



**TUGAS AKHIR - KI091391**

# **IMPLEMENTASI ALGORITMA ANT-INSPIRED UNTUK MENDETEKSI TEPI GAMBAR**

**IDA AYU PUTU KRISTANTARI**  
**NRP 5109100094**

**Dosen Pembimbing 1**  
**Ahmad Saikhu, S.Si., M.T**

**Dosen Pembimbing 2**  
**Anny Yuniarti, S.Kom., M.Comp.Sc.**

**JURUSAN TEKNIK INFORMATIKA**  
**Fakultas Teknologi Informasi**  
**Institut Teknologi Sepuluh Nopember**  
**Surabaya 2014**



**FINAL PROJECT - KI091391**

# **ANT-INSPIRED ALGORITHM IMPLEMENTATION FOR IMAGE EDGE DETECTION**

**IDA AYU PUTU KRISTANTARI  
NRP 5109100094**

**First Advisor  
Ahmad Saikhu, S.Si., M.T.**

**Second Advisor  
Anny Yuniarti, S.Kom., M.Comp.Sc.**

**DEPARTMENT OF INFORMATICS  
Faculty of Information Technology  
Sepuluh Nopember Institute of Technology  
Surabaya 2014**

# IMPLEMENTASI ALGORITMA ANT-INSPIRED UNTUK MENDETEKSI TEPI GAMBAR

Nama Mahasiswa : IDA AYU PUTU KRISTANTARI  
NRP : 5109 100 094  
Jurusan : Teknik Informatika FTIf-ITS  
Dosen Pembimbing 1 : Ahmad Saikhu, S.Si., M.T.  
Dosen Pembimbing 2 : Anny Yuniarti, S.Kom., M.Comp.Sc.

## Abstrak

*Deteksi tepi merupakan salah satu proses pengolahan yang sering dibutuhkan pada analisis citra. Untuk mendapatkan citra yang baik dan citra yang sesuai dengan keinginan, biasanya citra diproses terlebih dahulu dengan perbaikan kualitas citra. Tepi citra selalu memiliki informasi penting mengenai objek gambar.*

*Pada awalnya terdapat beberapa metode deteksi tepi tradisional seperti canny yang mengekstrak tepi dengan menggunakan template tertentu dengan kombinasi fungsi smoothing (penghalusan) gambar. Deteksi tepi metode tradisional ini kadangkala belum maksimal saat digunakan untuk mendeteksi tepi citra, seperti garis-garis tepi yang dihasilkan masih terlihat tidak menyambung satu sama lain.*

*Dalam tugas akhir ini, algoritma ant-inspired digunakan untuk melakukan deteksi terhadap tepi citra yang sudah ditransformasi terlebih dahulu ke dalam format citra keabuan. Transformasi ke citra keabuan diperlukan untuk mempermudah proses deteksi tepi citra. Deteksi tepi yang digunakan didasarkan pada pergerakan semut yang berjalan mencari makanan dengan meninggalkan feromon sebagai jejaknya. Ada beberapa tahapan dalam tugas akhir ini. Tahap pertama adalah melakukan konversi citra ke format keabuan. Tahap selanjutnya adalah proses perubahan citra menjadi*

*citra gradien. Tahap yang terakhir adalah melakukan proses deteksi tepi citra menggunakan algoritma ant-inspired.*

*Dari percobaan yang dilakukan, dapat dibuktikan bahwa algoritma ant-inspired dapat menghasilkan deteksi tepi citra yang lebih baik dan akurat dengan nilai akurasi rata-rata sebesar 88,09%. Berdasarkan eksperimen yang dilakukan semakin besar jumlah iterasi, maka semakin meningkat pula kemampuan algoritma ini dalam mendeteksi tepi citra. Perubahan nilai threshold juga memperlihatkan perbedaan yang signifikan pada hasil deteksi tepi citra.*

***Kata Kunci: Ant-inspired, Ant Colony, Deteksi Tepi, Gradien Gambar***

# ANT-INSPIRED ALGORITHM IMPLEMENTATION FOR IMAGE EDGE DETECTION

**Student's Name** : IDA AYU PUTU KRISTANTARI  
**Student's ID** : 5109 100 094  
**Department** : Informatics, FTIf-ITS  
**First Advisor** : Ahmad Saikhu, S.Si., M.T.  
**Second Advisor** : Anny Yuniarti, S.Kom., M.Comp.Sc.

## Abstract

*Edge detection is one of the important image digital processes that are often needed in image analysis. To get a good of the desired result, image is usually processed first with image quality improvement. The edges of an image have important information about the object of the images.*

*There are some traditional edge detection methods such as canny edges that extracted the edges using a specific template with a combination of smoothing functions images. Sometimes, traditional edge detection methods are not effective enough when it used to detect the edges of the image. The edges result of these methods still looks not connected to each other.*

*In this final project, ant-inspired algorithms are used to perform edge detection on the image that has been transformed first into grayscale image format. Transformation into gray image is needed in order to the process of image edge detection become easier. This detection method based on the movement of walking ants looking for food by leaving a trail of pheromones. There are several stages in the final project. The first stage is converted the image to a grayscale format. The next stage is the process of changing the image*

*into image gradient. The last stage is the process of image edge detection using ant-inspired algorithm.*

*From the result of experiments, it can be proven that the ant-inspired algorithm can produce better and accurate edge on image with 88,09% of accuracy rate. Based on the experiments, the greater the number of iterations, it also increases the ability of the algorithm to detect the edges of the image. Increasing the threshold value changing also showed a significant difference in detecting the edges of the image.*

***Keywords: Ant-inspired, Ant Colony, Edge Detection, Gradient Image***

## LEMBAR PENGESAHAN

### IMPLEMENTASI ALGORITMA ANT-INSPIRED UNTUK MENDETEKSI TEPI GAMBAR

#### TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat  
Memperoleh Gelar Sarjana Komputer  
pada  
Bidang Studi Komputasi Cerdas dan Visualisasi  
Program Studi S-1 Jurusan Teknik Informatika  
Fakultas Teknologi Informasi  
Institut Teknologi Sepuluh Nopember

Oleh

**IDA AYU PUTU KRISTIANTARI**  
NRP. 5109 100 094

Disetujui oleh Pembimbing Tugas Akhir

1. Ahmad Saikhu, S.Si., M.T.  
NIP: 19710718 2006 04 001 (Pembimbing 1)
2. Anny Yuniarti, S.Kom., M.Comp.Sc.  
NIP: 19810622 2005 01 2 002 (Pembimbing 2)

**SURABAYA**  
**JANUARI, 2014**

## KATA PENGANTAR

Puji syukur kehadapan Tuhan Yang Maha Esa, yang telah melimpahkan berkat-Nya sehingga penulis bisa menyelesaikan tugas akhir yang berjudul ***“Implementasi Algoritma Ant-inspired untuk Mendeteksi Tepi Gambar”*** dengan tepat waktu.

Pengerjaan tugas akhir ini merupakan suatu kesempatan yang sangat berharga bagi penulis, karena dengan pengerjaan tugas akhir ini, penulis bisa memperdalam, meningkatkan, serta mengimplementasikan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesainya buku tugas akhir ini, tidak terlepas dari bantuan dan dukungan semua pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Tuhan Yang Maha Esa atas limpahan berkat-Nya sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.
2. Bapak Ahmad Saikhu, S.Si., M.T. selaku dosen pembimbing 1, yang telah memberikan kepercayaan, bimbingan, dukungan, nasehat, perhatian, serta semangat dikala penulis sedang dalam kesulitan.
3. Ibu Anny Yuniarti, S.Kom., M.Comp.Sc. selaku Dosen Pembimbing 2, yang telah memberikan kepercayaan, motivasi, bimbingan, dukungan, nasehat, perhatian, serta semua bantuan yang telah diberikan kepada penulis.
4. Ibu Nanik Suciati, S.Kom., M.Kom. selaku ketua jurusan Teknik Informatika ITS, Bapak Ary Mazzaruddin Shiddieqi selaku dosen wali penulis, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
5. Pak Yudi, Pak Sugeng, dan segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada



penulis selama menjalani kuliah di Teknik Informatika ITS.

6. Orang tua tercinta, bapak, ibu, kedua adikku yang selalu memberikan dukungan jasmani dan rohani, kasih sayang yang tak terkira, semangat, perhatian, serta doa yang luar biasa yang selalu dipanjatkan untuk penulis.
7. Teman-teman penulis, Shabrina, Almira, Juwita, Adel, Yolanda, Septi, Beti, Ika, Nur, Dila, Millah, yang selalu ada dan setia mendengarkan semua keluh kesah penulis dan selalu sabar direpotkan. Nyoman, Krizal, Ami, Evita, Andra, Rohmad, Rani, Thari, Yuyun, Yandri, Fadhil, dan teman-teman seperjuangan lainnya atas semangat dan bantuannya.
8. Teman-teman laboratorium VIPG, Fiah, Iqbal Aghe, Rosiadi, Akbar, Nia, Winny, dan Reda yang sudah membantu penulis dalam mengerjakan tugas akhir.
9. Teman-teman seperjuangan angkatan 2009 serta kakak 2007, 2008, dan adik-adik 2010, 2011, dan 2012.
10. Teman-teman admin laboratorium Game, OS, @jK, NCC, IBS, LP, LP 2, RPL, GCL, dan Semantik untuk semua bantuannya selama ini.
11. Teman-teman kost G/14, Mike, Chintya, Vivin, Nanda, Erna, Sinta, Epru, Desy, Aya, dan Nora yang menemani penulis selama kost di Surabaya.
12. Juga tak lupa kepada semua pihak yang belum sempat disebutkan satu per satu disini yang telah membantu terselesaikannya tugas akhir ini.

Sebagai manusia biasa, penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan dan memiliki banyak kekurangan. Sehingga dengan segala kerendahan hati, penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Januari 2014

## DAFTAR ISI

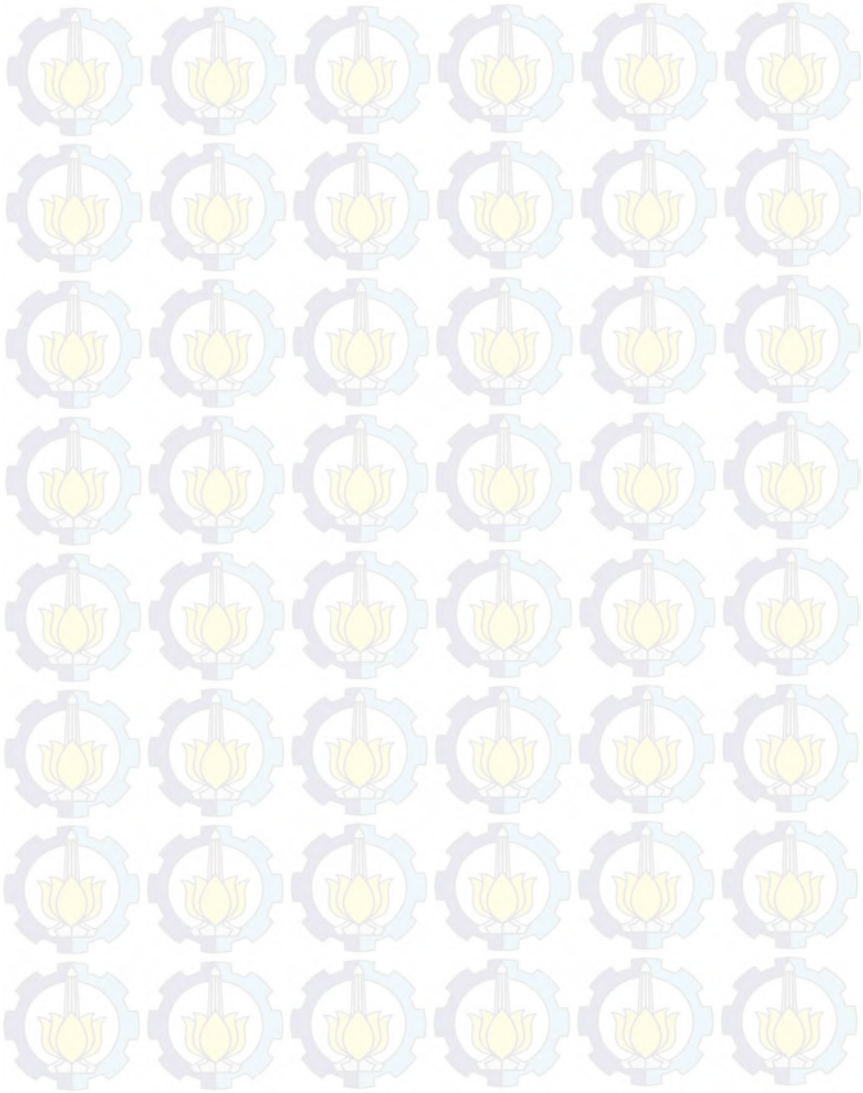
LEMBAR PENGESAHAN .....	v
Abstrak .....	vii
Abstract .....	ix
KATA PENGANTAR .....	xi
DAFTAR ISI .....	xiii
DAFTAR GAMBAR .....	xv
DAFTAR TABEL .....	xvii
BAB 1 PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	4
1.3 Batasan Masalah .....	5
1.4 Tujuan .....	5
1.5 Metodologi .....	5
1.6 Sistematika Penulisan .....	7
BAB 2 DASAR TEORI .....	9
2.1 Pengolahan Citra Digital .....	9
2.1.1 Citra .....	10
2.1.2 Citra Digital .....	10
2.1.3 Citra Biner .....	12
2.1.4 Citra Berwarna .....	13
2.1.5 Citra <i>Grayscale</i> .....	14
2.2 Gradien Gambar .....	16
2.3 Deteksi Tepi Citra .....	19
2.4 <i>Ant Colony</i> .....	26
2.4.1 <i>Ant Systems (AS)</i> .....	26
2.4.2 <i>Algoritma Ant Colony Optimization (ACO)</i> .....	32
2.5 Ketetangaan Piksel .....	40
2.6 <i>Thresholding</i> .....	41
2.6.1 <i>Otsu Thresholding</i> .....	43
2.7 <i>Algoritma Ant-inspired</i> .....	45
2.8 Perhitungan Akurasi .....	50
BAB 3 PERANCANGAN PERANGKAT LUNAK .....	51
3.1 Perancangan Data .....	51

3.1.1	Data Masukan .....	51
3.1.2	Data Proses .....	52
3.1.3	Data Keluaran .....	53
3.2	Perancangan Proses .....	54
3.2.1	Desain Secara Umum .....	54
3.2.2	Tahap Praproses.....	56
3.2.3	Tahap Proses.....	56
3.3	Perancangan Antarmuka .....	62
BAB 4 IMPLEMENTASI .....		65
4.1	Lingkungan Implementasi .....	65
4.2	Fungsi Utama.....	66
4.3	Tahap Deteksi Tepi.....	67
4.4	Tahap Menghitung Akurasi .....	70
BAB 5 UJI COBA DAN EVALUASI .....		73
5.1	Lingkungan Uji Coba .....	73
5.2	Data Uji Coba .....	74
5.3	Skenario Uji Coba.....	79
5.3.1	Perbandingan Citra Hasil dengan Nilai <i>Threshold</i> yang Berbeda-Beda .....	79
5.3.2	Perbandingan Citra Hasil Menggunakan Jumlah Iterasi yang Berbeda-Beda.....	84
5.3.3	Perbandingan Hasil Citra Menggunakan Nilai <i>Alpha</i> dan <i>Beta</i> yang Berbeda .....	89
5.3.4	Perbandingan Akurasi Citra Hasil dengan Algoritma Deteksi Tepi Lain .....	94
5.4	Evaluasi.....	95
BAB 6 KESIMPULAN DAN SARAN .....		97
6.1	Kesimpulan.....	97
6.2	Saran.....	97
DAFTAR PUSTAKA.....		99
LAMPIRAN HASIL UJI COBA.....		101
A. Citra-Citra Masukan dan Keluaran.....		101
BIODATA PENULIS.....		145

## DAFTAR TABEL

Tabel 3.1 Data Masukan .....	52
Tabel 3.2 Data Proses.....	52
Tabel 3.3 Data Keluaran .....	54
Tabel 4.1 Lingkungan Implementasi Perangkat Lunak.....	65
Tabel 5.1 Lingkungan Uji Coba.....	73
Tabel 5.2 Data Citra Uji Coba.....	74
Tabel 5.3 Akurasi Uji Coba <i>Threshold</i> 0,1 .....	82
Tabel 5.4 Akurasi Uji Coba <i>Threshold</i> 0,3 .....	83
Tabel 5.5 Akurasi Uji Coba Iterasi 50 .....	87
Tabel 5.6 Akurasi Uji Coba Iterasi 100.....	88
Tabel 5.7 Akurasi Uji Coba Iterasi 500.....	88
Tabel 5.8 Akurasi Uji Coba <i>Alpha</i> 0,5 dan <i>Beta</i> 0,5 .....	92
Tabel 5.9 Akurasi Uji Coba <i>Alpha</i> 1 dan <i>Beta</i> 0 .....	93
Tabel 5.10 Perbandingan Akurasi Citra Hasil dengan Metode <i>Canny</i> .....	94
Tabel A.1 Hasil Uji Coba Menggunakan <i>Threshold</i> 0,3 .....	101
Tabel A.2 Hasil Uji Coba Menggunakan <i>Threshold</i> 0,1 .....	107
Tabel A.3 Hasil Uji Coba Menggunakan 50 Iterasi .....	113
Tabel A.4 Hasil Uji Coba Menggunakan 100 Iterasi .....	118
Tabel A.5 Hasil Uji Coba Menggunakan 500 Iterasi .....	123
Tabel A.6 Hasil Uji Coba Menggunakan <i>Alpha</i> 0,5 dan <i>Beta</i> 0,5.....	128
Tabel A.7 Hasil Uji Coba Menggunakan <i>Alpha</i> 1 dan <i>Beta</i> 0 .....	134
Tabel A.8 Hasil Uji Coba Menggunakan Metode <i>Canny</i> ....	139

*[Halaman ini sengaja dikosongkan]*



## DAFTAR GAMBAR

Gambar 2.1 Representasi Citra Digital .....	11
Gambar 2.2 Citra Berwarna .....	14
Gambar 2.3 Citra Keabuan.....	15
Gambar 2.4 Arah Gradien Gambar .....	16
Gambar 2.5 (a) Citra Asli; (b) Citra Gradien Arah X; (c) Citra Gradien Arah Y.....	17
Gambar 2.6 Proses Deteksi Tepi Citra .....	19
Gambar 2.7 (a) Citra Asli; (b) Citra Hasil Deteksi Tepi Menggunakan Metode <i>Canny</i> .....	21
Gambar 2.8 (a) Citra Asli; (b) Citra Keabuan; (c) Citra Hasil Deteksi Tepi Metode <i>Canny</i> ; (d) <i>Prewitt</i> ; (e) <i>Sobel</i> .....	24
Gambar 2.9 Koloni Semut.....	27
Gambar 2.10 (a) Kelompok Semut Mencari Makan; (b) & (c) Semut Meninggalkan Jejak Feromon; (d) Semut Melewati Jalan Bawah yang Memiliki Lebih Banyak Feromon .....	29
Gambar 2.11 (a) Semut Menemukan Sumber Makanan; (b) Semut Berjalan Melewati Dua Kemungkinan Jalur; (c) Semut Mengambil Rute Terpendek.....	33
Gambar 2.12 Alur Algoritma ACO.....	34
Gambar 2.13 Ketetanggaan Antar Piksel.....	40
Gambar 2.14 Partisi Histogram dalam <i>Thresholding</i> Adaptif untuk Memperoleh Nilai <i>Threshold</i> .....	42
Gambar 2.15 (a) Citra Asli; (b) Citra Hasil <i>Otsu Thresholding</i> .....	44
Gambar 2.16 Tahapan Ekstraksi Tepi Gambar Menggunakan Algoritma <i>Ant-inspired</i> .....	48
Gambar 2.17 Algoritma <i>Ant-inspired</i> .....	49
Gambar 3.1 Citra Masukan .....	52
Gambar 3.2 Alur Proses Secara Umum .....	55
Gambar 3.3 Citra Hasil Praproses.....	56
Gambar 3.4 Diagram Alir Tahap Praproses .....	57
Gambar 3.5 (a) Gradien Gambar dengan Arah X; (b) Gradien Gambar dengan Arah Y.....	58

Gambar 3.6 Gambar Hasil Proses Perhitungan Global Stimulus.....	59
Gambar 3.7 Citra Hasil Deteksi Tepi .....	60
Gambar 3.8 Diagram Alir Tahap Proses Deteksi Tepi .....	61
Gambar 3.9 Rancangan Antarmuka Perangkat Lunak Implementasi Deteksi Tepi Gambar Menggunakan Algoritma <i>Ant-inspired</i> .....	62
Gambar 4.1 Kode Sumber Mengambil Data Citra Masukan. 66	
Gambar 4.2 Kode Sumber Bagian-Bagian yang Akan Diproses Ketika Tombol <i>Pop Up</i> Menu Dipilih .....	67
Gambar 4.3 Kode Gradien Gambar .....	68
Gambar 4.4 Kode Deteksi Tepi .....	70
Gambar 4.5 Kode Menghitung Skor dan Akurasi .....	71
Gambar 5.1 Citra hawk.jpg Menggunakan <i>Threshold</i> 0,1.....	80
Gambar 5.2 Citra hawk.jpg Menggunakan <i>Threshold</i> 0,3.....	81
Gambar 5.3 Citra butterfly.jpg Menggunakan <i>Threshold</i> 0,1	81
Gambar 5.4 Citra butterfly.jpg Menggunakan <i>Threshold</i> 0,3	82
Gambar 5.5 Citra church.jpg Menggunakan 50 Iterasi.....	85
Gambar 5.6 Citra church.jpg Menggunakan 100 Iterasi.....	85
Gambar 5.7 Citra church.jpg Menggunakan 500 Iterasi.....	85
Gambar 5.8 Citra bird.jpg Menggunakan 50 Iterasi .....	86
Gambar 5.9 Citra bird.jpg Menggunakan 100 Iterasi .....	86
Gambar 5.10 Citra bird.jpg Menggunakan 500 Iterasi .....	87
Gambar 5.11 Citra apple.jpg Menggunakan <i>Alpha</i> 0,5 dan <i>Beta</i> 0,5.....	90
Gambar 5.12 Citra apple.jpg Menggunakan <i>Alpha</i> 1 dan <i>Beta</i> 0.....	90
Gambar 5.13 Citra twoblack.jpg Menggunakan <i>Alpha</i> 0,5 dan <i>Beta</i> 0,5.....	91
Gambar 5.14 Citra twoblack.jpg Menggunakan <i>Alpha</i> 1 dan <i>Beta</i> 0.....	92

# BAB 1

## PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang yang mendukung penyusunan tugas akhir. Setelah penjelasan mengenai latar belakang, dalam sub bab ini akan dijelaskan juga mengenai rumusan masalah, batasan masalah, tujuan, metodologi, dan sistematika penulisan dari tugas akhir ini.

