* dan tidak perlu diganti nilainya kemudian, set $x_{ij} = 1$. Jika tidak, ganti y_{ij} dengan $S_{ij}^{med,w}$ dan set $x_{ij} = 0$.

2.2. Citra Digital

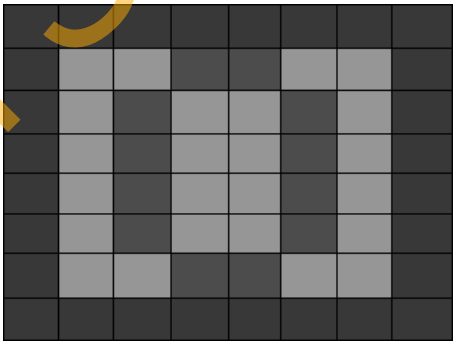
Citra (*image*) adalah bidang dalam dwimatra (dua dimensi) (Munir, 2004). Sebagai salah satu komponen multimedia, citra memegang peranan sangat penting sebagai bentuk informasi visual (Murinto, 2007). Seiring dengan perkembangan teknologi pengolahan citra (*image processing*) telah banyak dipakai di berbagai bidang. 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 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 $m \times n = (cij)$.

Gonzales dan Woods (1992) 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)* dan citra di titik tersebut.

Data atau informasi tidak hanya disajikan dalam bentuk tulisan, namun dapat berupa gambar, video ataupun audio. Ke-empat macam bentuk data atau informasi ini sering disebut multimedia (Munir, 2004).

Citra dapat berupa citra diam (*still images*) ataupun citra bergerak (*moving images*). Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan pada mata kita sebagai gambar yang bergerak.

Citra digital merupakan representasi dari citra yang diambil oleh mesin dengan bentuk pendekatan berdasarkan sampling dan kuantisasi (Basuki, 2005) Setiap citra digital memiliki beberapa karakteristik, antara lain ukuran citra, resolusi dan format nilainya. Umumnya citra digital berbentuk persegi panjang yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau piksel, sehingga ukuran citra selalu bernilai bulat.



Gambar 2.1 Citra dengan resolusi (Basuki, 2005)

Resolusi adalah kerapatan piksel dari citra yang berarti banyaknya jumlah piksel yang menyusun citra tersebut. Resolusi berbeda dengan ukuran panjang dan lebar pada umumnya. Pada setiap inchi atau cm bisa terdapat beberapa piksel,

seperti satuan dots per *inch* (dpi) yang berarti berapa titik/piksel pada tiap satu inchi-nya.

Untuk dapat diolah dengan komputer maka citra harus direpresentasikan secara numerik dalam bentuk matriks atau array. Citra dengan ukuran resolusi M x N (M = lebar, N = tinggi) dapat dinyatakan dengan array berukuran M x N. Sehingga dapat dikatakan bahwa array tersebut merupakan representasi citra dalam bentuk data nilai atau secara numerik. Representasi citra dari fungsi malar (kontinu) menjadi nilai-nilai diskrit disebut digitalisasi (Munir, 2004:18).

Contoh pada gambar 2.2 Citra abu-abu berukuran M x N tersebut dapat digambarkan dalam bentuk *array* atau matriks berukuran M x N sebagai berikut:

$$\begin{pmatrix} 56 & 56 & 56 & 56 & 56 & 56 & 56 & 56 \\ 56 & 150 & 150 & 76 & 76 & 150 & 150 & 56 \\ 56 & 150 & 76 & 150 & 150 & 76 & 150 & 56 \\ 56 & 150 & 76 & 150 & 150 & 76 & 150 & 56 \\ 56 & 150 & 76 & 150 & 150 & 76 & 150 & 56 \\ 56 & 150 & 76 & 150 & 150 & 76 & 150 & 56 \\ 56 & 150 & 150 & 76 & 76 & 150 & 150 & 56 \\ 56 & 56 & 56 & 56 & 56 & 56 & 56 & 56 \end{pmatrix}$$

