IMPLEMENTASI METODE CANNY PADA SEGMENTASI CITRA DIGITAL MATLAB 2016

Disusun dan Diajukan Untuk Memenuhi Persyaratan Ujian Akhir
Memperoleh Gelar Sarjana Komputer Pada Fakultas Sains dan Teknologi
Universitas Pembangunan Panca Budi
Medan

SKRIPSI

OLEH:

NAMA : **TRI AFNI TAMBUNAN**
NPM : **1414370502**
PROGRAM STUDI : **SISTEM KOMPUTER**

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS PEMBANGUNAN PANCA BUDI
MEDAN
2019

ABSTRAK

TRI AFNI TAMBUNAN
IMPLEMENTASI METODE CANNY PADA SEGMENTASI CITRA DIGITAL MATLAB 2016

Tahun2019

Proses segmentasi pada citra digital yang memisahkan sebuah objek dari background atau latar yang di dapat dari nilai RGB di setiap pixel pada citra digital sehingga objek tersebut dapat diproses untuk keperluan yang lain. Seiring dengan berkembangnya teknologi pada aplikasi yang memproses citra digital maka proses segmentasi menjadi semakin diperlukan. Hasil dari segmentasi juga harus semakin akurat karena jika hasil segmentasi tidak akurat maka akan mempergaruhi hasil proses selanjutnya.Secara umum proses segmentasi dibagi menjadi tiga bagian berdasarkan klasifikasi, berdasarkan tepi dan berdasarkan daerah.Sistem ini bertujuan untuk melakukan segmentasi objek image dengan menggunakan metode *Canny*. Untuk metode *Canny*, proses dimulai dengan menginputkan citra digital kemudian dilakukan proses *grayscale*. Selanjutnya pemilihan metode lalu melakukan proses deteksi tepi dengan operator *Canny* atau *Laplacian* dan terakhir adalah proses dilasi.Hasil segmentasi berhasil memisahkan objek dengan menggunakan deteksi tepi dengan operator *canny*. Sistem ini dikembangkan dengan menggunakan bahasa pemrograman Matlab 2016a.

Kata kunci :*Canny,Citra, Matlab 2016a, Segmentasi.*

DAFTAR ISI

Halaman

BAB III ANALISIS DAN PERANCANGAN SISTEM

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

BAB V PENUTUP

DAFTAR PUSTAKA

BIOGRAFI PENULIS

LAMPIRAN-LAMPIRAN

KATA PENGANTAR

Segala Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan segala rahmat-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Implementasi Metode Canny Pada Segmentasi Citra Digital Matlab 2016" guna memenuhi sebagian persyaratan untuk memperoleh gelar Sarjana (S-1) pada Fakultas Sains dan Teknologi Universitas Pembangunan Pancabudi Medan.

Penulis menyadari kelemahan serta keterbatasan yang ada sehingga dalam menyelesaikan skripsi ini memperoleh bantuan dari berbagai pihak, dalam kesempatan ini penulis menyampaikan ucapan terimakasih kepada :

1. Teristimewa untuk Ibu, suami dan seluruh Keluarga tercinta yang telah banyak memberikan doa dukungan dan memberi semangat, sehingga penulis dapat menyelesaikan penulisan skripsi ini.
2. Bapak Dr. H. Muhammad Isa Indrawan, S.E., M.M, selaku Rektor Universitas Pembangunan Panca Budi Medan.
3. Ibu Sri Shindi Indira, S.T., M.Sc, selaku Dekan Fakultas Sains Dan Teknologi Universitas Pembangunan Panca Budi Medan.
4. Bapak Muhammad Iqbal, S.Kom., M.Kom, selaku Ketua Program Studi Sistem Komputer Fakultas Sains dan Teknologi Universitas Pembangunan Panca Budi Medan dan juga selaku Dosen Pembimbing I yang telah memberikan arahannya dalam menyelesaikan penyusunan Skripsi ini.
5. Ibu, T.Henny Harumy S.Kom, M.Kom selaku Dosen Pembimbing II yang telah memberikan peluang waktu, arahannya dalam menyelesaikan penyusunan Skripsi ini.
6. Seluruh Dosen Fakultas Sains dan Teknologi Universitas Pembangunan Panca Budi Medan yang telah memberikan ilmunya kepada penulis.
7. Sahabat dan rekan seperjuangan tercinta yang tiada henti memberi dukungan dan motivasi kepada penulis.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan baik isi maupun susunannya. Semoga skripsi ini dapat bermanfaat tidak hanya bagi penulis juga bagi para pembaca.

Medan, 17 Desember 2018
Penulis

(Tri Afni Tambunan)
NPM : 1414370502

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Pemanfaatan pengolahan citra dalam era yang hampir serba digital ini sangat dibutuhkan dalam berbagai bidang, seperti dalam bidang kedokteran, forensik, hukum, perdangangan, Pendidikan maupun dalam kehidupan sehari hari, khusunya citra yang dapat menyampaikan informasi. Citra digital dapat diolah ataupun dimodifikasi menjadi citra digital yang lain, proses ini disebut proses pengolahan citra digital. Pengolahan citra memiliki tujuan untuk memperbaiki kualitas citra agar mudah di interpretasi oleh manusia atau komputer. Operasi pengolahan citra digital mencakup perbaikan citra, pemampaatan citra, segmentasi citra dan deteksi tepi citra.

Segmentasi citra dilakukan sebagai pemisahan objek yang satu dengan objek yang lain dalam suatu citra atau antara objek dengan latar yang terdapat dalam sebuah citra. Dengan proses segmentasi tersebut, masing-masing objek pada citra dapat diambil secara individu sehingga dapat digunakan sebagai input. Proses segmentasi pada citra digital dilakukan dengan deteksi tepi untuk menyediakan jalur target di sepanjang tepi (*edge*) objek citra. *Edge* pada gambar adalah area pada tepi objek *image* dengan intensitas *contrast* yang kuat, sehingga terlihat lompatan intensitas dari satu pixel ke pixel lainnya. *Edge* mendeteksi gambar dengan mengurangi secara signifikan jumlah data dan menyaring informasi yang tidak berguna (*filtering*) dan juga menjaga *property* penting yang diperlukan pada gambar.

Deteksi tepi (*Edge Detection*) pada suatu citra suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah untuk menandai bagian yang menjadi detail citra dan untuk memperbaiki detail dari citra yang kabur yang terjadi karena *error* atau adanya efek dari proses akusisi citra. Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya.

Ada banyak metode untuk pendeteksian *tepi* ini, tetapi metode untuk melokasikan *edge* ini merupakan karakteristik dari kategori "*gradient filter*" dan termasuk untuk pencarian segmentasi citra dengan menggunakan metode *canny*. Lokasi pixel dinyatakan/dideklarasikan jika nilai dari gradient melewati ambang batasan (x,y). Teknik *differensial* yang dikembangkan, yaitu *differensial* pada arah horizontal dan *differensial* pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan *differensial*. Teknik konversi biner yang disarankan adalah konversi biner dengan meratakan distribusi warna hitam dan putih. Metode *Canny* merupakan deteksi tepi yang optimal.

Adapun yang menjadi permasalahan pada segmentasi citra digital adalah mengetahui nilai pada setiap pixelnya pada citra asli dan harus dirubah ke citra *gryascale* untuk mendapatkan hasil deteksi tepi suatu objek. Maka dari itu digunakan metode *canny* untuk menyaring bagian citra dari citra awal/asli untuk mendapatkan hasil deteksi tepi yang jelas dan halus. Berdasarkan latar belakang masalah ini, penulis tertarik untuk memilih judul pada skripsi ini yaitu **"SEGMENTASI CITRA DIGITAL MENGGUNAKAN METODE CANNY "**

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan di atas, yang menjadi rumusan masalah adalah sebagai berikut:

1. Bagaimana membangun sebuah aplikasi yang mampu mendeteksi batas tepi menggunakan Program Matlab 2016a ?
2. Bagaimana proses melakukan segmentasi pendeteksian citra digital untuk menghasilkan citra *Grayscale* ?
3. Bagaimana cara mengimplementasikan pengolahan citra digital untuk mendeteksi tepi citra digital dengan metode *canny* ?

1.3 Batasan Masalah

Pada skripsi ini pembahasan akan dibatasi pada permasalahan-permasalahan sebagai berikut :

1. Citra *digital* yang diolah dalam program menggunakan format .Jpg.
2. Pengambilan citra / gambar dilakukan dengan jarak 1,5 meter untuk menjaga kualitas citra..
3. Pembahasan pada deteksi tepi dengan metode *Canny* menggunakan aplikasi program Matlab 2016a.

1.4 Tujuan Penelitian

Adapun tujuan dari dilakukannya penelitian dalam skripsi ini adalah sebagai berikut :

1. Memperbaiki dan meningkatkan penampakan garis batas suatu daerah pada objek citra *Grayscale*.

2. Menerapkan metode *canny* dalam sebuah prosedur program Matlab untuk mendeteksi tepi citra digital.
3. Untuk memperoleh informasi yang lebih jelas mengenai data batas tepi dalam sebuah citra digital menggunakan metode *canny*.

1.5 Manfaat Penelitian

Dengan penulisan skripsi ini diharapkan akan memberikan banyak manfaat, diantaranya sebagai berikut :

1. Agar dapat mempermudah dalam mendeteksi tepi objek pada citra yang dianggap sebagai tepi dari suatu objek yang dideteksi menggunakan metode canny dengan menggunakan program matlab.
2. Meningkatkan kualitas visual atau menonjolkan beberapa aspek informasi yang terkandung dalam citra digital serta mendeteksi tepi citra dengan mengunakan metode *canny*.
3. Dapat digunakan sebagai bahan referensi dengan pengetahuan yang sudah ada dan terkait dalam pembuatan skripsi ini, juga diharapkan pula dapat digunakan sebagai contoh dan acuan bagi para peneliti berikutnya.

1.6 Metode Penelitian

1.6.1 Metode Pengumpulan Data

Dalam rangka pengamatan untuk memperoleh data-data dan informasi yang sangat dibutuhkan dalam penulisan laporan ini, maka penulis akan menggunakan teknik pengumpulan data sebagai berikut :

1. Studi Pustaka (*Library Research*) yaitu dengan membaca jurnal-jurnal mupun buku-buku yang berhubungan dengan laporan, yang diperoleh dari berbagai sumber dan referensi sebagai bahan pembanding.
2. Studi Lapangan (*Field Research*)

   Informasi ini diperoleh secara langsung pada obyek yang diteliti dengan menggunakan dua metode yaitu :

   1) Metode Observasi

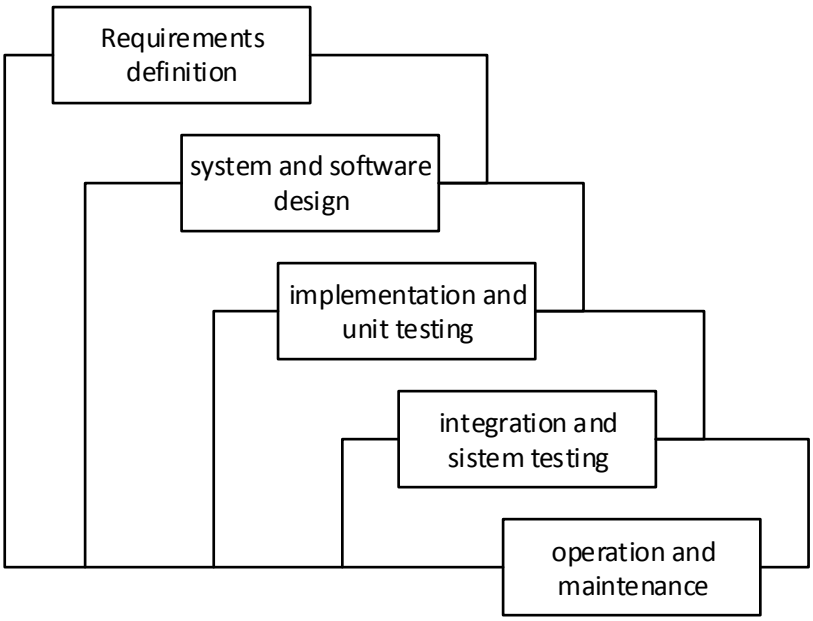
      Penulis melakukan pengamatan pada aplikasi yang serupa yang bersumber dari internet.

   2) Metode Wawancara (*Interview*)

      Penulis mengumpulkan informasi serta data dengan mengadakan tanya jawab langsung maupun tidak langsung pada programmer dan pihak-pihak yang lebih ahli tentang judul, agar penjelasan yang diterima lebih jelas dan dapat menanyakan langsung hal-hal yang kurang dimengerti mengenai judul yang dibahas.

1.6.2 Metode Pengembangan Sistem

Adapun metode yang penulis gunakan dalam mengembangkan perancangan sistem aplikasi ini adalah dengan pendekatan terstruktur seperti metode *waterfall.* Berikut adalah tahapan-tahapan yang terdapat dalam metode *waterfall* menurut Sukamto dan Shalahuddin (2013:28).



Gambar 1.1 Model *Waterfall*
Sumber : Sukamto dan Shalahuddin (2013:28)

1. Analisis Kebutuhan Perangkat Lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami seperti apa yang dibutuhkan oleh user. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu di dokumentasikan.

2. Desain

Desain perangkat lunak adalah langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak dan representasi antar muka. Tahap ini mentranslasi kebutuhan perangkat dari tahap analisis kebutuhan ke reprentasi desain agar dapat di implementasikan program pada tahap selanjutnya yang penting untuk di dokumentasikan.

3. Pembuatan Kode Program

Desain harus ditranslasikan kedalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

Pengujian fokus pada perangkat lunak dari segi logika dan fungsional. Hal ini dilakukan untuk meminimalisir kesalahan dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung dan Pemeliharaan

Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

1.7 Sistematika Penulisan

Tugas skripsi ini disusun dengan sistematika penulisan sebagai berikut :

BAB I : PENDAHULUAN

Membahas tentang latar belakang dilakukannya penelitian, rumusan masalah yang akan dibahas, maksud dan tujuan yang ingin dicapai, batasan masalah, dan sistematika dari penulisan.

BAB II : LANDASAN TEORI

Bab ini akan menjelaskan teori-teori yang berkaitan dengan segmentasi suatu objek, pengolahan citra antara lain pengertian *edge*

detection, pengertian citra serta membahas pengertian metode *canny* yang diangkat.

BAB III : ANALISA DAN PERANCANGAN

Berisi perancangan sistem, mulai dari proses yang terjadi di dalamnya pembuatan *usecase*, desain *interface*, hingga struktur hirarki dan menu program serta *uml* dan *activity diagram*.

BAB IV : ALGORITMA DAN IMPLEMENTASI

Memberikan penjelasan mengenai algoritma yang digunakan dalam pembahasan skripsi dan implementasi melalui program perangkat lunak yang sudah dibuat.

BAB V : PENUTUP

Berisi kesimpulan dari seluruh hasil penelitian skripsi ini serta saran-saran untuk pengembangan lebih lanjut.

BAB II

LANDASAN TEORI

2.1 Konsep Implementasi

Konsep implementasi semakin marak dibicarakan seiring dengan banyaknya pakar yang memberikan kontribusi pemikiran tentang definisi implementasi. Salah satunya adalah menurut Mulyadi (2015:12) menyatakan bahwa "Implementasi merupakan suatu kajian mengenai studi kebijakan yang mengarah pada proses pelaksanaan dari suatu kegiatan atau kebijakan yang telah di tetapkan". Sedangakan menurut Firendra mengutip dari Kamus Besar Bahasa Indonesia (KBBI) dari *web site* pribadinya menyatakan "implementasi merupakan pelaksanaan atau juga penerapan dan secara umum dan lebih luas di artikan sebagai sebuah tindakan yang dilakukan untuk melaksanakan sebuah rencana yang sudah dibuat atau disusun sebelumnya".

