BAB III

METODE ROBERTS DAN SOBEL DALAM MENDETEKSI TEPI SUATU CITRA DIGITAL

3.1 Tepi Objek

Pertemuan antara bagian obyek dan bagian latar belakang disebut tepi obyek. Dalam pengolahan citra, tepi obyek ditandai oleh titik yang nilai keabuannya memiliki perbedaan yang cukup besar dengan titik yang ada disebelahnya. Bila dua buah obyek atau lebih saling tumpang tindih maka mereka akan meninggalkan jejak tepi apabila intensitasnya tidak sama, sehingga dapat diketahui bahwa obyek yang satu berada di depan obyek yang lain atau sebaliknya (Sutoyo dkk., 2009:227).

Hal ini penting untuk memisahkan obyek-obyek yang tumpang tindih sehingga dapat dianalisis secara individu. Dengan demikian, tepi sebuah obyek dapat juga digunakan untuk memisahkan obyek-obyek yang saling bersinggungan sehingga tidak dianggap sebagai satu obyek yang besar, tetapi dapat dilacak atau dianalisis secara individu.

Ada tiga macam tepi di dalam citra digital, yaitu:

1. Tepi curam

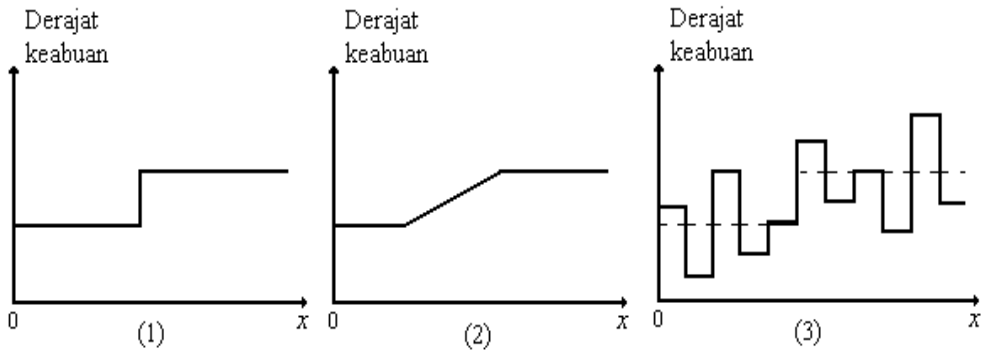
   Tepi curam adalah tepi dengan perubahan intensitas yang tajam. Arah tepi berkisar 90^o.

2. Tepi landai

   Tepi landai yaitu tepi dengan sudut arah yang kecil. Tepi landai dapat dianggap terdiri dari sejumlah tepi-tepi lokal yang lokasinya berdekatan.

3. Tepi yang mengandung *noise* (derau)

   Umumnya tepi yang terdapat pada aplikasi visi komputer mengandung *noise*.



Gambar 3.1 (1) Tepi curam (2) Tepi landai (3) Tepi curam yang mengandung *noise*

3.2 Deteksi Tepi

Menurut Hambali (2011:5) bahwa "Deteksi tepi yaitu proses untuk menentukan lokasi titik-titik yang merupakan tepi obyek". Sebuah operator deteksi tepi merupakan operasi bertetangga, yaitu sebuah operasi yang memodifikasi nilai keabuan sebuah titik berdasarkan nilai-nilai keabuan dari titik-titik yang ada di sekitarnya (tetangganya) yang masing-masing mempunyai bobot tersendiri. Bobot-bobot tersebut nilainya tergantung pada operasi yang akan dilakukan, sedangkan banyaknya titik tetangga yang terlibat biasanya adalah 2x2, 3x3, 7x7, dan sebagainya. Adapun tujuan dari deteksi tepi adalah sebagai berikut:

1. Untuk mendeteksi garis tepi yang membatasi dua wilayah citra, obyek dan latar belakangnya.
2. Untuk menemukan perubahan intensitas yang berbeda dalam sebuah bidang citra.