### 1.1 Latar Belakang

Pada zaman ini teknologi komputer telah berkembang pesat. Pada mulanya, teknologi komputer hanya berkembang pada proses pengolahan data saja. Namun seiring perkembangan zaman dan teknologi di bidang lain, maka teknologi komputer saat ini juga ikut melakukan perkembangan. Salah satu yang sedang pesat berkembang adalah teknologi komputer dalam bidang multimedia seperti, pengolahan gambar, audio, video, dan sebagainya.

Sejak kemunculan pertama gambar digital tahun 1920, terutama munculnya perangkat lunak dalam teknologi komputer dalam memproses gambar digital tahun 1960, pemrosesan sebuah gambar dalam komputer menjadi lebih mudah, sehingga pengolahan gambar dan visi komputer telah menjadi unsur yang sangat penting dalam dunia teknologi saat ini. Dengan berkembangnya teknologi, semakin berkembang pula metode-metode baru yang dapat digunakan pada permasalahan yang terjadi. Pengolahan citra sering kali menggunakan deteksi tepi untuk tujuan tertentu. Karena deteksi tepi merupakan langkah pertama untuk mengambil informasi penting dari sebuah citra.

Deteksi tepi merupakan salah satu proses *preprocessing* yang sering dibutuhkan pada analisis citra.



Proses tersebut bertujuan untuk meningkatkan penampakan garis pada citra. Untuk mendapatkan citra yang baik sesuai yang diinginkan biasanya citra diproses terlebih dahulu dengan perbaikan kualitas citra. Perbaikan kualitas citra dilakukan karena citra yang menjadi objek memiliki kualitas yang buruk, misalnya citra mengalami derau *gaussian*, *salt and pepper noise*, dan *blurring*. Deteksi tepi gambar merupakan salah satu permasalahan yang penting dalam pemrosesan citra digital. Gambar yang sudah memiliki tepi dapat diproses lebih lanjut untuk diketahui jenis gambar tergolong dalam suatu kelompok tertentu. Pendeteksian tepi pada citra digital bertujuan untuk mengenali suatu pola yang terdapat pada citra itu sendiri. Dengan dikenalnya pola pada sebuah citra akan mudah diperoleh informasi-informasi pada sebuah citra.

Pemilihan metode deteksi tepi yang tepat dan efisien sangat menentukan keberhasilan dari sistem pengenalan secara keseluruhan. Berbagai teknik dan metode telah digunakan mulai dari metode yang sederhana hingga metode yang rumit seperti *scale invariant feature transform* [1].

Tepi gambar selalu memiliki informasi penting mengenai objek gambar. Pada awalnya terdapat beberapa metode deteksi tepi tradisional seperti *sobel* dan *canny* yang mengekstrak tepi dengan menggunakan template tertentu dengan kombinasi fungsi *smoothing* (penghalusan) gambar. Deteksi tepi metode tradisional ini kadangkala belum maksimal saat digunakan untuk mendeteksi tepi citra, seperti garis-garis tepi yang dihasilkan masih terlihat tidak menyambung satu sama lain.

Beberapa metode baru pun dikembangkan untuk meningkatkan hasil deteksi tepi citra, diantaranya metode *ghita and whelan* yang menggunakan *mask* untuk

mendapatkan arah dari ujung ahir garis agar dapat mengestimasi *cost* dari garis tepi yang saling berhubungan [1]. *Cost* dan arah dari masing-masing garis tepi menentukan garis mana yang bisa dipilih sebagai tepi. Metode ini sebenarnya adalah metode yang sederhana, tetapi kekurangannya adalah metode ini mengembalikan nilai struktur tepi yang tidak lengkap.

Kemudian, beberapa pendekatan yang lain seperti transformasi *hough* pula dikembangkan pada tepi gambar. Metode ini melakukan pendekatan ekstraksi bentuk spesifik gambar untuk menyambungkan tepi yang tidak saling berhubungan. Akan tetapi permasalahannya terletak pada suatu hal, dimana tepi objek atau gambar tidak selalu memiliki bentuk yang tetap [1].

Tepi gambar yang rusak atau hancur sangat sulit untuk disambungkan kembali. Penyatuan tepi harus melibatkan paling tidak dua titik ujung agar dapat menghasilkan tepi yang saling berkaitan. Informasi lokal dari citra asli dapat dianalisis lebih dalam untuk membantu memperbaiki tepi yang rusak.

Suatu metode baru yang berdasarkan metodologi *swarm* pun dikembangkan. Salah satu metodologi *swarm* yang terkenal adalah *ant colony* [1]. Metode ini mendeteksi tepi gambar berdasarkan pengembangan pada metode *swarm ant colony*.

*Ant colony* adalah metode *heuristic* yang meniru perilaku semut dalam mengatasi permasalahan optimisasi diskrit. Semut-semut yang diciptakan berperilaku layaknya agen intelijen yang memiliki kemampuan untuk mengingat dan kemampuan untuk melihat. Semut ini mencari *path* yang optimal dengan melakukan iterasi pada setiap proses yang dilakukan. Karena posisi awal dari semut didefinisikan

terlebih dahulu, semut menggunakan aturan transisi posisi untuk menentukan solusi yang optimal. Jalan-jalan yang dilalui masing-masing semut bisa berbeda pada setiap iterasinya. Kondisi ini berbeda dengan metode komputasi lainnya seperti metode algoritma genetika.

Metode *ant colony* yang telah ada merupakan metode yang biasa digunakan untuk permasalahan mencari jarak terdekat seperti pada masalah *travelling salesman problem* (TSP) [2]. Algoritma *ant colony* merupakan salah satu metode metaheuristik yang menerapkan semut sebagai agen dan melakukan *update* terhadap feromon untuk dapat melakukan proses pencarian solusi yang efektif dan efisien.

Dalam tugas akhir ini penulis mengimplementasikan deteksi tepi citra dengan menggunakan algoritma *ant-inspired*. Berbeda dengan metode deteksi tepi lain, algoritma *ant-inspired* ini memanfaatkan tingkah laku semut dalam mencari makan dengan meninggalkan jejak berupa feromon untuk melakukan deteksi tepi pada citra.

## 1.2 Rumusan Masalah

Permasalahan yang diangkat dalam penyusunan tugas akhir ini adalah:

1. Bagaimana konsep algoritma *ant-inspired* dalam melakukan ekstraksi tepi gambar?
2. Bagaimana penggunaan algoritma *ant-inspired* pada gambar?
3. Bagaimana mengimplementasikan ekstraksi tepi gambar dengan menggunakan algoritma *ant-inspired* pada MATLAB?

### 1.3 Batasan Masalah

Permasalahan dalam tugas akhir ini dibatasi ruang lingkup pembahasannya sebagai berikut:

1. Sistem perangkat lunak dibangun dengan menggunakan perangkat lunak MATLAB R2008a.
2. Citra masukan yang digunakan merupakan citra berwarna dengan berbagai ukuran piksel.
3. Perangkat lunak ini merupakan implementasi dari algoritma *ant-inspired* untuk deteksi tepi pada citra berwarna yang diubah ke format citra *grayscale* (keabuan).

### 1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Mengetahui konsep algoritma *ant-inspired* dalam melakukan ekstraksi tepi gambar.
2. Mengimplementasikan algoritma *ant-inspired* untuk ekstraksi tepi gambar.
3. Mengimplementasikan algoritma *ant-inspired* untuk melakukan proses deteksi tepi pada citra yang sudah diubah ke dalam bentuk citra *grayscale* (keabuan).

### 1.5 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

- A. Studi literatur  
Pada tahap ini dilakukan pencarian informasi dan studi literatur yang diperlukan untuk pengumpulan data dan desain sistem yang akan dibuat. Informasi didapatkan dari buku dan materi-materi lain yang berhubungan dengan

algoritma *ant-inspired* yang didapat dari internet maupun buku acuan.

Literatur yang dipelajari adalah mengenai metode yang akan digunakan, diantaranya:

- a. Algoritma *ant-inspired* dan algoritma lainnya yang mendukung.
- b. Cara mendeteksi tepi citra menggunakan algoritma *ant-inspired* yang didasari oleh perilaku koloni semut dalam mencari makanan.
- c. Evaluasi hasil implementasi algoritma *ant-inspired*.

B. Pembuktian rumus secara analitis

Pada tahap ini dilakukan penerapan dasar teori dan algoritma yang telah dipahami untuk menyelesaikan proses deteksi tepi citra.

C. Perancangan perangkat lunak

Tahap ini adalah saat dimana sistem yang akan dibuat dirancang sesuai dengan hasil analisis dalam studi literatur. Sistem yang akan dibuat harus sesuai dengan langkah-langkah pengerjaan hasil analisis studi literatur.

D. Implementasi dan pembuatan sistem

Tahap ini adalah tahap dimana sistem yang telah dirancang pada tahap sebelum dibangun dan diimplementasikan menjadi sebuah perangkat lunak. Pada tahap ini dilakukan implementasi algoritma *ant-inspired* dari rancangan yang telah dibuat pada tahap sebelumnya dengan menggunakan MATLAB R2008a. Selain itu, proses pembangunan sistem dilanjutkan dengan pembuatan antarmuka pengguna. Hal ini dimaksudkan agar perangkat lunak terlihat menarik dan pengguna dapat menggunakannya dengan mudah.

E. Uji coba dan evaluasi

Pada tahap ini dilakukan uji coba dengan menggunakan bermacam citra masukan yang bervariasi untuk mencoba jalannya perangkat lunak telah sesuai dengan rancangan dan desain implementasi yang telah dibuat. Bagian ini bertujuan untuk mencari kesalahan-kesalahan program yang mungkin terjadi untuk selanjutnya dapat dilakukan penyempurnaan. Selain itu, pengujian ini juga bertujuan untuk mengamati kinerja dari perangkat lunak tersebut sehingga hambatan dan kesalahan yang mengganggu jalannya sistem dapat dikenali lalu diperbaiki.

F. Penyusunan laporan tugas akhir

Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan tugas akhir.

## 1.6 Sistematika Penulisan

Laporan tugas akhir ini dibagi menjadi 6 bab, yang terdiri dari:

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi yang digunakan, serta sistematika penulisan dalam penyusunan tugas akhir ini.

2. Bab II Dasar Teori

Bab ini membahas tentang teori penunjang yang berhubungan dengan pokok bahasan yang mendasari pembuatan tugas akhir. Bab ini terdapat penjelasan tentang pengolahan citra digital, citra, citra digital, citra biner, citra berwarna, citra keabuan, gradien gambar, deteksi tepi citra, *ant systems*, algoritma *ant colony optimization*, ketetanggaan piksel, *thresholding*, dan diakhiri dengan penjelasan tentang langkah-langkah untuk

melakukan deteksi tepi citra dengan menggunakan algoritma *ant-inspired*.

3. Bab III Perancangan Perangkat Lunak

Bab ini menjelaskan desain dari perangkat lunak yang akan dibuat meliputi perancangan fungsionalitas dan perancangan antarmuka perangkat lunak

4. Bab IV Implementasi Perangkat Lunak

Bab ini membahas implementasi dari rancangan perangkat lunak yang disertai dengan potongan kode sumber pokok pada perangkat lunak.

5. Bab V Uji Coba dan Evaluasi

Bab ini membahas uji coba dari perangkat lunak yang dibuat dengan melihat keluaran yang dihasilkan oleh perangkat lunak dan evaluasi untuk mengetahui kemampuan perangkat lunak.

6. Bab VI Penutup

Bab ini berisi kesimpulan dari hasil uji coba yang dilakukan serta saran untuk pengembangan perangkat lunak selanjutnya.

## **BAB 2**

### **DASAR TEORI**

Pada bab ini akan dibahas mengenai dasar teori yang mendukung penyusunan tugas akhir. Bab ini diawali dengan penjelasan tentang pengolahan citra digital, citra, citra digital, citra biner, citra berwarna, citra keabuan, gradien gambar, dan deteksi tepi citra. Selanjutnya akan dijelaskan mengenai algoritma *ant colony* konvensional yang digunakan sebagai algoritma dasar dalam proses deteksi tepi, ketetanggaan piksel, *thresholding*, dan diakhiri dengan penjelasan mengenai algoritma *ant-inspired* yang digunakan untuk mendeteksi tepi gambar pada tugas akhir ini. Materi-materi tersebut masing-masing akan dijelaskan dalam sub bab tersendiri.

#### **2.1 Pengolahan Citra Digital**

Pengolahan citra digital merupakan proses yang bertujuan untuk memanipulasi dan menganalisis citra dengan bantuan komputer [3]. Pengolahan citra digital dapat dikelompokkan dalam dua jenis kegiatan:

1. Memperbaiki kualitas suatu gambar, sehingga dapat lebih mudah diinterpretasi oleh mata manusia.
2. Mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis.

Bidang perangkat lunak kedua sangat erat hubungannya dengan ilmu pengenalan pola (*pattern recognition*) yang umumnya bertujuan mengenali suatu objek dengan cara mengekstrak informasi penting yang terdapat pada suatu citra. Bila pengenalan pola dihubungkan dengan pengolahan citra, diharapkan akan terbentuk suatu sistem yang dapat memproses citra masukan sehingga citra tersebut dapat dikenali polanya. Proses ini disebut pengenalan citra atau *image recognition*.



Proses pengenalan citra ini sering diterapkan dalam kehidupan sehari-hari. Pengolahan citra dan pengenalan pola menjadi bagian dari proses pengenalan citra. Kedua perangkat lunak ini akan saling melengkapi untuk mendapatkan ciri khas dari suatu citra yang hendak dikenali.

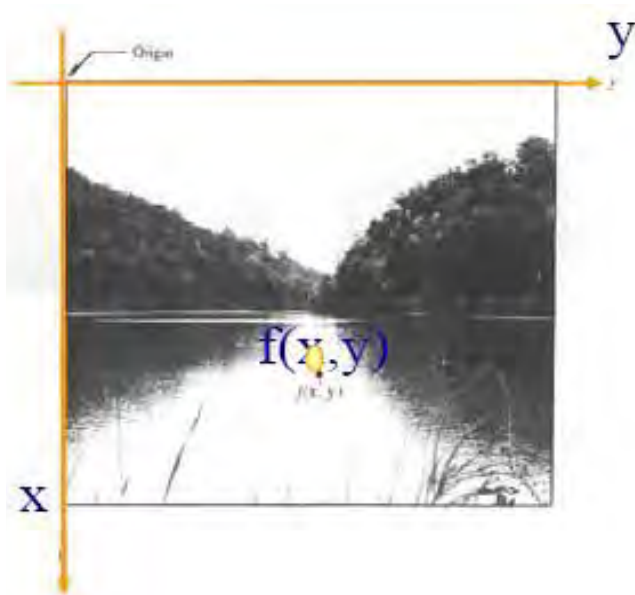
### **2.1.1 Citra**

Secara harfiah, dalam arti yang paling mendasar, pengertian citra (*image*) adalah gambar pada bidang dua dimensi [4]. Kemudian jika ditinjau dari sudut pandang matematis, citra merupakan fungsi kontinu dari intensitas cahaya pada bidang dua dimensi. Proses pembentukan suatu citra bisa direpresentasikan melalui objek yang diterangi sumber cahaya, kemudian objek memantulkan kembali sebagian berkas cahaya tersebut, sehingga pantulan cahaya ini dapat ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, *scanner*, dan alat-alat optik yang lainnya. Oleh karena itu dari proses tersebut terbentuk bayangan objek yang disebut citra yang dapat terekam [4].

Citra memiliki karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Maksudnya, sebuah gambar dapat memberikan informasi yang lebih banyak daripada informasi yang disajikan dalam bentuk teks.

### **2.1.2 Citra Digital**

Citra digital dapat didefinisikan sebagai fungsi dua variabel,  $f(x,y)$ , dimana  $x$  dan  $y$  adalah koordinat spasial dan nilai  $f(x,y)$  adalah intensitas citra pada koordinat tersebut [5], hal tersebut diilustrasikan pada Gambar 2.1.



**Gambar 2.1 Representasi Citra Digital**

Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*red, green, blue*). Sebuah citra diubah ke bentuk digital agar dapat disimpan dalam memori komputer atau media lain.

Proses mengubah citra ke bentuk digital bisa dilakukan dengan beberapa perangkat, misalnya *scanner*, kamera digital, dan *handycam*. Ketika sebuah citra sudah diubah ke dalam bentuk digital (selanjutnya disebut citra digital), bermacam-macam proses pengolahan citra dapat diperlakukan terhadap citra tersebut.

### 2.1.3 Citra Biner

Citra biner (*binary image*) adalah citra yang setiap pikselnya hanya memiliki dua kemungkinan derajat keabuan yakni 0 dan 1. Proses pembineran dilakukan dengan membulatkan ke atas atau ke bawah untuk setiap nilai keabuan dari piksel yang berada di atas atau bawah harga ambang. Metode untuk menentukan besarnya harga ambang disebut *thresholding*.

Citra biner diperoleh melalui proses pemisahan piksel-piksel berdasarkan derajat keabuan yang dimilikinya. Berdasarkan Persamaan 2.1, piksel yang memiliki derajat keabuan lebih kecil dari nilai batas yang ditentukan (*threshold*) akan diberikan nilai 0, sementara piksel yang memiliki derajat keabuan yang lebih besar dari batas akan diubah menjadi bernilai 1 [6]. Dimana  $f(x,y)$  merupakan citra biner dan  $T$  merupakan nilai ambang.

$$f(x, y) = \begin{cases} 0, & f(x, y) < T \\ 1, & f(x, y) \geq T \end{cases} \quad (2.1)$$

Citra biner disebut juga dengan citra *bi-level*. Citra ini adalah citra digital yang hanya mempunyai dua kemungkinan nilai pada piksel-pikselnya yaitu, nilai hitam atau putih. Citra biner biasanya dihasilkan melalui proses segmentasi ataupun proses *thresholding*, atau dihasilkan beberapa alat-alat *input* atau *output* yang hanya mendukung citra biner seperti printer, laser, dan mesin faks.

Jika nilai ambang  $T$  adalah 150, maka piksel-piksel dengan nilai intensitas di bawah 150 diubah menjadi hitam (nilai intensitas = 0), sedangkan piksel-piksel yang nilai intensitasnya di atas 150 diubah menjadi putih (nilai intensitas = 1).

#### 2.1.4 Citra Berwarna

Citra adalah suatu representasi, kemiripan, atau imitasi dari suatu objek yang terletak pada bidang dua dimensi yang dihasilkan dari gambar analog kontinu menjadi gambar diskrit. Citra terbagi dua, yaitu ada citra yang bersifat analog dan ada citra yang bersifat digital.

Citra dapat didefinisikan sebagai fungsi dua variabel,  $f(x,y)$ , dimana  $x$  dan  $y$  adalah koordinat spasial sedangkan nilai  $f(x,y)$  adalah intensitas citra pada koordinat tersebut. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*red, green, blue*).

Citra memiliki karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Maksudnya, sebuah gambar dapat memberikan informasi yang lebih banyak daripada informasi yang disajikan dalam bentuk teks. Citra dibagi menjadi empat jenis yaitu:

1. Citra diam, merupakan citra tunggal yang tidak bergerak.
2. Citra bergerak, merupakan rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan dimata sebagai gambar yang bergerak.
3. Citra analog, merupakan citra yang bersifat kontinu seperti gambar pada monitor televisi, foto sinar X, dan hasil CT *scan*. Citra analog dibagi menjadi N baris dan M kolom sehingga menjadi gambar diskrit [7].
4. Citra digital, merupakan citra yang dapat diolah oleh komputer.



**Gambar 2.2 Citra Berwarna**

Citra berwarna adalah citra yang nilai pikselnya merepresentasikan warna tertentu. Banyaknya warna yang mungkin digunakan bergantung kepada kedalaman piksel citra yang bersangkutan. Citra berwarna direpresentasikan dalam beberapa kanal (*channel*) yang menyatakan komponen-komponen warna penyusunnya. Banyaknya kanal yang digunakan bergantung pada model warna yang digunakan pada citra tersebut. Gambar 2.2 memperlihatkan contoh dari citra berwarna.

### **2.1.5 Citra *Grayscale***

Citra keabuan atau citra beraras keabuan adalah citra yang hanya menggunakan warna pada tingkatan warna abu-abu. Warna abu-abu adalah satu-satunya warna pada ruang berwarna 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. Gambar 2.3 memperlihatkan contoh dari citra keabuan.

Citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara hitam sebagai warna minimal dan putih sebagai warna maksimal, sehingga warna di antara keduanya adalah abu-abu. Namun dalam prakteknya warna yang dipakai tidak terbatas pada warna abu-abu, sebagai contoh dipilih warna minimalnya adalah putih dan warna maksimalnya adalah merah, maka semakin besar nilainya semakin besar pula intensitas warna merahnya.



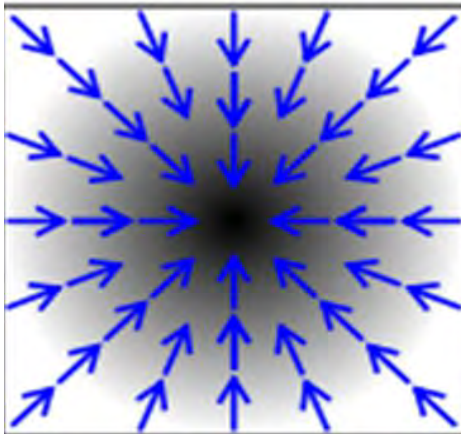
**Gambar 2.3 Citra Keabuan**

## 2.2 Gradien Gambar

Gradien gambar merupakan perubahan arah intensitas atau warna pada gambar. Sebuah gradien dapat digunakan untuk mengekstrak informasi gambar [8].

Dalam perangkat lunak grafis untuk mengedit gambar digital, istilah gradien digunakan untuk mencampurkan warna yang dianggap sebagai gradasi dari nilai yang rendah ke nilai yang tinggi. Gambar 2.4 merupakan gambar yang berubah dari piksel putih menuju piksel hitam.

Secara matematis, fungsi gradien dengan dua variabel pada setiap titik gambar adalah vektor 2D dengan komponen yang diperoleh dari hasil turunan gambar pada arah horizontal dan vertikal. Pada setiap gambar, vektor gradien menuju ke suatu arah dengan kemungkinan peningkatan intensitas yang terbesar, dan panjang gradien vektornya sesuai dengan tingkat perubahan ke arah tersebut.