Dari beberapa pengertian diatas dapat ditarik kesimpulan bahwa implementasi merupakan hal yang sangat penting dalam keseluruhan rangkaian kegiatan ataupun aktivitas yang dilakukan dari kebijakan yang susun sebelumnya, meliputi kebutuhan apa yang dibutuhkan, siapa pelaksananya, kapan pelaksanaanya serta kapan akan diselesaikan. Rencana yang dibuat dengan sangat baik tidak akan berarti apa-apa jika tidak dilaksanakan atau dilaksanakan dengan asal-asalan.

2.2 Pengertian Citra Digital

Citra adalah suatu representasi, kemiripan, atau imitasi dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpan. (Sukatmi, 2017)

Citra digital merupakan citra yang dihasilkan dari pengolahan dengan menggunakan komputer, dengan cara merepresentasikan citra secara numerik dengan nilai-nilai diskret. Pada umumnya citra digital berbentuk empat persegi panjang, dan dimensi ukurannya dinyatakan dalam tinggi kali lebar atau panjang kali lebar. (Winarno, 2014)

Menurut Fajar Astuti Hermawati (2013:3) "Citra atau gambar dapat didefinisikan sebagai sebuah fungsi dua dimensi, f(x,y), dimana x dan y adalah koordinat bidang datar, dan harga fungsi f disetiap pasanga koordinat (x,y) disebut intensitas atau level keabuan (*grey level*) dari gambar dititik itu dan Citra digital merupakan larik dua dimensi atau suatu matriks yang elemen-elemennya menyatakan tingkat keabuan dari elemen gambar. Citra digital dapat berupa keluaran proses penyamaran atau proses fotografi digital yang mengandung informasi".

Citra dapat dinyatakan dalam titik-titik koordinat pada kawasan ruang (spasial) atau bidang dan untuk menentukan warna atau menyatakan nilai keabuan suatu citra. Salah satu contoh bentuk citra digital adalah citra monokrom atau citra

hitam putih yang merupakan citra satu kanal, dimana citra f(x,y) merupakan fungsi tingkat keabuan dari hitam ke putih, x menyatakan variabel baris dan y menyatakan variabel kolom. Indeks baris dan kolom (x,y) dari sebuah piksel dinyatakan dalam bilangan bulat (*integer*).

Sebuah piksel merupakan sampel dari pemandangan yang mengandung intensitas citra yang dinyatakan dalam bilangan bulat. Untuk menunjukkan lokasi suatu piksel, koordinat (0,0) digunakan untuk posisi kiri atas dalam bidang citra, dan koordinat (m-1,n-1) digunakan untuk posisi kanan bawah dalam citra berukuran m x n piksel dimana m adalah kolom dan n adalah baris. Untuk menunjukkan tingkat pencahayaan suatu piksel, seringkali digunakan bilangan bulat yang besarnya delapan bit dengan lebar selang nilai 0-255 dimana 0 untuk warna hitam, 255 untuk warna putih, dan tingkat abu-abu berada di antara nilai 0 dan 255.

Citra dapat dinyatakan sebagai fungsi kontinu dari intensitas cahaya dalam bidang dua dimensi, $f(x,y)$, x dan y menyatakan koordinat ruang dan nilai f pada suatu koordinat (x,y)menyatakan kecerahan dan informasi warna citra. Secara matematis persamaan A untuk fungsi intensitas, $f(x,y)$ adalah : $0 < f(x,y) < \infty$

Pada hakikatnya citra yang dilihat oleh mata terdiri atas berkas-berkas cahaya yang dipantulkan oleh benda-benda sekitar kita. Jadi fungsi intensitas, $f(x,y)$ merupakan fungsi sumber cahaya, $i(x,y)$ yang menerangi objek serta jumlah cahaya yang dipantulkan, $r(x,y)$ oleh objek. Dengan demikian $f(x,y)$ dapat dinyatakan dengan persamaan B: $f(x,y) = i(x,y)r(x,y)$

dengan: $0 < i(x,y) < \infty$ (iluminasi sumber cahaya

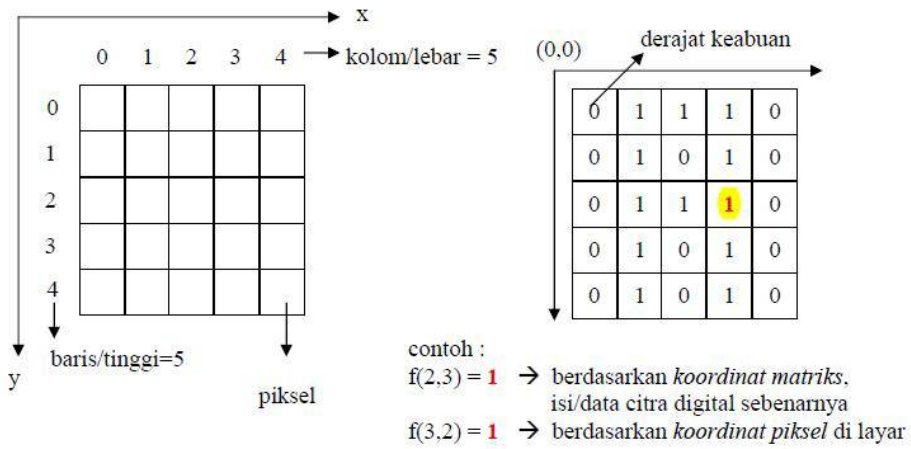
$0 < r(x, y) < 1$ (koefisien pantul cahaya)

Citra digital adalah sajian citra dalam bentuk diskret, baik pada koordinat ruang maupun nilai intensitas cahayanya. Dengan demikian citra digital dapat disajikan sebagai matriks berdimensi $\mathrm{M} \times \mathrm{N}$, dengan M menyatakan tinggi dan N menyatakan lebar dari citra. Aras keabuan (l) adalah nilai intensitas dari suatu piksel (x, y) yang terdapat dalam suatu citra monokrom f. Persamaan A dan B menunjukkan bahwa *l* terletak pada: $L\min \le l \le L\max$

Dengan: $L_{min} = i_{min}\ r_{min}$ atau $L_{max} = i_{max}\ r_{max}$

Selang $[L_{min}, L_{max}]$ disebut skala keabuan, yang secara praktis selang ini digeser ke selang $[0, L]$, dimana $l = 0$ menyajikan warna hitam dan $l = L$ menyajikan warna putih. Dengan demikian citra skala keabuan adalah format citra yang memiliki warna abu–abu bertingkat dari hitam menuju putih.
(http://id.wikipedia.org/wiki/Pengolahan_citra)

Gambar dibawah ini akan menunjukkan contoh posisi letak piksel dan bentuk matriks penyusunnya.



(a) (b)

Gambar 2.1 Posisi letak piksel dan bentuk Matriks penyusunannya

Sumber :http:// informatika.web.id/pengertian-citra-digital.html

2.3 Pengolahan Citra

Pengolahan citra merupakan proses untuk menghasilkan citra sesuai dengan keinginan atau kualitasnya menjadi lebih baik. Inputannya adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik dari pada citra masukan. Misal citra warnanya kurang tajam, kabur (*blurring*) dan mengandung *noise* (misal bintik-bintik putih) sehingga perlu ada pemrosesan untuk memperbaiki citra karena citra tersebut menjadi sulit di interpretasikan karena informasi yang disampaikan menjadi berkurang.

Menurut Abdul Kadir (2013:2), “pengolahan citra adalah istilah umum untuk berbagi teknik yang keberadaanya untuk memanipulasi dan memodifikasi citra dengan berbagai cara”.

Adapun contoh dari implementasi pengolahan citra digital seperti ditunjukkan pada Gambar berikut.

(a) (b)

Gambar 2.2 Citra yang kabur (a) , Citra yang diperbaiki (b)

Sumber : Abdul-Kadir, 2013,15

Umumnya, berbagai bentuk operasi-operasi pengolahan citra diterapkan pada citra apabila :

1. Perbaikan atau modifikasi citra untuk meningkatkan kualitas visual atau menonjolkan beberapa aspek informasi yang terkandung dalam citra.
2. Elemen di dalam citra perlu di kelompokkan, dicocokkan atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

2.4 Segmentasi Citra

Proses segmentasi citra merupakan proses yang membagi-bagi suatu citra menjadi daerah-daerah atau obyek-obyek yang dimilikinya. Menurut Abdul kadir (2013 : 336) “Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra kedalam beberapa daerah dengan setiap objek atau daerah memiliki kemiripan atribut”. Dalam konteks citra digital daerah hasil segmentasi tersebut merupakan

kelompok piksel yang bertetangga atau berhubungan. Segmentasi citra dapat dilakukan melalui beberapa pendekatan antara lain:

1. Pendekatan batas (*boundary approach*), pendekatan ini dilakukan untuk mendapatkan batas yang ada antar daerah.
2. Pendekatan tepi (*edge approach*), pendekatan tepi dilakukan untuk mengidentifikasi piksel tepi dan menghubungkan piksel-piksel tersebut menjadi suatu batas yang diinginkan
3. Pendekatan daerah (*region approach*), pendekatan daerah bertujuan untuk membagi citra dalam daerah-daerah sehingga didapatkan suatu daerah sesuai kriteria yang diinginkan.

Menurut Fajar Astuti Hermawati (2013:163) "segmentasi merupakan proses membagi citra ke dalam komponen-komponen region atau objek".

Proses segmentasi digunakan dalam berbagai penerapan, meskipun metode yang digunakan sangat bervariasi, semuanya memiliki tujuan yang sama mendapatkan representasi sederhana yang berguna dari suatu citra. Terdapat berbagai macam metode dalam melakukan segmentasi, cukup sulit untuk menentukan metode yang komprehensif, oleh karena itu pemilihan metode bergantung pada pendekatan yang akan digunakan dan fitur yang ingin diperoleh dari citra.

Berikut ini ada beberapa metode yang umum digunakan dalam segmentasi citra, yaitu :

1. *Thresholding*

Metode *thresholding* didasarkan pada pemisahan pixel ke dalam kelas yang berbeda tergantung pada tingkat keabuan masing-masing pixel. Intensitas citra medis seperti tumor dan jaringan pada otak biasanya sangat rumit dan memiliki tingkat keabuan yang sangat dekat sehingga menyebabkan kesulitan penentuan ambang batas (*threshold*). Metode thresholding tidak bisa diterapkan untuk citra dengan tingkat keabuan yang berdekatan sehingga biasanya di kombinasikan dengan metode lain.

2. *Region growing*

Metode *region growing* seperti menggabungan thresholding dengan kondisi konektivitas atau kriteria daerah homogenitas. Keberhasilan dari metode tersebut bergantung pada kepresisian informasi anatomi untuk meletakkan baik satu mapun beberapa pixel untuk masing-masing daerah homogen. Kelemahan lain dari metode *region growing* adalah metode tersebut hanya dapat bekerja dengan baik pada daerah yang homogen dan membutuhkan operator untuk menentukan daerah yang akan disegmentasi.

Metode *region growing* yang paling umum digunakan adalah *watershed*. Prinsip dasar dari watershed adalah merubah gradien tingkat keabuan citra menjadi permukaan topografi. Daerah minimum dari citra merupakan sumber dimana air meluap dan bentuk-bentuk "kolam" ("*catchment basin*") menggambarkan permukaan air. Algoritma ini akan berhenti bila dua "kolam" dari dua sumber yang berbeda bertemu. Jika pada citra terdapat

banyak pola dan noise, maka akan terbentuk banyak "kolam" sehingga terjadi segmentasi yang berlebihan.

Matei Mancas dan *Bernard Gosselin* memodifikasi algoritma *watershed* untuk menghidari masalah segmentasi berlebih pada citra tumor otak dengan mengunakan metode marker-based watershed dan gradien vector flow untuk komputasinya. Watershed pertama dihitung dari dua market set awal dan membagi citra menjadi 3 wilayah yaitu bagian luar, fuzzy dan bagian dalam. Algoritma yang sama kemudian digunakan untuk menghitung watershed yang kedua dan diperoleh 5 wilayah konsentris yang makin mendekati bagian tumor.

Salvador A Melo Júnior, et. al menyelesaikan masalah *over-segmentation* dengan menggunakan multi-state preprocessing untuk mengurangi noise citra, meningkatkan kontras, dan memodifikasi homotopi citra ventrikel kiri. Setelah tahap preprocessing, metode watershed berhasil melakukan segmentasi citra ventrikel kiri. Kontur akhir diperoleh dengan mengurangi ukuran wilayah tersegmentasi dengan algoritma kontur koreksi.

3. *Shapebased*

Metode shapebased juga memberikan pendekatan yang cukup sederhana dalam segmentasi citra namun sangat sulit dalam penentuan kontur awal sehingga ketidak tepatan dalam penentuan kontur awal dapat menyebabkan hasil segmentasi yang kurang memuaskan.

4. *Clustering*

Metode statistik atau clustering didasarkan pada distribusi parameter tertentu. Hal terpenting dalam metode ini adalah melakukan estimasi definisi awal dari parameter sehingga bagus tidaknya segmentasi tergantung pada seberapa baik distribusi yang diasumsikan mendekati distribusi dari data. Pada kenyataanya, secara umum citra medis mengandung noise dan ketidak pastian distribusi yang tidak dapat diketahui sebelumnya.

Metode segmentasi statistik mengklasifikasi dan melakukan pengelompokan piksel citra ke dalam wilayah terpadu menurutkriteria tertentu dengan menggunakan *pattern-classifier* tertentu dan teknik *post-processing* semisal *filter* morfologi. Algoritma K-*mean*, *fuzzy* c-*means* (FCM) dan *expectation–maximization* (EM) paling umum digunakan pada metode clustering.

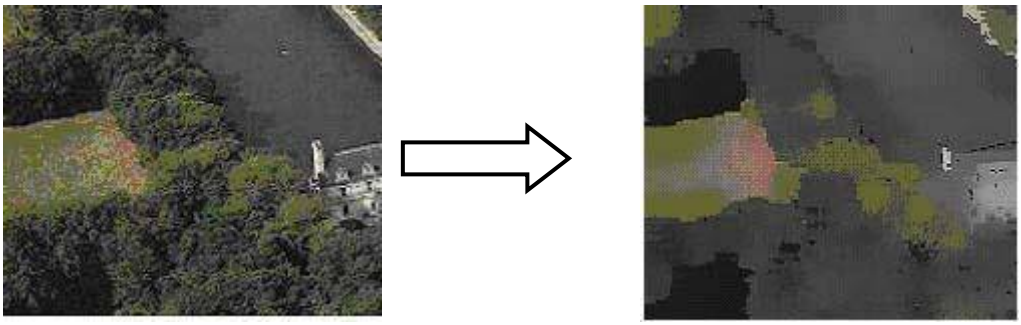
2.5 Segmentasi Citra Dengan Deteksi Warna

Segmentasi warna merupakan proses segmentasi dengan pendekatan daerah yangbekerja dengan menganalisis nilai warna dari tiap piksel pada citra dan membagi citra tersebut sesuai dengan fitur yang diinginkan. Segmentasi citra dengan deteksi warna menurut Abdul Kadir (2013:337) "Menggunakan dasar seleksi warna pada model warna dengan nilai toleransi tertentu".

Pada metode segmentasi dengan deteksi dilakukan pemililian sampel piksel sebagai acuan warna untuk membentuk segmen yang diinginkan. Citra digital

menggunakan model warna RGB sebagai standar acuan warna, oleh karena itu proses awal pada metode ini memerlukan konversi model warna RGB ke Grayscale. Untuk membentuk segmen sesuai dengan warna yang diinginkan maka ditentukan nilai toleransi pada setiap dimensi warna, kemudian nilai toleransi tersebut digunakan dalam perhitungan proses adaptive threshold. Hasil dari proses threshold tersebut akan membentuk segmen area dengan warna sesuai toleransi yang diinginkan.