Gambar 2.2 *Array* abu-abu dari gambar 2.1 (Basuki, 2005)

Karena indeks dari *array* yang penulis gunakan dimulai dari koordinat (0,0) pada pojok kiri atas, maka indeks *array* akan berakhir pada koordinat (M-1, N-1) di pojok kanan bawah. Dengan begitu, ukuran dari *array* tetap M x N. Berdasarkan gambar 2.2 kita dapat mengetahui nilai-nilai piksel penyusun citra. Misal, pada *array* dengan koordinat (0,0) memiliki nilai 56 dan pada koordinat (4,4) bernilai 150 juga.

2.2.1. Matriks Bitmap

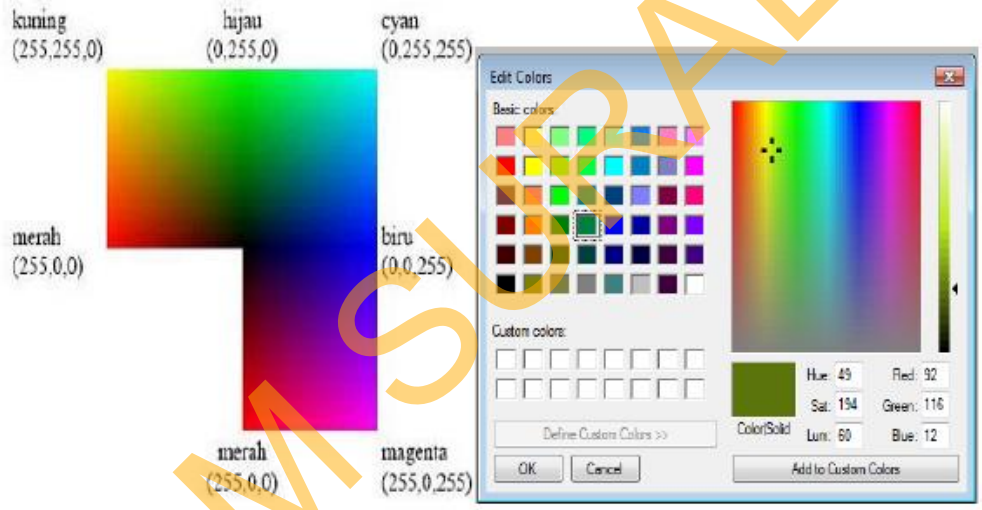
Citra disimpan di dalam berkas (*file*) dengan format tertentu (Munir, 2004). Format citra yang baku di lingkungan sistem operasi Microsoft Windows adalah berkas *bitmap* (*.bmp). Saat ini format BMP memang "kalah" populer dibandingkan dengan format JPG atau GIF. Hal ini karena berkas BMP pada umumnya tidak dimampatkan sehingga ukuran berkasnya relatif lebih besar daripada berkas JPG maupun GIF. Hal ini juga yang menyebabkan format BMP sudah jarang digunakan.

Meskipun format BMP tidak mangkus dari segi ukuran berkas, namun format BMP mempunyai kelebihan dari segi kualitas gambar. Citra dalam format BMP lebih bagus daripada citra dalam format yang lainnya, karena citra dalam format BMP umumnya tidak dimampatkan sehingga tidak ada informasi yang hilang. Terjemahan bebas *bitmap* adalah pemetaan bit. Artinya, nilai intensitas *pixel* di dalam citra dipetakan disejumlah bit tertentu. Peta bit yang umum adalah 8, artinya setiap *pixel* panjangnya 8 bit. Delapan bit ini merepresentasikan nilai intensitas *pixel*. Dengan demikian ada sebanyak 2^8 = 256 derajat keabuan, mulai dari 0-255.