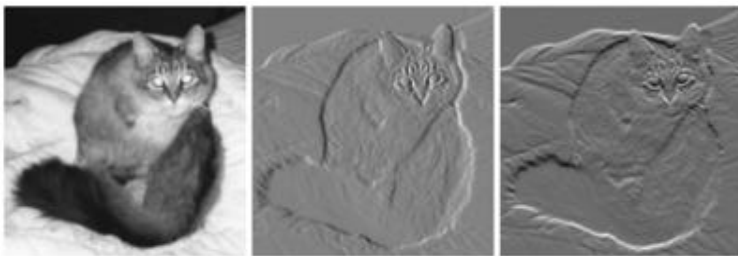


**Gambar 2.4 Arah Gradien Gambar**

Metode *sobel* merupakan pendekatan yang lumayan akurat untuk mencari gradien gambar, akan tetapi operator ini masih tidak efisien jika digunakan pada beberapa perangkat lunak. Lebih tepatnya, operator ini menggunakan nilai intensitas yang hanya menggunakan *mask* berukuran  $3 \times 3$  pada setiap gambar untuk memperkiraan gradien gambar yang sesuai.

Gradien gambar adalah hal yang sangat fundamental dalam pengolahan citra digital. Misalnya metode *canny* menggunakan gradien gambar untuk deteksi tepi. Pada metode ini, setelah nilai gradien gambar dihitung, piksel dengan nilai gradien terbesar dalam arah gradien menjadi tepi citra, dan tepinya dapat ditelusuri dalam arah tegak lurus yang sesuai dengan arah gradien.

Gradien gambar juga sering digunakan dalam peta dan representasi visual lainnya yang dapat menyediakan informasi tambahan. Alat *geographic information system* menggunakan alat *color progressions* untuk menunjukkan elevasi dan kepadatan penduduk. Gambar 2.5 (a) merupakan citra asli. Gambar 2.5(b) merupakan gradien gambar dalam arah  $x$ . Gambar 2.5(c) merupakan gradien gambar dalam arah  $y$ .



(a) (b) (c)  
**Gambar 2.5 (a) Citra Asli; (b) Citra Gradien Arah  $X$ ; (c) Citra Gradien Arah  $Y$**



Gradien gambar dapat digunakan untuk mengekstrak informasi dari gambar. Setiap piksel pada gradien gambar mengukur perubahan intensitas titik yang sama pada citra asli dalam arah tertentu. Untuk mendapatkan berbagai macam arah, gradien dalam arah  $x$  dan  $y$  harus dihitung.

Gradien gambar juga dapat digunakan untuk mencocokkan tekstur. Pencahayaan atau properti kamera yang berbeda dapat menyebabkan dua gambar yang sama memiliki nilai-nilai piksel yang sangat berbeda. Hal ini dapat menyebabkan algoritma yang digunakan gagal untuk mencocokkan fitur yang sangat mirip atau identik pada gambar tersebut. Salah satu cara untuk mengatasi ini adalah dengan menghitung tekstur berdasarkan gradien gambar dari citra asli.

Gradien dari citra  $f(x,y)$  pada lokasi  $(x,y)$  didefinisikan sebagai vektor pada Persamaan 2.2 [6].

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (2.2)$$

Dimana  $G_x$  adalah gradien terhadap sumbu  $x$  dan  $G_y$  adalah gradien terhadap sumbu  $y$ . Jarak vektor pada gradien didefinisikan sesuai dengan Persamaan 2.3.

$$mag(\nabla f) = \sqrt{G_x^2 + G_y^2} \quad (2.3)$$

Arah gradien pada  $(x,y)$  sesuai dengan Persamaan 2.4.

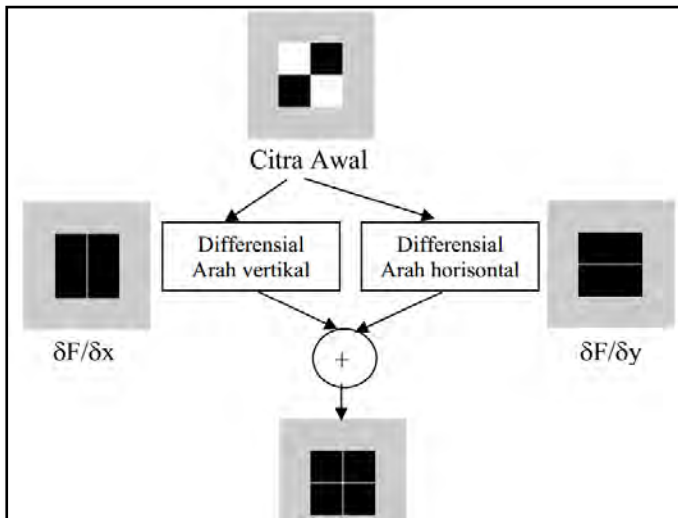
$$\alpha(x, y) = \tan^{-1} \frac{G_x}{G_y} \quad (2.4)$$

Dimana  $\alpha(x,y)$  menyatakan sudut vektor  $\nabla f$  di  $(x,y)$  terhadap sumbu  $x$ . Pendekatan lain untuk memperkirakan jarak vektor dapat dilihat pada Persamaan 2.5.

$$\nabla f \approx |G_x| + |G_y| \quad (2.5)$$

### 2.3 Deteksi Tepi Citra

Tepi adalah batas antara dua daerah dengan nilai *gray-level* yang relatif berbeda atau dengan kata lain tepi merupakan tempat-tempat yang memiliki perubahan intensitas yang besar dalam jarak yang pendek [9].



**Gambar 2.6 Proses Deteksi Tepi Citra**

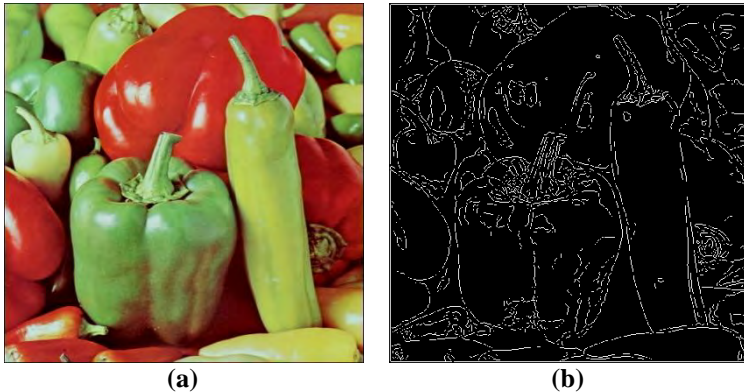
Suatu objek yang berada dalam bidang citra dan tidak bersinggungan dengan batas bidang citra memiliki arti bahwa objek tersebut dikelilingi daerah yang bukan objek, yaitu latar belakang. Pertemuan antara bagian objek dan bagian latar belakang disebut tepi objek. Demikian juga bila dua buah atau lebih objek saling tumpang tindih, bila intensitas mereka tidak sama, akan meninggalkan jejak tepi sehingga diketahui objek yang satu berada di depan objek yang lain atau sebaliknya [10]. Hal ini penting untuk mengembalikan atau merekonstruksi bentuk yang seharusnya dari objek yang berada di belakang objek lainnya, atau memisahkan objek yang tumpang tindih sehingga mereka dapat dianalisis secara individu. Proses ini disebut dengan deteksi tepi.

Deteksi tepi (*edge detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari objek-objek citra, tujuannya adalah [11]:

- a. Untuk menandai bagian yang menjadi detail citra.
- b. Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena kesalahan atau adanya efek dari proses akuisisi citra.

Suatu titik  $(x,y)$  dikatakan sebagai tepi dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar 2.6 menggambarkan bagaimana tepi suatu gambar diperoleh.

Deteksi tepi merupakan proses untuk menampilkan tepi-tepi dari objek-objek citra yang sebelumnya dilakukan proses pengubahan citra aras keabuan menjadi citra hitam putih. Tepi-tepi ini akan menandai bagian detail citra. Tepi-tepi pada gambar tersebut terletak pada titik-titik yang memiliki perbedaan tinggi, sehingga didapatkan citra yang telah terpisah antar latar depan dengan latar belakangnya.



**Gambar 2.7 (a) Citra Asli; (b) Citra Hasil Deteksi Tepi Menggunakan Metode *Canny***

Gambar 2.7(a) menunjukkan contoh citra asli dan Gambar 2.7(b) menunjukkan contoh citra yang telah mengalami proses deteksi tepi menggunakan metode *canny*.

Deteksi tepi atau *edge detection* digunakan untuk menentukan lokasi titik-titik yang merupakan tepi objek. Secara umum, tepi suatu objek dalam citra dinyatakan sebagai titik yang nilai keabuannya berbeda cukup besar dengan titik yang ada di sebelahnya. Banyak metode yang dipakai dalam proses ini, misalnya operator *robert*, *prewitt*, *sobel*, dan *canny*. Macam-macam metode untuk proses deteksi tepi, antara lain:

#### 1. Metode *Robert*

Merupakan metode pendektasian tepi dengan mencari perbedaan pada arah horizontal dan perbedaan pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan perbedaan. Agar mendapatkan tepi-tepi yang lebih baik, maka konversi biner dilakukan dengan meratakan distribusi warna hitam dan putih atau dengan kata lain objek gambar yang akan digunakan untuk metode ini sebaiknya adalah gambar hitam putih. Metode *Robert*

adalah nama lain dari teknik 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. Kernel filter yang digunakan dalam metode *Robert* ini terdapat pada Persamaan 2.6.

$$H = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } V = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad (2.6)$$

Dimana  $H$  adalah kernel filter untuk arah horizontal pada citra dan  $V$  adalah kernel filter untuk arah vertikal pada citra.

## 2. Metode *Prewitt*

Metode ini adalah pengembangan metode *robert* dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi *laplacian* yang dikenal sebagai fungsi untuk membangkitkan HPF. Sehingga tepi-tepi yang dihasilkan lebih banyak dari metode *robert*. Kernel filter yang digunakan dalam metode *prewitt* ini terdapat pada Persamaan 2.7.

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } V = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.7)$$

Dimana  $H$  adalah kernel filter untuk arah horizontal pada citra dan  $V$  adalah kernel filter untuk arah vertikal pada citra.

## 3. Metode *Sobel*

Metode ini merupakan pengembangan metode *Robert* dengan menggunakan filter HPF yang diberi satu angka

nol penyangga. Metode ini mengambil prinsip dari fungsi *laplacian* dan *gaussian* yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak dibanding 2 metode sebelumnya. Kernel filter yang digunakan dalam metode ini terdapat pada Persamaan 2.8.

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } V = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (2.8)$$

Dimana  $H$  adalah kernel filter untuk arah horizontal pada citra dan  $V$  adalah kernel filter untuk arah vertikal pada citra.

#### 4. Metode *Canny*

Salah satu algoritma deteksi tepi modern adalah deteksi tepi dengan menggunakan metode *canny*. Deteksi tepi *canny* ditemukan oleh Marr dan Hildreth yang meneliti pemodelan persepsi visual manusia [12]. Ada beberapa kriteria pendeteksi tepian paling optimum yang dapat dipenuhi oleh algoritma *canny*:

##### a. Mendeteksi dengan baik

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 ketebalan tepi sesuai yang diinginkan.

- b. Melokalisasi dengan baik  
Dengan menggunakan metode *canny* dimungkinkan dihasilkan jarak yang minimum antara tepi yang dideteksi dengan tepi yang asli.



(a)



(b)



(c)



(d)



(e)

**Gambar 2.8 (a) Citra Asli; (b) Citra Keabuan; (c) Citra Hasil Deteksi Tepi Metode Canny; (d) Prewitt; (e) Sobel**

c. Respon yang jelas

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.

Gambar 2.8(a) menunjukkan citra asli yang diambil dari perangkat lunak MATLAB R2008a. Gambar 2.8(b) menunjukkan citra yang telah diubah menjadi citra keabuan, Gambar 2.8(c) menunjukkan hasil deteksi tepi citra menggunakan metode deteksi tepi *canny*. Gambar 2.8(d) menunjukkan hasil deteksi tepi citra menggunakan metode *prewitt* dan Gambar 2.8(e) menunjukkan hasil deteksi tepi citra menggunakan metode *sobel*. Terdapat beberapa teknik untuk mendeteksi tepi, yaitu:

1. Operator gradien pertama, contoh beberapa gradien pertama yang dapat digunakan untuk mendeteksi tepi di dalam citra, yaitu operator *sobel*, operator *prewitt*, operator *roberts*, dan operator *canny*.
2. Operator turunan kedua, operator ini juga disebut sebagai operator *laplace*. Operator *laplace* mendeteksi lokasi tepi khususnya pada citra tepi yang curam. Pada tepi yang curam, turunan keduanya mempunyai persilangan nol, yaitu titik di mana terdapat pergantian tanda nilai turunan kedua, sedangkan pada tepi yang landai tidak terdapat persilangan nol. Contohnya adalah operator *laplacian gaussian* dan operator *gaussian*.
3. Operator kompas, digunakan untuk mendeteksi semua tepi dari berbagai arah di dalam citra. Operator kompas yang dipakai untuk deteksi tepi menampilkan tepi dari delapan macam arah mata angin yaitu utara, timur laut, timur, tenggara, selatan, barat, barat daya, dan barat laut. Deteksi tepi dilakukan dengan mengkonvolusikan citra dengan



berbagai *mask* kompas, lalu dicari nilai kekuatan tepi (*magnitude*) yang terbesar dan arahnya. Operator kompas yang dipakai untuk deteksi tepi menampilkan tepi dari delapan macam arah mata angin, yaitu utara, timur laut, timur, tenggara, selatan, barat, barat daya, dan barat laut.

Dalam tugas akhir ini, deteksi tepi yang digunakan adalah deteksi tepi dengan menggunakan algoritma *ant-inspired*. Algoritma ini masuk ke dalam jenis metode *swarm intelligence* yang dapat digunakan untuk mendeteksi tepi dengan menggunakan teknik probabilitas. Algoritma semut biasa digunakan untuk mencari optimasi jarak terdekat pada sebuah permasalahan, namun kali ini dengan berbagai macam pengembangan algoritma ini dapat digunakan untuk mendeteksi tepi citra.

## **2.4 Ant Colony**

Dalam sub bab ini akan dijelaskan mengenai dua dasar teori, yaitu *Ant Systems* (AS) dan algoritma *ant colony pptimization* (ACO).

### **2.4.1 Ant Systems (AS)**

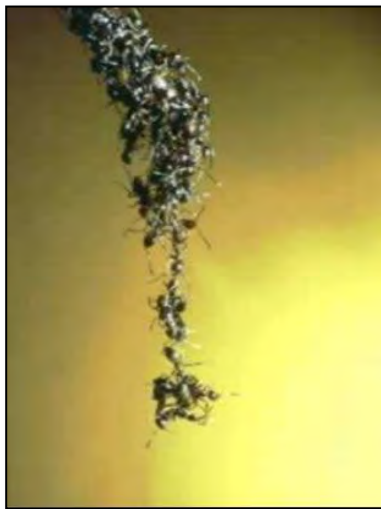
Pada kehidupan sebenarnya, semut-semut meninggalkan sarang untuk mencari makanan dan harus mencari kembali sarang mereka. Misalkan ada segerombolan semut yang mencari makanan, maka semut yang berada di depan harus memilih lintasan tertentu untuk dilewati.

Pada saat semut pertama berjalan, semut tersebut meninggalkan feromon yang dapat dicium oleh semut berikutnya, sehingga semut-semut berikutnya tahu apakah tempat tersebut sudah dilewati atau belum. Gambar 2.9

menunjukkan koloni semut yang berkumpul mencari sarang makanan.

Algoritma semut yang diperkenalkan oleh Moyson dan Mendrik dan secara meluas dikembangkan oleh Macro Dorigo ini, merupakan teknik probalilistik untuk menyelesaikan masalah komputasi dengan menemukan jalur terbaik melalui grafik. Algoritma ini terinspirasi oleh perilaku semut dalam menemukan jalur dari koloninya menuju makanan.

Secara alamiah koloni semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat-tempat sumber makanan. Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka semakin jelas bekas jejak kakinya.



**Gambar 2.9 Koloni Semut**

Hal ini menyebabkan lintasan yang dilalui semut dalam jumlah sedikit, semakin lama semakin berkurang kepadatan semut yang melewatinya atau bahkan tidak dilewati sama sekali. Sebaliknya lintasan yang dilalui semut dalam jumlah banyak, semakin lama semakin bertambah kepadatan semut yang melewatinya, atau bahkan semua semut melalui lintasan tersebut.

Pada saat semut berjalan ia meninggalkan sejumlah informasi yang disebut feromon dalam jumlah tertentu, ditempat yang dilaluinya dan menandai jalur tersebut. Dengan perantara feromon inilah terjadi komunikasi tidak langsung dan juga pertukaran informasi antar semut selama membangun suatu solusi. Bentuk komunikasi tidak langsung yang diperlihatkan oleh semut ini disebut *stigmergy*.

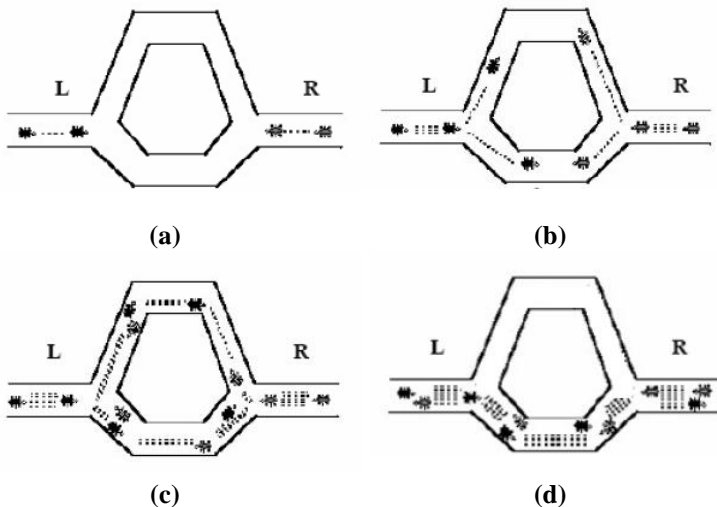
Gambar 2.10(a) mengilustrasikan proses dari *stigmergy*. Semut menggunakan feromon untuk menemukan jalur terpendek antara dua ujung yang dihubungkan dengan dua cabang yaitu bawah (yang lebih pendek) dan atas (yang lebih panjang).

Gambar 2.10(b) dan Gambar 2.10(c) menunjukkan bahwa kelompok semut berjalan menggunakan kecepatan yang sama dengan meninggalkan feromon atau jejak kaki di jalan yang telah dilalui. Feromon yang ditinggalkan oleh kumpulan semut yang melalui jalan atas telah mengalami banyak penguapan karena semut yang melalui jalan atas berjumlah lebih sedikit daripada jalan yang di bawah. Hal ini dikarenakan jarak yang ditempuh lebih panjang daripada jalan bawah.

Sedangkan feromon yang berada di jalan bawah, penguapannya cenderung lebih lama. Karena semut yang

melalui jalan bawah lebih banyak daripada semut yang melalui jalan atas.

Gambar 2.10(d) menunjukkan bahwa semut-semut yang lain pada akhirnya memutuskan untuk melewati jalan bawah karena feromon yang ditinggalkan masih banyak. Sedangkan feromon pada jalan atas sudah banyak menguap sehingga semut-semut tidak memilih jalan atas tersebut. Semakin banyak semut yang melalui jalan bawah maka semakin banyak pula semut yang mengikutinya. Demikian juga dengan jalan atas, semakin sedikit semut yang melalui jalan atas, maka feromon yang ditinggalkan semakin berkurang. Dari sinilah kemudian terpilih jalur terpendek antara sarang dan sumber makanan.



**Gambar 2.10 (a) Kelompok Semut Mencari Makan; (b) & (c) Semut Meninggalkan Jejak Feromon; (d) Semut Melewati Jalan Bawah yang Memiliki Lebih Banyak Feromon**

Hal ini berarti bahwa semakin banyak semut yang mengikuti sebuah jalur maka semakin bertambah menariklah jalur tersebut untuk dilalui. Probabilitas dimana seekor semut memutuskan untuk mengikuti suatu jalur meningkat dengan banyaknya semut yang lebih dulu menggunakan jalur tersebut.

Perilaku koloni semut telah menginspirasi munculnya sebuah metodologi baru yang di dalamnya terdapat sekumpulan semut buatan yang saling bekerja sama dalam mencari solusi terhadap suatu masalah optimisasi dengan cara bertukar informasi melalui feromon yang diletakkan pada ruas-ruas sebuah graf. Semut-semut buatan yang digunakan dalam sistem ini mempunyai perbedaan besar dengan hewan semut yang asli, antara lain semut ini memiliki memori. Semut ini juga tidak sepenuhnya buta, dan mereka akan berada pada lingkungan dimana waktunya adalah diskrit.

Secara informal, semut-semut buatan ini bekerja sebagai berikut. Setiap semut memulai turnya melalui sebuah titik yang dipilih secara acak (setiap semut memiliki posisi awal yang berbeda). Secara berulang kali, satu per satu titik yang ada akan dikunjungi oleh semut dengan tujuan untuk menghasilkan sebuah tur. Pemilihan titik-titik yang akan dilaluinya didasarkan pada suatu fungsi probabilitas, dinamai aturan transisi status (*state transition rule*), dengan mempertimbangkan visibilitas (invers dari jarak) titik tersebut dan jumlah feromon yang terdapat pada ruas yang menghubungkan titik tersebut.

Semut lebih suka bergerak menuju ke titik-titik yang dihubungkan dengan ruas yang pendek dan memiliki tingkat feromon yang tinggi. Setiap semut memiliki sebuah memori, yang diberi nama *tabulist*. *Tabulist* ini berisi semua titik yang telah dikunjungi oleh semut pada setiap tur yang dilakukan. *Tabulist* ini mencegah semut untuk mengunjungi titik-titik

yang sebelumnya telah dikunjungi selama tur tersebut berlangsung, yang membuat solusinya mendekati optimal.

Setelah semua semut menyelesaikan tur mereka dan *tabulist* mereka menjadi penuh, sebuah aturan pembaruan feromon global (*global pheromone updating rule*) diterapkan pada setiap semut. Penguapan feromon pada semua ruas dilakukan, kemudian setiap semut menghitung panjang tur yang telah mereka lakukan lalu meninggalkan sejumlah feromon pada tepi-tepi yang merupakan bagian dari tur mereka.

Semakin pendek sebuah tur yang dihasilkan oleh setiap semut, jumlah feromon yang ditinggalkan pada tepi-tepi yang dilaluinya pun semakin besar. Dengan kata lain, tepi-tepi yang merupakan bagian dari tur-tur terpendek adalah tepi-tepi yang menerima jumlah feromon yang lebih besar. Hal ini menyebabkan tepi-tepi yang diberi feromon lebih banyak akan lebih diminati pada tur-tur selanjutnya, dan sebaliknya tepi-tepi yang tidak diberi feromon menjadi kurang diminati. Dan juga, rute terpendek yang ditemukan oleh semut disimpan dan semua *tabulist* yang ada dikosongkan kembali.

