Prosiding
ANNUAL RESEARCH SEMINAR 2016
6 Desember 2016, Vol 2 No. 1

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

Segmentasi Citra Digital Menggunakan *Thresholding Otsu* untuk Analisa Perbandingan Deteksi Tepi

Ayu Ambarwati[1]
Sistem Komputer , Fakultas Ilmu Komputer , Universitas Sriwijaya
Jl. Raya Prabumulih-Palembang Km. 32 Indralaya Ogan Ilir 30662
email : ayuambarwati29@gmail.com ,

Rossi Passarella[2] dan Sutarno[3]
Dosen Sistem Komputer , Fakultas Ilmu Komputer , Universitas Sriwijaya
Jl. Raya Prabumulih-Palembang Km. 32 Indralaya Ogan Ilir 30662
email : [2]Passarella.rossi@gmail.com ,
[3]tarnoabufaruq@yahoo.com

***Abstrak*—Pendeteksian tepi menjadi salah satu tahapan penting pengolahan citra dalam proses segmentasi karena dapat mempertegas batas-batas antara objek dan latar belakang. Banyaknya metode deteksi tepi saat ini menimbulkan keraguan dalam memilih metode deteksi tepi yang tepat dan sesuai dengan kondisi citra. Berdasarkan masalah tersebut dilakukan penelitian untuk menganalisis kinerja metode deteksi tepi *Sobel, Prewitt, Roberts* dan *Canny* menggunakan *thresholding Otsu* berdasarkan nilai *threshold*, waktu proses dan pengamatan visual. Program dibuat menggunakan perangkat lunak Microsoft Visual *C#* 2010 *Express*. Hasil penelitian terhadap tiga citra uji bahwa metode *Canny* menghasilkan tepian yang tipis dan halus serta tidak menghilangkan informasi penting pada gambar meskipun membutuhkan waktu komputasi yang tidak sedikit. Nilai *threshold* yang diperoleh dari *Otsu* merupakan nilai ambang terbaik dan optimal untuk setiap metode.**

Keywords—Tepi, Sobel, Prewitt, Roberts, Canny, threshold, Otsu.

I. PENDAHULUAN

Penerapan pengolahan citra sudah banyak diimplementasikan di berbagai bidang seperti bidang kedokteran, pertanian dan sebagainya. Salah satu tahapan pengolahan citra yakni proses pendeteksian tepi. Deteksi tepi citra berperan penting terhadap keakuratan hasil segmentasi dan analisis citra[1]. Tepi menggambarkan batas-batas antara objek dan latar belakang gambar yang membantu dalam proses segmentasi[2]. Tepi sangat penting dalam pengenalan gambar dan analisis karena mengandung informasi penting yang berguna[3]. Ada 3 teknik dalam deteksi tepi yakni operator turunan pertama, turunan kedua dan kompas[4]. Banyaknya metode deteksi tepi saat ini menimbulkan keraguan dalam memilih metode yang tepat dan sesuai dengan kondisi citra[1].

Penelitian tentang "Performances Analysis of Different Edge Detection Methods on Road Images", melakukan perbandingan beberapa metode deteksi tepi terhadap citra jalan, hasil penelitian menunjukkan operator *Canny* menghasilkan tepi yang lebih baik dari operator lain meskipun sangat kompleks dan membutuhkan waktu komputasi lebih lama.

Terdapat beberapa faktor yang mempengaruhi kualitas hasil deteksi tepi seperti derau, objek dengan intensitas yang sama, dan kepadatan tepi[2]. Masalah tersebut dapat diatasi dengan menyesuaikan nilai-nilai tertentu dalam deteksi tepi seperti mengubah nilai *threshold* [2]. Nilai *threshold* diatur terlalu tinggi akan menghilangkan informasi penting pada gambar sedangkan jika terlalu rendah akan menyebabkan kesalahan mendeteksi (*false edge*)[5].

Analisis menerapkan *thresholding* merupakan cara mengenali citra berdasarkan nilai ambang. Salah satu metode mendapatkan nilai *threshold* adalah metode *Otsu*[6]. Metode *Otsu* merupakan metode popular diantara semua metode *thresholding* dan metode terbaik dalam mendapatkan nilai *threshold* secara otomatis[7]. Pada penelitian [3] menerapkan *thresholding Otsu* untuk meminimalisasikan varian dari *pixel* hitam dan putih mengenai perbandingan kinerja metode deteksi tepi pada citra wajah.

Tujuan penelitian ini melakukan simulasi deteksi tepi yang menerapkan metode deteksi tepi *Sobel, Prewitt, Roberts* dan *Canny* menggunakan *thresholding Otsu*

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

berdasarkan nilai *threshold*, waktu proses dan pengamatan visual.

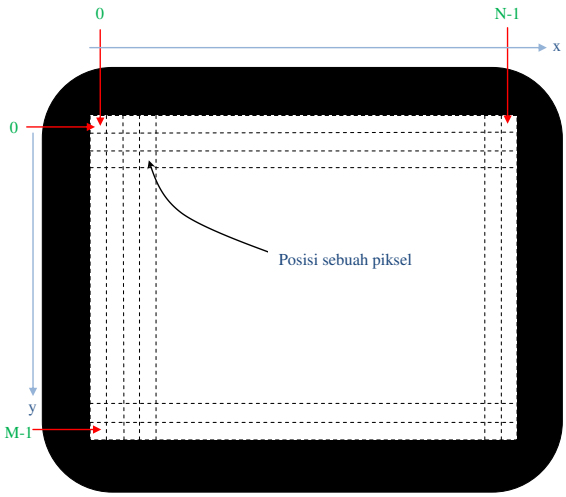
II. TINJAUAN PUSTAKA

2.1 Pengolahan Citra Digital

Pengolahan citra digital merupakan proses memanipulasi citra dengan bantuan komputer tujuannya yakni memperbaiki kualitas citra dan mengolah informasi pada gambar untuk keperluan pengenalan dan proses selanjutnya[9].