3. Untuk meningkatkan penampakan garis batas suatu daerah atau obyek di dalam citra.
4. Untuk mencirikan batas obyek dan berguna untuk proses segmentasi dan identifikasi obyek.

Manfaat yang bisa diperoleh dari deteksi tepi dalam berbagai bidang, misalnya yang paling banyak digunakan dalam bidang kedokteran adalah untuk menentukan stadium kanker, mendeteksi tepi citra USG janin, mendeteksi karies pada gigi, sehingga bentuk citra yang dihasilkan dapat terlihat lebih jelas. Di bidang lainnya, deteksi tepi digunakan untuk aplikasi pengenalan plat kendaraan, aplikasi pengenalan sidik jari, dan untuk membedakan uang asli dengan uang palsu.

Deteksi tepi merupakan komponen berfrekuensi tinggi, sehingga dibutuhkan *High Pass Filter* (HPF). Biasanya operator yang digunakan untuk mendeteksi tepi yang pertama adalah operator berbasis *Gradient* (turunan pertama), yaitu operator Roberts, Sobel, dan Prewitt. Sedangkan untuk mendeteksi tepi yang kedua adalah operator berbasis turunan kedua, yaitu operator Laplacian dan operator *Laplacian of Gaussian*. Turunan pertama menghasilkan tepi yang lebih tebal, sedangkan turunan kedua menghasilkan tepi yang lebih tipis.

3.3 Deteksi Tepi Berbasis Gradien

Deteksi tepi dapat dilakukan dengan menghitung selisih antara dua buah titik yang bertetangga sehingga didapat besar gradien citra. Gradien adalah turunan pertama dari persamaan dua dimensi yang didefinisikan sebagai vektor berikut.

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \quad \text{...(3-1)}$$

Gradien mempunyai dua sifat penting, yaitu:

1. Vektor $G[f(x,y)]$ menunjukkan arah penambahan laju maksimum dari fungsi $f(x,y)$.
2. Besar gradien sama dengan penambahan laju maksimum dari fungsi $f(x,y)$ per satuan jarak dalam arah G.

Besar gradien dihitung dengan persamaan berikut.

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2}. \quad \text{...(3-2)}$$

Untuk kebutuhan pengolahan citra, dalam praktiknya besar gradien dihitung sebagai berikut.

$$G[f(x,y)] = |G_x| + |G_y|. \quad \text{...(3-3)}$$

Sedangkan arahnya dapat dihitung dengan persamaan berikut.

$$\alpha(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right) \quad \text{...(3-4)}$$

di mana α diukur dari sumbu *x* sebagai garis acuan.

Untuk keperluan perhitungan pada citra digital, turunan pada persamaan 3-1 lebih mudah bila ditulis menggunakan pendekatan persamaan diferensial berikut.

$$G_x \cong f(x+1,y) - f(x,y) \quad \text{...(3-5)}$$

$$G_y \cong f(x,y+1) - f(x,y). \quad \text{...(3-6)}$$

3.4 Metode Roberts

Metode Roberts merupakan metode yang menggunakan operator Roberts. Operator Roberts adalah operator yang berbasis gradien yang menggunakan dua buah *kernel* yang berukuran 2x2 piksel. Operator ini mengambil arah diagonal untuk penentuan arah dalam penghitungan nilai gradien, sehingga sering disebut dengan operator silang. (Sutoyo dkk., 2009:228).

Perhitungan gradien dalam operator Roberts adalah sebagai berikut.

$$G = \sqrt{R_x^2 + R_y^2} \quad \text{...(3-7)}$$

dengan, G = besar gradien operator Roberts

R_x = gradien Roberts arah horisontal

R_y = gradien Roberts arah vertikal

di mana R_x dan R_y dihitung menggunakan *kernel* konvolusi sebagai berikut.

$$R_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad R_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Gambar 3.2 Operator Roberts

Sebenarnya, metode Roberts dalam mendeteksi tepi menghasilkan citra yang kurang memuaskan. Mungkin dikarenakan *kernel* yang digunakan berukuran 2x2 piksel dan penghitungan gradien hanya mengambil kedua arah diagonal.