Berikut contoh Gambar segmentasi mendekteksi warna dimana tiap piksel dalam suatu wilayah mempunyai kesamaan karakteristik atau propeti yang dapat dihitung (*computed property*), seperti : warna (*color*), intensitas (*intensity*), dan tekstur (*texture*).



Gambar 2.3 Segmentasi Deteksi Warna

Sumber :http:// *informatika.web.id/segmentasi-deteksi-warna.html*

Segmentasi wilayah merupakan pendekatan lanjutan dari deteksi tepi. Dalam deteksi tepi segmentasi citra dilakukan melalui identifikasi batas-batas objek (*boundaries of object*). Batas merupakan lokasi dimana terjadi perubahan intensitas. Dalam pendekatan didasarkan pada wilayah, maka identifikasi dilakukan melalui wilayah yang terdapat dalam objek tersebut.

Segmentasi memegang peranan yang sangat penting dalam pengolahan citra. Berbagai algoritma telah dikembangkan untuk melakukan segmentasi citra medis dengan kelebihan dan kekurangan masing-masing. Karena tidak ada solusi umum dalam penyelesaian segmentasi citra, metode-metode tersebut sering kali harus dikombinasikan satu dengan yang lainnya agar dapat memecahkan masalah segmentasi citra secara efektif.

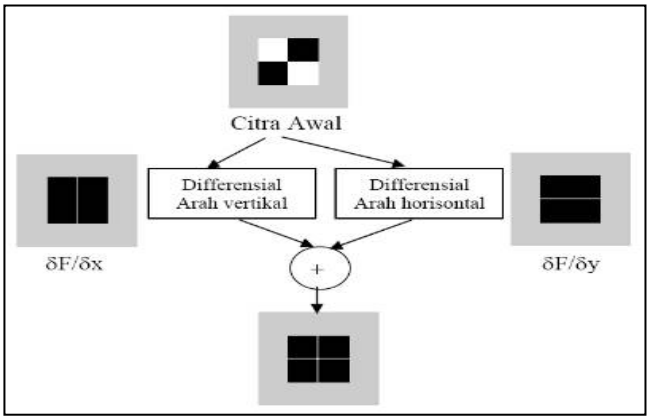
2.6 *Edge Detection*

Edge detection adalah pendekatan yang paling umum digunakan untuk mendeteksi diskontinuitas graylevel. Hal ini disebabkan karena titik ataupun garis yang terisolasi tidak terlalu sering dijumpai dalam aplikasi praktis. Suatu edge adalah batas antara dua region yang memiliki *graylevel* yang relatif berbeda. Pada dasarnya ide yang ada di balik sebagian besar teknik edge-detection adalah menggunakan perhitungan *local derivative operator*. Gradien dari suatu citra f(x,y) pada lokasi (x,y) adalah vector.

Menurut Abdul Kadir (2013:342) “deteksi tepi menggunakan dua macam detektor yaitu detektor baris (Hy) dan kolom (Hx). Tepi (edge) adalah perubahan nilai intensitas derajat keabuan yang cepat/tiba-tiba (besar) dalam jarak yang singkat. Sedangkan deteksi tepi (*edge detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah untuk menandai bagian yang menjadi detail citra, untuk memperbaiki detail dari citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra.

Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya".

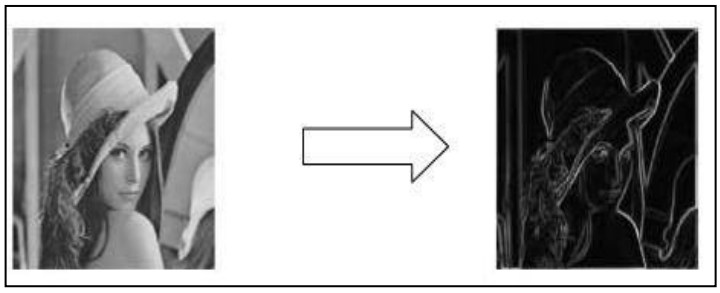
Pada Gambar di bawah ini dapat dilihat proses yang dilakukan untuk memperoleh tepi gambar dari suatu citra yang ada.



Gambar 2.4 Proses Deteksi Tepi Citra

Sumber :http:// informatika.web.id/pengertian-citra-digital.html

Pada Gambar 6 terlihat bahwa hasil deteksi tepi berupa tepi-tepi dari suatu gambar. Bila diperhatikan bahwa tepi suatu gambar terletak pada titik-titik yang memiliki perbedaan tinggi.



Gambar 6. Hasil Deteksi Tepi

Sumber : Abdul_Kadir, 2013, 343

Proses deteksi tepi (*edge detection*) sendiri masih dapat dikelompokkan berdasarkan operator atau metode yang di gunakan dalam proses pendeteksian tepi suatu citra untuk memperoleh citra hasil.

Edge Linking Secara ideal, teknik yang digunakan untuk mendeteksi diskontinuitas seharusnya hanya menghasilkan pixel-pixel yang berada pada batas region. Namun dalam prakteknya hal ini jarang terjadi karena adanya noise, batas yang terpisah karena pencahayaan yang tidak merata, dan efek lain yang mengakibatkan variasi intensitas. Untuk itu algoritma *edge-detection* biasanya dilanjutkan dengan prosedur edge-linking untuk merangkai pixel-pixel tersebut menjadi satuk esatuan sehingga memberikan suatu informasi yang berarti. Salah satu teknik yang dapat digunakan untuk *edge-linking* adalah *local processing*, yaitu dengan menganalisa karakteristik pixel-pixel di dalam suatu *neighborhood* pada semua titik (x,y) di dalam citra yang telah mengalami *edge-detection*. Selanjutnya semua titik yang sejenis dihubungkan sehingga membentuk kumpulan pixel yang memiliki sifat-sifat yang sama.

Dua sifat utama yang digunakan untuk menentukan kesamaan edge pixel dalam analisa inia dalah:

1. Besarnya respon *gradient* operator yang digunakan.
2. Arah *gradient*

Sifat yang pertama dinyatakan dengan nilai $\tilde{N}f$ yang telah dibahas sebelumnya. Jadi suatu *edge* pixel dengan koordinat (x',y') dan bertetangga dengan (x,y), dikatakan memiliki *magnitude* sama dengan pixel di (x,y) jika:

$$|\nabla f(x,y) - \nabla f(x',y')| \leq T$$

dimana T adalah *threshold* positif. Sedangkan arah vektor gradient dinyatakan dengan a(x,y) yang juga telah dibahas sebelumnya. Suatu *edge* pixel dengan koordinat (x',y') dan bertetangga dengan (x,y), dikatakan memiliki sudut yang sama dengan pixel di (x,y) jika:

$$|\mathrm{a}(x,y) - \mathrm{a}(x',y')| \leq A$$

dimana A adalah *threshold* sudut. Suatu titik yang menjadi tetangga dari (x,y) dihubungkan dengan titik (x,y) jika memenuhi kedua kriteria di atas, baik *magnitude* maupun sudutnya. Proses *linking* ini diulang untuk seluruh lokasi titik yang ada di dalam citra.

2.7 Operasi Pixel (*Picture Element*)

Agar citra dapat diolah oleh sebuah komputer digital, maka sebuah citra harus disimpan pada format yang dapat diolah oleh sebuah program komputer. Cara yang paling praktis yang dapat dilakukan adalah dengan membagai citra menjadi sekumpulan sel-sel diskret, yang disebut piksel.

Citra digital tersusun dari pixel-pixel. Pixel disebut juga dengan dot. Pixel berbentuk bujur sangkar dengan ukuran relatif kecil yang merupakan penyusun/pembentuk gambar bitmap. Banyaknya pixel tiap satuan luas tergantung pada resolusi yang digunakan. Keanekaragaman warna pixel tergantung pada bit depth yang dipakai. Semakin banyak jumlah pixel tiap satu satuan luas, semakin baik kualitas gambar yang dihasilkan dan tentu akan semakin besar ukuran filenya.

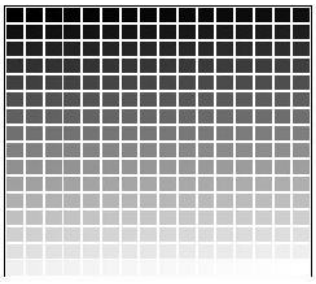
Menurut Abdul kadir (2013:36), "Operasi piksel adalah pengolahan citra yang memetahkan hubungan setiap pixsel yang bergantung pada pixsel itu sendiri". Pada umumnya sebuah citra dibagi menjadi kisi-kisi persegi, sehingga piksel sendiri adalah sebuah kisi-kisi persegi yang kecil. Selanjutnya setiap piksel diberi

nilai yang menyatakan warna atau menyatakan tingkat kecerahan piksel yang bersangkutan, yang sering disebut dengan intensitas piksel.

2.8 Citra Keabuan

Menurut Marvin ch. Wijaya (2007:62) "Citra keabuan adalah citra yang hanya menggunakan warna yang merupakan tingkatan warna abu-abu. Warna abu-abu adalah satu-satunya warna pada ruang RGB dengan komponen merah, hijau, dan biru mempunyai intensitas yang sama. Pada citra beraras keabuan hanya perlu menyatakan nilai intensitas untuk tiap piksel sebagai nilai tunggal, sedangkan pada citra berwarna perlu tiga nilai intensitas untuk tiap pikselnya".

Intensitas citra beraras keabuan disimpan sebagai integer 8 bit sehingga memberikan 28 = 256 tingkat keabuan dari warna hitam sampai warna putih. Dengan menggunakan pola 8-bit ini citra beraras keabuan membutuhkan ruang memori dan waktu pengolahan yang lebih sedikit dari pada citra berwarna (RGB). Pada Gambar berikut diperlihatkan visualisasi 256 aras keabuan.



Gambar 2.5 Visualisasi 256 Aras Keabuan

Sumber : Marvin ch. Wijaya, 2007, 62

2.9 Format File

Format file menentukan bagaimana informasi data dipresentasikan dalam suatu file. Informasi tersebut meliputi ada tidaknya kompresi, program aplikasi (feature) yang di support, penggunaan enkripsi dan lain-lain. Tiap format file memiliki kelebihan dan kelemahan pada masing-masing format tersebut. Dalam sistem operasi Windows biasanya format file dapat dibedakan dari namanya yaitu diakhiri titik dan diikuti dengan tiga atau empat huruf terakhir (misal .txt, .doc, .html dan lain-lain). (http// id. Wikipedia.org/wiki/citra).

1. Bitmap / bmp adalah standar file bitmap/raster pada sistem operasiberbasis Windows. Biasanya mempunyai ukuran file yang relatif besar.
2. GIF (*Graphics Interchange Format*) menggunakan maksimal 8 bit warna (28 = 256 warna) pada gambar dan melakukan kompresi dengan LZW *compression* yang merupakan kompresi *loseless*. Kompresi *Loseless* ini berarti tidak ada data yang dibuang. Sifatnya dapat menjadi lossy compression jika ada informasi warna pada gambar yang hilang. Format GIF mendukung gambar transparansi dan animasi.
3. JPEG / JPG (*Joint Photographic Experts Group*) menggunakan 24 bitwarna (224 = 16 juta warna) dan melakukan kompresi dengan caramembuang data pada gambar (bersifat *lossy compression*). Semakin kecil file yang diinginkan semakin kecil data yang akan dibuang sehingga kualitasnya akan semakin menurun. Format JPEG tidak mendukung transparansi dan animasi.

4. PNG-8 menggunakan 8 bit warna, kurang kompatibel (tidak didukung)oleh browser, tetapi biasanya mempunyai hasil kompresi yang lebih kecil dari format GIF. Berbeda dengan GIF yang telah di patenkan. Format PNG besifat bebas paten.
5. PNG-24 menggunakan 24 bit warna, hampir sama dengan PNG-8 tetapi mempunyai ukuran yang lebih besar dan warna yang lebih banyak.

2.10 *Bit Depth*

Bit Depth (Kedalaman Warna) yang sering disebut juga dengan pixel depth atau *color depth*. Bit Depth menentukan berapa banyak informasi warna yang tersedia untuk ditampilkan/dicetak dalam setiap pixel. Semakin besar nilainya semakin bagus kualitas gambar yang dihasilkan. Tentu ukurannya juga semakin besar. Misalkan suatu gambar mempunyai *bit depth* = 1. Ini berarti hanya ada 2 kemungkinan warna (21=2) yang ada pada gambar tersebut yaitu hitam dan putih. *Bit depth* = 24 berarti mempunyai kemungkinan warna 224=16 juta warna. (http// id. Pegolahan citra.org/wiki/citra).

2.11 Resolusi

Resolusi adalah jumlah pixel per satuan luas yang ada suatu gambar. Satuan pixel yang sering dipakai adalah dpi (*dot per inch*) atau ppi (*pixel per inch*). Satuan DPI menentukan jumlah pixel yang ada setiap satu satuan luas. Yang dalam hal ini adalah satu inch kuadrat. Resolusi sangat berpengaruh pada detil dan perhitungan gambarnya. Untuk memahami pentingnya resolusi coba perhatikan contoh berikut.

Jika suatu gambar dengan luas 1inch kuadrat dan jumlah dot adalah70 x 70 (yang berarti mempunyai resolusi 4900 dpi) diperbesar menjadi 10 inch maka jumlah pixel tetap 4900 dpi. Tetapi resolusinya berubah menjadi 4900:10 = 490 dpi. Hal ini mengakibatkan gambar menjadi kabur dan kasar. (http// id.Wikipedia.org/wiki/citra).

2.12 *Paint*

Paint adalah salah satu program (*software*) pengolah gambar, program paint merupakan salah satu program bawaan dari *micorosft windows*. Jadi program windows selalu ada program pengolah gambar yaitu paint. Setiap windows tampilan program paintnya berbeda, tetapi pada prinsipnya cara penggunaannya adalah sama. *Paint* merupakan program graphics *painting* sederhana yang terintegrasi dengan hampir seluruh versi *Microsoft Windows*, sejak perilisan pertamanya. Sering dirujuk sebagai MS Paint atau Microsoft Paint. Program ini dapat membuka dan menyimpan gambar dalam berbagai format, yaitu BMP, JPEG, GIF, PNG, dan TIFF. (http://www.tanyapedia.com/apa-itu-paint/)

2.13 Metode *Canny*

Operator *Canny* yang dikemukakan oleh Jhon *Canny* pada tahun 1986, terkenal sebagai operator deteksi tepi yang optimal, Abdul Kadir (2013:361). Salah satu algoritma deteksi tepi modern adalah deteksi tepi dengan menggunakan metode *Canny*.

Ada beberapa kriteria pendeteksi tepian paling optimum yang dapat dipenuhi oleh algoritma *Canny*:

1. Mendeteksi dengan baik (kriteria deteksi)

   Kemampuan untuk meletakkan dan menandai semua tepi yang ada sesuai dengan pemilihan parameter-parameter konvolusi yang dilakukan. Sekaligus juga memberikan fleksibilitas yang sangat tinggi dalam hal menentukan tingkat deteksi ketebalan tepi sesuai yang diinginkan.

2. Melokalisasi dengan baik (kriteria lokalisasi)

   Dengan *Canny* di mungkinkan dihasilkan jarak yang minimum antara tepi yang di deteksi dengan tepi yang asli.

3. Respon yang jelas (kriteria respon)

   Hanya ada satu respon untuk tiap tepi. Sehingga mudah dideteksi dan tidak menimbulkan kerancuan pada pengolahan citra selanjutnya.

Pemilihan parameter deteksi tepi *Canny* sangat mempengaruhi hasil dari tepian yang dihasilkan. Beberapa parameter tersebut antara lain :

1. Nilai *Standart Deviasi Gaussian*
2. Nilai Ambang

Pendekatan algoritma *Canny* dilakukan dengan konvolusi fungsi gambar dengan operator Gaussian dan turunan-turunannya. Turunan pertama dari fungsi citra yang dikonvolusikan dengan fungsi gaussian,

$$g(x,y) = D[gauss(x,y) * f(x,y)]$$

Ekivalen dengan fungsi citra yang dikonvolusikan dengan turunan pertama dari fungsi gaussian,

$$g(x,y) = D[gauss(x,y)] * f(x,y)$$

Algoritma ini dikenal sebagai algoritma yang baik untuk mendeteksi batas. Deteksi tepi ini pertama-tama menghaluskan gambar dengan menyingkirkan *noise*. Kemudian ditemukannya gradient gambarnya untuk menyoroti daerah yang mempunyai ruang kepalsuan yang tinggi. Algoritma ini lalu mengambil jalur sepanjang ruang tersebut dan menekan pixel yang tidak maximum (penekanan *non-maximum*).