2.2 Citra Digital

Citra digital dibentuk dari sekumpulan piksel (*pixel* atau "picture element")[8]. Setiap piksel digambarkan sebagai kotak kecil. Citra digital didefinisikan sebagai fungsi *f(x,y)* ukuran N kolom dan M baris, x y merupakan koordinat spasial dan *f* adalah titik koordinat yang menunjukkan nilai keabuan dititik tersebut ditunjukkan pada gambar 2.1.



Gambar 2.1. Sistem koordinat citra berukuran M x N

Berdasarkan nilai *pixel*, citra digital dikelompokkan kedalam tiga jenis citra yaitu :

1. Citra Warna

Citra Warna (RGB) memiliki 3 komponen warna yakni *Red, Green* dan *Blue* disetiap *pixel*[8]. Setiap komponen warna menggunakan 8 bit (nilai berkisar 0 sampai 255). Dengan demikian, kemungkinan warna yang bisa disajikan mencapai 255x255x255 atau 16.581.375 warna[10].

2. Citra Aras Keabuan

Citra aras keabuan memiliki satu nilai kanal disetiap *pixel*, dengan kata lain nilai kanal *red* = *green* = *blue* [10]. Nilai tersebut menunjukkan tingkat intensitas. Tingkat intensitas dimulai dari warna dimulai dari hitam, keabuan sampai putih. Citra aras keabuan memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan) diperoleh menggunakan persamaan 2.1. berikut :

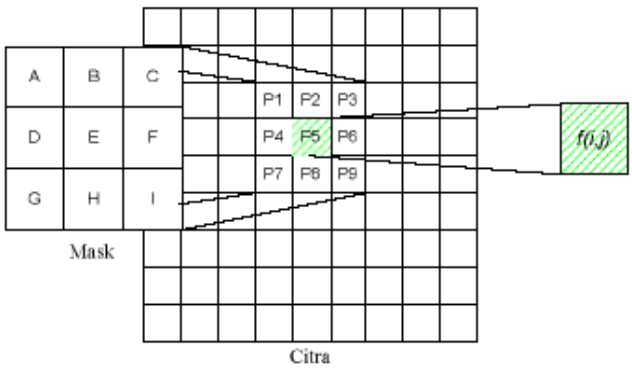
Aras Keabuan = ((*Red* * 0,299) + (*Green* * 0,587) + (*Blue* * 0,114))

3. Citra Biner

Citra biner memiliki dua kemungkinan nilai *pixel* yakni hitam dan putih atau 0 dan 1[8]. Citra biner sering muncul sebagai hasil proses pengambangan *(thresholding)*, segmentasi ataupun morfologi[10].

2.3 Konvolusi

Konvolusi merupakan proses memperoleh suatu piksel baru berdasarkan nilai piksel itu sendiri dan tetangganya, dengan melibatkan kernel[8]. Kernel yang digunakan ukuran 3x3 atau 5x5. Konvolusi dilakukan dengan menggeser kernel sepanjang baris dan kolom sehingga diperoleh nilai yang baru. Ilustrasi konvolusi dijelaskan pada gambar 2.2.



$$f(i,j) = AP_1 + BP_2 + CP_3 + DP_4 + EP_5 + FP_6 + GP_7 + HP_8 + IP_9$$

Gambar 2.2. Contoh Konvolusi

2.4 Segmentasi Citra

Segmentasi citra bertujuan mendapatkan objek-objek citra dengan cara membagi citra ke dalam beberapa daerah yang memiliki kemiripan atribut[8].

Terdapat dua teknik segmentasi citra yaitu mendeteksi diskontinuitas dan similaritas. Pendekatan diskontinuitas membagi citra berdasarkan perubahan intensitas yang tiba-tiba seperti deteksi titik, deteksi garis, dan deteksi tepi. Sedangkan pendekatan similaritas memecah citra ke dalam daerah yang sama menurut beberapa kriteria yang sudah ditentukan, seperti proses *thresholding, region growing,* dan *region splitting* dan *merging*[11].

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

2.5 *Thresholding*

Thresholding mengubah citra keabuan menjadi citra biner bergantung pada nilai *threshold* (T) sehingga dapat diketahui daerah mana yang termasuk objek dan latar belakang[12]. Jika nilai *pixel* lebih besar dari *threshold* diatur menjadi 1 sebaliknya jika kurang dari *threshold* diatur menjadi 0. Secara umum proses *thresholding* ditunjukkan pada persamaan 2.2.

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) \geq T \\ 0 \; if \; f(x,y) < T \end{cases}$$

Persamaan 2.2 menjelaskan bahwa *g(x,y)* adalah citra biner dari citra aras keabuan *f(x,y)* dan T menyatakan nilai ambang.