2.2.2. Citra Warna

RGB adalah suatu model warna yang terdiri dari merah, hijau, dan biru, digabungkan dalam membentuk suatu susunan warna yang luas. Setiap warna dasar, misalnya merah, dapat diberi rentang nilai. Untuk monitor komputer, nilai rentangnya paling kecil =0 dan paling besar= 255. Pilihan skala 256 ini adalah didasarkan pada cara mengungkap 8 digit bilangan biner yang digunakan oleh mesin komputer. Dengan cara ini, akan diperoleh warna campuran sebanyak

256x256x256 = 16777216 jenis warna. Sebuah jenis warna, dapat dibayangkan sebagai suatu vektor di ruang 3 dimensi yang biasanya dipakai dalam matematika, koordinatnya dinyatakan dalam bentuk tiga bilangan, yaitu komponen-x, komponen-y, dan komponen-z. misalkan sebuah vektor dituliskan sebagai r = (x,y,z). untuk warna, komponen-komponen tersebut digantikan oleh komponen R(ed), G(reen), B(lue). Jadi, sebuah jenis warna dapat dituliskan sebagai berikut : warna RGB (30, 75, 255). Putih = RGB (255, 255, 255), sedangkan untuk hitam=RGB (0, 0, 0). Gambar 2.3 menunjukkan citra warna



Gambar 2.3 Citra Warna (RGB), (Tidak ada nama,

http://repository.usu.ac.id/bitstream/123456789/29726/4/Chapter%20II.pdf

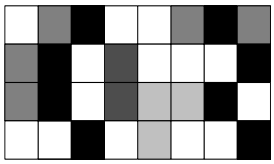
,diakses tanggal 12 Januari 2013)

2.2.3. Citra Skala Keabuan (*Grayscale*)

Citra skala keabuan memberi kemungkinan warna yang lebih banyak dari pada citra biner, karena terdapat kemungkinan nilai-nilai lain antara nilai minimum (0) hingga nilai maksimum. Banyaknya kemungkinan nilai tergantung

dari jumlah *bit* yang digunakan. Contoh, jika skala keabuan yang digunakan bernilai 4 *bit*, maka jumlah kemungkinan nilai adalah 2^4 = 16, dan nilai maksimum adalah 2^4-1 = 15. Sedangkan untuk skala keabuan 8 *bit*, maka jumlah kemungkinan nilainya adalah 2^8 = 256, dengan nilai maksimumnya 2^8-1 = 255

Format citra ini umumnya memiliki warna antara hitam sebagai warna minimal dan warna putih sebagai warna maksimal, sedangkan warna diantaranya adalah warna kelabu.



= 255 100 0 255 255 100 0 100
= 100 0 255 0 255 255 255 0
= 100 0 255 0 200 200 0 255
= 255 255 0 255 200 255 255 0

Gambar 2.4 Citra abu-abu 8 *bit* dan representasinya dalam data *digital* (Achmad, 2005)

Dalam prakteknya warna yang dipakai tidak terbatas pada warna kelabu, sebagai contoh dipilih warna minimalnya adalah warna putih dan warna maksimalnya adalah warna merah, maka semakin besar nilainya maka semakin besar pula intensitas warna merahnya. Beberapa buku menyebut format citra ini sebagai citra intensitas (Achmad, 2005).

2.2.4. *Pixel*

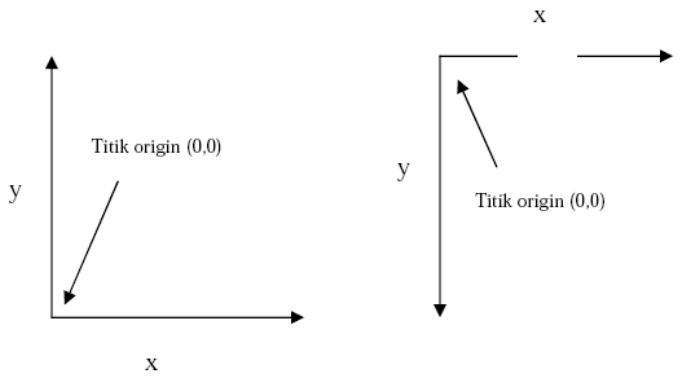
Pixel (Picture Elements) adalah nilai tiap-tiap entri matriks pada *bitmap*. Rentang nilai-nilai *pixel* ini dipengaruhi oleh banyaknya warna yang dapat ditampilkan. Jika suatu *bitmap* dapat menampilkan 256 warna maka nilai-nilai *pixel* nya dibatasi dari 0-255. Suatu *bitmap* dianggap mempunyai ketepatan yang tinggi jika dapat menampilkan lebih banyak warna. Prinsip ini dapat dilihat dari