Dengan menggunakan metode *Canny* tepian yang dihasilkan lebih jelas, perbedaan tepian dengan *background* citra terlihat nyata. Tapi di dalam *Canny* sendiri, noise dibagian tertentu tidak bisa hilang begitu saja. Pengujian lain terhadap deteksi tepi adalah dengan menggunakan ketahanannya terhadap gangguan (*noise*). Gangguan pada citra masukan (*input*) dapat dijadikan sebagai salah satu parameter yang menentukan tingkat tampilan dari beberapa metode untuk melacak tepian suatu objek. Nilai *grayscale* level pada suatu tepian objek akan berubah sehingga akan semakin sulit bagi operator deteksi untuk menentukan batas tepian suatu objek.

Berikut ini akan disertakan juga langkah-langkah cara mendeteksi tepi batas/tepi menggunakan *Canny*, yaitu :

a. Langkah pertama yang harus dilakukan adalah menyaring dan membuang noise dengan merubah objek ke citra *grayscale* pada gambar asli sebelum mencoba untuk menetapkan dan mendeteksi tepian. Untuk menghilangkan noise kita dapat memakai *Gaussian filter*, tapi ini juga digunakan secara eksklusif dalam algoritma *Canny*.

$$f_o(X,Y) = \frac{f_i^R(X,Y) + f_i^G(X,Y) + f_i^B(X,Y)}{3}$$

b. Setelah menghaluskan gambar dan menyingkirkan noise, langkah selanjutnya adalah menemukan tepi dengan menggunakan gradient dari gambar tersebut. Operator *Canny* membuat sebuah ukuran gradient ruang pada gambar tersebut. Lalu, besarnya gradient pada setiap poin dapat dicari. Operator *Canny* menggunakan sepasang matrik 3×3 konvolusi, satu matrik untuk mengkalkulasi gradient di x (kolom) dan yang satu untuk mengkalkulasi gradient di y (baris).

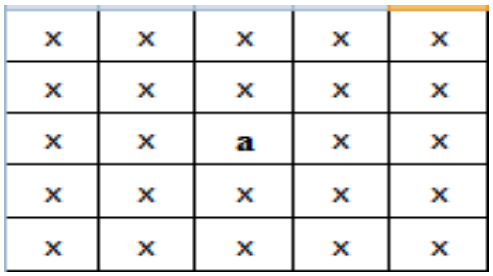
	-1	0	1		1	2	1
Gx=	-2	0	2	Gy=	0	0	0
	-1	0	1		-1	-2	-1

Gambar 2.6 Matrix (x,y) Operator Canny

http://masters.donntu.org/2010/fknt/chudovskaja/library/article5.htm

Rumus untuk menghitung gradient : |G| = |Gx| + |Gy|

c. Setelah arah tepian ditemukan, langkah selanjutnya adalah merelasikan arah tepiannya ke arah yang dapat ditrace pada sebuah gambar. Jika pada sebuah gambar mempunyai pixel 5×5 :



Gambar 2.7 Matrix 5x5

Sumber : http:// en.wikipedia.org/wiki/*Canny*_operator

d. Pada proses akhir akan menghasilkan sebuah gradient dari konvulasi yang mengetahui garis tepi. Sehingga setiap pixel yang ada terhubung satu sama lain. Oleh karena itu, memungkinkan untuk mengkombinasikan tingkat kehalusan dan pendektesian tepi kedalam suatu konvolusi dalam satu dimensi dengan dua arah yang berbeda (vertikal dan horizontal).

2.14 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, artifact tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelanbisnis dan sistem non perangkat lunak lainnya.

Menurut Rosa A.S dan M. Shalahudin (2014) "UML merupakan gabungan dari metode *Booch*, *Rumbaugh* (OMT) dan *Jacobson*". Tetapi uml ini akan mencakup lebih luas dari pada OOA&D. Pada pertengahan pengembangan uml dilakukan standarisasi proses dengan MG *Management Group* dengan harapan uml akan menjadi bahasa standar pemodelan pada masa yang akan datang. uml disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat".

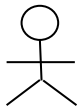
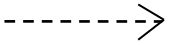
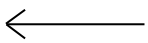
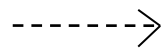
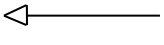
Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika anda ingin berdiskusi tentang desain denganseseorang, maka anda hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain. Untuk mendapatkan banyak pandangan terhadap sistem informasi yang akan dibangun, uml menyediakan beberapa diagram visual yang menunjukkan berbagai aspek dalam sistem. Ada beberapa diagram yang disediakan dalam UML antara lain :

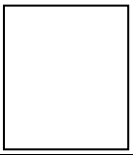
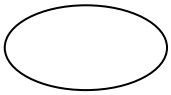
2.14.1 Diagram *use case* (*use case diagram*)

Use case Diagram digunakan untuk mengambarkan interaksi antara pengguna sistem (*actor*) dengan kasus (*use case*) yang disesuaikan dengan langkah-langkah (*scenario*) yang telah ditentukan. Sejak tahun 1992, dengan adanya pengembang uml, yaitu *Jacob Et All*, menjadikan *Use case* sebagai model utama atau yang dibutuhkan (*Requeirment Model*) pada UML Rosa dan M. Shalahudin (2014:155).

Use Case Diagram adalah suatu representasi / model yang digunakan pada rekayasa perangkat lunak yang menunjukkan sekumpulan *use case* dan aktor serta hubungan diantara keduanya. Diagram use case menyajikan interakasi antara *use case* dan aktor. Dimana, aktor dapat berupa orang, peralatan, atau sistem lain yang berinteraksi dengan sistem yang sedang dibangun. *Use case* menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai.

Tabel 2.1 Simbol-Simbol *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*
2.		*Dependecy*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*indepent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*)
3.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasanya objek induk (*ancestor*)
4.		*Include*	Menspesifikasikan bahwa *use case* sumber secara eksplisit.
5.		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

6.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.		*System*	Mespesifikasikan paket yang menampilkan sitem secara terbatas.
8.		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem secara yang menghsilkan suatu hasil yang terukur bagi suatu aktor
9.		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi
10.		*Collaboration*	Interaksi aturan-aturan dan elemen menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya(sinergi).

Sumber : Rosa A.S dan M. Shalahudin, 2014:156

2.14.2 Diagram aktivitas (*activity diagram*)

Rosa dan M. Shalahudin (2014:161), "Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu di perhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Berikut adalah simbol-simbol yang digunakan dalam diagram aktivitas :

Tabel 2.2 Simbol-Simbol *Activity Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1.		*Acitify*	Memperlihat bagaimana masing-masig kelas antara muka saling berinteraksi satu sama lain
2.		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3.		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4.		*Actifity Fial Node*	Bagaimana Objek dibentuk dan dihancurkan
5.		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.

Sumber : Rosa A.S dan M. Shalahudin, 2014:162

2.14.3 Tujuan Penggunaan UML

Adapun tujuan dari penggunaan *Unified Modelling Language* (UML) adalah sebagai berikut :

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

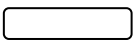
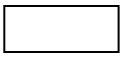
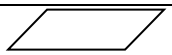
3. Memberikan model yang siap pakai, bahasa pemodelan visual yang eksperesif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah (*blue print*) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bias diketahui informasi secara detail tentang coding program atau bahkan membawa program dan menginterpretasikan kembali ke dalam bentuk diagram (*reserve enginering*).

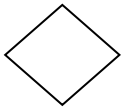
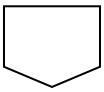
2.15 *Flowchart*

Flowchart adalah bagan alur yang menggambarkan langkah-langkah penyelesaian suatu masalah dengan simbol-simbol tertentu yang menggambarkan proses (intruksi) dengan proses lainnya di dalam program secara mendetail.

Adapun simbol dari *flowchart* yaitu :

Tabel 2.3 Simbol-simbol *flowchart*

Simbol	Nama	Fungsi
	Terminator	Permulaan atau akhir program
	Garis Alir/ Flow Line	Arah aliran program
	Proses	Proses perhitungan atau pengolahan data
	Input / Output Data	Proses input atau output data, informasi, parameter
	Predefined Proses	Permulaan sub program atau proses menjalankan sub program

	Decision	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk angkah
	One page Connector	Penghubung bagian-bagian flowchart yang ada dalam satu halaman
	Off Page Connector	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

(sumber : Supardi, 2013:58)

2.16 Bahasa Pemograman *Matlab*

Dikutip dari jurnal (Cahyono, 2013) yang berjudul "Penggunaan Software Matrix Laboratory (*Matlab*) Dalam Pembelajaran Aljabar Linier" menyatakan bahwa *MATLAB* (*Matrix Laboratory*) adalah suatu program untuk analisis dan komputasi numerik dan merupakan suatu bahasa pemrograman matematika lanjutan yang dibentuk dengan dasar pemikiran menggunakan sifat dan bentuk matriks. Pada awalnya, program ini merupakan interface untuk koleksi rutin-rutin numeric dari proyek *LINPACK* dan *EISPACK*, dan dikembangkan menggunkan bahasa *FORTRAN* namun sekarang merupakan produk komersial dari perusahaan *Mathworks*, *Inc*.yang dalam perkembangan selanjutnya dikembangkan menggunakan bahasa C++ dan *assembler* (utamanya untuk fungsi-fungsi dasar MATLAB).

MATLAB telah berkembang menjadi sebuah environment pemrograman yang canggih yang berisi fungsi-fungsi built-in untuk melakukan tugas pengolahan sinyal, aljabar linier, dan kalkulasi matematis lainnya. MATLAB juga berisi *toolbox* yang berisi fungsi-fungsi tambahan untuk aplikasi khusus . MATLAB

bersifat extensible, dalam arti bahwa seorang pengguna dapat menulis fungsi baru untuk ditambahkan pada *library* ketika fungsi-fungsi *built-in* yang tersedia tidak dapat melakukan tugas tertentu. Kemampuan pemrograman yang dibutuhkan tidak terlalu sulit bila Anda telah memiliki pengalaman dalam pemrograman bahasa lain seperti C++, PASCAL, atau FORTRAN.

MATLAB merupakan merk *software* yang dikembangkan oleh *Mathworks.Inc*.(lihat http://www.mathworks.com) merupakan *software* yang paling efisien untuk perhitungan numeric berbasis *matriks*. Dengan demikian jika di dalam perhitungan kita dapat menformulasikan masalah ke dalam format matriks maka *MATLAB* merupakan software terbaik untuk penyelesaian numericnya. *MATLAB* yang merupakan bahasa pemrograman tingkat tinggi berbasis pada *matriks* sering digunakan untuk teknik komputasi numerik, untuk menyelesaikan masalah-masalah yang melibatkan operasi matematika elemen, matrik, optimasi, proksimasi dan lain-lain. Sehingga Matlab banyak digunakan pada :

1. Matematika dan Komputansi,
2. Pengembangan dan Algoritma,
3. Pemrograman modeling,
4. simulasi,
5. Pembuatan prototipe,
6. Analisa Data,
7. Eksplorasi dan visualisasi,
8. Analisis numerik dan statistic, dan pengembangan aplikasi teknik.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem (*System analyst*) dapat di definisikan sebagai penguraian dari suatu sistem informasi yang utuh kedalam beberapa bagian komponen dengan maksud untuk mengidentifikasi permasalahan-permasalahan, yang terjadi dan kebutuhan yang di harapkan sehingga dapat diusulkan perbaikan dari sistem sebelumnya.

Pada bab ini akan dijelaskan mengenai analisis segmentasi citra digital dengan menggunakan metode *canny*. Aplikasi ini pada akhirnya akan menampilkan hasil deteksi tepi berupa garis tepi dari hasil operator *canny*. Tahapan yang digunakan untuk mendeteksi citra menggunakan beberapa tahapan yaitu tahap ekstraksi warna, tahap *croping*, tahap deteksi tepi menggunakan metode *canny*, tahap operasi dilatasi citra, tahap pengisian objek, tahap segmentasi citra dan tahap pembuatan garis tepi.

3.1.1 Analisis Masalah

Pada segmentasi citra terdiri dari beberapa langkah yang dapat digambarkan menjadi *block* diagram dengan model seperti dibawah ini :

Gambar 3.1 *Diagram Block System*

Adapun dari masing-masing bagian dalam diagram blok diatas adalah sebagai berikut :

1. Citra fotografi digunakan sebagai input sistem.
2. Melakukan *image proccesing* pada citra fotografi, proses awal dan proses segmentasi.
3. Citra fotografi yang telah melalui proses pegolahan citra pada sistem yang menghasilkan suatu citra fotografi yang telah tersegmentasi untuk selanjutnya dapat dilakukan proses yang lain sesuai dengan kebutuhan.

Berikut ini adalah algoritma dari setiap tahap penentuan pendeteksian segmentasi citra sebagai berikut :

1. Ekstraksi Warna

   Warna pokok dalam pengelolaan citra terdiri dari 3 (tiga) unsur, yaitu *Red* (R), *Green* (H), dan *Blue* (B). Jika warna-warna pokok tersebut akan digabungkan, maka dapat menghasilkan warna-warna yang berbeda/lain. Penggabungan tersebut bergantung pada warna pokok, yang tiap-tiap warna memiliki nilai 256 (8bit). Warna yang dideskripsikan dengan *RGB* adalah pemetaan yang mengacu pada panjang gelombang dari nilai *RGB*. Pemetaan menghasilkan nuansa warna untuk masing-masing R, G dan B. Masing-masing R, G dan B didiskritkan dalam skala 256, sehingga *RGB* akan memiliki indeks antara 0 sampai 255. Ekstraksi warna adalah mengambil fitur warna berupa R, G, dan B dari gambar.

2. Deteksi Tepi Metode *Canny*

Satu cara untuk menghindari gradien yang dihitung pada titik interpolasi dari piksel-piksel yang terlibat adalah dengan menggunakan jendela 3x3 untuk perhitungan gradien, sehingga perkiraan gradien berada tepat di tengah jendela. Operator *canny* adalah operator yang paling banyak digunakan sebagai pelacak tepi karena kesederhanaan dan keampuhannya.

3. Operasi Dilatasi Citra

Operasi dilatasi citra adalah memperjelas pada citra dengan menghilangkan batasan dari citra yang diperoleh dari fungsi *strel* (*structuring element*) dengan metode *line* untuk menentukan garis-garis yang terdapat dari sebuah citra sehingga mudah untuk dilakukan segmentasi, fungsi tersebut hanya terdapat pada matlab.

4. Pengisian Area Objek

Tahap selanjutnya adalah pemberian warna ke dalam garis-garis yang telah terhubung dengan warna putih dan garis yang tidak terhubung diabaikan. Proses pengisian pada area objek/gambar ini menggunakan fungsi *imfill* (*image fill*).

Fungsi *holes* adalah fungsi yang dapat mengartikan bahwa objek yang sudah dideteksi akan diberi semacam lubang putih yang mengisi area objek di dalamnya.

5. Segmentasi Citra

Segmentasi citra adalah proses pemisahan objek dari latarnya. Dimulai dengan membangun struktur dengan fungsi *strel* yang menggunakan bentuk disk dan objek. Nilai *disk* yang digunakan adalah 3. Fungsi *disk* pada *Matlab* bernilai ganjil dan yang terbaik, untuk aplikasi ini adalah 3 karena apabila nilainya berbeda maka hasil yang di dapat kurang maksimal. Proses penghalusan (*smoothing*) pada citra untuk mempertajam kualitas citra yang telah di segmentasi menggunakan fungsi *imrode*.