2.5.1 Metode Otsu

Konsep *thresholding Otsu* pertama kali diperkenalkan oleh *Nobuyuki Otsu* (1979) untuk pengelompokkan citra biner berdasarkan bentuk histogram secara otomatis, mengasumsikan bahwa citra berisi dua kelas dasar dengan bentuk histogram bimodal (*foreground* dan *background*)[12].

Tujuan dari metode *Otsu* membagi histogram citra keabuan kedalam dua daerah yang berbeda secara otomatis tanpa membutuhkan bantuan user. Pendekatan dengan melakukan analisis diskriminan yakni menentukan suatu variabel lalu memaksimumkan variabel tersebut agar dapat memisahkan objek dengan latar belakang[10][13].

Berikut algoritma Otsu menentukan *threshold* (k)[10]. Nilai k berkisar antara 0 sampai 255.

1. Probabilitas setiap *pixel* pada *gray level* i
$$pi = \frac{ni}{N}$$
n_i = jumlah *pixel* pada *level* ke *i*
2. Jumlah kumulatif (zerothCM)
$$\omega(k) = \sum_{i=0}^{k} pi$$
3. Rerata kumulatif (firstCM)
$$\mu(k) = \sum_{i=0}^{k} i.\,pi$$
4. Rerata intensitas global (tMean)
$$\mu T = \sum_{i=0}^{L} i.\,pi$$
5. Nilai ambang k ditentukan dengan memaksimumkan persamaan :
$$\sigma_B{}^2(k^*) = \max(\sigma_B{}^2(k))$$

Dengan

$$\sigma_B{}^2(k) = \frac{[\mu T\,\omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]}$$

Metode *Otsu* merupakan metode popular diantara semua metode *thresholding* dan metode terbaik dalam mendapatkan nilai *threshold* secara otomatis[7].

2.6 Deteksi Tepi

Deteksi tepi digunakan untuk memperoleh tepi-tepi dari objek[8]. Deteksi tepi menggunakan perubahan nilai intensitas yang drastis pada batas dua area. Tepi didefinisikan sebagai perubahan nilai intensitas derajat keabuan yang mendadak (besar) dalam jarak yang singkat[4]. Tepi mengandung informasi penting berupa bentuk maupun ukuran objek.

Terdapat beberapa teknik dalam mendeteksi tepi yaitu orde pertama menggunakan turunan pertama seperti operator *Roberts*, *Prewitt*, *Sobel* lalu orde kedua menggunakan turunan kedua seperti *Laplacian of Gaussian* (LoG) dan operator kompas yakni mendeteksi tepi kesegala arah mata angin seperti *Kirsh, Robinson*[8].

Deteksi tepi turunan orde pertama menggunakan operator gradient. Operator gradient didefinisikan sebagai vektor berikut pada persamaan 2.3.

$$G[f(x,y] = G\begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix}$$

Besar gradient magnitude dihitung dengan persamaan 2.4.

$$G[f(x,y] = \sqrt{Gx^2 + Gy^2}$$

Arah gradient dihitung dengan persamaan 2.5.

$$\theta(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

2.6.1 *Sobel*

Sobel menggunakan dua buah kernel horizontal dan vertikal berukuran 3x3 sehingga perkiraan gradien berada tepat ditengah jendela[3][5]. Operator ini sensitif terhadap *noise*[3]. Keuntungan operator ini memberikan hasil deteksi tepi yang baik dan mudah dalam implementasi[3]. Operator

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

Sobel sensitif terhadap arah tepi diagonal daripada tepi vertikal dan horizontal[8]. Operator *Sobel* diterapkan dalam kernel berikut:

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_7 & (x,y) & a_3 \\ a_6 & a_5 & a_4 \end{bmatrix}$$

Gambar 2.3. Kernel ukuran 3x3

Didefinisikan Gx dan Gy pada persamaan 2.6. berikut :

$$G_x = (a_2 - a_0) + (a_4 - a_6) + c(a_3 - a_7)$$
$$G_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$G_y = (a_0 - a_6) + (a_2 - a_4) + c(a_1 - a_5)$$
$$G_y = (a_0 + ca_1 + a_2) - (a_4 + ca_5 + a_6)$$

c adalah konstanta yang bernilai 2.

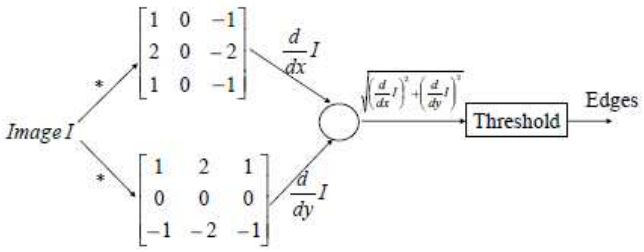
-1	0	1
-2	0	2
-1	0	1

(a) G_x

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

(b) G_y

Gambar 2.4. Kernel *Sobel*



Gambar 2.5. Deteksi Tepi *Sobel*

2.6.2 *Prewitt*

Prewitt menggunakan perhitungan gradient yang sama dengan *Sobel* yakni diperolej dari hasil gabungan konvolusi tepi arah vertikal dan horizontal, tetapi menggunakan nilai konstanta c = 1[8]. Operator ini tidak menekankan pembobotan pada *pixel* yang lebih dekat dengan titik pusat kernel[14]. Operator *Prewitt* sensitif terhadap arah tepi vertikal dan horizontal[8].