Algoritma metode Roberts dalam mendeteksi tepi suatu citra digital adalah sebagai berikut:

Citra masukan berupa citra *grayscale*.

1. Konvolusikan citra *grayscale* dengan *kernel* Roberts horisontal (R_x) = $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ dan *kernel* Roberts vertikal (R_y) = $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.
2. Hitung besar gradien dengan rumus $G = \sqrt{R_x^2 + R_y^2}$.
3. Citra keluaran merupakan hasil dari besar gradien (G).

Berikut sintaks deteksi tepi metode Roberts menggunakan Matlab.

```
guidata(hObject,handles);
handles.current_data1=handles.data1;
I=handles.current_data1;
I=rgb2gray(I);
I=double(I);
robertshor = [1 0; 0 -1];
robertsver = [0 -1; 1 0];
Rx = conv2(I,robertshor,'same');
Ry = conv2(I,robertsver,'same');
hasil = sqrt((Rx.^2)+(Ry.^2));
set(handles.text20,'String','Citra *Roberts*');
axes(handles.axes2);
imagesc(hasil),colormap('gray'),colorbar('vert');
```

Contoh kasus:

1. Citra masukan berupa citra *grayscale* berukuran 5x5 piksel, untuk mempermudah dan menyederhanakan penghitungan.

255	255	235	196	36
251	255	221	106	30
254	250	168	35	26
255	217	84	27	20
247	171	28	32	17

2. Konvolusikan citra *grayscale* dengan *kernel* Roberts horisontal (R_x) = $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.

255	255	235	196	36	0
251	255	221	106	30	0
254	250	168	35	26	0
255	217	84	27	20	0
247	171	28	32	17	0
0	0	0	0	0	0

-1	0
0	1

3. Hasil konvolusi citra *grayscale* dengan *kernel* Roberts horisontal.

0	-34	-129	-166	-36
-1	-87	-186	-80	-30
-37	-166	-141	-15	-26
-84	-189	-52	-10	-20
-247	-171	-28	-32	-17

4. Konvolusikan citra *grayscale* dengan *kernel* Roberts vertikal (R_y) = $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

255	255	235	196	36	0
251	255	221	106	30	0
254	250	168	35	26	0
255	217	84	27	20	0
247	171	28	32	17	0
0	0	0	0	0	0

0	1
-1	0

5. Hasil konvolusi citra *grayscale* dengan *kernel* Roberts vertikal.

4	-20	-25	-70	-30
1	-29	-62	-5	-26
-5	-49	-49	-1	-20
-30	-87	-1	-12	-17
171	28	32	17	0

6. Hitung besar gradien citra dengan rumus $G = \sqrt{R_x^2 + R_y^2}$, diperoleh hasil akhir seperti berikut.

4	39.45	131.4	180.2	46.86
1.414	91.71	196.1	80.16	39.7
37.34	173.1	149.3	15.03	32.8
89.2	208.1	52.01	15.62	26.25
300.4	173.3	42.52	36.24	17

3.5 Metode Sobel

Metode Sobel merupakan metode yang menggunakan operator Sobel. Operator ini menggunakan dua buah *kernel* yang berukuran 3x3 piksel untuk penghitungan gradien sehingga perkiraan gradien berada tepat di tengah jendela. (Sutoyo dkk., 2009:229).