Peranan utama dari penguapan feromon adalah untuk mencegah stagnasi, yaitu situasi dimana semua semut berakhir dengan melakukan tur yang sama. Proses di atas kemudian diulangi sampai tur-tur yang dilakukan mencapai jumlah maksimum atau sistem ini menghasilkan perilaku stagnasi dimana sistem ini berhenti untuk mencari solusi alternatif. Seiring waktu, bagaimanapun juga jejak feromon akan menguap dan akan mengurangi kekuatan daya tariknya.

Semakin cepat setiap semut pulang pergi melalui jalur tersebut, maka feromon yang menguap lebih sedikit. Begitu

pula sebaliknya jika semut semakin lama pulang pergi melalui jalur tersebut, maka feromon yang menguap lebih banyak.

Aturan transisi status diambil jika semut harus mengeksplorasi simpul yang belum pernah dikunjungi sama sekali. Kemungkinan ini merupakan probabilitas dari semut  $m$  pada titik  $i$  yang memilih untuk menuju ke titik  $j$ .

Setelah semua semut menyelesaikan turnya masing-masing maka feromon diperbaharui. Untuk memastikan bahwa semut mengunjungi  $n$  titik yang berbeda, diberikan *tabulist* pada masing-masing semut, yaitu sebuah struktur data yang menyimpan titik-titik yang telah dikunjungi semut dan melarang semut mengunjungi kembali titik-titik tersebut sebelum mereka menyelesaikan sebuah tur. Ketika sebuah tur selesai, *tabulist* digunakan untuk menghitung solusi yang ditemukan semut pada tur tersebut. *Tabulist* kemudian dikosongkan dan semut kembali bebas memilih titik tujuannya pada tur berikutnya.

#### **2.4.2 Algoritma *Ant Colony Optimization* (ACO)**

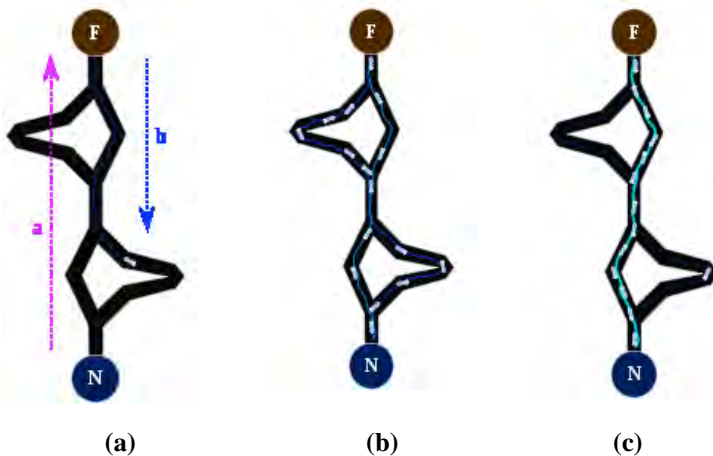
Algoritma ACO diperkenalkan oleh Marco Dorigo bersama dua orang temannya yaitu Colorni dan Maniezzo pada tahun 1991 sebagai bagian dari thesis untuk gelar PhD-nya. Algoritma ini terinspirasi dari tingkah laku semut untuk menawarkan solusi baru mengenai optimisasi.

*Ant colony optimization* (ACO) adalah sebuah metodologi yang dihasilkan melalui pengamatan terhadap semut. Pada algoritma ACO, semut berfungsi sebagai agen yang ditugaskan untuk mencari solusi terhadap suatu masalah optimisasi [13]. ACO telah diterapkan dalam berbagai bidang, salah satunya adalah untuk mencari solusi optimal pada *traveling salesman problem* (TSP). Dengan memberikan

sejumlah  $n$  titik, TSP dapat didefinisikan sebagai suatu permasalahan dalam menemukan jalur terpendek dengan mengunjungi setiap titik yang ada hanya sekali.

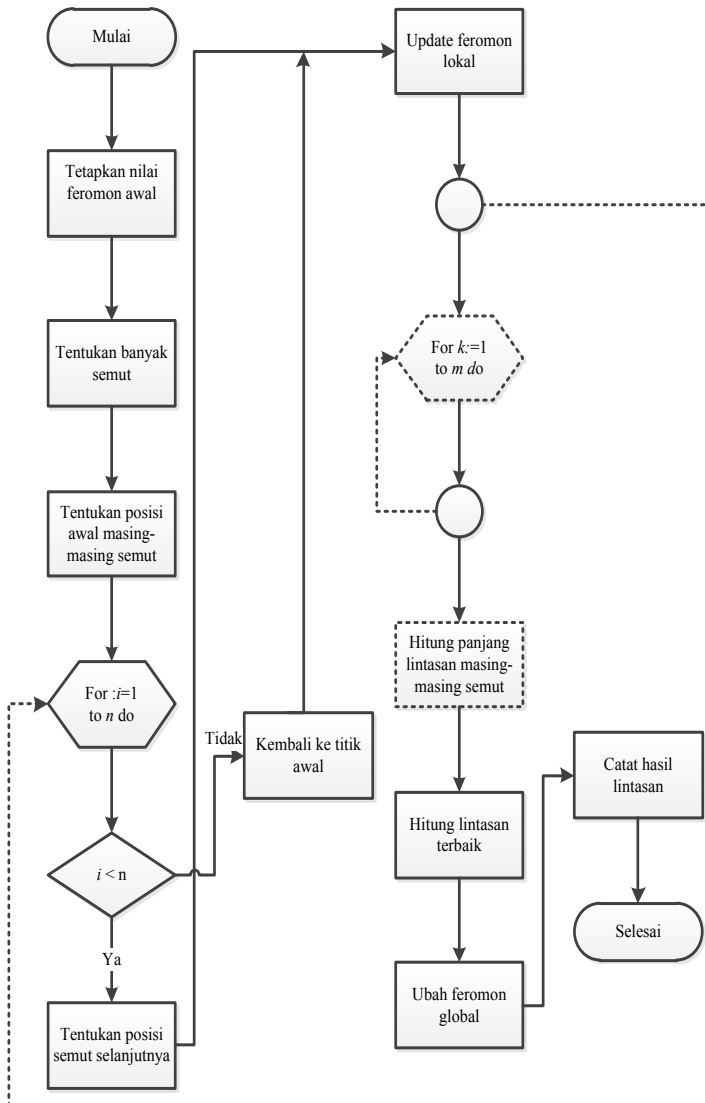
Algoritma *Ant colony optimization* (ACO) merupakan pengembangan dari AS dengan maksud untuk meningkatkan efisiensinya ketika diterapkan pada TSP yang kompleks. ACO memiliki aturan pembaruan feromon global yang hanya dilakukan pada ruas-ruas yang merupakan bagian dari tur terbaik. Disaat semut membangun sebuah solusi, diterapkan suatu aturan pembaruan feromon lokal (*local pheromone updating rule*).

ACO bekerja sebagai berikut, pertama kali sejumlah  $m$  semut ditempatkan pada sejumlah  $n$  kota berdasarkan beberapa aturan inisialisasi secara acak. Setiap semut membuat sebuah tur berupa sebuah solusi TSP yang mungkin dengan menerapkan sebuah aturan transisi status secara berulang kali.



**Gambar 2.11 (a) Semut Menemukan Sumber Makanan; (b) Semut Berjalan Melewati Dua Kemungkinan Jalur; (c) Semut Mengambil Rute Terpendek**





**Gambar 2.12** Alur Algoritma ACO

Pada saat membangun turnya, seekor semut juga memodifikasi jumlah feromon pada ruas-ruas atau tepi-tepi yang dikunjunginya dengan menerapkan aturan pembaruan feromon lokal. Setelah semua semut mengakhiri tur mereka, jumlah feromon yang ada pada ruas-ruas dimodifikasi kembali dengan menerapkan aturan pembaruan feromon global. Gambar 2.11 menunjukkan perjalanan semut saat mencari sarang makanan.

Pada Gambar 2.11(a) Semut pertama menemukan sumber makanan pada (F) melalui berbagai cara, kemudian kembali lagi ke sarang mereka yaitu (N) dengan meninggalkan jejak berupa feromon yang ditunjukkan dengan tanda panah berwarna biru (b). Kemudian pada Gambar 2.11(b) semut-semut mengikuti kemungkinan jalur yang bisa dilewati, feromon yang tersisa paling banyak membuat sebuah jalur menjadi rute terpendek. Selanjutnya pada Gambar 2.11(c) semut mengambil rute terpendek tersebut, sedangkan jalur-jalur lain akan kehilangan feromon akibat proses penguapan feromon.

Semut mampu mengindra lingkungannya yang kompleks untuk mencari makanan dan kemudian kembali ke sarangnya dengan meninggalkan zat feromon pada jalur-jalur yang mereka lalui. Feromon adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan oleh makhluk hidup untuk mengenali sesama jenis, individu lain, kelompok, dan untuk membantu proses reproduksi.

Berbeda dengan hormon, feromon menyebar ke luar tubuh sehingga, dapat mempengaruhi dan dikenali oleh individu lain yang sejenis (satu spesies). Proses peninggalan feromon ini dikenal sebagai *stigmergy*, sebuah proses memodifikasi lingkungan yang tidak hanya bertujuan untuk mengingat jalan pulang ke sarang, tetapi juga memungkinkan

para semut berkomunikasi dengan koloninya. Seiring waktu, jejak feromon akan menguap dan akan mengurangi kekuatan daya tariknya. Semakin lama seekor semut pulang pergi melalui jalur tersebut, semakin lama pula feromon menguap.

Agar semut mendapatkan jalur optimal, diperlukan beberapa proses, yaitu pada awalnya semut berkeliling secara acak hingga menemukan makanan. Ketika menemukan makanan mereka kembali ke koloninya sambil memberikan tanda dengan jejak feromon. Jika semut-semut lain menemukan jalur tersebut, mereka tidak akan bepergian dengan acak lagi, melainkan akan mengikuti jejak tersebut. Seekor semut yang secara tidak sengaja menemukan jalur optimal akan menempuh jalur ini lebih cepat dari rekan-rekannya, melakukan *round-trip* lebih sering, dan dengan sendirinya meninggalkan feromon lebih banyak dari jalur-jalur yang lebih lambat ditempuh. Feromon yang berkonsentrasi tinggi pada akhirnya akan menarik semut-semut lain untuk berpindah jalur, menuju jalur paling optimal, sedangkan jalur lainnya akan ditinggalkan.

Dalam membuat tur, semut dipandu oleh informasi heuristik dan feromon. Sebuah ruas dengan jumlah feromon yang tinggi merupakan pilihan yang tepat. Aturan pembaruan feromon ini dirancang agar semut cenderung untuk memberi lebih banyak feromon pada ruas-ruas yang harus mereka lewati. Berikutnya akan dibahas mengenai tiga karakteristik utama dari ACO, yaitu aturan transisi status, aturan pembaharuan feromon global, dan aturan pembaharuan feromon lokal. Gambar 2.12 menunjukkan alur algoritma ACO.

Aturan transisi status yang berlaku pada ACO dimana semut  $m$  yang berada di titik  $i$ , akan memilih titik berikutnya

yaitu  $j$  didefinisikan menggunakan probabilitas yang ditunjukkan pada Persamaan 2.9.

$$p_{(ij)}^m = \frac{(\tau_{ij}^\alpha)(\eta_{ij}^\beta)}{\sum_{j \in \text{allowed}_i} (\tau_{ij}^\alpha)(\eta_{ij}^\beta)} \quad (2.9)$$

Dimana  $\tau_{ij}$  adalah jumlah feromon yang diletakkan saat perpindahan dari piksel  $i$  ke piksel  $j$  (dari kota  $i$  ke kota  $j$ ).  $0 \leq \alpha$  adalah parameter untuk mengontrol  $\tau_{ij}$ .  $\eta_{ij}$  adalah kemampuan untuk berpindah dari piksel  $i$  ke piksel  $j$ .  $\beta \geq 1$  adalah parameter untuk mengontrol  $\eta_{ij}$ .  $\tau_{ij}$  dan  $\eta_{ij}$  merepresentasikan daya tarik dan tingkat jejak untuk kemungkinan perpindahan antar kota lainnya.

Aturan transisi status yang dihasilkan dari Persamaan 2.9 dinamakan aturan proporsional acak semu (*pseudorandom-proportional rule*). Aturan transisi status ini, sebagaimana dengan aturan proporsional acak terdahulu, mengarahkan semut untuk bertransisi ke kota-kota yang dihubungkan dengan ruas-ruas yang pendek dan memiliki jumlah feromon yang besar.

Selanjutnya setelah membahas aturan untuk transisi status, maka aturan untuk memperbarui feromon global akan dibahas pada bagian ini. Pada sistem ini, pembaruan feromon secara global hanya dilakukan oleh semut yang membuat tur terpendek sejak percobaan pertama kali dilakukan. Pada akhir sebuah iterasi, setelah semua semut menyelesaikan tur mereka, sejumlah feromon diletakkan pada ruas-ruas yang dilewati oleh seekor semut yang telah menemukan tur terbaik (ruas-ruas yang lain tidak diubah). Tingkat feromon itu diperbarui

dengan menerapkan aturan pembaruan feromon global yang ditunjukkan pada Persamaan 2.10 dan 2.11.

$$\tau(i, j) \leftarrow (1 - \alpha) \cdot \tau(i, j) + \alpha \cdot \Delta\tau(i, j) \quad (2.10)$$

$$\Delta\tau(i, j) = \begin{cases} \frac{1}{L_{gb}} & \text{Jika } (i, j) \in \text{global best tour} \\ 0 & \text{sebaliknya} \end{cases} \quad (2.11)$$

$L_{gb}$  adalah panjang dari tur terbaik secara global sejak permulaan percobaan. Seperti yang terjadi pada *ant colony system*, pembaruan feromon global dimaksudkan untuk memberikan feromon yang lebih banyak pada tur yang lebih pendek. Persamaan *update* jejak feromon ini, dilakukan pada akhir sebuah iterasi algoritma saat semua semut telah menyelesaikan sebuah tur. Persamaan diperangkat lunakkan ke ruas yang digunakan semut saat menemukan lintasan keseluruhan yang terbaik sejak awal percobaan. Tujuan pemberian nilai ini adalah memberi sejumlah jejak feromon pada lintasan terpendek, dimana tur terbaik (lintasan dengan panjang terpendek) mendapat jumlah feromon yang lebih banyak.

Selanjutnya, selama melakukan tur untuk mencari solusi dari TSP, semut-semut mengunjungi ruas-ruas dan mengubah tingkat feromon pada ruas-ruas tersebut dengan menerapkan aturan pembaruan feromon lokal yang ditunjukkan oleh Persamaan 2.12.

$$\tau(i, j) \leftarrow (1 - \rho) \cdot \tau(i, j) + \rho \cdot \Delta\tau(i, j) \quad (2.12)$$

Pada Persamaan 2.12,  $0 < \rho < 1$  adalah sebuah parameter. Peranan dari aturan pembaruan feromon lokal ini adalah untuk mengacak arah tur yang sedang dibangun, sehingga kota-kota yang telah dilewati sebelumnya oleh tur seekor semut mungkin akan dilewati kemudian oleh tur yang dimiliki semut

lainnya. Dengan kata lain, pengaruh dari pembaruan lokal ini adalah untuk membuat tingkat ketertarikan ruas-ruas yang ada berubah secara dinamis. Maksud dinamis disini berarti setiap kali seekor semut menggunakan sebuah ruas maka ruas ini dengan segera akan berkurang tingkat ketertarikannya (karena ruas tersebut kehilangan sejumlah feromonnya), secara tidak langsung semut yang lain akan memilih ruas-ruas lain yang belum dikunjungi. Konsekuensinya, semut tidak akan memiliki kecenderungan untuk berkumpul pada jalur yang sama.

Filosofi dari algoritma *ant colony* adalah pemecahan masalah yang merupakan pemodelan dari realita koloni semut dalam mencari makanan. Jumlah alternatif solusi sebagai pemecahan masalah adalah merupakan jumlah semut di dalam koloni tersebut, dan populasi dari jumlah alternatif solusi adalah masing-masing saat koloni pergi mencari makanan dan kembali ke sarang mereka.

Algoritma ini merupakan sebuah fenomena alam menarik yang telah terimplementasi di dalam praktek algoritma program komputer di dunia teknologi informasi. Manfaat algoritma koloni semut ini sangat banyak, selain digunakan untuk mencari sebuah *path* atau jalur yang optimal baik jarak, waktu, dan biaya di dalam sebuah jaringan distribusi barang, algoritma ini dapat juga dimanfaatkan untuk mencari nilai optimal lain yang terjadi di lapangan, seperti permasalahan kombinasi, nilai produksi *forecasting*, bahkan sebagai deteksi tepi citra.

Semut sangat hebat dalam melakukan *trace-back* jalur tercepat yang telah dan pernah dilalui, karena telapak pada kaki semut mengeluarkan sejenis zat kimia berupa feromon yang mudah menguap sehingga, kekuatan bau zat tersebut harus ditambah oleh kaki semut berikutnya. Jalur yang sering

dilewati itulah diprediksi merupakan jalur yang paling optimal atau jalur yang mendekati optimal antara sarang semut dengan sumber makanan.

*Ant colony optimization algorithm* ini telah memperkaya jenis algoritma pencarian nilai optimal yang telah ada di bidang keilmuan teknologi informasi dan pemrograman komputer. Analisis mendalam dari serangga kecil yang bernama semut ini saja, mampu memberikan ilmu pengetahuan dan teknologi yang sangat bermanfaat bagi umat manusia sedunia.

## 2.5 Ketetangaan Piksel

Ketetangaan piksel adalah salah satu fitur yang sangat penting dalam pengolahan dan analisis citra digital. Untuk membentuk sebuah grup piksel, pemrosesan terhadap sebuah citra dilakukan dan terdapat dua kriteria dasar yang harus diperhatikan.

Pertama adalah keseluruhan piksel harus memiliki nilai intensitas yang sama dan saling terhubung satu sama lain. Jika tidak, maka proses pembentukan sebuah grup piksel tidak dapat dilakukan. Kedua adalah masing-masing piksel harus merupakan citra berwarna atau citra keabuan.

$(x-1,y-1)$	$(x-1,y)$	$(x-1,y+1)$
$(x,y-1)$	$(x,y)$	$(x,y+1)$
$(x+1,y-1)$	$(x+1,y)$	$(x+1,y+1)$

**Gambar 2.13 Ketetangaan Antar Piksel**

Mengekstraksi dan menandai berbagai macam komponen terhubung dan tidak terhubung pada gambar merupakan proses yang sangat penting pada berbagai perangkat lunak analisis dan pengolahan citra [14]. Proses ini disebut *connected component labeling*.

Sebuah piksel  $p$  pada koordinat  $(x,y)$  mempunyai empat tetangga vertikal dan horisontal, dimana koordinat tersebut dinyatakan oleh:  $(x-1,y)$ ,  $(x+1,y)$ ,  $(x,y-1)$ ,  $(x,y+1)$ . Satuan piksel ini dinamakan 4-tetangga dari  $p$  yang dinotasikan dengan  $N4(p)$ . Sedangkan empat tetangga diagonal  $p$  mempunyai koordinat:  $(x+1,y+1)$ ,  $(x+1,y-1)$ ,  $(x-1,y+1)$ , dan  $(x-1,y-1)$ . Satuan piksel dinamakan 8-tetangga dari  $p$ , yang dinotasikan dengan  $ND(p)$ . Gambar 2.13 menunjukkan hubungan ketetanggaan antar piksel. Ketetanggaan piksel ini terdiri dari 3 jenis yaitu:

1. Ketetanggaan 4 piksel  
Setiap piksel memiliki empat tetangga sehingga keempat tetangga menyentuh tepi piksel.
2. Ketetanggaan 6 piksel  
Setiap piksel memiliki enam tetangga sehingga keenam tetangga menyentuh tepi piksel.
3. Ketetanggaan 8 piksel  
Setiap piksel memiliki delapan tetangga sehingga kedelapan tetangga menyentuh tepi piksel.

## 2.6 *Thresholding*

Dalam pemrosesan citra, mengubah gambar menjadi biner artinya mengubah warna tiap-tiap piksel pada gambar menjadi hanya bernilai 0 atau 255, sehingga hanya berwarna hitam dan putih. Dengan normalisasi gambar bisa juga

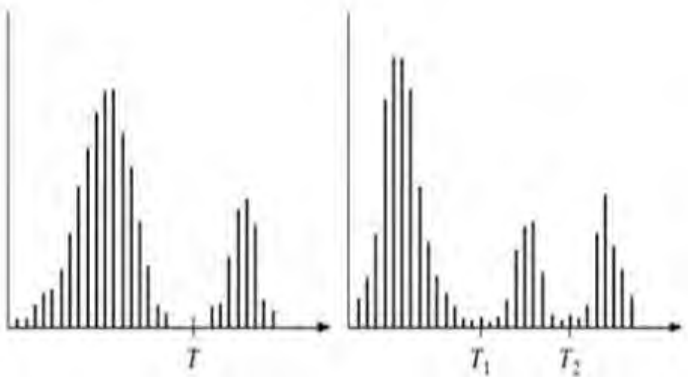


dinyatakan dengan warna tiap-tiap piksel pada gambar 0 atau 1 [10].

*Thresholding* adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk objek dan *background* dari citra secara jelas. Citra hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan objek serta ekstraksi fitur. Metode *thresholding* secara umum dibagi menjadi dua [15], yaitu:

1. *Thresholding* global

*Thresholding* dilakukan dengan mempartisi histogram dengan menggunakan sebuah *threshold* (batas ambang) global  $T$ , yang berlaku untuk seluruh bagian pada citra. *Thresholding* dikatakan global jika nilai *threshold*  $T$  hanya bergantung pada  $f(x,y)$ , yang melambangkan tingkat keabuan pada titik  $(x,y)$  dalam suatu citra. Salah satu *thresholding* global adalah *otsu thresholding*.



**Gambar 2.14 Partisi Histogram dalam *Thresholding* Adaptif untuk Memperoleh Nilai *Threshold***

## 2. *Thresholding* adaptif

*Thersholding* dilakukan dengan membagi citra menggunakan beberapa sub citra. Lalu pada setiap sub citra, segmentasi dilakukan dengan menggunakan *threshold* yang berbeda [3].