contoh pada gambar 4 yang memberikan contoh dua buah *bitmap* dapat memiliki perbedaan dalam menangani transisi warna putih ke warna hitam.

Gambar 2.5 Perbedaan ketepatan warna *bitmap* (Jannah, 2008)

Perbedaan ketepatan warna *bitmap* pada gambar 2.5 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 dalam 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 bergerk ke kanan dan indeks y bergerak ke bawah. Konversi 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 kenyataandan 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 2.6 Perbedaan Letak Titik Origin Pada Koordinat Grafik dan pada Citra (Jannah, 2008)

2.2.5. Dimensi dan Resolusi

Dimensi *bitmap* adalah ukuran *bitmap* yang dinotasikan dengan menulis lebar x tinggii *bitmap*. Satuan ukur dimensi *bitmap* adalah berupa satuan ukur metris maupun *pixel*. Dimensi yang digunakan oleh *bitmap* mewakili ordo matriks citra itu sendiri. 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.

2.3. Pengolahan Citra

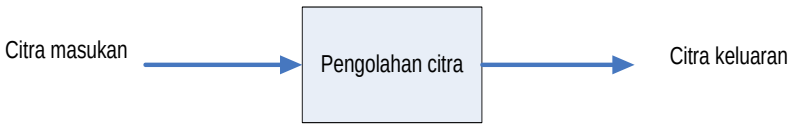
2.3.2. Definisi Pengolahan Citra

Image processing atau pengolahan citra adalah salah bidang dalam dunia komputer yang mulai berkembang sejak manusia memahami bahwa komputer tidak hanya mampu menangani data teks, tetapi juga data citra (Ahmad, 2005:4). Terminologi pengolahan citra dipergunakan bila hasil pengolahan data yang berupa citra, adalah juga berbentuk citra yang lain, yang mengandung atau memperkuat informasi khusus pada citra hasil pengolahan sesuai dengan tujuan pengolahannya.

Sesuai dengan perkembangannya, pengolahan citra mempunyai dua tujuan utama, yakni sebagai berikut:

1. Memperbaiki kualitas citra, dimana citra yang dihasilkan dapat menampilkan informasi secara jelas atau dengan kata lain manusia dapat melihat informasi yang diharapkan dengan menginterpretasikan citra yang ada. Dalam hal ini interpretasi terhadap informasi yang ada tetap dilakukan oleh manusia (*human percpetion*).
2. Mengekstraksi informasi ciri yang menonjol pada suatu citra, dimana hasilnya adalah informasi citra dimana manusia mendapat informasi ciri dari citra secara numerik atau dengan kata lain komputer (mesin) melakukan interpretasi terhadap informasi yang ada pada citra melalui

besaran-besaran data yang dapat dibedakan secara jelas (besaran-besaran ini berupa besaran numerik).



Gambar 2.7 Pengolahan citra

Secara umum, operasi-operasi pada pengolahan citra diterapkan pada citra bila:

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokkan, dicocokkan, diukur
3. Sebagian citra perlu digabung dengan citra yang lain.