6. Garis Luar

Garis luar (*outline*) adalah proses pembuatan garis luar dari objek hasil segmentasi, sehingga garis luar tersebut dapat terlihat pada citra asli. Pembuatan garis luar ini menggunakan fungsi BW *outline* yang hanya terdapat pada aplikasi *matlab*.

3.1.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem berfungsi untuk mengidentifikasi data dan proses yang dibutuhkan pada pengembangan atau perancangan sistem yang akan di usulkan. Adapun tujuan dari analisis kebutuhan sistem ini harus sesuai dengan kebutuhan pada kebutuhan proses implementasi nantinya dengan tujuan dapat membantu dalam merancang dan membangun proses implementasi jadi lebih lancar, akurat, efisien dan efektif dalam melakukan kegiatan yang diperlukan nantinya.

1. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan suatu gambaran dari serangkaian informasi yang dipakai pada sistem yang akan dibangun yang meliputi :

1) Perangkat Keras (*Hardware*)

Kebutuhan perangkat keras dalam melakukan perancangan dan implementasi metode *canny* adalah sebagai berikut:

Tabel 3.1 Spesifikasi Perangkat Keras (*Hardware*)

Laptop		
No	Spesifikasi Hardware	
1	Monitor	1366 x 768 (64 bit) (60 Hz) 14"
2	*Processor*	*Intel® Core™ i5 CPU M 460 @ 2.53GHz (4 CPUs)*
3	RAM	6144 MB
4	VGA	*AMD Mobility Radeon HD 5000 Series*
5	Hardisk	640 GB

2) Perangkat Lunak (*Software*)

Kebutuhan perangkat lunak untuk melakukan perancangan dan implementasi metode *canny* adalah sebagai berikut:

Tabel 3.2 Spesifikasi Perangkat Lunak (*Software*)

Laptop		
No	Spesifikasi Software	
1	*Windows 10*	Sebagai *System* operasi
2	*Matlab R2016a*	Sebagai *Software* Komputasi, Visualisasi, dan Pemograman
3	*Paint*	Sebagai aplikasi pemotong citra gambar

2. Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional merupakan tahap dimana pembangunan sebuah perangkat lunak menganalisis sumber daya yang akan digunakan pada perangkat yang dibangunnya. Kebutuhan non-fungsional yang harus dimiliki oleh sistemnya adalah sebagai berikut :

1) Waktu pemrosesan data (*running time*) yang cepat sehingga efektif ketika digunakan oleh *user*.
2) Tampilan antarmuka (*user interface*) yang mudah dipahami oleh pengguna (*user friendly*).

3. Analisis Kebutuhan Informasi

Analisis kebutuhan informasi yang akan diterapkan dalam kebutuhan implementasi metode *canny* adalah sebagai berikut :

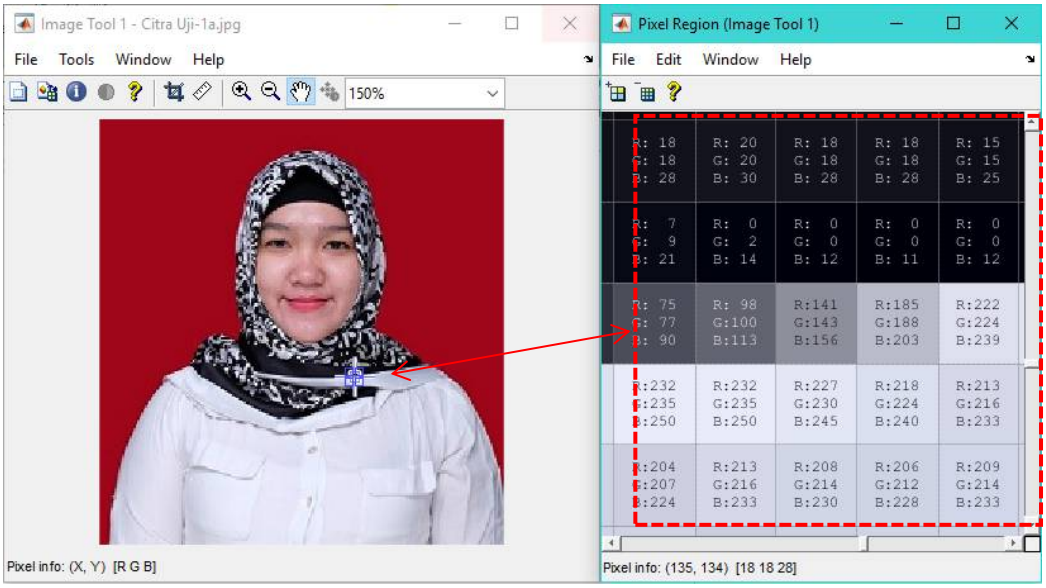
1) Kebutuhan masukan data citra gambar yang akan di tetili,
2) Kebutuhan proses dari citra yang telah dipilih, serta
3) Kebutuhan keluaran dari proses penggunaan metode *canny* dalam mendeteksi batasan tepi suatu citra.

3.2 Proses Segmentasi Citra

Segmentasi yaitu mereduksi citra menjadi objek atau *region*, misalnya memisahkan objek-objek yang berbeda dengan mengekstraksi batas-batas objek (*boundary*). Segmentasi merupakan suatu proses untuk mendapatkan area atau obyek yang diinginkan pada suatu citra dengan memisahkan antara area atau objek

dari latar belakangnya. Proses segmentasi memisahkan objek dari latar belakangnya pada tepi yang terhubung, jika tepi tidak terhubung maka objek tidak tersegmentasi. Citra hasil segmentasi berbentuk citra biner terbagi atas dua bagian, yaitu bagian hitam dan bagian putih, angka 0 menyatakan warna latar belakang (*background*) dan angka 1 menyatakan warna tinta atau objek atau dalam bentuk angka 0 untuk hitam dan angka 1 untuk warna putih. Proses penghalusan (*smoothing*) pada citra untuk mempertajam kualitas citra yang telah disegmentasi menggunakan fungsi imrode.

Berikut ini uji coba yang menunjukkan hasil segmentasi di konvulasikan dengan R,G,B sehingga menghasilkan nilai 0 sampai 255.



Gambar 3.2 Citra Awal dan Hasil Konvulasi Ke R,G,B.

Pada gambar diatas diketahui nilai konvulasinya. Tanda panah pada gambar menunjukaan nilai R,G,B. sebagai sampel pada gambar yang akan di proses ke citra *grayscale* dengan menggunakan metode *canny* untuk proses pendeteksi tepi.

3.3 Penerapan Metode *Canny*

Pada Citra warna bisa diubah menjadi citra *grayscale* dengan cara menghitung rata-rata elemen warna Red, Green, Blue. Secara matematis rumus perhitungannya adalah sebagai berikut.

$$f_o(X,Y) = \frac{f_i^R(X,Y) + f_i^G(X,Y) + f_i^B(X,Y)}{3} \dots\dots\dots\dots\dots \quad (1)$$

Pada gambar citra asli diatas dapan kita lihat nilai pixsel R,G,B seperti pada gambar 11 Dari hasil piksel R,G,B akan diubah menjadi pixsel *grayscale* Perhitungan fungsi negasi dilakukan seperti berikut.

Setiap titik yang terletak di posisi (X,Y), nilai-nilai komponen *Red, Green* dan *Blue* ditambahkan, kemudian hasilnya dibagi 3. Berikut ini contoh perhitungannya:

1. $f_o = \frac{18+18+28}{3} = 21$
2. $f_o = \frac{20+20+30}{3} = 23$
3. $f_o = \frac{18+18+28}{3} = 21$
4. $f_o = \frac{18+18+28}{3} = 21$
5. $f_o = \frac{15+15+25}{3} = 18$
6. $f_o = \frac{7+9+21}{3} = 6$
7. $f_o = \frac{0+2+14}{3} = 5$
8. $f_o = \frac{0+0+12}{3} = 4$
9. $f_o = \frac{0+0+11}{3} = 3$
10. $f_o = \frac{0+0+12}{3} = 4$
11. $f_o = \frac{75+77+90}{3} = 80$
14. $f_o = \frac{185+188+203}{3} = 192$
15. $f_o = \frac{222+224+239}{3} = 228$
16. $f_o = \frac{232+235+250}{3} = 239$
17. $f_o = \frac{232+235+250}{3} = 239$
18. $f_o = \frac{227+230+245}{3} = 234$
19. $f_o = \frac{218+224+240}{3} = 227$
20. $f_o = \frac{213+216+233}{3} = 220$
21. $f_o = \frac{204+207+224}{3} = 211$
22. $f_o = \frac{213+216+233}{3} = 220$
23. $f_o = \frac{208+214+230}{3} = 217$
24. $f_o = \frac{206+212+228}{3} = 215$

12. $f_o = \frac{98+100+113}{3} = 103$ 25. $f_o = \frac{209+214+233}{3} = 218$

13. $f_o = \frac{141+143+156}{3} = 146$

Maka hasil yang didapat dari pixel warna ke *grayscale* seperti pada tabel dibawah ini.

Tabel 3.3 Hasil Citra *Grayscale*

21	23	21	21	18
6	5	4	3	4
80	103	146	192	228
239	239	234	227	220
211	220	217	215	218

Matriks bobot 5x5 merupakan matriks simeteris yang akan dilakukan perkalian terhadap matriks A yaitu setiap pixel (posisi tengah) yang ada pada citra beserta 24 pixel tetangga yang mengelilingi. Matriks akan dinormalisasi dengan cara membagi setiap isi matriks dengan jumlah seluruh isi matriks.

Untuk mendeteksi tepi dengan metode *Canny*, kita akan menggunakan gradien G(x,y) yang merupakan sebuah vektor yang terdiri dari dua unsur yaitu Gx dan Gy. Deteksi tepi dilakukan dengan cara membaca setiap pixel pada citra dengan cara membaca dari pixel paling kiri atas (timur utara) dan bergerak ke pixel paling kanan bawah (barat selatan). Oleh karena itu, untuk membantu penelusuran tepi, gradien Gx dan Gy masing-masing dihitung dengan matriks operator *Canny* Mask 3x3. Operator ini mengambil prinsip dari fungsi *laplacian* dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari Operator *Canny* ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan

deteksi tepi. Nilai C konstanta bernilai dua, sehingga terbentuk matriks operator *Canny* seperti tabel di bawah ini:

Tabel 3.4 Matrix 3x3 Operator *Canny*

Sx=	-1	0	1
	-2	0	2
	-1	0	1

dan

Sy=	1	2	1
	0	0	0
	-1	-2	-1

Sehingga besar gradient dapat di hitung dengan menggunakan persamaan:

Sx = (p1 + cp2 + p3) + (p7 + cp6 + p5)

Sy = (p1 + cp8 + p7) + (p3 + cp4 + p5)

$|S| = |S_x| + |S_y|$ ..... (2)

Berikut adalah hasil perhitungan perkalian matrix penggunaan operator *Canny* untuk mengetahui segmentasi sebuah gambar citra dengan pixel 5x5. Konvolusi pertama dilakukan terhadap pixel yang bernilai 1 (titik pusat mask).

Tabel 3.5 Citra *Grayscale* Matrix 3x3

21	23	21	21	18
6	5	4	3	4
80	103	146	192	228
239	239	234	227	220
211	220	217	215	218

a. Nilai pada citra hasil konvolusi pertama didapatkan dengan perhitungan :

Sx = (21)(-1) + (6)(-2) + (80)(-1) + (21)(1) +(4)(2) + (146)(1) = 62

Sy = (21)(1) + (23)(2) + (21)(1) + (80)(-1) +(103)(-2) + (146)(-1) = -344 = 0

b. Konvolusi kedua dilakukan terhadap pixel yang bernilai 4 (titik pusat mask)

Sx = (23)(-1) + (5)(-2) + (103)(-1) + (21)(1) +(3)(2) + (192)(1) = 83
Sy = (23)(1) + (21)(2) + (21)(1) + (103)(-1) +(146)(-2) + (192)(-1) = -501 = 0

c. Konvolusi ketiga dilakukan terhadap pixel yang bernilai 3 (titik pusat mask)

Sx = (21)(-1) + (4)(-2) + (146)(-1) + (18)(1) +(4)(2) + (228)(1) = 79
Sy = (21)(1) + (21)(2) + (18)(1) + (146)(-1) +(192)(-2) + (228)(-1) = -677 = 0

d. Konvolusi keempat dilakukan terhadap pixel yang bernilai 103 (titik pusat mask)

Sx = (6)(-1) + (80)(-2) + (239)(-1) + (4)(1) +(146)(2) + (234)(1) = -21 = 0
Sy = (6)(1) + (5)(2) + (4)(1) + (239)(-1) +(239)(-2) + (234)(-1) = -692 = 0

e. Konvolusi kelima dilakukan terhadap pixel yang bernilai 146 (titik pusatmask)

Sx = (5)(-1) + (103)(-2) + (239)(-1) + (3)(1) +(192)(2) + (227)(1) = -28 = 0
Sy = (5)(1) + (4)(2) + (3)(1) + (239)(-1) +(234)(-2) + (227)(-1) = -684=0

f. Konvolusi keenam dilakukan terhadap pixel yang bernilai 192 (titik pusat mask)

Sx = (4)(-1) + (146)(-2) + (234)(-1) + (4)(1) +(228)(2) + (220)(1) = 150
Sy = (4)(1) + (3)(2) + (4)(1) + (234)(-1) +(227)(-2) + (220)(-1) = 667 = 0

g. Konvolusi ketujuh dilakukan terhadap pixel yang bernilai 239 (titik pusat mask)

Sx = (80)(-1) + (239)(-2) + (211)(-1) + (146)(1) +(234)(2) + (217)(1) = -172 =0
Sy = (80)(1) + (103)(2) + (146)(1) + (211)(-1) +(220)(-2) + (217)(-1) = -216 =0

h. Konvolusi kedelapan dilakukan terhadap pixel yang bernilai 234 (titik pusat mask)

Sx = (103)(-1) + (239)(-2) + (220)(-1) + (192)(1) +(227)(2) + (215)(1) = 60

Sy = (103)(1) + (146)(2) + (192)(1) + (220)(-1) +(217)(-2) + (215)(-1)=-282 =0

i. Konvolusi kesembilan dilakukan terhadap pixel yang bernilai 227 (titik pusat mask)

Sx = (146)(-1) + (234)(-2) + (217)(-1) + (228)(1) + (220)(2) + (218)(1) = 55
Sy = (146)(1) + (192)(2) + (228)(1) + (217)(-1) + (215)(-2) + (218)(-1) =107=0

Tabel 3.6 Hasil Konvulasi Tepi Citra Matrix 3x3

*	*	*	*	*
*	62	83	79	*
*	*	*	150	*
*	*	60	55	*
*	*	*	*	*

Maka hasil deteksi tepi yang didapat dari perhitungan matrix untuk mengetahui segmentasi pada gambar menggunakan metode *canny*, hasil yang didapat pada perkalian matrix 3x3 dengan citra *grayscale* matrix 5x5 pada tabel diatas.