Histogram yang berada di sisi kiri pada Gambar 2.14 mewakili citra  $f(x,y)$  yang tersusun atas objek terang di atas *background* gelap. Pikel-pikel objek dan *background* dikelompokkan menjadi dua bagian yang dominan. Cara untuk mengekstrak objek dari *background* adalah dengan memilih nilai *threshold*  $T$  yang memisahkan dua mode tersebut. Kemudian untuk sembarang titik  $(x,y)$  yang memenuhi  $f(x,y) > T$  disebut titik objek, selain itu disebut titik *background*. Kesuksesan *thresholding* bergantung pada seberapa bagus teknik partisi histogram. Citra hasil *thresholding* dapat didefinisikan seperti pada Persamaan 2.13.

$$g(x, y) = \begin{cases} 1, & f(x, y) > T \\ 0, & f(x, y) \leq T \end{cases} \quad (2.13)$$

### 2.6.1 *Otsu Thresholding*

*Otsu* adalah salah satu metode yang umum digunakan untuk melakukan proses *thresholding* histogram. Algoritma ini mengasumsikan bahwa gambar yang akan di-*threshold* berisi dua kelas piksel (objek gambar dan latar belakangnya). Metode *otsu* dinamai sesuai dengan penemunya Nobuyuki Otsu. Metode *otsu* bekerja dengan mencari *threshold* yang meminimumkan varians intra-kelas, yang didefinisikan sebagai jumlah dari varians dari dua kelas, sesuai dengan Persamaan 2.14.

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \quad (2.14)$$

Bobot  $\omega_i$  adalah probabilitas dari dua kelas yang dipisahkan oleh sebuah *t threshold* dan varians dari kelas-kelas. *Otsu* menunjukkan bahwa meminimalkan varians intra-kelas adalah sama dengan memaksimalkan varians inter-kelas, sesuai dengan Persamaan 2.15.

$$\sigma_b^2(t) = \sigma^2 - \sigma_o^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \quad (2.15)$$

Varians inter-kelas dinyatakan dalam probabilitas kelas dan kelas rata-rata dapat diperbarui secara iteratif [3]. *Otsu thresholding* dapat diamati pada Gambar 2.15(a) dan Gambar 2.15(b).



**Gambar 2.15 (a) Citra Asli; (b) Citra Hasil *Otsu Thresholding***

## 2.7 Algoritma *Ant-inspired*

Pada bagian ini akan dijelaskan mengenai algoritma *ant-inspired* yang digunakan dalam tugas akhir ini. Deteksi tepi adalah sebuah teknik yang digunakan untuk menandai perubahan intensitas ketajaman suatu citra dan sangat penting untuk melakukan analisis yang lebih jauh tentang *content* sebuah citra [16]. Deteksi tepi dengan menggunakan metode konvensional selalu memberikan hasil berupa struktur gambar yang rusak dan garis-garis tepi yang tidak terhubung satu sama lain.

Mendeteksi tepi pada sebuah gambar merupakan hal yang sangat penting pada pengolahan citra. Hal ini memerlukan pendekatan yang berbeda untuk beberapa gambar. Beberapa metode yang sudah ada sebelumnya yaitu *canny*, *sobel*, *prewitt*, dan beberapa teknik lainnya telah digunakan secara luas. Akan tetapi metode ini belum bisa dengan baik digunakan untuk semua jenis gambar. Sehingga diperlukan sebuah ekstraksi fitur dengan metode baru untuk mendapatkan hasil deteksi yang lebih baik.

Algoritma ini masuk ke dalam jenis metode *swarm intelligence* yang dapat digunakan untuk mendeteksi tepi dengan menggunakan teknik probabilitas. Pengertian *swarm intelligence* adalah metode penyelesaian masalah yang memanfaatkan perilaku dari sekumpulan agen yang saling bekerjasama [17].

Tepi gambar dapat diumpamakan sebagai sarang makanan yang akan dicari oleh semut. Dalam algoritma ini sejumlah semut tersebar secara acak pada piksel gambar dengan *range* nilai dari 0 sampai 255 dalam format citra keabuan. Pada setiap iterasi, semut berpindah ke salah satu piksel yang berdekatan dan memperkuat jejak piksel tersebut

dengan cara meninggalkan zat kimia berupa feromon pada piksel tersebut.

Tahapan ekstraksi tepi gambar menggunakan algoritma *ant-inspired* dapat dilihat pada Gambar 2.16. Tahapan ini terdiri dari berbagai macam proses yang akan dijelaskan di tulisan selanjutnya.

*Ant colony* adalah sebuah teknik yang ditemukan oleh Dorigo yang menggunakan pergerakan semut. *Ant colony* merupakan metode yang berdasarkan pergerakan semut yang digunakan untuk memecahkan *travelling salesman problem* (TSP). Secara alamiah, semut mampu menemukan rute terpendek dalam perjalanan saat mencari sumber makanan.

Koloni semut dapat menemukan rute terpendek antara sarang dan sumber makanan berdasarkan jejak kaki pada lintasan yang telah dilalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas bekas jejak kakinya, sehingga metode ini secara tepat dapat digunakan pada TSP. Algoritma ini digunakan untuk mencari jarak terdekat dari sebuah graf yang biasanya diterapkan pada permasalahan TSP. Metode ini didasarkan pada pergerakan semut yang akan meninggalkan feromon pada saat mencari makanan.

Algoritma ini memiliki perbedaan dengan algoritma ACO seperti yang dijelaskan pada sub bab 2.4.2. Pada algoritma ini, rumus yang digunakan hanya rumus probabilitas dan rumus untuk *update* feromon global. Sedangkan pada algoritma ACO terdapat rumus untuk *update* feromon lokal dan terdapat *tabulist* yang berguna untuk menyimpan posisi semut sebelumnya. ACO juga menyimpan rute-rute terpendek yang berhasil dilakukan oleh semut saat melakukan perjalanan dari titik awal ke titik tujuan. Pada algoritma ini pergerakan

semut menggunakan rumus probabilitas dengan aturan ketetanggaan *von neumann*.

Pada algoritma ini, masing-masing semut dapat bergerak dari piksel  $i$  ke  $j$  dengan probabilitas  $p_{ij}$  sesuai dengan Persamaan 2.16.

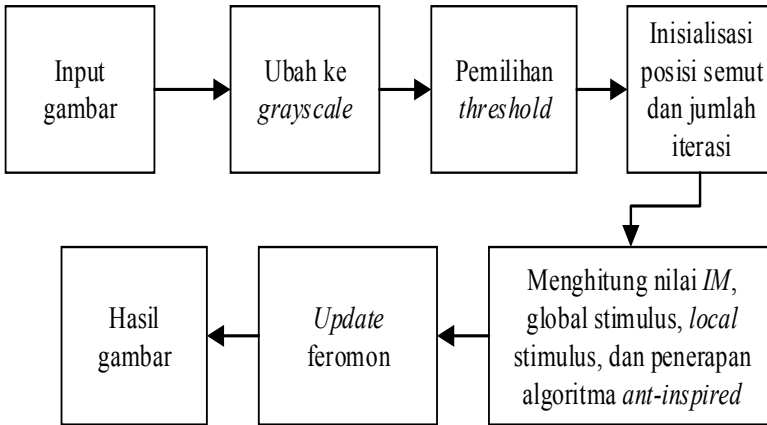
$$p_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_k (\tau_{i,k}^{\alpha}) (\eta_{i,k}^{\beta})} \quad (2.16)$$

Dimana  $\tau_{ij}$  adalah jumlah feromon antara piksel  $i$  dan  $j$ ,  $\eta_{ij}$  adalah pemilihan *path* (jalur), sedangkan  $\alpha$  merupakan koefisien *influence* untuk teta dan  $\beta$  adalah koefisien *influence* untuk eta. Kemudian, feromon akan di-*update* berdasarkan Persamaan 2.17.

$$\tau_{ij} = (1 - \rho) \tau_{ij} + \lambda \quad (2.17)$$

Dimana koefisien  $\rho$  merupakan nilai penguapan feromon dan  $\lambda$  adalah feromon yang disebarakan semut saat bergerak dari piksel  $i$  ke  $j$ . Secara garis besar algoritma untuk proses ekstraksi fitur dijabarkan pada Gambar 2.16 [1].

Dalam algoritma ini digunakan dua jenis feromon. Tipe feromonnya adalah feromon tipe 1 dan feromon tipe 2 [1]. Untuk feromon tipe 1, nilai  $\lambda$  biasanya diset 0,02 dan  $\rho$  diset  $< 1$  (disesuaikan dengan nilai  $\lambda$ ). Setiap akhir proses iterasi, masing-masing semut meninggalkan feromon dengan intensitas  $\lambda$  dan berpindah ke piksel tetangga. Selain itu, setiap akhir proses iterasi semua feromon tipe 1 semakin berkurang nilainya berdasarkan rumus  $1-\rho$  dari Persamaan 2.17. Nilai  $\alpha$  diset 1 dan nilai  $\beta$  diset 0. Koefisien  $\mu$  merupakan koefisien yang digunakan sebagai pengingat tempat terakhir yang dikunjungi semut.



**Gambar 2.16 Tahapan Ekstraksi Tepi Gambar Menggunakan Algoritma *Ant-inspired***

Feromon tipe 2 adalah komponen yang bertanggung jawab untuk proses penentuan keputusan semut. Agar memudahkan, feromon tipe 1 disebut “feromon” dan feromon tipe 2 disebut “stimulus”.

1	Mengubah gambar ke citra keabuan
2	Menentukan $T$
3	Menentukan angka dan posisi awal dari setiap semut
4	Menentukan jumlah iterasi
5	Menghitung $IM$ menggunakan persamaan 2.18
6	Menghitung <i>matrix</i> stimulus global menggunakan persamaan 2.19
7	<b>for</b> iterasi < max.iterasi <b>do</b>
8	<b>for all</b> semut <b>do</b>
9	Menghitung stimulus lokal menggunakan persamaan 2.21
10	Pindahkan semut menggunakan persamaan 2.16
11	Meletakkan feromon tipe 1

12	<b>end for</b>
13	<i>Update</i> $\Delta$ (feromon) menggunakan persamaan 2.17
14	Iterasi = iterasi + 1
15	<b>end for</b>

**Gambar 2.17** Algoritma Ant-inspired

Selain feromon yang disebarkan semut saat bergerak, gambar yang akan diekstrak fiturnya harus mengandung karakteristik yang mirip dengan feromon agar semut tidak bergantung terhadap nilai  $\lambda$ . Untuk menyelesaikan hal ini, maka gradien dari gambar dihitung menggunakan Persamaan

2.18. Dimana  $\frac{\partial IM}{\partial x} \hat{i}$  adalah gradien gambar terhadap sumbu

$x$  dan  $\frac{\partial IM}{\partial y} \hat{j}$  adalah gradien gambar terhadap sumbu  $y$ .

$$\nabla IM = \frac{\partial IM}{\partial x} \hat{i} + \frac{\partial IM}{\partial y} \hat{j} \quad (2.18)$$

Global stimulus ( $S$ ) dihitung menggunakan persamaan (2.19) yang merupakan *Euclidean norm* dari gradien gambar (Persamaan 2.18).

$$S = \|\nabla IM\| \quad (2.19)$$

Dimana  $\|\nabla IM\|$  didefinisikan seperti pada Persamaan 2.20.

$$\|\nabla IM\| = \sqrt{\left(\frac{\partial IM}{\partial x}\right)^2 + \left(\frac{\partial IM}{\partial y}\right)^2} \quad (2.20)$$



Setelah mendapatkan hasil berdasarkan Persamaan 2.20, langkah selanjutnya yaitu menghitung stimulus  $S_{i,j}$  masing-masing semut pada  $(i,j)$  berdasarkan Persamaan 2.21.

$$S_{i,j} = \sigma(i, j, i', j') = S((neighbors(i, j)) \wedge \neg(i', j')) \quad (2.21)$$

Dimana piksel  $(i,j)$  adalah semua tetangga dari piksel  $(i,j)$  dan  $(i',j')$  adalah lokasi sebelumnya yang dikunjungi oleh semut.

## 2.8 Perhitungan Akurasi

Akurasi menyatakan seberapa dekat nilai hasil pengukuran dengan nilai sebenarnya (*true value*) atau nilai yang dianggap benar (*accepted value*). Jika tidak ada data atau nilai sebenarnya yang dianggap benar, maka tidak dapat ditentukan akurasi dari suatu pengukuran.

Perhitungan skor dan akurasi deteksi tepi citra dapat dihitung dengan menggunakan sesuai dengan Persamaan 2.22 dan Persamaan 2.23.

$$Score = tp + tn \quad (2.22)$$

$$accuracy = \left( \frac{tp}{tp + fn} \right) + \left( \frac{tn}{tn + fp} \right) \times 100 \% \quad (2.23)$$

Dimana  $tp$  adalah jumlah *true positive*,  $tn$  adalah jumlah *true negative*,  $fn$  adalah jumlah *false negative*, dan  $fp$  adalah jumlah *false positive* dari proses deteksi tepi citra. Dari hasil perhitungan akurasi tersebut akan didapatkan hasil akurasi dengan *range* antara 0% sampai 100%.

## **BAB 3**

### **PERANCANGAN PERANGKAT LUNAK**

Pada bab ini akan diuraikan mengenai perancangan sistem perangkat lunak agar dapat mencapai tujuan dari tugas akhir. Perancangan yang dibuat meliputi perancangan data, perancangan proses, dan perancangan antarmuka. Perancangan data terdiri dari data masukan, data proses, dan data keluaran. Perancangan proses yang dilakukan meliputi desain secara umum, tahap pra-proses, dan tahap proses.

#### **3.1 Perancangan Data**

Perancangan data merupakan hal penting untuk diperhatikan karena diperlukan data yang tepat agar perangkat lunak beroperasi secara benar. Data yang dibutuhkan dalam membangun perangkat lunak untuk proses deteksi tepi citra menggunakan algoritma *ant-inspired* terdiri dari data masukan berupa data yang akan dimasukkan dari pengguna, kemudian data proses berupa data-data yang dibutuhkan dan dihasilkan selama menjalankan proses eksekusi perangkat lunak, serta data keluaran yang memberikan hasil berupa bentuk tepi citra yang terdeteksi.

##### **3.1.1 Data Masukan**

Data masukan adalah data awal yang akan diproses pada implementasi tugas akhir ini. Data tersebut berupa citra masukan yang akan diproses untuk mendapatkan deteksi tepi dari citra tersebut. Data ini berupa citra berwarna dengan berbagai ukuran. Untuk data masukan dapat dilihat pada Tabel 3.1. Nama data dalam tabel tersebut dijelaskan dalam keterangannya. Contoh citra berwarna yang digunakan sebagai data masukan ditunjukkan pada Gambar 3.1.



**Gambar 3.1 Citra Masukan**

**Tabel 3.1 Data Masukan**

No.	Nama Data	Keterangan
1.	input	Citra berwarna dengan ukuran tertentu yang akan digunakan dalam proses deteksi tepi citra

### 3.1.2 Data Proses

Data proses adalah data yang digunakan dalam proses pengoperasian perangkat lunak. Data proses yang digunakan dalam proses deteksi tepi menggunakan algoritma *ant-inspired* ditunjukkan pada Tabel 3.2.

**Tabel 3.2 Data Proses**

No.	Nama Data	Tipe Data	Keterangan
1.	imgAbu	double	Citra hasil dari proses pengolahan dari citra berwarna ke citra keabuan
2.	gx_2	double	Gradien citra arah x

No.	Nama Data	Tipe Data	Keterangan
3.	gy_2	double	Gradien citra arah y
4.	imgGra	double	Citra hasil gabungan dari citra gradien $x$ dan citra gradien $y$
5.	jumlah_semut	uint8	Jumlah semut yang diletakkan pada piksel citra untuk melakukan proses deteksi tepi
6.	posisi_semut	double	Posisi semut yang digenerate secara random pada piksel citra untuk melakukan proses deteksi tepi
7.	iterasi	uint8	Jumlah iterasi yang digunakan untuk melakukan proses deteksi tepi
8.	probabilitas	uint8	Nilai probabilitas yang digunakan untuk menentukan langkah semut selanjutnya

### 3.1.3 Data Keluaran

Data keluaran adalah data dari hasil proses deteksi tepi dengan menggunakan algoritma *ant-inspired* dengan tepi yang berhasil dideteksi. Citra ini akan disimpan di direktori perangkat lunak. Citra ini nantinya bisa digunakan sebagai citra masukan saat melakukan pemrosesan citra lanjut atau keperluan lain. Untuk data keluaran dapat dilihat pada Tabel 3.3. Nama data dalam tabel tersebut akan dijelaskan dalam keterangannya.

**Tabel 3.3 Data Keluaran**

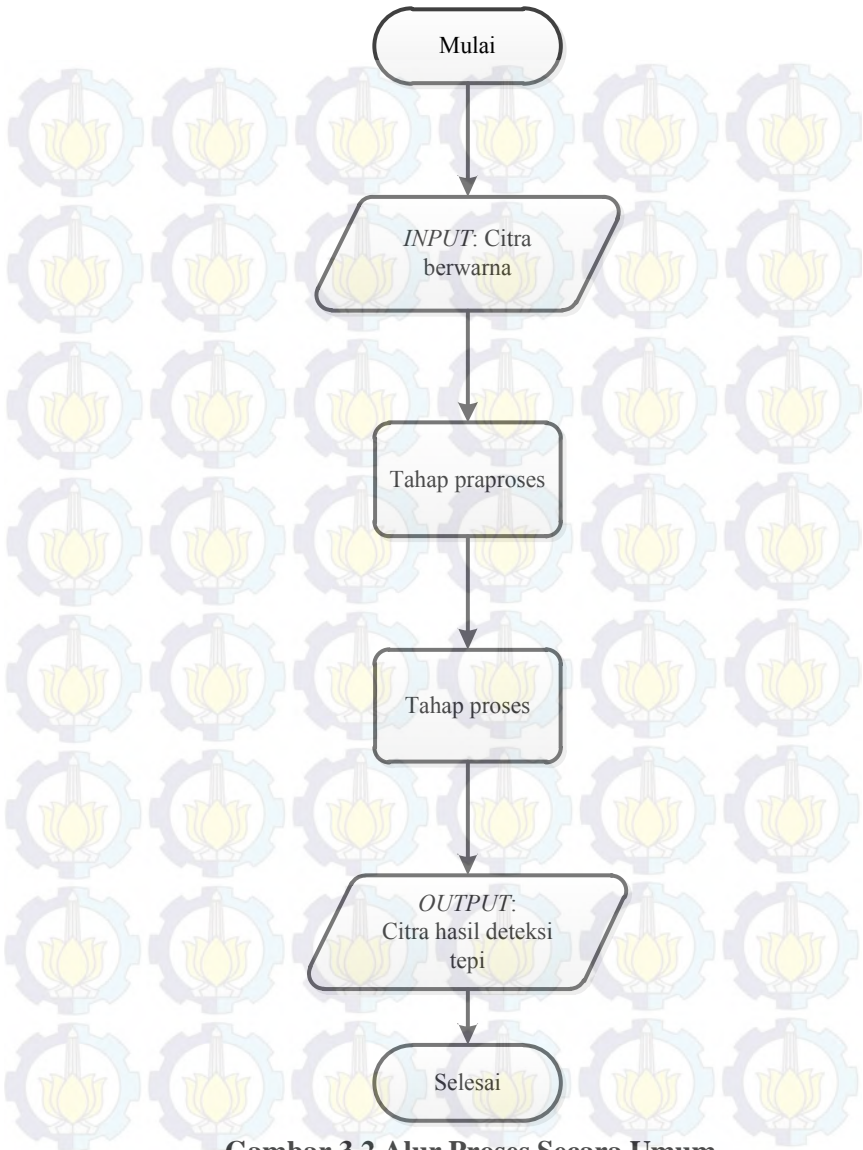
No.	Nama Data	Keterangan
1.	imgAbu	Citra hasil dari proses pengolahan dari citra berwarna ke citra keabuan
2.	hasil	Citra hasil dari proses deteksi tepi

### 3.2 Perancangan Proses

Perancangan proses dilakukan untuk mengetahui alur dalam penerapan algoritma yang nantinya akan dipakai dalam tahap implementasi pada tugas akhir ini. Alur tersebut akan ditampilkan dalam diagram alir sesuai masing-masing proses.

#### 3.2.1 Desain Secara Umum

Secara umum citra hasil deteksi tepi dengan menggunakan algoritma *ant-inspired* ini dimulai dengan memasukkan citra masukan. Citra masukan yang berupa format berwarna ini akan diubah terlebih dahulu ke format citra keabuan untuk memudahkan proses deteksi tepi. Setelah mendapatkan citra keabuan, maka citra ini akan diubah ke format citra biner. Setelah itu dilanjutkan dengan mengubah citra ke dalam format `double` sebagai syarat untuk melakukan perhitungan gradien citra. Nilai dari gradien citra memiliki kemungkinan menghasilkan nilai yang negatif, sehingga perlu untuk dilakukan proses normalisasi dengan memangkatkan hasil dari gradien citra tersebut. Hasil keluaran dari sistem ini adalah citra yang sudah mengalami pendeteksian pada tepinya. Keseluruhan tahapan dalam tugas akhir ini akan digambarkan pada diagram alir pada Gambar 3.2.