-1	0	1
-1	0	1
-1	0	1

(a) G_x

1	1	1
0	0	0
-1	-1	-1

(b) G_y

Gambar 2.6. Kernel *Prewitt*

2.6.3 *Roberts*

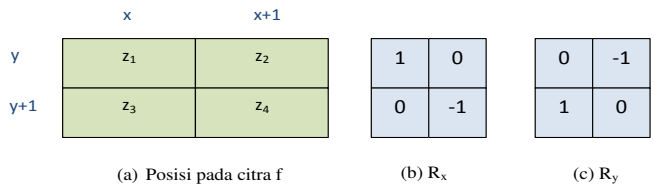
Roberts disebut operator *Roberts Cross* (silang) karena menggunakan kernel ukuran 2x2 *pixel*, sehingga tepi yang dihasilkan berada pada tepi atas atau tepi bawah[8]. Ukuran filter kecil membuat komputasi sangat cepat, namun menimbulkan kelemahan yakni sensitif terhadap derau jika tepi itu lemah. Operator *Roberts* mengambil arah diagonal untuk penentuan arah dalam perhitungan nilai gradient menggunakan persamaan 2.7.

$$Gx = f(x, y) - f(x+1, y+1)$$
$$Gy = f(x+1, y) - f(x, y+1)$$

Dalam bentuk konvolusi diimplementasikan sebagai berikut :

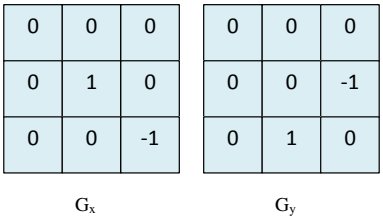
2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id



Gambar 2.7. Kernel *Roberts*

Pada simulasi penelitian ini kernel dimodifikasi menjadi kernel 3x3 dengan memberikan nilai 0 pada elemen kernel tambahan[14].



Gambar 2.8. Kernel *Roberts*

2.6.4 *Canny*

Canny dikembangkan oleh *John Canny* pada tahun 1986, menggunakan algoritma multi-tahap dalam mendeteksi tepi citra. Algoritma ini memiliki 3 kriteria yakni memberikan tingkat kesalahan yang seminimum mungkin, melokalisasi titik-titik tepi (jarak piksel-piksel tepi yang ditemukan deteksi dan tepi yang sesungguhnya sangat pendek), dan hanya memberikan satu respon untuk satu tepi[8]. Terdapat enam tahap dalam mendeteksi tepi *Canny* sebagai berikut :

1. Mengimplementasikan tapis *Gaussian* untuk mereduksi *noise* dan meningkatkan kualitas detail citra. Proses ini menghasilkan citra tampak sedikit lebih buram yang. Tujuan proses ini mendapatkan tepian citra yang sebenarnya sehingga garis-garis halus yang berada di area gradasi pada citra masih bisa dideteksi. Filter *Gaussian* 2D dinyatakan persamaan 2.8. berikut :

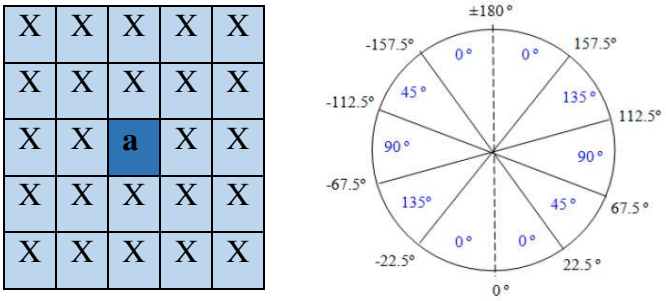
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

dimana *G(x,y)* elemen matriks *gaussian* pada posisi (x,y), π=22/7, e = 2.71828182846, σ ialah nilai standar deviasi. Berdasarkan rumus diatas, salah satu contoh filter *Gaussian* dengan σ =1.4 dan ukuran kernel 5x5 :

$$\frac{1}{115}$$

Gambar 2.9. Filter *Gaussian*

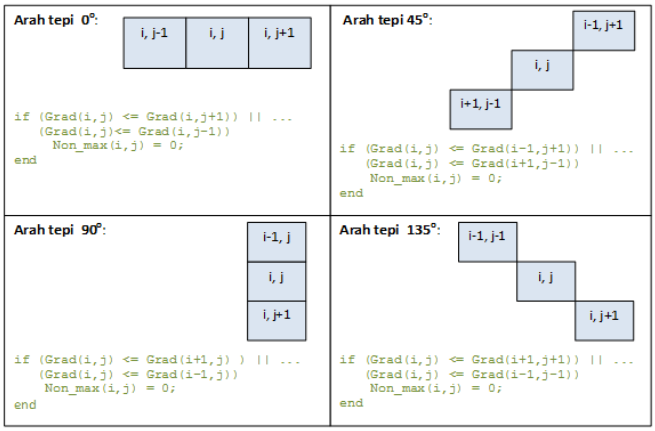
2. Melakukan perhitungan gradient magnitude dan arah tepi. Perhitungan gradient magnitude menggunakan salah satu kernel *Roberts, Prewitt,* atau *Sobel*. Dalam penelitian ini menggunakan kernel *Sobel*. Sedangkan perhitungan arah tepi menggunakan persamaan 2.5. Selanjutnya melakukan orientasi tepi dengan menghubungkan arah tepi ke arah yang dapat ditelusuri dalam citra. Sebagai contoh, terdapat susunan *pixel* berukuran 5x5. Kemudian dengan melihat piksel a, hanya terdapat empat kemungkinan arah yakni $0^0, 45^0, 90^0, 135^0$.