2.3.2. Operasi Pengolahan Citra

Operasi-operasi pengolahan citra diklasifikasikan dalam beberapa jenis sebagai berikut (Munir, 2004):

1. Perbaikan kualitas citra (*image enhancement*)

   Bertujuan untuk memperbaiki kualitas yang dimiliki citra dengan cara memanipulasi parameter-parameter citra, sehingga ciri-ciri khusus yang terdapat pada citra dapat ditonjolkan. Contoh-contoh operasi perbaikan citra:

   a. Perbaikan kontras gelap/terang
   b. Perbaikan tepian objek
   c. Penajaman

d. Pemberian warna semu

e. Penapisan derau

2. Pemugaran citra (*image restoration*)

   Bertujuan menghilangkan atau meminimumkan cacat pada citra. Dengan operasi ini penyebab degradasi gambar dapat diketahui. Contoh-contoh operasi pemugaran citra:

   a. penghilangan kesamaran (*deblurring*)

   b. penghilangan derau (*noise*)

3. Pemampatan citra (*image compression*)

   Bertujuan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam operasi ini adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus. Contoh metode pemampatan citra adalah metode JPEG.

4. Segmentasi citra (*image segmentation*)

   Tujuan dari operasi ini untuk memecah suatu citra ke dalam beberapa segmen dengan kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

5. Pengorakan citra (*image analysis*)

   Bertujuan menghitung besaran kuantitatif dari citra untuk menghilangkan deskripsinya. Teknik pengorakan citra mengektraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Contoh-contoh operasi pengorakan citra:

   a. Pendeteksian tepi objek (*edge detection*)

b. Ekstraksi batas (*boundary*)

c. Representasi daerah (*region*)

6. Rekontruksi citra (*image reconstruction*)

   Bertujuan membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekontruksi citra banyak digunakan dalam bidang medis. Contohnya beberapa foto rontgen dengan sinar X digunakan untuk membentuk ulang gambar organ tubuh.

2.4. Pemrosesan Citra Digital

2.4.1. *Filter*

Filtering merupakan suatu proses yang mengambil sebagian sinyal frekuensi tertentu dan membuang sinyal pada frekuensi lain (Sigit, dkk ,2005). *Filtering* pada citra menggunakan prinsip sama, yaitu mengambil fungsi citra pada frekuensi-frekuensi tertentu dan membuang fungsi citra pada frekuensi-frekuensi lain.

Dari sifat-sifat citra pada bagian frekuensi, prinsip-prinsip *filtering* dapat dikembangkan menjadi berikut :

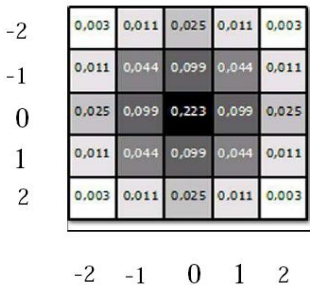
1. Bila ingin mempertahankan gradasi atau banyaknya level warna pada suatu citra, maka kita mempertahankan frekuensi rendah dan membuang frekuensi tinggi. Prinsip ini dinamakan *Low Pass Filter* dan banyak digunakan untuk reduksi *noise* dan proses blur.
2. Bila ingin mendapatkan *threshold* atau citra biner yang menunjukkan bentuk suatu gambar, maka kita mempertahankan frekuensi tinggi dan membuang frekuensi rendah. Prinsip ini dinamakan *High Pass Filter*

dan banyak digunakan untuk menentukan garis tepi (*edge*) atau sketsa citra.

3. Bila ingin mempertahankan gradasi dan bentuk dengan tetap mengurangi banyaknya bidang frekuensi (*bandwidth*) dan membuang sinyal yang tidak perlu maka kita mempertahankan frekuensi rendah dan frekuensi tinggi, sedangkan frekuensi tengahan dibuang. Prinsip ini dinamakan *Band Stop Filter*. Teknik yang dikembangkan menggunakan *Wavelet Transform* yang banyak digunakan untuk kompresi, restorasi, dan *denoising.*

2.4.2. Kernel *Filter*

Kernel atau *mask* memberikan petunjuk tentang apa yang harus dilakukan *filter* terhadap data. Pada umumnya kernel mempunyai panjang danlebar 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 dengan (*i,j*) yang berjalan dari -2 hingga 2 dan (*x,y*) yang berjalan dari 0 sampai 4.