Misalkan susunan piksel-piksel di sekitar piksel (*x*,*y*) seperti berikut.

a_0	a_1	a_2
a_7	*(x,y)*	a_3
a_6	a_5	a_4

Gambar 3.3 Susunan piksel-piksel di sekitar piksel (*x*,*y*)

Berdasarkan susunan piksel tetangga tersebut, besaran gradien yang dihitung menggunakan operator Sobel adalah sebagai berikut.

$$G = \sqrt{S_x^2 + S_y^2} \quad ...(3\text{-}8)$$

dengan, G = besar gradien operator Sobel

S_x = gradien Sobel arah horizontal

S_y = gradien Sobel arah vertikal

di mana G adalah besar gradien di titik tengah *kernel* dan turunan parsial dihitung menggunakan persamaan berikut.

$$S_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \quad ...(3\text{-}9)$$

$$S_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \quad ...(3\text{-}10)$$

di mana *c* adalah konstanta yang bernilai 2. S_x dan S_y diimplementasikan menjadi *kernel* berikut.

S_x =

-1	0	1
-2	0	2
-1	0	1

S_y =

1	2	1
0	0	0
-1	-2	-1

Gambar 3.4 Operator Sobel

Algoritma metode Sobel dalam mendeteksi tepi suatu citra digital adalah sebagai berikut:

Citra masukan berupa citra *grayscale*.

1. Konvolusikan citra *grayscale* dengan *kernel* Sobel horisontal (S_x) = $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ dan *kernel* Sobel vertikal (S_y) = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & 0 & 2 \\ -1 & -2 & -1 \end{bmatrix}$.
2. Hitung besar gradien dengan rumus $G = \sqrt{S_x^2 + S_y^2}$.
3. Citra keluaran merupakan hasil dari besar gradien (G).

Berikut sintaks deteksi tepi metode Sobel menggunakan Matlab.

```
guidata(hObject,handles);
handles.current_data1=handles.data1;
I=handles.current_data1;
I=rgb2gray(I);
I=double(I);
sobelhor = [-1 0 1; -2 0 2; -1 0 1];
sobelver = [1 2 1; 0 0 0; -1 -2 -1];
Sx = conv2(I,sobelhor,'same');
Sy = conv2(I,sobelver,'same');
hasil = sqrt((Sx.^2)+(Sy.^2));
set(handles.text20,'String','Citra *Sobel*');
axes(handles.axes2);
imagesc(hasil),colormap('gray'),colorbar('vert');
```

Contoh kasus:

1. Citra masukan berupa citra *grayscale* berukuran 5x5 piksel, untuk mempermudah dan menyederhanakan penghitungan.

255	255	235	196	36
251	255	221	106	30
254	250	168	35	26
255	217	84	27	20
247	171	28	32	17

2. Konvolusikan citra *grayscale* dengan *kernel* Sobel horisontal (S_x) = $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$.

0	0	0	0	0	0	0
0	255	255	235	196	36	0
0	251	255	221	106	30	0
0	254	250	168	35	26	0
0	255	217	84	27	20	0
0	247	171	28	32	17	0
0	0	0	0	0	0	0

1	0	-1
2	0	-2
1	0	-1

3. Hasil konvolusi citra *grayscale* dengan *kernel* Sobel horisontal.

-765	70	267	589	498
-1015	166	572	723	443
-972	373	769	539	203
-855	647	734	281	121
-559	609	468	86	91

4. Konvolusikan citra *grayscale* dengan *kernel* Sobel vertikal (S_y) = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & 0 & 2 \\ -1 & -2 & -1 \end{bmatrix}$.

0	0	0	0	0	0	0
0	255	255	235	196	36	0
0	251	255	221	106	30	0
0	254	250	168	35	26	0
0	255	217	84	27	20	0
0	247	171	28	32	17	0
0	0	0	0	0	0	0

-1	-2	-1
0	0	0
1	2	1

5. Hasil konvolusi citra *grayscale* dengan *kernel* Sobel vertikal.

757	982	803	463	166
-7	-78	-300	-399	-181
-30	-209	-391	-305	-99
-93	-305	-362	-155	-21
-727	-773	-412	-158	-67

6. Hitung besar gradien citra dengan rumus $G = \sqrt{S_x^2 + S_y^2}$, diperoleh hasil akhir seperti berikut.

1076	984.5	846.2	749.2	524.9
1015	183.4	645.9	825.8	478.5
972.5	427.6	862.7	619.3	225.9
860	715.3	818.4	320.9	122.8
917.1	984.1	623.5	179.9	113

Dari hasil deteksi tepi, nilai yang diperoleh dari kedua buah metode masing-masing berbeda, dikarenakan operator dan ukuran kernel yang digunakan berbeda. Metode Roberts lebih menekankan penghitungan gradien arah diagonal sedangkan metode Sobel lebih ke arah vertikal dan horisontalnya.