Gambar 2.10. Pembagian sudut gradient penentu arah

3. Memperkecil garis tepi yang muncul dengan menerapkan *nonmaximum suppresion* sehingga menghasilkan garis tepian yang lebih ramping. *Nonmaximum Suppresion* dilaksanakan dengan memperhatikan dua titik tetangga yang terletak pada arah tepi. Jika nilai *pixel* titik perhatian lebih besar daripada gradien kedua tetangga, nilainya akan dipertahankan. Sebaliknya, jika *pixel* titik perhatian lebih kecil daripada nilai salah satu atau kedua gradien tetangga, nilainya akan diubah menjadi 0.

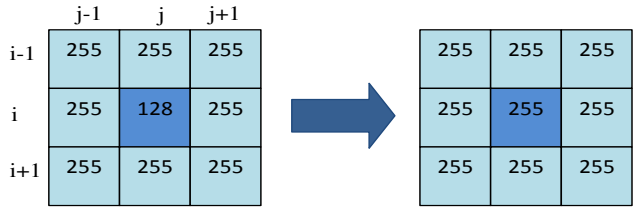
ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id



Gambar 2.11. *Nonmaximum Suppresion*

4. Menerapkan dua buah *threshold* (*double thresholding*). Sederhananya *double thresholding* bertujuan klasifikasi dua buah nilai *High-threshold* (T_2) dan *Low-threshold* (T_1), dengan $T_2 \approx 2T_1$[7]. Jika nilai *pixel* lebih besar atau sama dengan T_2 maka diatur nilai 255, jika nilai *pixel* kurang dari atau sama dengan T_1 maka diatur menjadi 0. *Pixel* diantara T_1 dan T_2 disebut kandidat *pixel* tepi maka sementara diberi nilai 128.

5. *Edge tracking by hysteresis* bertujuan memperoleh tepian final dengan menekan semua sisi yang tidak terhubung pada tepian yang sangat kuat[15]. Nilai 128 selanjutnya dilakukan pengecekan pada *pixel* dari 8 arah tetangganya, sehingga *pixel* hanya bernilai 0 atau 255. Perubahan nilai 128 menjadi nilai 255 apabila semua kondisi terpenuhi yakni jika salah satu atau semua *pixel* pada 8 arah tetangganya bernilai 255. Proses pengujian dilakukan sampai tidak ada lagi perubahan dari nilai 128 menjadi 255. Selanjutnya, semua *pixel* yang bernilai 128 yang tersisa diubah menjadi 0.

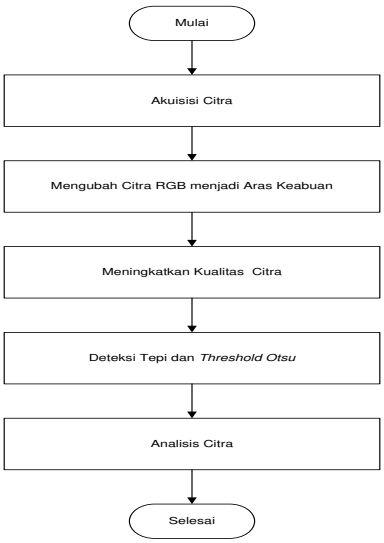


Gambar 2.12. Pengujian mengubah nilai 128 menjadi 255

III. PERANCANGAN PROGRAM

3.1 Perancangan Sistem

Aplikasi dibuat menggunakan perangkat lunak Microsoft Visual Studio *C#* 2010 *Express*. Berikut gambar 3.1. diagram alir program yang akan dibuat :



Gambar 3.1. Diagram Alir Program

3.2 Akuisisi Citra

Citra yang digunakan berupa citra diam tunggal yang diperoleh dari sumber internet (https://homepages.cae.wisc.edu/~ece533/images/) kemudian di-*resize* menjadi ukuran 320x240 *pixel* dalam format JPEG. Terdapat 3 buah citra yang digunakan dalam penelitian:

(1) Citra Lena

(2) Citra Barbara

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

(3) Citra Cameramen

Gambar 3.2. Citra digital

3.3 Pengubahan ke aras keabuan

Proses pengolahan citra warna lebih sulit karena mengandung tiga komponen warna sehingga perlu diubah menjadi citra aras keabuan. Pengubahan citra RGB bertujuan menyederhanakan citra RGB yang tadinya 3 *channels* berubah menjadi 1 *channels* sehingga memudahkan proses selanjutnya. Citra aras keabuan diperoleh menggunakan persamaan 2.1.