**Gambar 3.2 Alur Proses Secara Umum**

### 3.2.2 Tahap Praproses

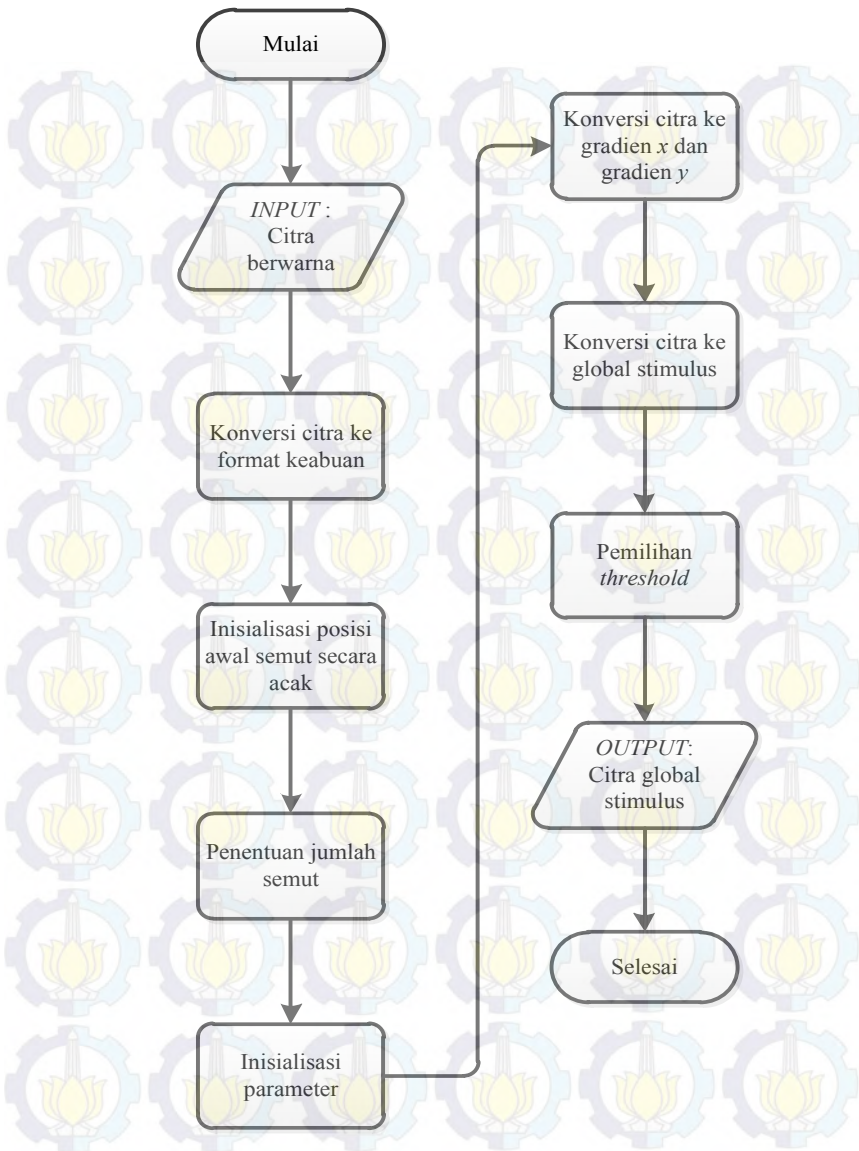
Tahap pra-proses dilakukan sebelum proses inti dilakukan. Pada tahap ini dilakukan konversi gambar dari citra berwarna ke citra keabuan. Selain itu dilakukan pula penentuan jumlah semut yang digunakan, posisi semut, jumlah iterasi, nilai *threshold*, dan parameter lainnya. Jumlah semut yang digunakan untuk masing-masing gambar adalah jumlah kolom dikalikan dengan jumlah baris dibagi sembilan untuk masing-masing gambar. Peletakan posisi semut dilakukan secara random dengan menggunakan fungsi yang ada pada perangkat lunak yang digunakan yaitu MATLAB R2008a. *Threshold* yang digunakan merupakan nilai yang lebih besar sama dengan 0,1. Untuk jumlah iterasi bisa menyesuaikan dengan maksimum iterasi sejumlah lima ratus iterasi. Gambar 3.3 menunjukkan citra hasil praproses. Secara keseluruhan tahap ini dapat dilihat pada diagram alir Gambar 3.4.

### 3.2.3 Tahap Proses

Tahapan proses ini terdiri dari beberapa tahapan yaitu perhitungan gradien gambar, perhitungan global stimulus, mencari piksel yang merupakan tepi, dan proses *update* feromon. Tahapan-tahapan ini akan dijelaskan pada sub bab selanjutnya.

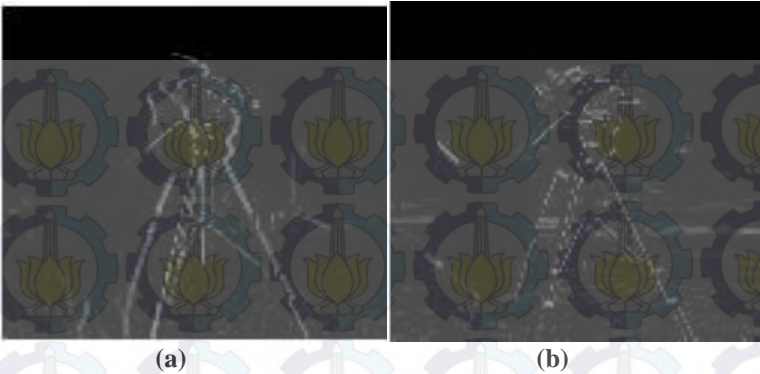


**Gambar 3.3 Citra Hasil Praproses**



**Gambar 3.4 Diagram Alir Tahap Praproses**





**Gambar 3.5 (a) Gradien Gambar dengan Arah  $X$ ; (b) Gradien Gambar dengan Arah  $Y$**

### 3.2.3.1 Perhitungan Gradien Gambar

Pada proses perhitungan gradien gambar, gradien dari fungsi dengan dua variabel pada setiap titik gambar adalah vektor 2D dengan komponen yang diperoleh dari hasil turunan gambar pada arah horizontal dan vertikal.

Pada setiap gambar vektor gradien menuju ke suatu arah dengan kemungkinan peningkatan intensitas yang terbesar, dan panjang gradien vektornya sesuai dengan tingkat perubahan ke arah tersebut. Proses ini dihitung dengan menggunakan rumus yang ada pada Persamaan 2.18. Proses ini dapat dilihat pada diagram alir Gambar 3.5. Gambar 3.5(a) menunjukkan hasil gambar yang telah melalui proses perhitungan gradien pada arah  $x$  dan Gambar 3.5(b) menunjukkan hasil gambar yang telah melalui proses perhitungan gradien pada arah  $y$ .

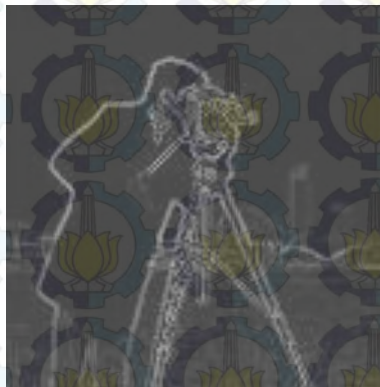
### 3.2.3.2 Perhitungan *Global Stimulus* Gambar

Setelah melakukan perhitungan terhadap gradien gambar, maka dilakukan perhitungan global stimulus gambar.

Perhitungan ini bertujuan agar hasil dari perhitungan gradien gambar tidak memiliki hasil yang negatif. Proses ini dihitung dengan menggunakan rumus yang ada pada Persamaan 2.20 dan diagram alir dapat dilihat pada Gambar 3.7. Gambar 3.6 menunjukkan hasil gambar yang telah melalui proses perhitungan global stimulus.

### 3.2.3.3 Proses Deteksi Tepi Citra

Setelah mendapatkan nilai dari perhitungan gradien dan global stimulus gambar, maka proses inti yaitu mendeteksi tepi citra dilakukan. Proses dimulai dengan menentukan nilai awal untuk feromon tipe 1 yang dibawa oleh semut, kemudian melakukan penyimpanan terhadap posisi semut yang sekarang. Selanjutnya adalah penentuan sistem konektivitas piksel yang menggunakan sistem delapan konektivitas. Untuk menghindari adanya nilai piksel yang melewati ukuran atau *range* dari gambar, maka dilakukan filterisasi terlebih dahulu agar semut bergerak sesuai dengan aturan konektivitas yang telah ditentukan.

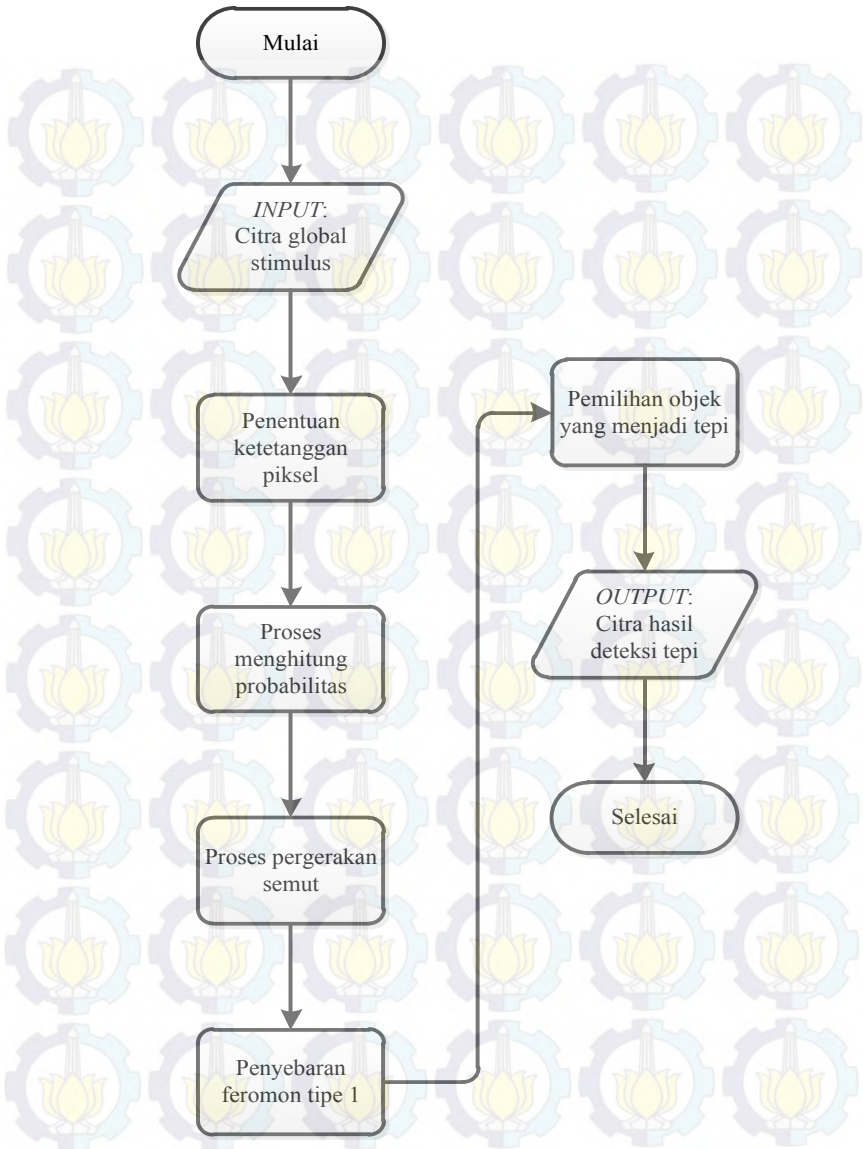


**Gambar 3.6 Gambar Hasil Proses Perhitungan Global Stimulus**

Setelah melakukan filterisasi, maka langkah selanjutnya adalah menghitung probabilitas perpindahan semut dari posisi semut yang sekarang. Probabilitasnya dihitung dengan menggunakan rumus pada Persamaan 2.16. Dari empat konektivitas yang merupakan kandidat piksel untuk pergerakan semut, piksel tersebut masing-masing memiliki nilai probabilitas yang berbeda. Keempat nilai probabilitas tersebut disimpan yang kemudian akan digunakan untuk proses selanjutnya. Empat nilai hasil probabilitas tersebut dipilih secara acak untuk menentukan piksel mana yang akan ditempati oleh semut. Setelah pemilihan piksel secara acak berdasarkan nilai probabilitasnya, maka dilakukan penyimpanan terhadap posisi semut yang baru. Kemudian pemberian nilai terhadap feromon yang disebarkan oleh semut untuk setiap pergerakan dari piksel satu ke yang lainnya. Hasil gambar yang telah terdeteksi tepinya ditunjukkan pada Gambar 3.7. Proses ini secara garis besar dapat dilihat pada diagram alir pada Gambar 3.8. Setelah semut bergerak dari satu piksel menuju piksel yang lainnya, feromon yang dibawa oleh semut diperbaharui.



**Gambar 3.7 Citra Hasil Deteksi Tepi**



**Gambar 3.8 Diagram Alir Tahap Proses Deteksi Tepi**

### 3.2.3.4 Proses *Update* Feromon

Terdapat dua jenis feromon pada algoritma ini dimana kedua feromon tersebut tidak saling bergantung dan tidak dapat digabungkan. Feromon tipe 1 adalah feromon yang dibawa oleh semut saat melakukan perpindahan dari piksel satu ke piksel lainnya yang diberi nilai satu. Feromon ini akan diperbaharui dengan menggunakan rumus pada Persamaan 2.17. Feromon tipe 2 adalah karakteristik yang mirip dengan feromon pada gambar yang digunakan untuk menuntun semut agar tidak bergantung pada feromon tipe 1. Feromon ini tidak perlu diperbaharui dan diperoleh dengan cara mencari gradien dari gambar seperti yang telah dijelaskan pada sub bab 2.7.

### 3.3 Perancangan Antarmuka

Gambar 3.9 memperlihatkan rancangan antarmuka untuk perangkat lunak proses deteksi tepi gambar dengan menggunakan algoritma *ant-inspired*.



**Gambar 3.9 Rancangan Antarmuka Perangkat Lunak Implementasi Deteksi Tepi Gambar Menggunakan Algoritma *Ant-inspired***

Yang pertama kali perlu dilakukan saat menjalankan perangkat lunak ini adalah menentukan citra yang akan dideteksi tepinya. Terdapat menu *pop up* yang digunakan untuk melihat citra hasil dari algoritma yang digunakan.

Setelah citra asli (*input*) muncul maka akan ditampilkan citra yang sudah terdeteksi tepinya dengan memilih menu *pop up* "Citra Keluaran" pada GUI. Disini juga akan ditampilkan citra yang telah diproses menjadi format keabuan.

## BAB 4 IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi perangkat lunak yang meliputi algoritma dan kode program yang terdapat dalam perangkat lunak. Pada tahap implementasi dari tiap fungsi akan dijelaskan mengenai parameter *input*, *output*, dan beberapa keterangan yang berhubungan dengan program dan teori.

### 4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1. Pada tabel ini akan dijelaskan mengenai prosesor, memori, sistem operasi, dan perangkat pengembang yang digunakan selama proses implementasi tugas akhir ini.

**Tabel 4.1 Lingkungan Implementasi Perangkat Lunak**

Perangkat	Spesifikasi
Perangkat Keras	Prosesor: Intel(R) Core(TM)2 Duo CPU E7300 @2.66GHz(2 CPUs), ~2.7GHz
	Memori: 2 GB
Perangkat Lunak	Sistem Operasi: Windows 7 Professional 64-bit (6.1, Build 7601)
	Perangkat Pengembang: MATLAB R2008a, Microsoft Visio 2010, Microsoft Word 2010

## 4.2 Fungsi Utama

Fungsi utama yang digunakan adalah `main_GUI`. Fungsi tersebut memiliki operasi pengolahan data yang berupa pengambilan data gambar citra.

Nama *File* : `main_GUI.m`

Nama Fungsi : `main_GUI`

1.	<code>function</code> varargout = <code>main_GUI</code> (varargin)
2.	...
3.	<code>function</code> pushbutton1_Callback(hObject, eventdata, handles)
4.	[filename, pathname] = <code>uigetfile</code> ({'*.jpg'; '*.png'; '*.tif'; '*.tiff'});
5.	...
6.	<code>input</code> = <code>imread</code> ([pathname filename]);
7.	<code>handles.input</code> = <code>input</code> ;
8.	<code>axes</code> ( <code>handles.axes1</code> );
9.	<code>imshow</code> ( <code>input</code> );

**Gambar 4.1 Kode Sumber Mengambil Data Citra Masukan**

Gambar 4.1 merupakan fungsi untuk mengambil data masukan gambar sebelum citra tersebut akan di proses selanjutnya. Setelah mendapatkan data masukan kemudian akan diproses menggunakan fungsi pada Gambar 4.2.

1.	<code>function</code> popupmenu1_Callback(hObject, eventdata, handles)
2.	...
3.	<code>if</code> <code>idMode</code> ==1
4.	<code>axes</code> ( <code>handles.axes1</code> );
5.	<code>imshow</code> ( <code>input</code> );



6.	<code>elseif idMode ==2</code>
7.	<code>axes(handles.axes1);</code>
8.	<code>imshow(imgAbu);</code>
9.	<code>elseif idMode ==3</code>
10.	<code>axes(handles.axes1);</code>
11.	<code>imshow(hasil);</code>
12.	<code>end</code>

**Gambar 4.2 Kode Sumber Bagian-Bagian yang Akan Diproses Ketika Tombol *Pop Up* Menu Dipilih**

### 4.3 Tahap Deteksi Tepi

Fungsi deteksi tepi yang digunakan adalah `antinspired.m`. Fungsi tersebut memiliki operasi pengolahan data yang berupa inisialisasi posisi semut, jumlah semut, operasi untuk menemukan tepi citra, dan *update* feromon.

Nama File : `antinspired.m`

Nama Fungsi : `antinspired`

Input : `input`

Output : `hasil_gambar`

1.	<code>function [hasil_gambar] = antinspired(input)</code>
2.	<code>...</code>
3.	<code>[baris, kolom] = size(imgAbu);</code>
4.	<code>...</code>
5.	<code>jumlah semut temp = (baris*kolom);</code>
6.	<code>jumlah semut = round(jumlah semut temp/9);</code>
7.	<code>...</code>
8.	<code>posisi semut = zeros(jumlah semut, 2);</code>
9.	<code>posisi semut sblm = posisi semut;</code>
10.	<code>...</code>
11.	<code>%inisialisasi posisi semut secara random</code>
12.	
13.	<code>rand('state', sum(clock));</code>
14.	<code>temp = rand(jumlah semut,2);</code>
15.	

16.	<code>%posisi semut</code>
17.	<code>posisi_semut(:,1) = round(1+ (baris-1) * temp(:,1)); %baris</code>
18.	<code>posisi_semut(:,2) = round(1+ (kolom-1) * temp(:,2)); %kolom</code>
19.	<code>...</code>
20.	<code>%5.Menghitung IM</code>
21.	<code>[gx, gy] = gradient(imgAbu);</code>
22.	<code></code>
23.	<code>%6.Menghitung matrix global stimulus / norm %euclidean</code>
24.	<code>...</code>
25.	<code>gx 2 = gx.^2;</code>
26.	<code>gy 2 = gy.^2;</code>
27.	<code>imgGra = sqrt(gx 2 + gy 2);</code>
28.	<code></code>
29.	<code>p = imgGra;</code>
30.	<code>v = imgGra;</code>

**Gambar 4.3 Kode Gradien Gambar**

Untuk tahap mengambil gradien gambar dan stimulus gambar ditunjukkan pada Gambar 4.3. Pada gambar tersebut terdapat beberapa langkah proses mencari gradien gambar dan stimulus gambar yang dimulai pada baris 21 sampai 27.

31.	<code>...</code>
32.	<code>for jumlah_iterasi = 1 : (total_jumlah_iterasi)</code>
33.	<code>...</code>
34.	<code>...</code>
35.	<code>for semut id = 1:jumlah semut</code>
36.	<code>...</code>
37.	<code>baris semut skrg = posisi semut(semut id,1);</code>
38.	<code>kolom semut skrg = posisi semut(semut id,2);</code>
39.	<code></code>
40.	<code>...</code>
41.	<code>%(ketetapan Vonn Neuman)</code>
42.	<code>range_pencarian_temp = [baris_temp-1 kolom temp; baris_temp kolom temp+1; baris_temp+1 kolom temp; baris temp kolom temp-1];</code>
43.	<code></code>
44.	<code>temp = find(range_pencarian_temp(:,1)&gt;=1 &amp; range_pencarian_temp(:,1)&lt;=baris &amp; range_pencarian_temp(:,2)&gt;=1 &amp; range_pencarian temp(:,2)&lt;=kolom);</code>
45.	<code>range pencarian = range pencarian temp(temp, :);</code>

46.	...
47.	
48.	local_stimulus_v = zeros(size(range_pencarian,1),1);
49.	local_stimulus_p = zeros(size(range_pencarian,1),1);
50.	
51.	for kk = 1:size(range_pencarian,1)
52.	local_stimulus_v(kk) = v(range_pencarian(kk,1), range_pencarian(kk,2));
53.	local_stimulus_p(kk) = p(range_pencarian(kk,1), range_pencarian(kk,2));
54.	end
55.	probabilitas = (local_stimulus_v.^alpha) .* (local_stimulus_p.^beta) ./ (sum(sum((local_stimulus_v.^alpha) .* (local_stimulus_p.^beta))));
56.	
57.	posisi_semut_sblm(semut_id,1) = posisi_semut(semut_id,1);
58.	posisi_semut_sblm(semut_id,2) = posisi_semut(semut_id,2);
59.	
60.	rand('state', sum(100*clock));
61.	tempC = find(cumsum(probabilitas)>=rand(1), 1);
62.	
63.	baris_semut_slnjt = range_pencarian(tempC,1);
64.	kolom_semut_slnjt = range_pencarian(tempC,2);
65.	
66.	if length(baris_semut_slnjt) == 0
67.	baris_semut_slnjt = baris_semut_skrng;
68.	=kolom_semut_slnjt = kolom_semut_skrng;
69.	end
70.	
71.	If baris_semut_slnjt ~= posisi_semut_sblm(semut_id,1) kolom_semut_slnjt ~= posisi_semut_sblm(semut_id,2)
72.	posisi_semut(semut_id,1) = baris_semut_slnjt;
73.	posisi_semut(semut_id,2) = kolom_semut_slnjt;
74.	end
75.	
76.	end
77.	
78.	p = ((1-rho).*p) + (lambda);
79.	
80.	end
81.	...

82.	<code>p = abs(p&gt;=T);</code>
83.	<code>p = p.*255;</code>
84.	<code>Hasil gambar = p;</code>

#### Gambar 4.4 Kode Deteksi Tepi

Untuk tahap deteksi tepi gambar ditunjukkan pada Gambar 4.4. Pada gambar tersebut terdapat beberapa langkah proses deteksi tepi yang dimulai pada baris 31 sampai 87. Pada baris 58 merupakan implemmentasi probabilitas yang digunakan semut untuk bergerak dari satu piksel ke piksel lainnya sesuai dengan Persamaan 2.16. Setelah semut bergerak dari satu piksel menuju piksel yang lainnya, feromon yang dibawa oleh semut diperbaharui sesuai dengan Persamaan 2.17 yang diimplementasikan pada baris 81.

#### 4.4 Tahap Menghitung Akurasi

Dalam sub bab ini, akan diimplementasikan proses perhitungan akurasi seperti dalam Persamaan 2.23. Kode perhitungan akurasi dari hasil segmentasi citra seperti terdapat dalam Gambar 4.5.

Nama *File* : `accuracy.m`

Nama Fungsi : `accuracy`

*Input* : `segmen, hasil`

*Output* : `score, nilaiakurasi`

1.	<code>function [score,nilaiakurasi] = accuracy(pathname, filename, hasil)</code>
2.	<code>...</code>
3.	<code>tp = sum(sum(and(hasil,segmen)));</code>
4.	<code>e = sum(sum(segmen&gt;0));</code>
5.	<code>fn = e-tp; %</code>
6.	<code>tn = sum(sum((or(hasil,segmen)-1)*-1));</code>
7.	<code>fp = sum(sum(((hasil&gt;0)-(segmen&gt;0))&gt;0));</code>

8.	...
9.	$\text{score} = \text{tp} + \text{tn};$
10.	$\text{nilaiakurasi} = (\text{tp} + \text{tn}) / (\text{tp} + \text{fp} + \text{fn} + \text{tn}) * 100;$

**Gambar 4.5 Kode Menghitung Skor dan Akurasi**

Akurasi deteksi tepi mempunyai *range* antara 0% sampai 100%, dengan ketentuan semakin mendekati 100% maka akan semakin baik. Untuk mengetahui tingkat kebenaran akurasi citra dilakukan dengan membandingkan citra hasil deteksi tepi yang ditunjukkan dengan variabel hasil dengan citra manual segmentasi dari *database* yang disimpan dalam variabel *segmen*.