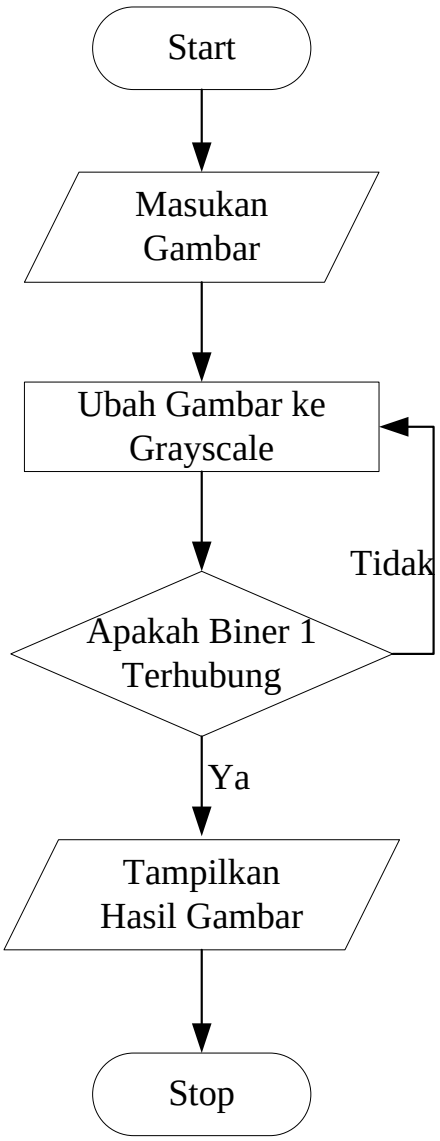
3.4 Perancangan Sistem

Perancangan sistem bertujuan untuk memberikan gambaran secara umum kepada pemakai dalam pembuatan rancangan dari pengembangan sistem untuk mempermudah dalam pengolahan data. Sehingga nantinya diharapkan aplikasi yang dibuat lebih baik dan dapat membantu dalam proses Analisa citra.

3.4.1 *Flowchart*

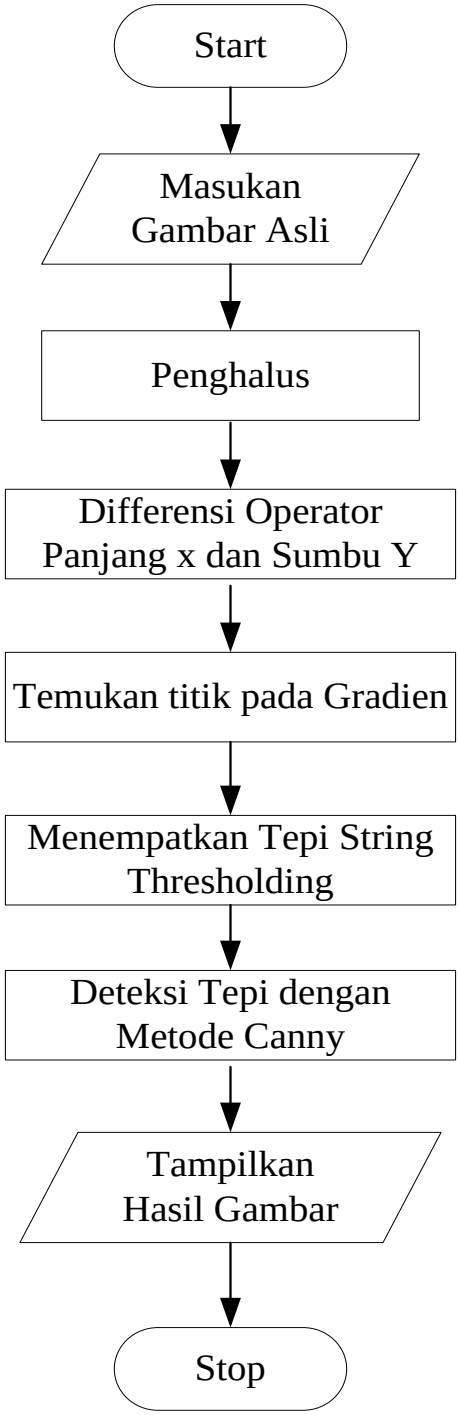
Flowchart adalah bagan atau *chart* yang menunjukan prosedur kerja sistem secara logika.

1. *Flowchart* Segmentasi Citra



Gambar 3.3 *Flowchart* Segmentasi Citra

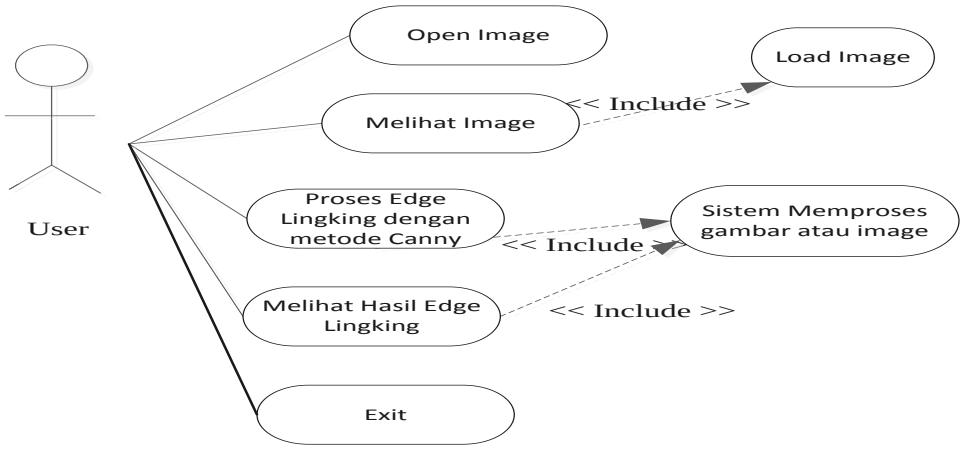
2. *Flowchart* Deteksi Tepi



Gambar 3.4 *Flowchart* Deteksi Tepi

3.4.2 *Usecase Diagram*

Diagram *use case* menyajikan interakasi antara *use case* dan aktor. Dimana, aktor dapat berupa orang, peralatan, atau sistem lain yang berinteraksi dengan sistem yang sedang dibangun. *Use case* menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai.

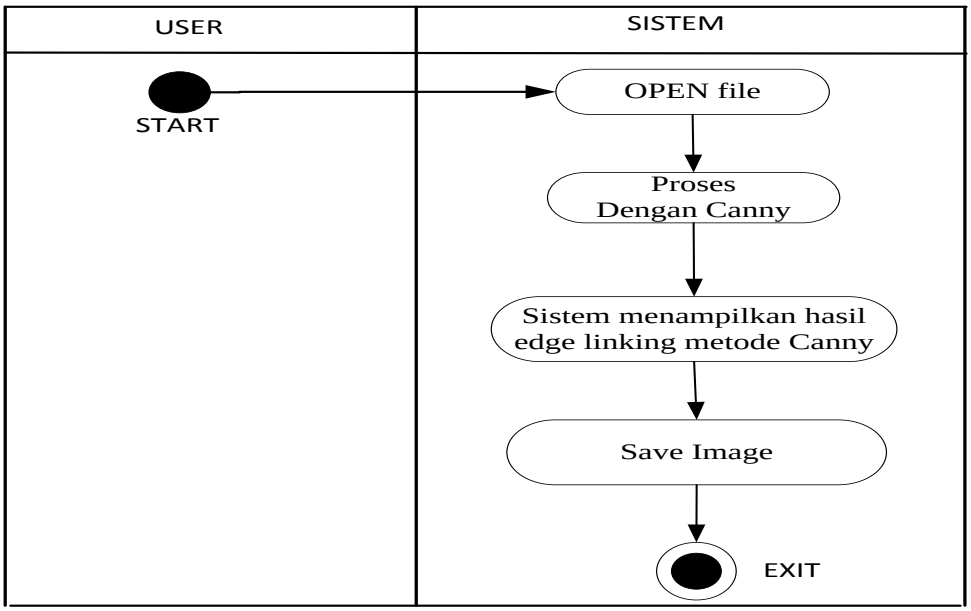


Gambar 3.5 Perancangan *Usecase Diagram*

Perancangan perangkat lunak dilakukan dengan orientasi terhadap objek menggunakan UML. Dari diagram *use case* dibuat diagram *activity*, yang menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.

3.4.3 *Activity Diagram*

Diagram aktivitas atau *activity* diagram menggambarkan aliran fungsionalitas sistem. Pada tahap pemodelan bisnis, diagram aktivitas dapat digunakan untuk menujukkan aliran kerja bisnis *(business work flow*). Dapat juga digunakan untuk menggambarkan aliran kejadian (*flow of event*) dalam *use case*.



Gambar 3.6 Perancangan *Activity Diagram*

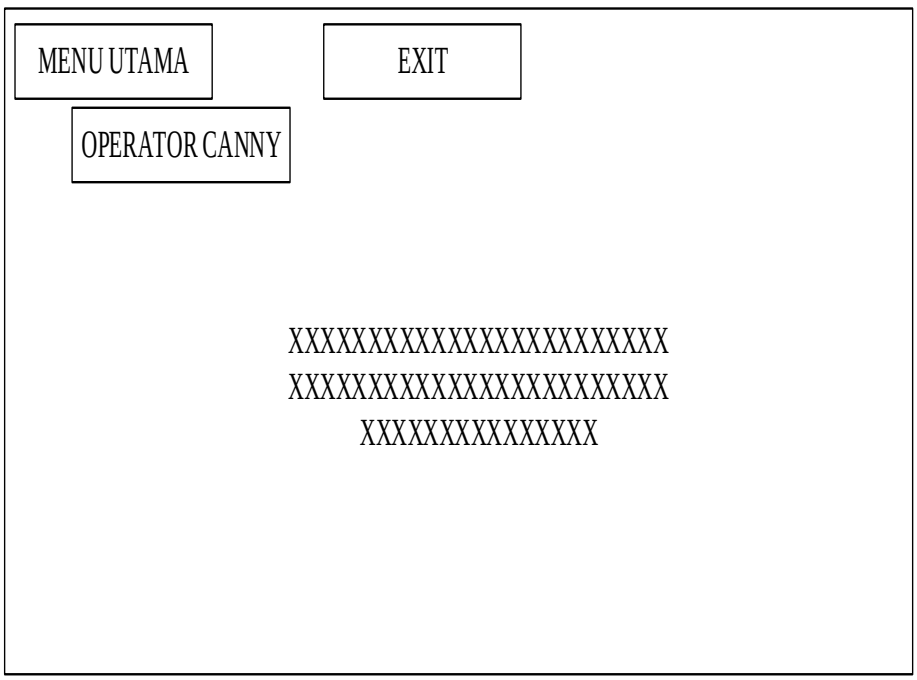
3.5 Perancangan Antarmuka (*interface*)

Perancangan antarmuka berfungsi untuk memberikan gambaran pembuatan tampilan untuk mempermudah komunikasi antara sistem dengan *user*. Hal yang dibutuhkan dalam pembuatan perancangan antar muka yang menarik dan tidak membosankan serta mudah dimengerti oleh *user*.

Antar muka perangkat lunak pegolahan citra digital mengunakan metode *canny* dalam mengimplementasikan segmentasi metode *canny* ke dalam bentuk program diperlukan rancangan *interface* seperti berikut :

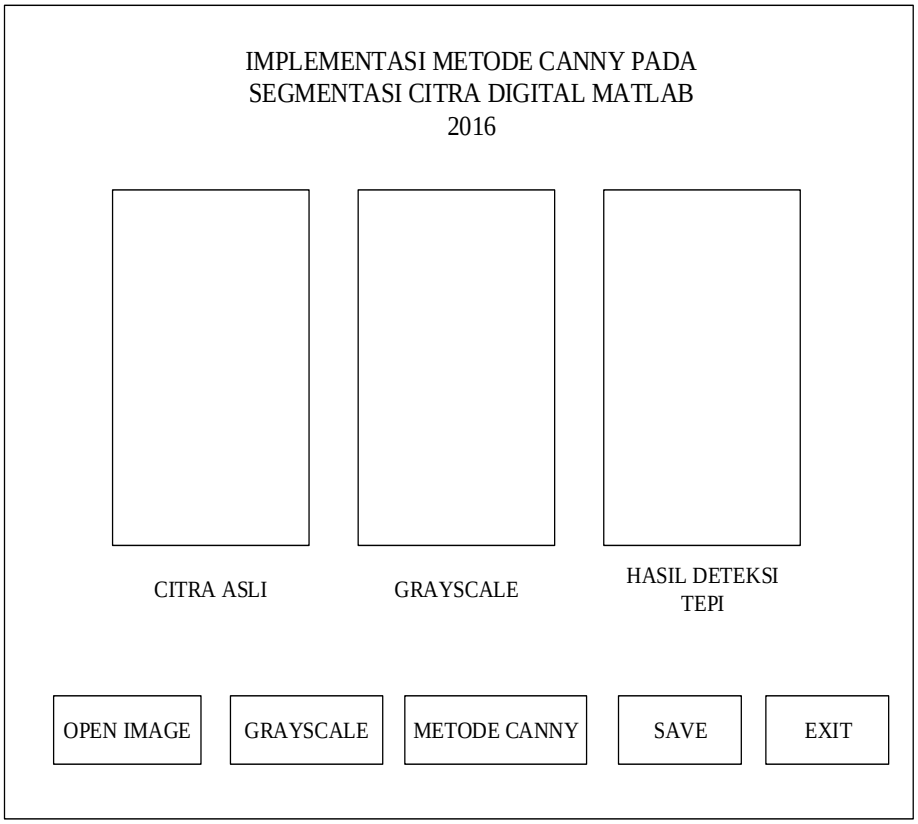
1. *Form* Menu Utama

Form menu utama merupakan *form* yang berfungsi untuk melakukan proses awal dalam memulai sebuah program, dimana pada *form* utama pada program segmentasi citra digital dengan metode *canny* ini memiliki menu utama dan exit, serta pada menu utama memiliki satu submenu untuk memanggil *form* operator *canny*.



Gambar 3.7 Perancangan *Form* Menu Utama

2. *Form* Deteksi Tepi Dengan Metode *Canny*



Gambar 3.8 Rancangan *Form* Deteksi Tepi pada Matlab GUI

Pada rancangan *form* deteksi tepi pada matlab GUI seperti gambar diatas terdapat 5 buah tombol yang berfungsi untuk mengelolah data yang terinput didalam program matlab, dimana fungsi setiap tombol berbeda seperti dijelaskan dibawah ini :

1) *Open Image*

Fungsi dari sub menu open ini adalah untuk mencari di direktori mana gambar yang akan ditampilkan di *toolbox picture* untuk di proses selanjutnya.

2) *Grayscale*

Fungsi dari sub menu *Grayscale* adalah untuk mengubah citra awal ke warna keabuan pada tingkat warna hitam sampai warna putih.

3) Metode *Canny*

Fungsi dari sub menu metode *canny* berfungsi untuk mendeteksi tepi citra pada gambar yang telah di pilih untuk mengetahui hasil dari proses metode *canny*.

4) *Save*

Fungsi dari sub menu *save* ini adalah untuk menyimpan gambar yang sudah diproses *edge lingking* dengan menggunakan metode *canny*, dan *user* dapat menyimpan gambar tersebut ke direktori dimana gambar tersebut disimpan.

5) *Exit*

Fungsi dari sub menu *exit* adalah, jika *user* ingin keluar dari proses aplikasi operator *canny* pada program matlab

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Algoritma

Algoritma merupakan langkah-langkah ataupun susunan yang logis yang digunakan untuk memecahkan masalah dalam perancangan program deteksi tepi dengan metode *canny* pada skripsi ini. Adapun algoritma yang dibuat dalam langkah-langkah segmentasi citra digital dengan metode *canny* dalam pengolahan citra adalah sebagai berikut :

1. Algoritma segmentasi citra semula dirubah ke citra *grayscale*

   x(Grsc) *Grayscale*
   Input : R,G,B
   Proses : R ← *Red*
   G ← *Green*
   B ← *Blue*
   Grsc ← 1/3(R+G+B)
   Output : Grsc

2. Algoritma Horizontal

   Sx (Horizontal)

   Input : P1,Cp2,P3,P5,Cp6,P7

   Proses : P1 ← Horizontal
   Cp2 ← Horizontal
   P3 ← Horizontal
   P5 ← Horizontal

Cp6← Horizontal

P7 ← Horizontal

Sx ←(P1+Cp2+P3)-(P7+Cp6+P5)

Output : Sx

3. Algoritma Vertikal

Input : P1,P3,Cp4,P5,P7,Cp8

Proses : P1 ← Vertikal

P3 ← Vertikal

Cp4←Vertikal

P5 ←Vertikal

P7 ←Vertikal

Cp8← Vertikal

Sy ←(P1+Cp8+P7)-(P3+Cp4+P5)

Output : Sy

4. Algoritma Deteksi Tepi Menggunakan *Canny*

Hasil Citra (Hc1)

Input : Grsc, Sx,

Proses : Grsc ←*grayscale*

Sx ←Horizontal

Hc ←Grsc*Sx

Output : Hc(1)

Hasil Citra (Hc2)

Input : Grsc, Sy

Proses : Grsc ← *grayscale*

Sy ← Vertikal

Hc ← Grsc*Sy

Output : Hc(2)

Hasil *Canny*

Input : Hc(1), Hc(2)

Proses : Hc(1) ← Hasil Citra (1)

Hc (2)← Hasil Citra (2)

Hasil *Canny*← Hc(1) + Hc(2)

Output : Hasil *Canny*

4.2 Implementasi Sistem

Setelah sistem dianalisis dan didesain secara rinci, maka akan menuju pada tahapan implementasi. Implementasi merupakan tahap meletakan sistem sehingga siap untuk di operasikan. Adapun tujuan dari implementasi adalah untuk menkonfirmasi modul-modul perancangan, sehingga *user* dapat memberikan masukan dalam pengembangan lebih lanjut nantinya.