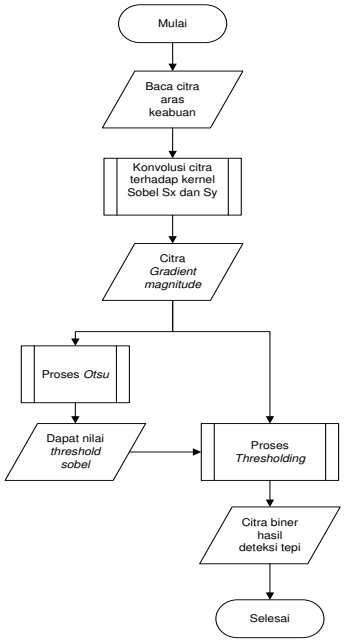
3.4 Peningkatan Kualitas Citra

Pada kenyataannya tidak bisa dipungkiri bahwa citra yang diambil dari kamera akan berisi sejumlah *noise*. Untuk mencegah *noise* salah dideteksi perlu adanya peningkatan kualitas citra dilakukan dengan melakukan penapisan citra untuk menghilangkan *noise* pada citra menggunakan tapis *Median* ukuran matriks 3x3. Semakin besar ukuran matriks yang digunakan maka akan menghasilkan citra yang kabur tetapi faktanya akan menghilangkan tepi-tepi objek pada citra[3].

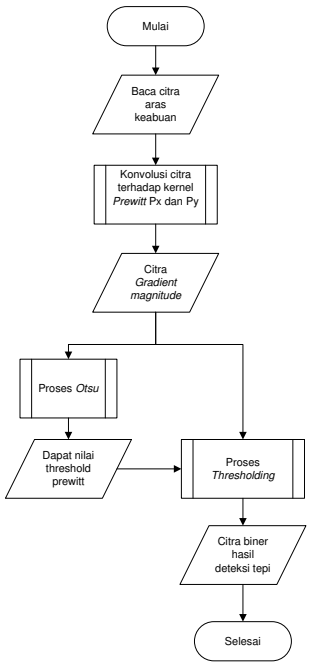
3.5 Deteksi Tepi dan *Threshold Otsu*

Tahapan pertama mendeteksi tepi yakni konvolusi citra terhadap kernel 3x3 *Sobel/Prewitt/Roberts* sehingga menghasilkan citra gradient magnitude. Selanjutnya lakukan proses *Otsu* pada citra gradient magnitude untuk menghasilkan nilai *threshold* sehingga diperoleh nilai *threshold* berbeda tiap operator. *Threshold* digunakan dalam mempertegas tepian.

Pada *Canny* memiliki banyak tahapan akan dijabarkan pada penjelasan berikutnya. Berikut gambar 3.3. flowchart tahapan deteksi tepi *Sobel*. Pada *Prewitt* atau *Roberts* tinggal diubah kernelnya pada proses konvolusi menggunakan kernel *Prewitt* atau *Roberts*.

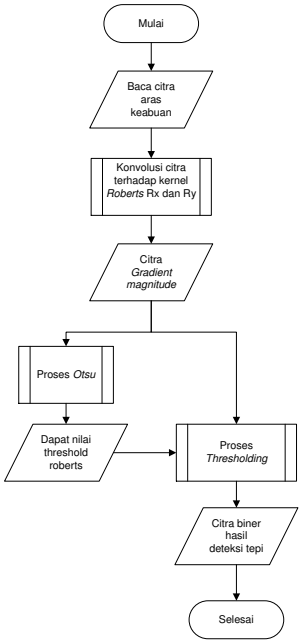


Gambar 3.3. Flowchart deteksi Tepi *Sobel*

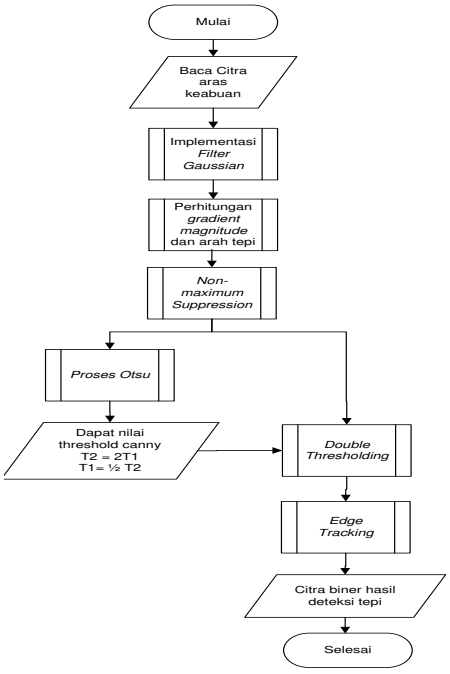


Gambar 3.4. Flowchart Deteksi Tepi *Prewitt*

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id



Gambar 3.5. Flowchart Deteksi Tepi *Roberts*



Gambar 3.6. Flowchart Deteksi Tepi *Canny*

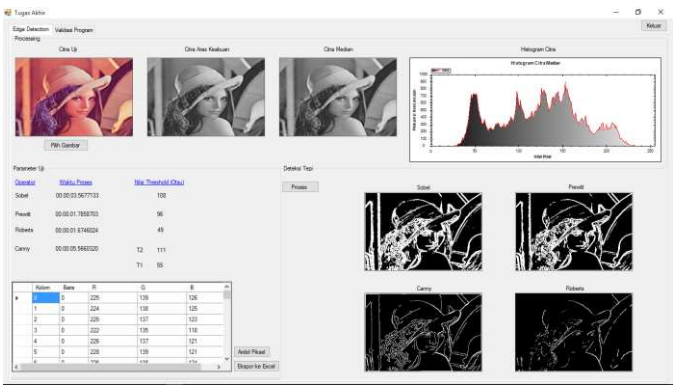
3.6 Analisis Citra

Analisis citra berdasarkan pengukuran subjektif yakni pengamatan visual dari hasil deteksi tepi citra yang diperoleh dan pengukuran objektif yakni waktu proses dan nilai *threshold* tiap metode deteksi tepi.