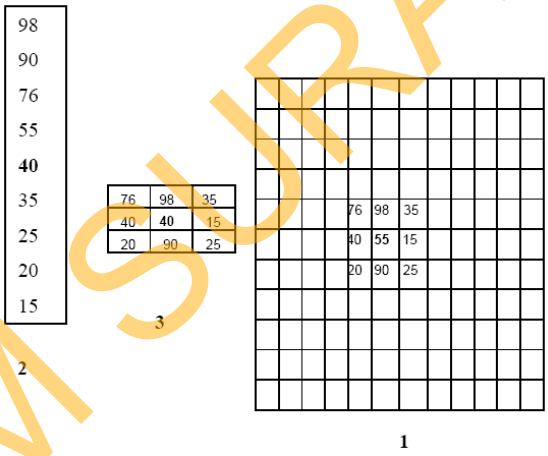
Gambar 2.8 Penentuan Lokasi Entri pada Kernel *Filter* (Jannah, 2008)

2.4.3. *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 3x3 *pixel* terhadap *bitmap* 2 dimensi.



Gambar 2.9 Ilustrasi Penerapan *Filter* Median Berukuran 3x3 *Pixel* (Jannah, 2008)

Cara mencari nilai median di atas adalah :

1. Baca nilai *pixel* yang akan diproses 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.

Penggunaan median *filter* itu sendiri juga mempunyai suatu kelemahan yaitu gambar yang sudah diproses akan tampak sedikit blur atau kabur. (Sulistyo, Wiwin, dkk, 2011).

Median filter adalah merupakan filter spasial *non-liner*, yang hasil prosesnya berdasarkan pada peringkat (rangking) nilai piksel. Secara statistik median mencari nilai yang berada ditengah deretan semua angka yang telah diurutkan. Cara pengurutan yang diambil dalam aplikasi ini adalah dengan menggunakan *bubble sort*.

Algoritma *bubble sort* adalah salah satu algoritma pengurutan yang paling sederhana, baik dalam hal pengertian maupun penerapannya. Ide dari algoritma ini adalah mengulang proses pembandingan antara tiap-tiap elemen *array* dan menukarrnya apabila urutannya salah. Pembandingan elemen-elemen ini akan terus diulang hingga tidak perlu dilakukan penukaran lagi. Algoritma ini termasuk dalam golongan algoritma *comparison sort*, karena menggunakan perbandingan dalam operasi antar elemennya. Berikut ini adalah gambaran dari algoritma *bubble sort*.

Pass Pertama

(**4 2** 5 3 9) menjadi (**2 4** 5 3 9)

(2 **4 5** 3 9) menjadi (2 **4 5** 3 9)

(2 4 **5 3** 9) menjadi (2 4 **3 5** 9)

(2 4 3 **5 9**) menjadi (2 4 3 **5 9**)

Pass Kedua

(**2 4** 3 5 9) menjadi (**2 4** 3 5 9)

(2 **4 3** 5 9) menjadi (2 **3 4** 5 9)

(2 3 **4 5** 9) menjadi (2 3 **4 5** 9)

(2 3 4 **5 9**) menjadi (2 3 4 **5 9**)

Pass Ketiga

(**2 3** 4 5 9) menjadi (**2 3** 4 5 9)

(2 **3 4** 5 9) menjadi (2 **3 4** 5 9)

(2 3 **4 5** 9) menjadi (2 3 **4 5** 9)

(2 3 4 **5 9**) menjadi (2 3 4 **5 9**)

Dapat dilihat pada proses di atas, sebenarnya pada pass kedua, langkah kedua, *array* telah terurut. Namun, algoritma tetap dilanjutkan hingga pass kedua berakhir. Pass ketiga dilakukan karena definisi terurut dalam algoitma *bubble sort* adalah tidak ada satupun penukaran pada suatu pass, sehingga pass ketiga dibutuhkan untuk mem-verifikasi keurutan *array* tersebut.