4.2.1 Kebutuhan Implementasi Sistem

Adapun kebutuhan implementasi sistem yang akan digunakan dalam pengujian segmentasi citra digital menggunakan metode *canny* adalah sebagai berikut :

1. Perangkat Keras (*Hardware*)

   Berikut adalah spesifikasi perangkat keras yang digunakan saat pengembangan dan pengujian sistem segmentasi citra digital menggunakan metode *canny*.

Tabel 4.1 Perangkat keras

Perangkat Keras	Spesifikasi
Processor	Core i5 M460@2.53GHz
RAM	6 GB
Hardisk	640 GB
VGA	AMD Radeon 5000 S
Monitor	14"
Mouse	Standar
Keyboard	Standar

2. Perangkat Lunak (*Software*)

   Berikut adalah spesifikasi perangkat lunak yang digunakan saat pengembangan dan pengujian sistem segmentasi citra digital menggunakan metode *canny*.

Tabel 4.2 Spesifikasi Perangkat Lunak (*Software*)

Spesifikasi Software	
Windows 10	Sebagai system operasi
Matlab R2016a	Sebagai *Software* Komputasi, Visualisasi, dan Pemograman
Paint	Sebagai aplikasi pemotong citra gambar

3. Kebutuhan Sumber Daya Manusia (*Brainware*)

Pada implementasi ini dibutuhkan seorang pengguna (*user*) yang mampu menjalankan atau mengoperasikan komputer dan menjalankan sistem yang telah dibuat sehingga penerapan implementasi sistem dapat berjalan dengan baik.

4.2.2 Implementasi Perancangan Aplikasi

Implementasi tampilan aplikasi akan di ambil dari *screenshot* aplikasi ketika aplikasi dijalankan. Berikut ini adalah implementasi tampilan aplikasi sistem segmentasi citra digital menggunakan metode *canny*.

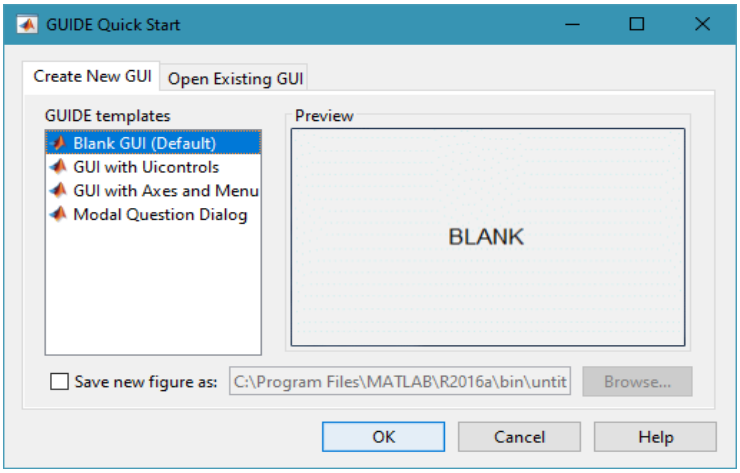
1. Pembuatan *GUIDE*

Pembuatan *GUIDE* atau *GUI builder* merupakan sebuah *user interface* (*GUI*) yang dibangun dengan obyek grafik seperti tombol (*button*), kotak teks, *slider*, menu dan lain-lain. Untuk Memulai *GUIDE* Matlab dapat dilakukan dengan dua cara, yaitu:

1) Melalui *command* yang terletak pada sisi bawah matlab dengan mengetikkan: >> *guide* (enter)

Gambar 4.1 Memulai *Guide Matlab*

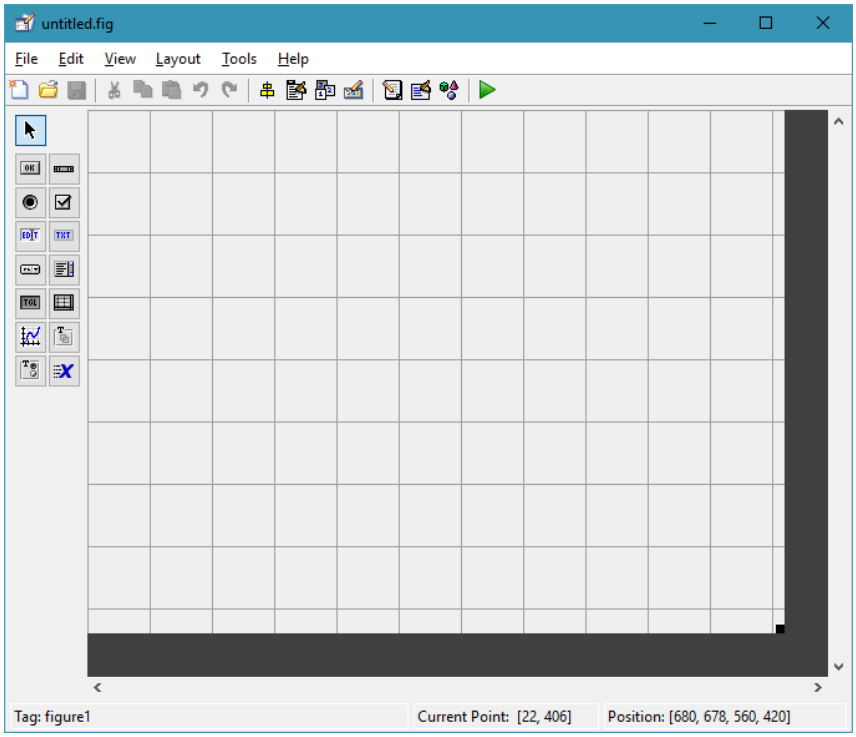
2) Melalui klik tombol *Start Matlab* atau menu *home* dan pilihlah *New* lalu pilih *app* dan pilih *GUIDE* (*GUI Bulder*). Lalu akan muncul *window* seperti di bawah ini:



Gambar 4.2 Pemilihan *Guide Matlab*

Adapun langkah-langkah menggunakan *windows Guide* pada *Matlab 2016a* adalah sebagai berikut :

1) Klik *Blank Gui* (*Default*)
2) Lalu klik OK
3) Setelah itu muncullah jendela seperti gambar berikut ini.



Gambar 4.3 Tampilan Utama *Guide*

2. Tampilan *Form* Utama

Tampilan *form* utama yang dihasilkan merupakan tampilan yang menyerupai bentuk perancangan *form* utama yang dirancang sebelumnya. Pada tampilan *form* utama terdapat dua buah menu utama dan satu sub menu dimana menu utama yang berfungsi untuk menampilkan tampilan utama sebelum mamasuki *form* deteksi tepi dengan metode *canny*, sedangkan sub menu operator *canny* yang berfungsi menampilkan *form* operator *canny* dan menu keluar berfungsi untuk keluar dari program segmentasi citra digital dengan metode *canny*.

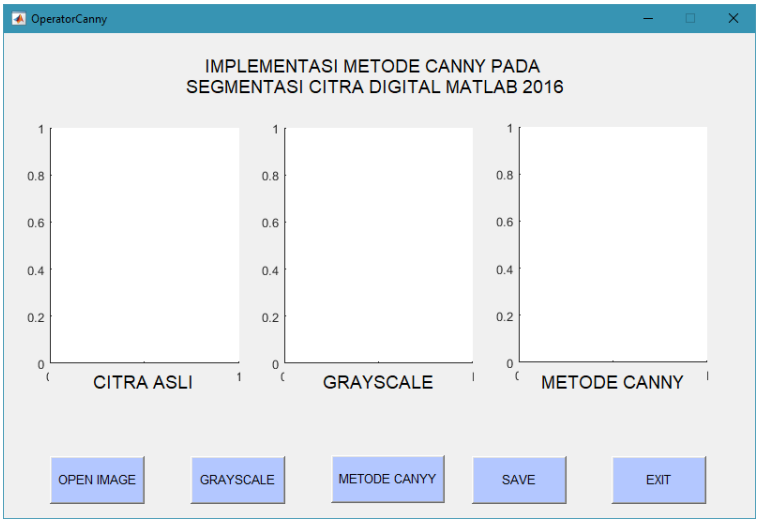
Form utama dapat dilihat pada gambar berikut ini :

Gambar 4.4 Tampilan *Form* Utama

3. Tampilan *Form* Operator *Canny*

Tampilan *form* operator *canny* dibawah ini memiliki tiga buah *axes* yang berfungsi untuk menampilkan gambar yang diiginkan dalam proses segmentasi citra dengan cara mendeteksi tepi pada sebuah objek. Dimana *axes* pertama menampilkan hasil citra asli yang digunakan, dan akses yang kedua menampilkan hasil *grayscale* dari citra asli/awal sebelumnya dan akses ke tiga menampilkan hasil deteksi tepi dengan metode *canny*.

Berikut adalah tampilan *form* operator *canny* pada sistem segmentasi citra digital menggunakan metode *canny*.



Gambar 4.5 Tampilan Operator *Canny*

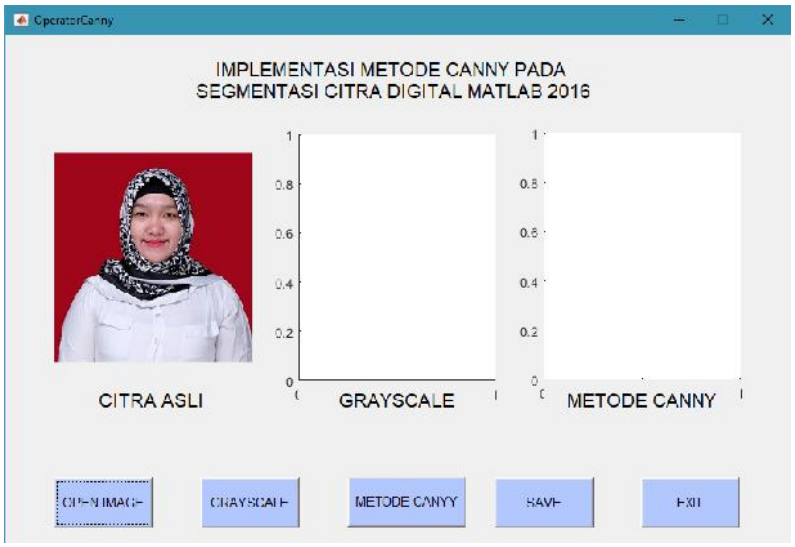
Pada *form* operator *canny* ini juga memiliki 5 buah Push Button dimana setiap push button memiliki fungsi masing-masing. Push button open image berfungsi untuk memanggil gambar yang diniginkan oleh pengguna untuk dideteksi dan akan tampil pada akses pertama, push button grayscale berfungsi untuk merubah citra asli R,G,B menjadi citra keabuan atau citra grayscale dan akan tampil pada akses dua. Push button metode *canny* berfungsi untuk menghasilkan hasil dari citra asli mnjadi titik-titik disetiap tepi dari citra asli sebelumnya dan ditampilkan pada akses ketiga, sedangkan tombol *save* berfungsi untuk menyimpan data/gambar hasil dari deteksi tepi dan tombol exit berfungsi untuk keluar dari *form* operator *canny*.

4.3 Pengujian Sistem

Setelah mendapatkan hasil tampilan perangkat lunak, selanjutnya dilakukan pengujian sistem tersebut. Adapun metode pengujian sistem yang penulis lakukan adalah metode *black box testing*. Uji coba yang akan dilakukan bersifat mandiri dan di uji coba langsung dengan memperhatikan rancangan yang dibuat. Berikut adalah beberapa tahapan pengujian.

4.3.1 Pengujian Proses *Open Image*

Pengujian proses *open image* dilakukan untuk melihat apakah gambar yang akan di segmentasi bisa ditampilkan pada citra asli/awal. Caranya adalah dengan mengklik tombol *open image* pada form operator *canny* dengan memilih objek/gambar yang di pilih dalam pengolahan citra dan akan dideteksi tepinya dengan metode *canny* yang akan di proses dari citra asli menjadi citra *grayscale*.

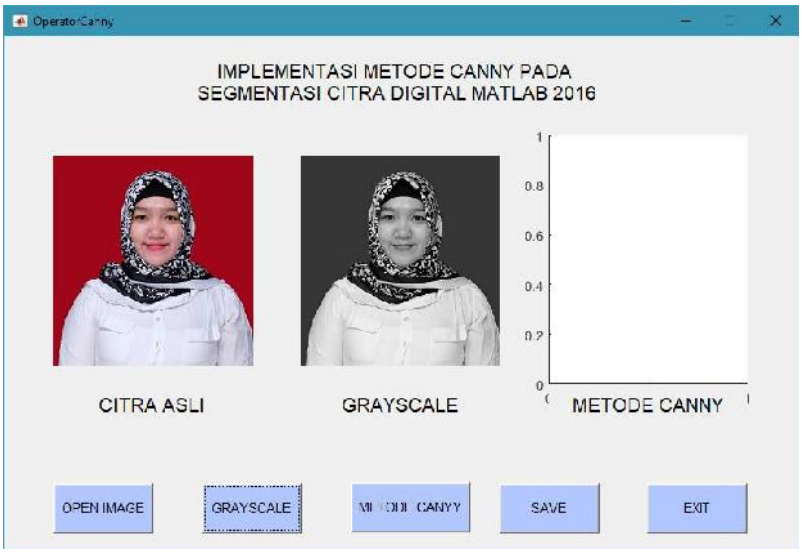


Gambar 4.6 Pengujian *Open Image*

Dari hasil pengujian yang dilakukan dapat diketahui bahwa tidak terjadi *error* pada proses pemilihan gambar dengan format Jpg. Dimana gambar yang dimasukan pada proses segmentasi telah berhasil di proses dengan memilih foto dengan format Jpg untuk disegmentasi menggunakan metode *canny* seperti dapat dilihat pada gambar diatas.

4.3.2 Pengujian Proses *Grayscale*

Pada *form grayscale* ini berfungsi untuk merubah citra asli ke citra keabuan atau *grayscale*. Gambar yang di input pada akses pertama adalah citra asli/awal dan pada saat *push button grayscale* di klik akan menghasilkan data keabu-abuan dan tampil pada akses ke dua seperti pada gambar dibawah ini:



Gambar 4.7 Pengujian Proses *Grayscale*

Hasil pengujian yang dilakukan dapat diketahui bahwa tidak terjadi *error* pada proses segmentasi citra asli menjadi *grayscale* menggunakan metode *canny*.