IV. HASIL PENELITIAN

4.1 Pengujian Program

Program aplikasi deteksi tepi digunakan untuk mengolah citra masukan menjadi citra deteksi tepi dengan operator deteksi tepi *Sobel, Prewitt, Roberts* dan *Canny*. Selain itu program melakukan perhitungan waktu proses dan nilai *threshold* tiap operator.



Gambar 4.1. Contoh tampilan penggunaan aplikasi deteksi tepi

4.2 Pengubahan ke Aras Keabuan

Proses pertama kali setelah mendapatkan citra RGB ialah proses *grayscalling* mengubah citra RGB menjadi citra aras keabuan dengan tujuan mempermudah proses pengolahan. Berikut gambar 4.2. hasil citra aras keabuan pada citra uji.

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

Gambar 4.2. Citra Aras Keabuan

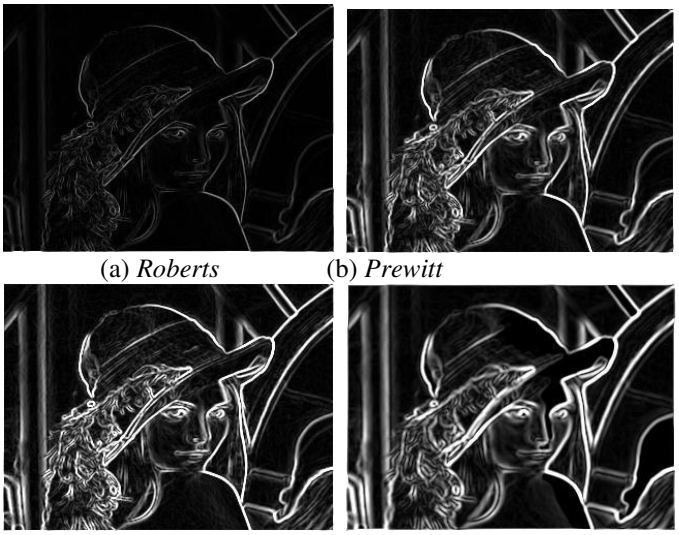
4.3 Peningkatan Kualitas Citra

Peningkatan kualitas citra dilakukan untuk memperoleh keindahan citra dan mengoreksi citra dari segala *noise* akibat perekaman data. Peningkatan kualitas citra dilakukan dengan menerapkan tapis *Median* ukuran 3x3.

Gambar 4.3. Hasil peningkatan mutu citra menggunakan tapis *Median*

4.4 Deteksi Tepi dan *Threshold Otsu*

Deteksi tepi dilakukan pada citra yang telah ditingkatkan mutunya. Tahapan pertama mendeteksi tepi yakni melakukan konvolusi citra terhadap masing-masing kernel yang menghasilkan citra baru yakni citra gradient magnitude.



(a) *Roberts* (b) *Prewitt*

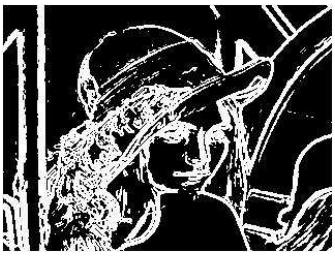
(c) *Sobel* (d) *Canny*

Gambar 4.3. Citra gradient magnitude

Untuk menentukan suatu nilai cukup signifikan bisa menggunakan *threshold*. Suatu nilai *pixel* pada citra gradient magnitude merupakan bagian dari tepi jika nilainya lebih besar dari *threshold*. Perhitungan nilai *threshold* berdasarkan metode *Otsu* menghasilkan nilai ambang optimal. Nilai ambang tersebut digunakan dalam *thresholding* untuk mendefinisikan tepi. Pada *Canny* memiliki banyak tahapan, langkah pertama yang dilakukan citra diperhalus menggunakan *Gaussian filter* menggunakan persamaan 2.8. Selanjutnya citra dilakukan konvolusi menggunakan kernel *Sobel* menghasilkan citra gradient magnitude. Kemudian dilanjutkan tahap *nonmaxima supression,* proses *Otsu, double thresholding* dan terakhir *edge tracking* seperti ditunjukkan pada flowchart gambar 3.4.

(a) *Roberts* T = 49 (b) *Prewitt* T = 96

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id



(c) *Sobel* T = 109 (d) *Canny* T = 111

Gambar 4.4. Hasil deteksi tepi citra Lena dengan nilai *threshold Otsu*

(a) *Roberts* T = 35 (b) *Prewitt* T = 97

(c) *Sobel* T = 111 (d) *Canny* T = 111

Gambar 4.5. Hasil deteksi tepi citra Barbara dengan nilai *threshold Otsu*

(a) *Roberts* T = 52 (b) *Prewitt* T = 113

(c) *Sobel* T = 120 (d) *Canny* T = 124

Gambar 4.6. Hasil deteksi tepi citra Cameramen dengan nilai *threshold Otsu*

4.5 Analisis Citra

Berdasarkan hasil pengujian terhadap tiga citra uji secara visual memberikan hasil bahwa metode deteksi tepi *Canny* menghasilkan tepian citra yang nyata, tipis dan halus. *Canny* mampu mendeteksi garis halus pada gambar yang tidak dapat dideteksi oleh operator lain. Selain itu tidak menghilangkan informasi penting yang terkandung pada gambar. Sementara pada *Sobel, Prewitt* maupun *Roberts* menghasilkan tepian yang tebal sehingga menimbulkan kerancuan dan juga menghilangkan informasi penting yang terkandung pada gambar.