Beberapa kelebihan dari algoritma *Bubble Sort* adalah sebagai berikut :

- Algoritma yang sederhana
- Mudah untuk diubah menjadi kode
- Definisi terurut terdapat dengan jelas dalam algoritma
- Cocok untuk pengurutan data dengan elemen kecil telah terurut

Algoritma yang sederhana. Hal ini dilihat dari proses pengurutan yang hanya menggunakan rekurens dan perbandingan, tanpa penggunaan proses lain.

Algoritma pengurutan lain cenderung menggunakan proses lain, misalnya proses partisi pada algoritma *Quick Sort*.

Mudah untuk diubah menjadi kode. Hal ini diakibatkan oleh sederhananya *bubble sort* , sehingga kecil kemungkinan terjadi kesalahan sintax dalam pembuatan kode.

Definisi terurut terdapat dengan jelas dalam algoritma. Definisi terurut ini adalah tidak adanya satu kalipun *swap* pada satu kali pass. Berbeda dengan algoritma lain yang seringkali tidak memiliki definisi terurut yang jelas tertera algoritmanya, misalnya *Quick Sort* yang hanya melakukan partisihingga hanya dua buah nilai yang dibandingkan.

Cocok untuk pengurutan data dengan elemen kecil telah terurut. Algoritma *bubble sort* memiliki kondisi *best case* dengan kompleksitas algoritma.

Beberapa kekurangan dari algoritma *bubble sort* adalah sebagai berikut :

- Tidak efektif dalam pengurutan data berskala besar
- Langkah pengurutan yang terlalu panjang

2.5. *Noise*

Noise adalah citra atau gambar atau piksel yang mengganggu kualitas citra. *Noise* dapat disebabkan oleh gangguan fisis (*optik*) pada alat akuisisi maupun secara disengaja akibat proses pengolahan yang tidak sesuai, selain itu *noise* juga dapat disebabkan oleh kotoran-kotoran yang terjadi pada citra. Terdapat beberapa *noise* sesuai dengan bentuk dan karakteristik jenis, yaitu *salt&pepper*, *gaussian*, *uniform*, dan *noise speckle*. Banyak metode yang ada

dalam pengolahan citra yang bertujuan untuk mengurangi atau menghilangkan *noise*.

Noise muncul biasanya sebagai akibat dari pembelokkan yang tidak bagus (sensor *noise*, *photographic gain noise*). Gangguan tersebut umumnya berupa variasi intensitas suatu piksel dengan piksel-piksel tetangganya. Secara visual, gangguan mudah dilihat oleh mata karena tampak berbeda dengan piksel tetangganya. Piksel yang mengalami gangguan umumnya memiliki frekuensi tinggi. Komponen citra yang berfrekuensi rendah umumnya mempunyai nilai piksel konstan atau berubah sangat lambat. Operasi *denoise* dilakukan untuk menekan komponen yang berfrekuensi tinggi dan meloloskan komponen yang berfrekuensi rendah (Munir, 2004)

Reduksi *noise* adalah suatu proses menghilangkan atau mengurangi *noise* dari suatu *signal*. Reduksi *noise* secara konsep hampir sama penerapannya pada setiap jenis *signal*, tetapi untuk implementasinya, reduksi *noise* tergantung dari jenis *signal* yang akan diproses.

Secara umum metode untuk mereduksi *noise* dapat dilakukan dengan cara melakukan operasi pada citra digital dengan menggunakan suatu jendela ketetanggan , kemudian jendela tersebut diterapkan dalam citra. Proses tersebut dapat juga disebut proses *filtering*.

2.5.1. *Noise Uniform*

Noise Uniform seperti halnya *Noise Gaussian* dapat dibangkitkan dengan cara membangkitkan bilangan acak [0,1] dengan distribusi *uniform.* Kemudian untuk titik-titik yang terkena *noise* , nilai fungsi citra ditambahkan dengan *noise* yang ada, atau dirumuskan dengan :

$$y(i,j) = x(i,j) + p.a$$ .....(2.2)

Dimana :

a = nilai bilangan acak berdistribusi *uniform* dari *noise*

p = persentase *noise*

y(i,j) = nilai citra terkena *noise*

x(i,j) = nilai citra sebelum terkena *noise*

Noise Uniform merupakan *noise* sintesis yang sebenarnya dalam penerapannya jarang digunakan, tetapi secara pemrograman pembangkitan *noise uniform* ini merupakan jenis pembangkitan *noise* yang paling mudah.