## BAB 5

### UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Pembahasan yang dikemukakan meliputi data uji coba, hasil uji coba, dan evaluasi.

#### 5.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan dalam pembuatan tugas akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk proses deteksi tepi citra dengan menggunakan algoritma *ant-inspired*. Lingkungan uji coba merupakan komputer atau perangkat keras tempat uji coba sistem. Berikut adalah lingkungan uji coba yang digunakan pada tugas akhir ini yang ditunjukkan pada Tabel 5.1.

**Tabel 5.1 Lingkungan Uji Coba**

Perangkat	Spesifikasi
Perangkat Keras	Prosesor: Intel(R) Core(TM)2 Duo CPU E7300 @2.66GHz(2 CPUs), ~2.7GHz
	Memori: 2 GB
Perangkat Lunak	Sistem Operasi: Windows 7 Enterprise 32-bit (6.1, Build 7600)
	Perangkat Pengembang: MATLAB R2008a, Microsoft Visio 2010, Microsoft Word 2010

## 5.2 Data Uji Coba

Data yang digunakan pada uji coba ini adalah citra berbagai ukuran dan berupa citra berwarna. Citra yang akan digunakan ini diambil dari *website Berkeley* [18] dan beberapa data ada yang diambil dari data uji citra pada *website* yang berkaitan dengan segmentasi dan deteksi tepi citra. Adapun data uji coba ditunjukkan pada Tabel 5.2.

**Tabel 5.2 Data Citra Uji Coba**

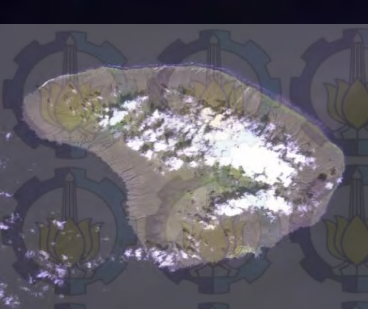

Nama Citra	Citra
hawk.jpg	
butterfly.jpg	
church.jpg	


Nama Citra	Citra
bird.jpg	 A photograph of a dark bird, possibly a crow or raven, in flight against a solid blue background. The bird is captured in mid-flight, with its wings spread and tail feathers visible.
apple.jpg	 A photograph of the Apple logo, which is a blue, glossy, three-dimensional apple with a bite taken out of it. The logo is centered against a plain white background.
box.jpg	 A photograph of an open cardboard box, likely for an Apple product. The box is brown and is open, revealing a black surface with the white Apple logo. A small key is attached to the bottom of the box.
twoglass.jpg	 A photograph of two black mugs. The mugs are cylindrical with a handle on the right side. They are placed on a light-colored surface against a light blue background.



Nama Citra	Citra
wolf.jpg	 A photograph of a wolf standing on a snowy slope at night. The wolf is illuminated by a light source, possibly a fire or a lamp, creating a strong contrast against the dark background. The snow is bright white, and the background is dark and textured.
woman.jpg	 A close-up portrait of a woman wearing a light-colored, knitted headscarf. She is looking directly at the camera with a neutral expression. Her hands are resting near her face, and she is wearing a dark-colored top.
swan.jpg	 A photograph of a black swan swimming in a pond. The swan is the central focus, with its long neck curved upwards. The water is calm, and the swan's reflection is visible in the water below it.
pulau1.jpg	 A 3D topographic map of a green island. The island is shown in a perspective view, with the terrain colored in shades of green and brown to represent elevation. The island is surrounded by a dark blue ocean.

Nama Citra	Citra
pulau2.jpg	
pulau3.jpg	
pulau4.jpg	
pulau5.jpg	

Nama Citra	Citra
pulau6.jpg	
pulau7.jpg	
pulau8.jpg	
pulau9.jpg	

Nama Citra	Citra
pulau10.jpg	

### 5.3 Skenario Uji Coba

Pada bagian ini dijelaskan mengenai skenario uji coba yang telah dilakukan. Terdapat beberapa skenario, diantaranya yaitu:

1. Perbandingan nilai akurasi dari citra hasil deteksi tepi dengan menggunakan nilai *threshold* yang berbeda-beda.
2. Perbandingan nilai akurasi citra hasil deteksi tepi dengan menggunakan jumlah iterasi yang berbeda-beda.
3. Perbandingan nilai akurasi citra hasil deteksi tepi dengan menggunakan nilai *alpha* dan *beta* yang berbeda-beda.
4. Perbandingan nilai akurasi citra hasil deteksi tepi dengan algoritma deteksi tepi *canny*.

Keempat skenario tersebut akan diujicobakan pada perangkat lunak MATLAB R2008a dan dijelaskan pada sub bab berikut ini.

#### 5.3.1 Perbandingan Citra Hasil dengan Nilai *Threshold* yang Berbeda-Beda

Pada skenario uji coba yang pertama ini akan dibandingkan nilai akurasi citra yang dihasilkan dengan

menggunakan nilai *threshold* yang berbeda. Uji coba pertama skenario ini akan diujikan pada citra hawk.jpg. Pada skenario citra hawk.jpg, masing-masing nilai *threshold* yang digunakan untuk mendeteksi tepi citra adalah 0,1 dan 0,3, iterasi yang digunakan sejumlah 500, nilai *alpha* dan *beta* yang digunakan adalah 1 dan 0.

- Nilai *Threshold* 0,1

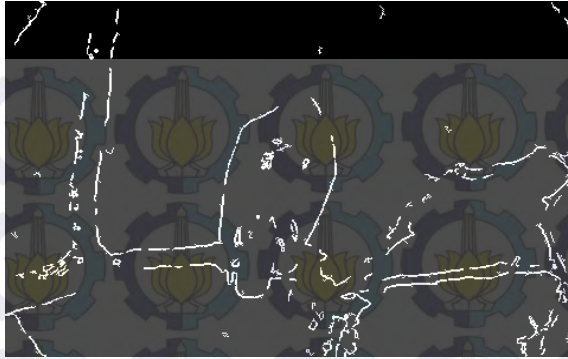
Hasil deteksi tepi citra hawk.jpg yang menggunakan nilai *threshold* sebesar 0,1 ditunjukkan pada Gambar 5.1. Terlihat pada citra ini garis tepi sudah tersambung dengan baik dan akurasi menunjukkan persentase sebesar 87,50%.

- Nilai *Threshold* 0,3

Hasil deteksi tepi citra hawk.jpg menggunakan nilai *threshold* sebesar 0.3 ditunjukkan pada Gambar 5.2. Dari uji coba tersebut didapatkan nilai akurasi sebesar 76,63%. Uji coba lainnya dilakukan pada citra butterfly.jpg yang ditunjukkan pada Gambar 5.3 dan Gambar 5.4.



**Gambar 5.1** Citra hawk.jpg Menggunakan *Threshold* 0,1



**Gambar 5.2** Citra hawk.jpg Menggunakan *Threshold* 0,3

- Nilai *Threshold* 0,1

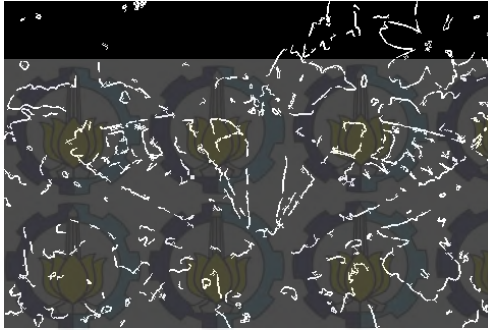
Hasil deteksi tepi citra butterfly.jpg yang menggunakan nilai *threshold* sebesar 0.1 ditunjukkan pada Gambar 5.3. Terlihat pada citra ini garis tepi sudah tersambung dengan baik dan akurasi menunjukkan persentase sebesar 71,45%.

- Nilai *Threshold* 0,3

Hasil deteksi tepi citra butterfly.jpg yang menggunakan nilai *threshold* sebesar 0,3 ditunjukkan pada Gambar 5.4.



**Gambar 5.3** Citra butterfly.jpg Menggunakan *Threshold* 0,1



**Gambar 5.4 Citra butterfly.jpg Menggunakan *Threshold* 0,3**

Dari uji coba tersebut didapatkan nilai akurasi sebesar 70,33%. Dari hasil uji coba pada skenario yang pertama ini dapat dilihat pada Tabel 5.3 dan Tabel 5.4 bahwa citra hasil deteksi tepi yang menggunakan *threshold* yang mendekati nol akan menghasilkan nilai akurasi yang semakin tinggi.

Dari Tabel 5.3 dan Tabel 5.4 dapat ditunjukkan bahwa nilai akurasi rata-rata untuk citra masukan dengan menggunakan *threshold* 0,1 sebesar 82,38% dan nilai akurasi rata-rata untuk citra masukan dengan menggunakan *threshold* 0,3 sebesar 71,92%.

**Tabel 5.3 Akurasi Uji Coba *Threshold* 0,1**

Citra Masukan	Akurasi (%)
hawk.jpg	87,50
butterfly.jpg	71,45
church.jpg	85,61
bird.jpg	96,98
apple.jpg	96,94
box.jpg	86,74
twoglass.jpg	93,99
wolf.jpg	85,54
woman.jpg	78,49

Citra Masukan	Akurasi (%)
swan.jpg	96,79
pulau1.jpg	70,62
pulau2.jpg	79,16
pulau3.jpg	79,93
pulau4.jpg	77,44
pulau5.jpg	78,58
pulau6.jpg	62,57
pulau7.jpg	70,50
pulau8.jpg	84,60
pulau9.jpg	96,22
pulau10.jpg	78,43
<b>Rata-rata</b>	<b>82,38</b>

**Tabel 5.4 Akurasi Uji Coba *Threshold* 0,3**

Citra Masukan	Akurasi (%)
hawk.jpg	76,63
butterfly.jpg	70,33
church.jpg	85,10
bird.jpg	79,71
apple.jpg	87,45
box.jpg	73,34
twoglass.jpg	61,91
wolf.jpg	77,55
woman.jpg	72,48
swan.jpg	89,16
pulau1.jpg	63,98
pulau2.jpg	62,00
pulau3.jpg	62,24
pulau4.jpg	67,57
pulau5.jpg	61,18
pulau6.jpg	54,63
pulau7.jpg	62,61
pulau8.jpg	75,63
pulau9.jpg	86,26
pulau10.jpg	68,59
<b>Rata-rata</b>	<b>71,92</b>



### 5.3.2 Perbandingan Citra Hasil Menggunakan Jumlah Iterasi yang Berbeda-Beda

Pada skenario uji coba yang kedua ini akan dibandingkan nilai akurasi citra yang dihasilkan dengan menggunakan jumlah iterasi yang berbeda. Uji coba pertama skenario ini akan diujikan pada citra church.jpg. Pada skenario citra church.jpg, jumlah iterasi yang digunakan untuk mendeteksi tepi citra adalah 50, 100, dan 500, *threshold* yang digunakan adalah 0,1, nilai *alpha* dan *beta* yang digunakan masing-masing adalah 1 dan 0.

- Jumlah Iterasi 50

Hasil deteksi tepi citra yang menggunakan nilai iterasi sejumlah 50 yang ditunjukkan pada Gambar 5.5. Terlihat pada citra ini garis tepi belum tersambung dengan baik dan akurasi menunjukkan persentase sebesar 85,27%.

- Jumlah Iterasi 100

Hasil deteksi tepi citra yang menggunakan nilai iterasi sejumlah 100 ditunjukkan pada Gambar 5.6. Terlihat pada citra ini garis tepi belum tersambung dengan baik dan akurasi menunjukkan persentase sebesar 85,47%.

- Jumlah Iterasi 500

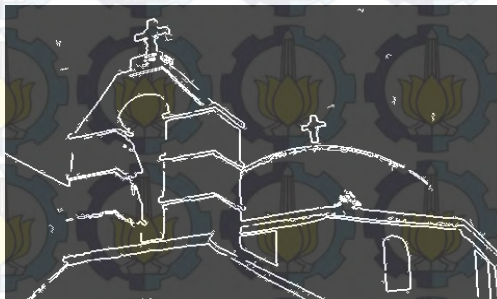
Hasil deteksi tepi citra yang menggunakan nilai iterasi sejumlah 500 ditunjukkan pada Gambar 5.7. Dari uji coba tersebut didapatkan nilai akurasi sebesar 85,61%. Uji coba lainnya dilakukan pada citra bird.jpg yang ditunjukkan pada Gambar 5.8, Gambar 5.9, dan Gambar 5.10.



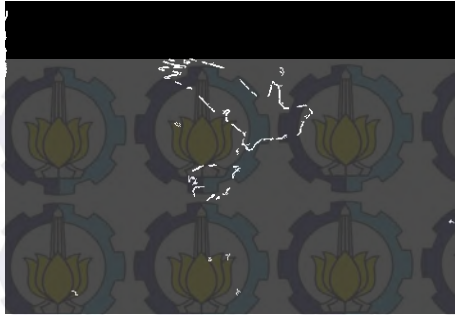
**Gambar 5.5 Citra church.jpg Menggunakan 50 Iterasi**



**Gambar 5.6 Citra church.jpg Menggunakan 100 Iterasi**



**Gambar 5.7 Citra church.jpg Menggunakan 500 Iterasi**



**Gambar 5.8 Citra bird.jpg Menggunakan 50 Iterasi**

- Jumlah Iterasi 50

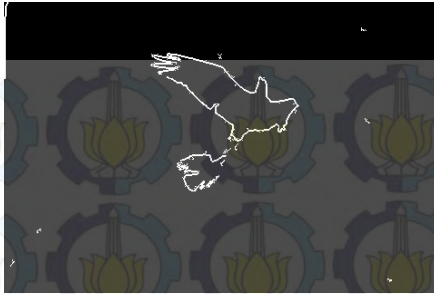
Hasil deteksi tepi citra yang menggunakan nilai iterasi sejumlah 50 ditunjukkan pada Gambar 5.8. Terlihat pada citra ini garis tepi belum tersambung dengan baik dan akurasi menunjukkan persentase sebesar 86,78%.

- Jumlah Iterasi 100

Hasil deteksi tepi citra yang menggunakan nilai iterasi sejumlah 100 ditunjukkan pada Gambar 5.9. Terlihat pada citra ini garis tepi belum tersambung dengan baik dan akurasi menunjukkan persentase sebesar 96,87%.



**Gambar 5.9 Citra bird.jpg Menggunakan 100 Iterasi**



**Gambar 5.10 Citra bird.jpg Menggunakan 500 Iterasi**

- Jumlah Iterasi 500

Hasil deteksi tepi citra yang menggunakan nilai iterasi sejumlah 500 ditunjukkan pada Gambar 5.10. Dari uji coba tersebut didapatkan nilai akurasi sebesar 96,98%. Dari hasil uji coba pada skenario yang kedua ini dapat dilihat pada Tabel 5.5, Tabel 5.6, dan Tabel 5.7 bahwa citra dengan iterasi yang semakin besar akan menghasilkan nilai akurasi yang semakin tinggi.

**Tabel 5.5 Akurasi Uji Coba Iterasi 50**

Citra Masukan	Akurasi (%)
hawk.jpg	86,73
butterfly.jpg	70,34
church.jpg	85,27
bird.jpg	86,78
apple.jpg	78,23
box.jpg	66,66
twoglass.jpg	78,36
wolf.jpg	78,57
woman.jpg	69,19
swan.jpg	76,99
pulau1.jpg	53,30
pulau2.jpg	51,57
pulau3.jpg	52,00
pulau4.jpg	57,45

Citra Masukan	Akurasi (%)
pulau5.jpg	50,88
pulau6.jpg	54,53
pulau7.jpg	52,22
pulau8.jpg	65,47
pulau9.jpg	66,28
pulau10.jpg	58,52
<b>Rata-rata</b>	<b>66,97</b>

**Tabel 5.6 Akurasi Uji Coba Iterasi 100**

Citra Masukan	Akurasi (%)
hawk.jpg	87,26
butterfly.jpg	70,98
church.jpg	85,47
bird.jpg	96,87
apple.jpg	87,16
box.jpg	76,78
twoglass.jpg	89,22
wolf.jpg	88,15
woman.jpg	72,93
swan.jpg	92,21
pulau1.jpg	61,59
pulau2.jpg	60,17
pulau3.jpg	60,79
pulau4.jpg	67,45
pulau5.jpg	69,35
pulau6.jpg	63,25
pulau7.jpg	61,26
pulau8.jpg	64,97
pulau9.jpg	86,24
pulau10.jpg	68,83
<b>Rata-rata</b>	<b>75,55</b>

**Tabel 5.7 Akurasi Uji Coba Iterasi 500**

Citra Masukan	Akurasi (%)
hawk.jpg	87,50
butterfly.jpg	71,45

Citra Masukan	Akurasi (%)
church.jpg	85,61
bird.jpg	96,98
apple.jpg	96,94
box.jpg	86,74
twoglass.jpg	93,99
wolf.jpg	85,54
woman.jpg	78,49
swan.jpg	96,79
pulau1.jpg	70,62
pulau2.jpg	79,16
pulau3.jpg	79,93
pulau4.jpg	77,44
pulau5.jpg	78,58
pulau6.jpg	62,57
pulau7.jpg	70,50
pulau8.jpg	84,60
pulau9.jpg	96,22
pulau10.jpg	78,43
<b>Rata-rata</b>	<b>82,38</b>

Dari Tabel 5.5, Tabel 5.6, dan Tabel 5.7 dapat ditunjukkan bahwa nilai akurasi rata-rata untuk citra masukan dengan menggunakan iterasi 50 adalah sebesar 66,97%. Nilai akurasi rata-rata untuk citra masukan dengan menggunakan iterasi 100 adalah sebesar 75,55% dan nilai akurasi rata-rata untuk citra masukan dengan menggunakan iterasi 500 adalah sebesar 82,38%.

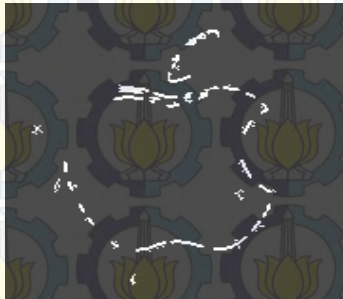
### 5.3.3 Perbandingan Hasil Citra Menggunakan Nilai *Alpha* dan *Beta* yang Berbeda

Pada skenario uji coba yang ketiga ini citra akan dibandingkan nilai akurasi citra yang dihasilkan dengan menggunakan nilai *alpha* dan *beta* yang berbeda. Uji coba pertama skenario ini akan diujikan pada citra apple.jpg. Pada

skenario citra *apple.jpg*, nilai *alpha* yang digunakan untuk mendeteksi tepi citra adalah 0.5 dan nilai *beta* yang digunakan adalah 0.5. Untuk uji coba parameter berikutnya, nilai *alpha* yang digunakan adalah 1 dan nilai *beta* yang digunakan adalah 0. Iterasi yang digunakan sejumlah 500 dan nilai *threshold* yang digunakan adalah 0,1.

- Nilai *Alpha* 0,5 dan *Beta* 0,5

Hasil deteksi tepi citra yang menggunakan nilai *alpha* dan *beta* masing-masing sebesar 0,5 dan 0,5 ditunjukkan pada Gambar 5.11.



**Gambar 5.11** Citra *apple.jpg* Menggunakan *Alpha* 0,5 dan *Beta* 0,5



**Gambar 5.12** Citra *apple.jpg* Menggunakan *Alpha* 1 dan *Beta* 0

Terlihat pada Gambar 5.11 garis tepi belum tersambung dengan baik dan akurasi menunjukkan persentase sebesar 67,12%.

- Nilai *Alpha* 1 dan *Beta* 0

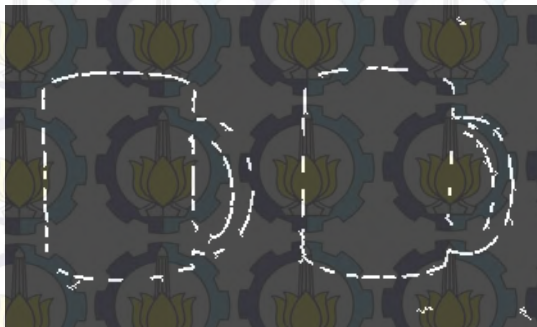
Hasil deteksi tepi citra yang menggunakan nilai *alpha* dan *beta* masing-masing sebesar 1 dan 0 ditunjukkan pada Gambar 5.12. Dari uji coba tersebut didapatkan nilai akurasi sebesar 96,94%. Uji coba lainnya dilakukan pada citra twoglass.jpg yang ditunjukkan pada Gambar 5.13 dan Gambar 5.14.

- Nilai *Alpha* 0.5 dan *Beta* 0.5

Hasil deteksi tepi citra yang menggunakan nilai *alpha* dan *beta* masing-masing sebesar 0,5 dan 0,5 ditunjukkan pada Gambar 5.13. Dari uji coba tersebut didapatkan nilai akurasi sebesar 76,28%.

- Nilai *Alpha* 1 *Beta* 0

Hasil deteksi tepi citra yang menggunakan nilai nilai *alpha* dan *beta* masing-masing sebesar 1 dan 0 ditunjukkan pada Gambar 5.14. Dari uji coba tersebut didapatkan nilai akurasi sebesar 93,99%.



Gambar 5.13 Citra twoblack.jpg Menggunakan *Alpha* 0,5 dan *Beta* 0,5





**Gambar 5.14** Citra twoblack.jpg Menggunakan  $\alpha$  1 dan  $\beta$  0

**Tabel 5.8** Akurasi Uji Coba  $\alpha$  0,5 dan  $\beta$  0,5

Citra Masukan	Akurasi (%)
hawk.jpg	86,63
butterfly.jpg	70,24
church.jpg	85,13
bird.jpg	74,63
apple.jpg	67,12
box.jpg	76,59
twoglass.jpg	76,28
wolf.jpg	88,85
woman.jpg	70,11
swan.jpg	86,94
pulau1.jpg	63,29
pulau2.jpg	61,48
pulau3.jpg	71,95
pulau4.jpg	77,48
pulau5.jpg	70,35
pulau6.jpg	64,49
pulau7.jpg	72,05
pulau8.jpg	75,42
pulau9.jpg	76,42
pulau10.jpg	68,58
<b>Rata-rata</b>	<b>74,20</b>

**Tabel 5.9 Akurasi Uji Coba  $\alpha$  1 dan  $\beta$  0**

Citra Masukan	Akurasi (%)
hawk.jpg	87,50
butterfly.jpg	71,45
church.jpg	85,61
bird.jpg	96,98
apple.jpg	96,94
box.jpg	86,74
twoglass.jpg	93,99
wolf.jpg	85,54
woman.jpg	78,49
swan.jpg	96,79
pulau1.jpg	70,62
pulau2.jpg	79,16
pulau3.jpg	79,93
pulau4.jpg	77,44
pulau5.jpg	78,58
pulau6.jpg	62,57
pulau7.jpg	70,50
pulau8.jpg	84,60
pulau9.jpg	96,22
pulau10.jpg	78,43
<b>Rata-rata</b>	<b>82,38</b>

Dari hasil uji coba pada skenario yang ketiga ini dapat dilihat pada Tabel 5.8 dan Tabel 5.9 citra dengan nilai  $\alpha$  1 dan  $\beta$  0 akan menghasilkan nilai akurasi yang lebih tinggi.