3.6 Perancangan Program Deteksi Tepi

Untuk mengimplementasikan metode Roberts dan Sobel ke dalam bentuk program diperlukan beberapa perangkat pendukung diantaranya sebagai berikut.

3.6.1 Perangkat Keras

Perangkat keras yang digunakan untuk membuat dan menjalankan program deteksi tepi adalah satu set komputer (PC) dengan spesifikasi sebagai berikut:

1. Sistem Komputer: Intel(R) Atom (TM) CPU N280 @ 1.66GHz
2. Sistem Operasi: Microsoft Windows XP
3. Media masukan: papan ketik (*keyboard*) dan *mouse*
4. *Hardisk*: 250 GB
5. Memori: 0.99 GB RAM

3.6.2 Perangkat Lunak

Pada skripsi ini, penulis menggunakan Matlab versi 7.0 untuk membuat program deteksi tepi pada suatu citra, sedangkan untuk membuat tampilan antarmuka dengan menggunakan GUIDE (*Graphical User Interface Development Environment*) yang ada dalam program Matlab. Diagram alir pembuatan program untuk mendeteksi tepi suatu citra ditunjukkan pada Gambar 3.5.

Gambar 3.5 Diagram alir pembuatan program deteksi tepi

3.6.3 Perancangan Tampilan Antarmuka

Dalam membuat program deteksi tepi akan dirancang tampilan GUI sebanyak empat buah fig-*file* dan *property inspector*-nya harus diatur lagi. Simpan keempat fig-*file* dengan nama 'menu_utama.fig', 'program.fig', 'tentang_program.fig', dan tentang_saya.fig'. Masing-masing fig-*file* ini menghasilkan m-*file* dengan nama yang sama, tapi ekstensinya beda, yaitu 'menu_utama.m', 'program.m', 'tentang_program.m', dan tentang_saya.m'. Berikut adalah tabel yang berisi perancangan dalam pembuatan program deteksi tepi.

Tabel 3.1 Perancangan program deteksi tepi

No.	Fig-*file*	Komponen	Nama	Fungsi
1	menu_utama	*Static Text*	Program Deteksi Tepi	Menampilkan tulisan (judul)
		Push Button	Program	Menampilkan *file* program
			Tentang_Program	Menampilkan *file* tentang_program
			Tentang_Saya	Menampilkan *file* tentang_saya
2	program	*Static Text* (32)	'disesuaikan'	Menampilkan tulisan
		Panel	Informasi	Mengelompokkan tulisan yang berisi informasi citra
		Button Group	Metode	Mengelompokkan semua metode deteksi tepi
			Tombol	Mengelompokkan tombol Reset, Hapus, dan Kembali
		Radio Button (2)	(Roberts, Sobel)	Menjalankan program sesuai metode deteksi tepi
		Push Button	Tampil Semua	Menampilkan semua hasil metode deteksi tepi

			Buka	Membuka *file* citra
			Reset	Mengulang proses dari awal
			Hapus	Menghapus citra dan informasi citra
			Kembali	Kembali ke menu_utama
		Axes (3)	-	Menampilkan citra
3	tentang_program	*Push Button*	Kembali	Kembali ke menu_utama
		Static Text (2)	'disesuaikan'	Menampilkan tulisan
4	tentang_saya	*Push Button*	Kembali	Kembali ke menu utama
		Static Text (7)	'disesuaikan'	Menampilkan tulisan
		Axes (3)	-	Menampilkan citra