2.5.2. *Noise Gaussian*

Noise Gaussian merupakan model *noise* yang mengikuti distribusi normal standar dengan rata-rata nol dan standar deviasi 1. Efek dari *noise* ini adalah munculnya titik-titik berwarna yang jumahnya sama dengan persentase *noise*.

Dengan rumus :

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

Noise Gaussian dapat dibangkitkan dengan cara membangkitkan bilangan acak [0,1]dengan distribusi *Gaussian*. Kemuadian titik-titik yang terkena *noise*, nilai fungsi citra ditambahkan dengan *noise* yang ada, atau dirumuskan dengan :

$$y(i,j) = x(i,j) + p.a$$ .....(2.4)

Dimana :

a = nilai bilangan acak berdistribusi *Gaussian*

p = persentase *noise*

y (i,j) = nilai citra terkena *noise*

x (i,j) = nilai citra sebelum terkena *noise*

Untuk membangkitkan bilangan acak berdistribusi *Gaussian* , tidak dapat langsung menggunakan fungsi rnd, tetapi diperlukan suatu metode yang digunakan untuk mengubah distribusi bilangan acak ke dalam fungsi f tertentu. Dalam buku ini digunakan metode *rejection* untuk memudahkan dalam alur pembuatan programnya. Metode *rejection* dikembangkan dengan cara membangkitkan dua bilangan acak (x,y) dan ditolak bila y>f(x).

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

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

Tingkat keberhasilan dan peforma dari suatu metode *filtering* pada citra dihitung dengan menggunakan *Peak Signal to Noise Ratio* atau biasa disingkat dengan PSNR. Meskipun peforma 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 peforma yang baik.

Peak Signal to Noise Ratio (PSNR) adalah sebuah perhitungan yang menentukan nilai dari sebuah citra yang dihasilkan. Nilai PSNR ditentukan oleh besar atau kecilnya nilai MSE yang terjadi pada citra. Semakin besar nilai PSNR, semakin baik pula hasil yang diperoleh pada tampilan citra hasil. Sebaliknya, semakin kecil nilai PSNR, maka akan semakin buruk pula hasil yang diperoleh pada tampilan citra hasil. Satuan nilai dari PSNR sama seperti MSE, yaitu decibel

(dB). Jadi hubungan antara nilai PSNR dengan nilai MSE adalah semakin besar nilai PSNR, maka akan semakin kecil nilai MSE-nya. PSNR secara umum digunakan untuk mengukur kualitas pada penyusunan ulang citra. Hal ini lebih mudah didefinisikan dengan *Mean Square Error* (MSE).

Mean Square Error (MSE) adalah kesalahan kuadrat rata-rata. Nilai MSE didapat dengan membandingkan nilai selisih *pixel-pixel* citra asal dengan citra hasil pada posisi *pixel* yang sama. Semakin besar nilai MSE, maka tampilan pada citra hasil akan semakin buruk. Sebaliknya, semakin kecil nilai MSE, maka tampilan pada citra hasil akan semakin baik. (Lestari, 2006)

Misal I (x,y) adalah citra masukan I'(x,y) adalah citra keluaran, keduanya memiliki M baris dan N kolom, maka didefinisikan sebagai berikut :

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}[I(x,y) - I'(x,y)]^2 \quad \text{.....(2.5)}$$

Rumus menghitung PSNR adalah :

$$PSNR = 20 \times log10\left(255/\sqrt{MSE}\right) \quad \text{.....(2.6)}$$

Dimana : x = ukuran baris dari citra

y = ukuran kolom dari citra

I = matriks citra awal

I'= matriks citra hasil