4.3.3 Pengujian Proses Metode *Canny*

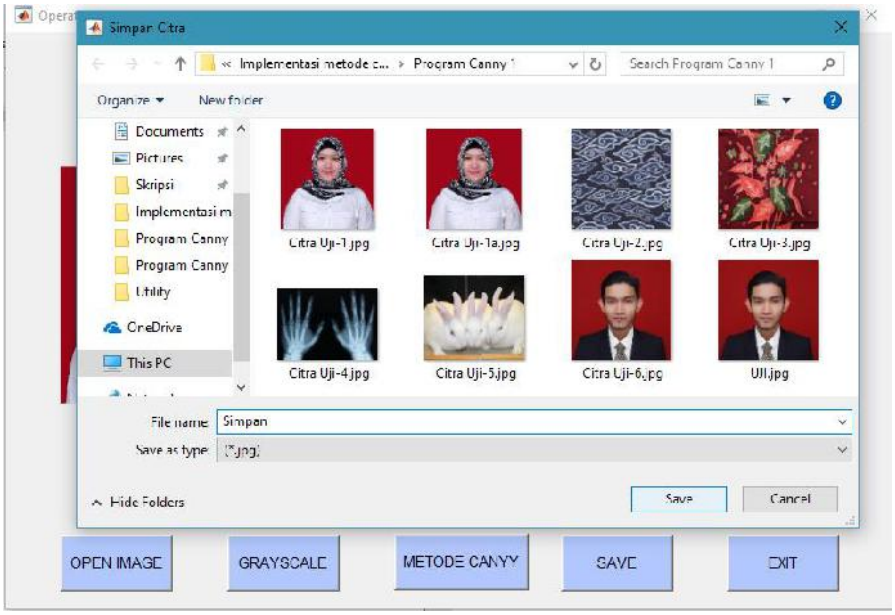
Pada *form* metode *canny* dibawah ini akan menghasilkan garis-garis disetiap tepi objek yang dideteksi, dan latar objek akan dipisahkan dengan segmentasi sehingga tampilan seperti akses ketiga, sehingga muncul garis-garis disetiap tepi objek.

Gambar 4.8 Pengujian Proses Metode *Canny*

Pengujian proses metode *canny* yang dilakukan pada gambar diatas dapat diketahui bahwa sudah tidak ada lagi noise pada sebuah gambar yang telah diubah ke citra grayscale sehingga dapat di proses tepi pada objek. Dimana hasil pengujian pada metode *canny* telah berhasil dideteksi tepi citra pada objek secara jelas dan halus sesuai dengan konvolusi yang dilakukan.

4.3.4 Pengujian Proses *Save*

Setelah semua proses dijalankan dengan menghasilkan gambar yang dideteksi tepinya, selanjutnya gambar dapat di simpan/di *save* di tempat folder yang ingin di masukan untuk tempat menyimpan data seperti gambar di bawah ini:



Gambar 4.9 Pengujian Proses *Save*

Pengujian proses *save* yang dilakukan pada gambar diatas dapat diketahui bahwa gambar sudah tersimpan di tempat folder gambar. Dimana hasil citra deteksi tepi yang tersimpan juga menggunakan format Jpg.

Adapun kesimpulan dari pengujian sistem yang penulis lakukan dengan menggunakan metode *black box testing* atau hanya mengamati hasil eksekusi melalui data yang di uji serta memeriksa fungsional dari perangkat lunak dapat dilihat pada tabel berikut ini :

Tabel 4.3 Pengujian Sistem (*black box*)

No	Kasus/Form yang diuji	Skenario Uji	Hasil yang diharapkan	Hasil Pengujian
1	Menjalankan Program *(Running Menu)*	Menjalankan *source code* program menggunakan kompiler program aplikasi Matlab R2016a	Ketika program di jalankan maka akan menampilkan tampilan dan informasi program seperti pada saat perancangan.	[v] Berhasil [ ] Gagal
2	Operator *Canny* pada sub menu utama program	Memilih operator *canny* pada sub menu utama program.	Ketika memilih operator *canny* maka secara otomatis program akan membuka jendela baru tempat untuk memasukan citra	[v] Berhasil [ ] Gagal
3	Tombol *open image* pada operator *canny*	Memilih tombol *open image* dan memilih citra asli yang nantinya akan di tampilkan pada program operator *canny*.	Ketika berhasil menekan tombol *open image* maka akan menampilkan jendela untuk memilih citra asli yang kemudian akan di tampilkan pada program.	[v] Berhasil [ ] Gagal

4	Tombol *grayscale* pada program.	Memilih tombol *grayscale* pada program maka program akan melakukan perubahan pada citra	Ketika berhasil maka citra asli akan berubah menjadi citra *graysclace* atau abu-abu.	[v] Berhasil [ ] Gagal
5	Tombol Metode *Canny*	Memilih tombol Metode *canny* pada program, maka diharapkan citra yang sebelumnya *grayscale* akan di konvulasi	Ketika meng klik tombol metode *canny* maka program akan melakukan konvulasi pada citra sebelumnya dan menampilkan hasil pada tampilan axes sebelah kanan program	[v] Berhasil [ ] Gagal
6	Tombol *Save* pada program	Mengklik tombol *save* pada program.	Ketika meng klik tombol *save*, maka program akan menampilkan jendela baru menunjukan lokasi penyimpanan.	[v] Berhasil [ ] Gagal
7	Tombol *Exit* pada program	Mengklik tombol *exit*	Maka akan keluar dari semua proses yang di lakukan program	[v] Berhasil [ ] Gagal

4.4 Kesimpulan dan Evaluasi Sistem

Pengujian yang telah dilakukan menggunakan metode *black box* merupakan jenis pengujian sistem yang bertujuan untuk menemukan pesan kesalahan atau kekurangan pada perangkat lunak pada saat dijalankan. Berdasarkan pada pengujian sistem informasi yang telah dilakukan sebelumnya dapat disimpulkan bahwa sistem yang telah di implementasikan sesuai dengan perancangan sistem pada bab sebelumnya serta telah memenuhi standar pengujian.

Hasil yang didapatkan dari pembahasan yang ada adalah terciptanya sebuah sistem *Edge Lingking* dalam proses metode *canny* yang menghasilkan nilai tepi pada gambar melalui segmentasi citra yang menggunakan program matlab 2016a. Dalam implementasi program *matlab* ini menjelaskan tentang alur pembuatan dan kegunaan program dilengkapi dengan antar muka program atau sering disebut dengan *Interface*.

Selain itu juga pemilihan *software* aplikasi yang tepat, yang memiliki kemampuan untuk menyelesaikan permasalahan yang ada sekarang ini sering digunakan oleh mahasiswa dalam mendeteksi tepi gambar. *Software* yang digunakan dalam menyelesaian permasalahan yang ada yaitu *Software Matlab* 2016a adalah *interactive* program untuk *numerical computation* dan data *visualization* digunakan secara *extensif* oleh *control engineers* untuk *analysis* dan *design*. Terdapat banyak *toolboxes* yang tersedia yang terdiri dari *basic functions* di *Matlab* dalam aplikasi yang berbeda. Ide pada tutorial ini adalah pengguna dapat melihat *Matlab* pada satu window ketika menjalankan Matlab di window yang lain.

Pengguna dapat membuat *plot* dan menggunakan program yang tersedia dalam *m-file.*

Adapun beberapa kelebihan dalam implementasi metode canny mengunakan Matlab adalah sebagai berikut :

1. Kelebihan dari metode *canny* ini adalah kemampuannya dalam mengurangi noise sebelum melakukan perhitungan deteksi tepi sehingga sehingga tepi tepi yang di hasilkan lebih banyak.
2. Metode *canny* dilakukan dengan konvolusi fungsi gambar dengan operator *Gaussian* dan turunan-turunannya. Turunan pertama dari fungsi citra yang dikonvolusikan dengan fungsi *gaussian* di mungkinkan dihasilkan jarak yang minimum antara tepi yang di deteksi dengan tepi yang asli.
3. *Edge Lingking image* dengan metode *canny* dapat menghasilkan garis tepi setelah diproses dengan program matlab serta dapat mendeteksi pengolahan citra dengan baik sehingga hasil deteksi tepi dapat lebih halus.
4. Program *Matlab* dapat digunakan dalam pengolahan citra untuk mendeteksi citra tepi dengan mengunakan metode *canny* dan dapat merubah citra R,G,B menjadi citra *grayscale.*

Selain memiliki kelebihan, juga memiliki beberapa kekurangan yang dapat dijabarkan sebagai berikut :

1. Operator *canny* tidak bekerja terlalu baik pada citra asli dengan ukuran dan pixel yang tinggi karena mengakibatkan permasalahan pada sudut dan

persimpangan yang mengakibatkan titik pola tidak terhubung satu dengan yang lain yang menjadikan hasil citra menjadi kurang tampak.

2. Ukuran perbaikan citra tergantung pada ukuran dan resolusi citra yang dapat menyebabkan lokasi tepi mungkin mati atau tidak tampak.
3. Sudut pixel menampilkan citra dan arah yang salah jika menggunakan citra percobaan dengan ukuran dan resolusi yang tinggi.
4. Matlab adalah bahasa pemrograman komputer yang tidak memerlukan definisi variabel secara khusus.
5. Dalam mendefinisikan sebuah variabel harus penuh ketelitian, karena jika variabel yang dimasukan kurang satu kata maka program matlab tidak dieksekusi atau tidak jalan.

BAB V

PENUTUP

1. Kesimpulan

Setelah melakukan analisa hasil pengujian pada pegolahan citra digital tentang segmentasi citra digital dengan metode *canny* maka dapat diambil kesimpulan seperti dibawah ini:

a. Implementasi Metode *canny* pada segmentasi citra digital telah berhasil bangun untuk mendeteksi batas tepi menggunakan program aplikasi *Matlab* R2016a.
b. Proses segmentasi citra digital menggunakan operator *canny* sangat bergantung pada gradiasi dan isi parameter citra yang di uji.
c. Proses segmentasi citra digital dalam mendeteksi tepi gambar dengan format JPG menghasilkan tepi-tepi dari obyek-obyek citra untuk menandai bagian yang menjadi detail citra, memperbaiki detail dari citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra pada citra *grayscale*.
d. Proses segmentasi citra digital menggunakan operator *canny* akan memperhalus ujung ke ujung sebelum dapat mendeteksi semua ujung serta waktu dan ukuran filter yang sesuai akan mendeteksi semua sisi tanpa kesalahan.

2. Saran

Dalam Pembuatan Skripsi ini, masih banyak kekurangan yang dapat diperbaiki untuk pengembangan berikutnya. Beberapa saran yang dapat diberikan adalah :

a. Pengembangan selanjutnya, dapat menggunakan program aplikasi selain matlab serta sistem bisa mengenali gambar tidak hanya *.jpg, tapi juga gambar yang berekstensi lain seperti format *.gif, *.bmp, *.png maupun video dari berbagai ukuran sehingga lebih cepat untuk diidentifikasi.
b. Untuk pemilihan citra yang akan di uji harap untuk memperhatikan jarak objek pada citra, ukuran dan kualitas untuk mengapatkan hasil yang sempurna, karena operator *canny* tidak bekerja terlalu baik pada citra asli dengan ukuran dan resolusi yang tinggi karena mengakibatkan titik-titik pada citra tidak bersambung dengan baik yang menjadikan hasil konvulasi citra menjadi kurang jelas.
c. Perlu dikembangkan penelitian dengan metode pengenalan yang lain, misalnya menggunakan jaringan syaraf tiruan.
d. Pemilihan citra/sampel yang baik untuk proses pelatihan mutlak dilakukan karena sangat akan mempengaruhi hasil akhir.

DAFTAR PUSTAKA

Abdul Kadir & Adhi Susanto (2013). *Teori dan Aplikasi Pengolahan Citra*. PenerbitAndi, Yogyakarta.

A.S Rosa , dan M.Shalahuddin. (2014). *Rekayasa Perangkat Lunak Struktur dan Berorientasi Objek*. Bandung : Informatika.

Bill Green. (2010). *Cara Kerja Deteksi Canny Edge*. Magister Teknik Nasional, Universitas Donetsk. | *Lengkap*. Available at: http://masters.donntu.org/2010/fknt/chudovskaja/library/article5.htm (Accessed: 12 Desember 2018).

Cahyono, B. (2013). *Penggunaan Software Matrix Laboratory (Matlab) Dalam Pembelajaran Aljabar Linier*. Jurnal PHENOMENON, Vol. 1. No.1, 47-62

Erika, Winda, Heni Rachmawati, and Ibnu Surya. "Enkripsi Teks Surat Elektronik (E-Mail) Berbasis Algoritma Rivest Shamir Adleman (RSA)." *Jurnal Aksara Komputer Terapan* 1.2 (2012).

Firendra (2018). *Implementasi Adalah? Berikut 10 Pengertian Implementasi Menurut Para Ahli | Lengkap*. Available at: http://blogpengertian.com/arti-implementasi- adalah/ (Accessed: 22 November 2018).

Hafni, layla, and rismawati rismawati. "analisis faktor-faktor internal yang mempengaruhi nilai perusahaan pada perusahaan manufaktur yang terdaftar di bei 2011-2015." bilancia: jurnal ilmiah akuntansi 1.3 (2017): 371-382.

Hamdi, nurul. "model penyiraman otomatis pada tanaman cabe rawit berbasis programmable hamdi, muhammad nurul, evi nurjanah, and latifah safitri handayani. "community development based onibnu khaldun thought, sebuah interpretasi program pemberdayaan umkm di bank zakat el-zawa." el muhasaba: jurnal akuntansi (e-journal) 5.2 (2014): 158-180.

Hermawati, Fajar Astuti. (2013). *Konsep dan Teori Pengolahan Citra Digital*,Yogyakarta : Andi Offset

Indra permana, a. M. I. N. U. D. D. I. N. "sistem pakar mendeteksi hama dan penyakit tanaman kelapa sawit pada pt. Moeis kebun sipare-pare kabupaten batubara." (2013).

Mulyadi (2015). *Akuntansi Biaya, Edisi 5*. Yogyakarta : Sekolah Tinggi Ilmu Manajemen YKPN.

Muttaqin, muhammad. "analisa pemanfaatan sistem informasi e-office pada universitas pembangunan panca budi medan dengan menggunakan metode utaut." *jurnal teknik dan informatika* 5.1 (2018): 40-43.

Perwitasari, I. D. (2018). Teknik Marker Based Tracking Augmented Reality untuk Visualisasi Anatomi Organ Tubuh Manusia Berbasis Android. *INTECOMS: Journal of Information Technology and Computer Science*, *1*(1), 8-18.

Prijono, Agus dan Marvin Ch. Wijaya. *Pengolahan Citra Digital Menggunakan Matlab Image Processing Toolbox*. Cetakan Pertama. 2007. Bandung:INFORMATIKA

Puspita, Khairani, and Purwa Hasan Putra. "Penerapan Metode Simple Additive Weighting (SAW) Dalam Menentukan Pendirian Lokasi Gramedia Di Sumatera Utara." Seminar Nasional Teknologi Informasi Dan Multimedia, ISSN. 2015.

Rizal, Chairul. "Pengaruh Varietas dan Pupuk Petroganik Terhadap Pertumbuhan, Produksi dan Viabilitas Benih Jagung (Zea mays L.)." ETD Unsyiah (2013).

Sukatmi. (2017). Perbandingan Deteksi Tepi Citra Digital dengan Metode Prewitt, Sobel dan Canny. KOPERTIP, 01(01), 1–4.

Supardi. (2013). *Aplikasi Statistika dalam Penelitian Konsep Statistika yang Lebih Komprehensif*. Jakarta: Change Publication

Syahputra, rizki, and hafni hafni. "analisis kinerja jaringan switching clos tanpa buffer." *journal of science and social research* 1.2 (2018): 109-115.

Wahyuni, Sri. "Implementasi Rapidminer Dalam Menganalisa Data Mahasiswa Drop Out." *Jurnal Abdi Ilmu* 10.2 (2018): 1899-1902

Winarno, E. (2014). *Aplikasi Deteksi Tepi pada Realtime Video menggunakan Algoritma Canny Detection.* Jurnal Teknologi Informasi DINAMIK, 16, 44-49.