Tabel 4.1. Waktu proses metode deteksi tepi

Citra Uji	Operator			
	Roberts	*Prewitt*	*Sobel*	*Canny*
Lena	1,67	1,79	3,58	5,58
Barbara	1,69	1,95	3,74	5,59
Cameramen	1,39	1,53	3,05	4,62
Rata-rata	1,58	1,76	3,46	5,26

Dari tabel 4.1 dapat dilihat waktu proses paling cepat adalah *Roberts* dengan rata-rata 1,58 detik, sementara *Canny* membutuhkan waktu yang cukup lama yakni rata-rata 5,26 detik. Hal itu disebabkan *Canny* menggunakan algoritma multi-tahap sehingga memerlukan komputasi yang cukup banyak. Selain itu banyak sedikitnya waktu pemrosesan juga bergantung pada ukuran kernel yang digunakan, pada penelitian ini menggunakan ukuran kernel 3x3.

ISBN : 979-587-626-0 | UNSRI http://ars.ilkom.unsri.ac.id

Tabel 4.2. Nilai *Threshold Otsu*

Citra Uji	*Threshold*			
	Roberts	*Prewitt*	*Sobel*	*Canny*
Lena	49	96	109	111
Barbara	35	97	111	111
Cameramen	53	113	120	124

Dari tabel 4.2 bahwa keempat metode deteksi tepi memberikan nilai threshold berbeda-beda terhadap ketiga citra uji. Nilai threshold Otsu digunakan untuk mendefinisikan tepi sehingga citra deteksi tepi berupa citra biner yang memiliki nilai 0 atau 255.

V. PENUTUP

Setelah dilakukan pengujian dan analisa dari hasil simulasi dapat disimpulkan sebagai berikut:

1. *Canny* memberikan hasil tepian yang tipis dan halus yang tidak bisa oleh dideteksi oleh metode deteksi yang lain. Selain itu *Canny* tidak menghilangkan informasi penting gambar.
2. *Roberts* membutuhkan waktu komputasi sangat cepat dibandingkan dengan yang lain, sementara *Canny* membutuhkan banyak waktu dalam komputasinya.

Nilai *threshold Otsu* merupakan nilai yang optimal. Nilai *threshold* diatur terlalu tinggi akan menghilangkan informasi pada gambar sedangkan jika terlalu rendah akan menyebabkan kesalahan mendeteksi tepian.

REFERENSI

[1] Febriani and L. ETP, "Analisis Penelusuran Tepi Citra Menggunakan Detektor Tepi Sobel dan Canny," *Proceeding, Semin. Ilmu Nas. Komput. dan Sist. Intelijen*, pp. 20–21, 2008.

[2] M. Avlash and L. Kaur, "Performances Analysis of Different Edge Detection Methods on Road Images," *Int. J. Adv. Res. Eng. Appl. Sci.*, vol. 2, no. 6, pp. 27–38, 2013.

[3] S. B. Kutty, S. Saaidin, P. Nor, A. Megat, and S. A. Hassan, "Evaluation of Canny and Sobel Operator for Logo Edge Detection," vol. 2, no. Istmet, pp. 153–156, 2014.

[4] D. Agushinta R and A. Diyanti, "Perbandingan Kinerja Metode Deteksi Tepi pada Citra Wajah," 2004.

[5] Z. Othman, M. Rafiq, and A. Kadir, "Comparison of Canny and Sobel Edge Detection in MRI Images," pp. 133–136.

[6] R. F. Gusa, "Pengolahan Citra Digital untuk Menghitung Luas Daerah Bekas Penambangan Timah," *J. Nas. Tek. Elektro*, vol. 2, no. 2, pp. 27–34, 2013.

[7] M. Fang, G. Yue, and Q. Yu, "The Study on An Application of Otsu Method in Canny Operator," *Int. Symp. Inf. ...*, vol. 2, no. 4, pp. 109–112, 2009.

[8] A. Kadir and A. Susanto, *Teori dan Aplikasi Pengolahan Citra*. Yogyakarta, Indonesian: Andi, 2013.

[9] M. Kamal and R. Suko Basuki, "Segmentasi Citra Daun Tembakau Berbasis Deteksi Tepi Menggunakan Algoritma Canny," 2012.

[10] D. Putra, *Pengolahan Citra Digital*. Yogyakarta, Indonesian: Andi, 2010.

[11] E. Prasetyo, *Pengolahan Citra Digital dan Aplikasinya Menggunakan MatLab*. Yogyakarta, Indonesian: Andi, 2011.

[12] V. Sucharita, S. Jyothi, and D. M Mamatha, "Evaluation of the Digital images of Penaeid Prawns Species Using Canny Edge Detection and Otsu Thresholding Segmentation," *Int. J. Emerg. Technol. Comput. Appl. Sci. (IJETCAS)*, pp. 117–121, 2013.

[13] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man. Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[14] M. Yunus, "Perbandingan Metode-metode Edge Detection untuk Proses Segmentasi Citra Digital," *J. Teknol. Inf.*, vol. 3, no. 2, pp. 146–160.

[15] C. Danil, "Edge Detection dengan Algoritma Canny," pp. 197–203.