Dari Tabel 5.8 dan 5.9 dapat ditunjukkan nilai akurasi rata-rata untuk citra masukan dengan menggunakan iterasi  $\alpha$  0,5 dan  $\beta$  0,5 adalah sebesar 74,20%. Nilai akurasi rata-rata untuk citra masukan dengan menggunakan  $\alpha$  1 dan  $\beta$  0 adalah sebesar 82,38%.

### 5.3.4 Perbandingan Akurasi Citra Hasil dengan Algoritma Deteksi Tepi Lain

Pada skenario uji coba ini akan dibandingkan nilai akurasi citra yang dihasilkan dengan algoritma deteksi tepi konvensional, yaitu *canny*. Pada scenario ini, nilai *threshold* yang digunakan untuk mendeteksi tepi citra adalah 0,1, iterasi yang digunakan sejumlah 500, nilai *alpha* dan *beta* yang digunakan masing-masing adalah 1 dan 0. Metode *canny* yang digunakan adalah fungsi *canny* yang ada pada perangkat lunak MATLAB R2008a. Untuk lebih jelasnya Tabel 5.10 menunjukkan perbandingan akurasi beserta *running time* antara metode *ant-inspired* dan metode *canny*.

Dari hasil uji coba pada skenario yang keempat ini dapat dilihat pada Tabel 5.10 bahwa metode *ant-inspired* memiliki rata-rata nilai akurasi yang sedikit lebih tinggi jika dibandingkan dengan metode *canny*. Untuk *running time* dapat dilihat bahwa metode *ant-inspired* memiliki *running time* yang sangat lambat jika dibandingkan dengan *running time* metode *canny* yang sangat cepat.

**Tabel 5.10 Perbandingan Akurasi Citra Hasil dengan Metode Canny**

Nama Citra	Akurasi Citra Hasil Deteksi Tepi (%)	Running Time (s)	Akurasi Citra Hasil Metode Canny (%)	Running Time (s)
hawk.jpg	87,50	682,13	85,43	0,182514
butterfly.jpg	71,45	653,69	68,33	0,194533
church.jpg	85,61	616,24	83,14	0,190011
bird.jpg	96,98	616,97	86,65	0,187003
apple.jpg	96,94	146,93	86,53	0,046562
box.jpg	86,74	1095,38	80,14	0,282692
twoglass.jpg	93,99	350,24	84,73	0,079024

Nama Citra	Akurasi Citra Hasil Deteksi Tepi (%)	Running Time (s)	Akurasi Citra Hasil Metode Canny (%)	Running Time (s)
wolf.jpg	85,54	607,99	83,29	0,189314
woman.jpg	78,49	591,15	78,93	0,446784
swan.jpg	96,79	348,69	88,74	0,138302
pulau1.jpg	70,62	758,45	73,20	0,479686
pulau2.jpg	79,16	650,32	73,18	0,209622
pulau3.jpg	79,93	1023,58	63,11	0,336291
pulau4.jpg	77,44	973,18	75,86	0,313234
pulau5.jpg	78,58	959,65	70,28	0,263566
pulau6.jpg	62,57	1332,92	73,88	0,458218
pulau7.jpg	70,50	1550,90	61,05	0,562460
pulau8.jpg	84,60	492,62	75,80	0,150695
pulau9.jpg	96,22	1185,94	74,26	0,392363
pulau10.jpg	78,43	407,71	67,67	0,131105
<b>Rata-rata</b>	<b>82,38</b>	<b>752,234</b>	<b>75,92</b>	<b>0,261699</b>

#### 5.4 Evaluasi

Dari hasil uji coba yang telah dilakukan, beberapa parameter yang digunakan selama uji coba memberikan pengaruh terhadap hasil deteksi tepi citra dengan menggunakan algoritma *ant-inspired*.

Perubahan nilai *threshold* memberikan pengaruh terhadap hasil deteksi tepi citra yang diolah. Semakin kecil ukuran *threshold* yang dipilih maka kualitas hasil deteksi tepi terhadap citra akan semakin baik.

Perubahan jumlah iterasi juga memberikan pengaruh terhadap hasil deteksi tepi dan *running time* dari citra. Semakin banyak jumlah iterasi yang dimasukkan, maka semakin baik hasil deteksi tepi citra. Akan tetapi *running time* yang dihasilkan juga lebih lama. Semakin sedikit jumlah

iterasi yang dimasukkan, maka semakin jelek hasil dari deteksi tepi citra dengan *running time* yang semakin singkat.

Perubahan nilai *alpha* dan *beta* memberikan pengaruh pula terhadap hasil deteksi tepi dari citra. Dari hasil uji coba nilai *alpha* 0,5 dan *beta* 0,5 memberikan nilai akurasi yang lebih rendah jika dibandingkan dengan citra hasil deteksi tepi dengan nilai *alpha* 1 dan *beta* 0.

Selain itu, perbandingan hasil deteksi tepi citra menggunakan metode *ant-inspired* dan metode *canny* dapat dikatakan bahwa akurasinya lebih besar walaupun metode tersebut mahal dari segi komputasi. Waktu eksekusi sangat bergantung pada ukuran citra masukan dan keadaan citra. Semakin besar ukuran citra atau semakin rumit bentuk objek dalam citra maka waktu eksekusi semakin lama.

## BAB 6

### KESIMPULAN DAN SARAN

#### 6.1 Kesimpulan

Berdasarkan hasil uji coba yang telah dilakukan, terdapat beberapa kesimpulan yang dapat diambil, yaitu:

1. Nilai *threshold* 0,1 menghasilkan ekstraksi tepi citra yang optimum dengan nilai rata-rata akurasi sebesar 82,38%.
2. Berdasarkan eksperimen, semakin besar jumlah iterasi, maka semakin meningkat pula kemampuan algoritma ini dalam mendeteksi tepi. Namun waktu komputasi yang diperlukan juga semakin lama seiring dengan peningkatan jumlah iterasi.
3. Nilai *alpha* 1 dan *beta* 0 menghasilkan nilai akurasi dan ekstraksi fitur yang optimum.
4. Metode *ant-inspired* memiliki nilai akurasi yang lebih tinggi sebesar 8,68% jika dibandingkan dengan metode deteksi tepi *canny* walaupun metode *ant-inspired* memiliki waktu komputasi yang jauh lebih lambat.
5. Metode *ant-inspired* cocok untuk mendeteksi citra dengan tingkat detail citra yang rendah (sederhana) seperti citra hawk.jpg, bird.jpg, dan apple.jpg. Pada citra tersebut *background* gambar hanya berupa awan dan memiliki warna yang homogen. Pada citra yang kompleks seperti butterfly.jpg *background* gambar berupa tumbuhan yang memiliki daun, kayu, dan sebagainya sehingga detail citra ini lebih tinggi. Akurasi yang dihasilkan pada citra butterfly.jpg ini lebih rendah jika dibandingkan dengan citra hawk.jpg, bird.jpg, dan apple.jpg.

#### 6.2 Saran

Adapun saran yang ingin disampaikan penulis terkait dengan pengerjaan tugas akhir ini adalah:

1. Metode *ant-inspired* dengan citra yang sudah dikonversi ke format keabuan dapat diterapkan pada citra berwarna yang dapat diproses dengan menggunakan pendekatan algoritma lain.
2. Untuk ke depannya diharapkan algoritma ini dapat diimplementasikan pada sebuah *graphics processing unit* dengan menggunakan CUDA (arsitektur komputasi paralel yang dimiliki oleh perusahaan Nvidia) dimana secara signifikan akan dapat meningkatkan kecepatan proses algoritma *ant-inspired*. Implementasi algoritma ini pada *graphics processing unit* mampu meningkatkan kecepatan komputasi sebanyak 100 kali lipat.

## DAFTAR PUSTAKA

- [1] S. Ali Etemad and Tony White, "An Ant-inspired Algorithm For Detection of Image Edge Features," *Science Direct, Applied Soft Computing*, vol. 11, pp. 4883-4893, 2011.
- [2] Z. C. S. S Hlaing and M. A. Khine, "An Ant Colony Optimization Algorithm for Solving Travelling Salesman Problem," *International Conference on Information Communication and Management*, vol. 16, pp. 54-59, 2011.
- [3] R.C Gonzales and R.E Woods, *Digital Image Processing Using MATLAB*. New Jersey, USA: Pearson Prentice-Hall, 2004.
- [4] Munir Rinaldi, *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Bandung, Indonesia: Informatika Bandung, 2004.
- [5] Ian T. Young, Jan J., and Lucas J., "Fundamentals of Image Processing," Delft University of Technology, Delft, 1995.
- [6] Robyn Owens. (1997, Oktober) Binary Images. [Online]. [http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/OWENS/LECT2/node3.html](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html)
- [7] Rafael C. Gonzalez and Richard E. Woods, "Introduction," in *Digital Image Processing*. Upper Saddle River, New Jersey, United States of America: Pearson Education, Inc, 2008, ch. 1.
- [8] David Jacobs, "Image Gradients," University of Maryland, College Park, Class Notes 2005.
- [9] G. T. Shrivakshan and Dr. C. Chandrasekar, "A Comparison of Various Edge Detection techniques used in Image Processing," *IJCSI International Journal of Computer Science*, vol. 9, no. 1, pp. 269-276, 2012.



- [10] De-Sian Lu and Chien-Chang Chen, "Edge Detection Improvement by Ant Colony Optimization," *Science Direct, Pattern Recognition Letters*, vol. 9, no. 1, pp. 416-425, 2008.
- [11] Ramesh C. Jain, "Edge Detection," in *Machine Vision*. San Diego, USA: McGraw-Hill Higher Education, ch. 5.
- [12] Ehsan N., Sara S., and Hamid. H, "Edge Detection Techiques: Evaluations and Comparisons," *Applied Mathematical Science*, vol. 2, no. 31, pp. 1507-1520, 2008.
- [13] M. Dorigo, V.Maniezzo, and A. Colorni, "Ant System: Optimization By A Colony of Cooperating Agents," *IEEE Transactions on System, Man, and Cybernetics-Part B*, vol. 26, no. 1, pp. 29-41, 1996.
- [14] R. Fisher, S.Perkins, A. Walker, and E. Wolfart. (2003) Connected Component Labelling. [Online]. <http://homepage.inf.ed.ac.uk/rbf/HIPR2/label.htm#1>
- [15] W.K Pratt, *Digital Image Processing*. Wiley, Hoboken, United States of America, 2007.
- [16] S.Y Zhu, K.N. Plataniotis, and A.N. Venetsanopoulos, "Comprehensive Analysis of Edge Detection In Color Image Processing," *Optical Engineering*, vol. 38, no. 4, pp. 612-625, 1999.
- [17] Mehdi Hosseinzadeh Aghdam, Nasser Ghassem Aghae, and Mohammad Ehsan Basiri, "Text Feature Selection Using Ant Colony Optimization," *Science Direct*, March 2009.
- [18] David R. Martin. (2009) The Berkeley Segmentation Dataset and Benchmark. [Online]. <http://www.eecs.berkeley.edu/Research/Projects/CS/vison/bsds/>

## BIODATA PENULIS






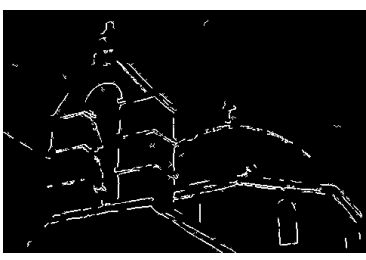





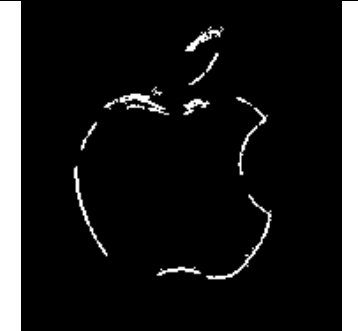

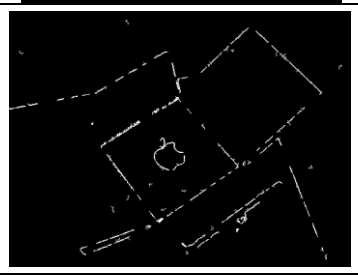

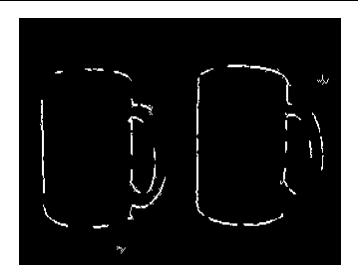
Ida Ayu Putu Kristiantari, lahir di Denpasar pada tanggal 21 Oktober 1991 dan dibesarkan di kota Kupang, merupakan anak pertama dari tiga bersaudara. Penulis telah menempuh pendidikan di SDN Bonipoi 1 Kupang (1997), SDN 1 Sumerta Denpasar (1997-2000), SDN Bertingkat Naikoten 1 Kupang (2000-2003), SMPN 1 Kupang (2003-2006), SMAN 4 Denpasar (2006-2009), dan terakhir sebagai mahasiswa Teknik Informatika ITS (2009-2014). Selama kuliah, penulis aktif menjadi anggota Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) dan juga sebagai staf Departemen Pengembangan dan Profesi. Semasa di perkuliahan penulis pernah diberi amanah untuk menjadi asisten Mata Kuliah Matematika Diskrit (2011), Aljabar Linier (2011) dan juga pernah menjadi Finalis Lomba Simulasi Bisnis geMasTIK V di Institut Teknologi Bandung. Di sela-sela kesibukan kampus, penulis suka menghabiskan waktu bersama teman-teman dan membaca buku. Penulis dapat dihubungi melalui e-mail: [kristiantari21@gmail.com](mailto:kristiantari21@gmail.com).

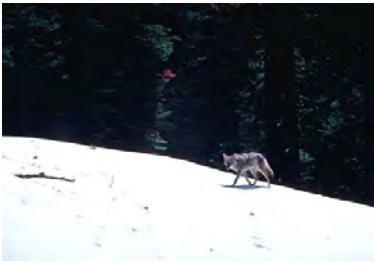



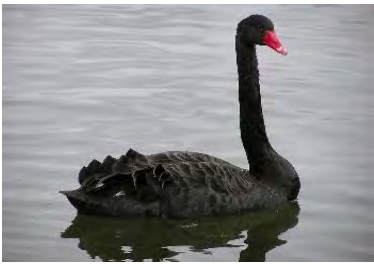

**LAMPIRAN  
HASIL UJI COBA**


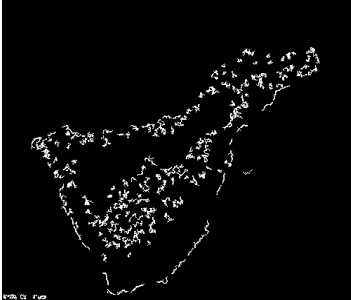




**A. Citra-Citra Masukan dan Keluaran**






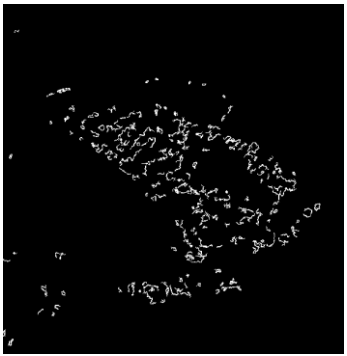
**Tabel A.1 Hasil Uji Coba Menggunakan *Threshold 0,3***









No	Citra Masukan	Citra Keluaran
1		
2		
3		

No	Citra Masukan	Citra Keluaran
4		
5		
6		
7		

No	Citra Masukan	Citra Keluaran
8		
9		
10		








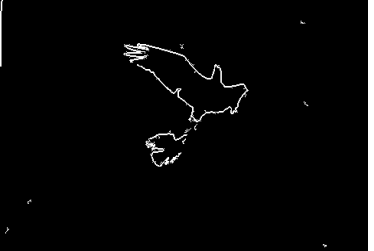
No	Citra Masukan	Citra Keluaran
11		
12		
13		




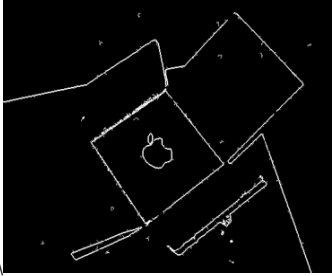


No	Citra Masukan	Citra Keluaran
14		
15		
16		


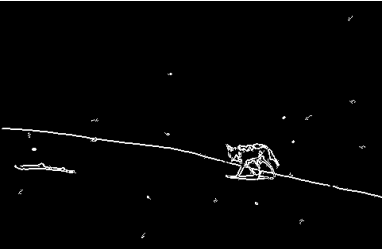


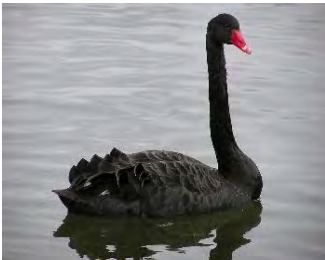
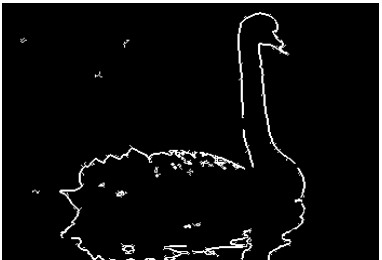
No	Citra Masukan	Citra Keluaran
17		
18		
19		
20		


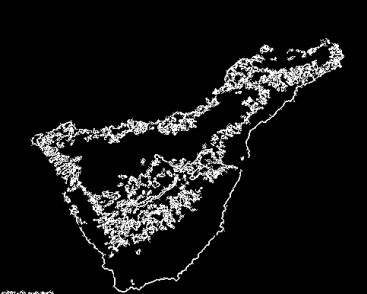












Tabel A.2 Hasil Uji Coba Menggunakan *Threshold* 0,1


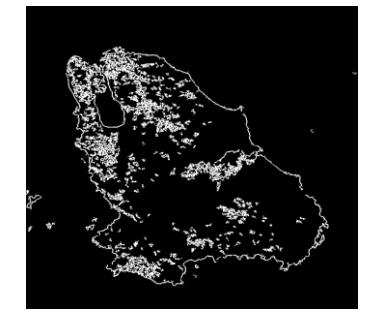

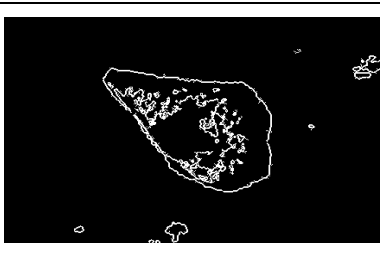



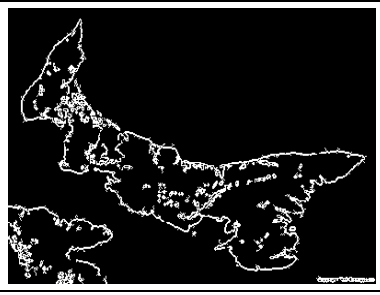
No	Citra Masukan	Citra Keluaran
1		
2		
3		
4		

No	Citra Masukan	Citra Keluaran
5		
6		
7		









No	Citra Masukan	Citra Keluaran
8	 A photograph of a small, light-colored dog standing on a snowy slope. The background is dark and appears to be a forest or wooded area.	 The edge detection result of the input image. The dog and the snowy slope are highlighted in white against a black background, showing the boundaries and features of the scene.
9	 A close-up portrait of a woman with dark hair, wearing a light-colored, textured headscarf. She is resting her chin on her hands.	 The edge detection result of the input image. The woman's face, headscarf, and hands are highlighted in white against a black background, showing the contours and features.
10	 A photograph of a black swan swimming in a body of water. The swan is facing right, and its reflection is visible in the water.	 The edge detection result of the input image. The swan and its reflection are highlighted in white against a black background, showing the outline and features of the bird.


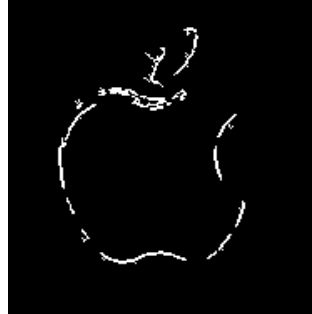



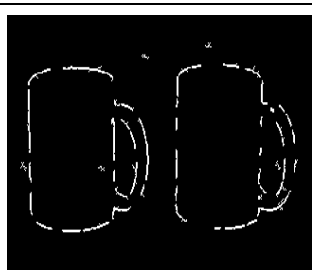
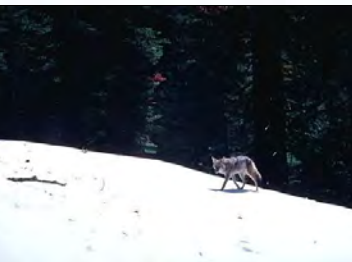

No	Citra Masukan	Citra Keluaran
11		
12		
13		

No	Citra Masukan	Citra Keluaran
14		
15		
16		








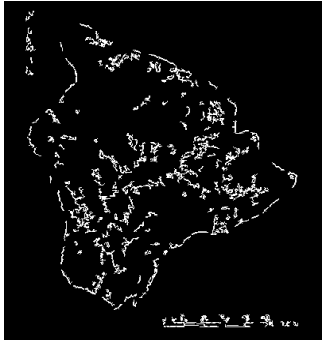
No	Citra Masukan	Citra Keluaran
17		
18		
19		
20		







Tabel A.3 Hasil Uji Coba Menggunakan 50 Iterasi


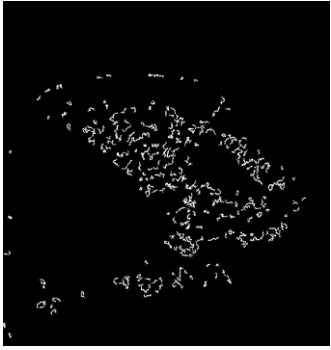



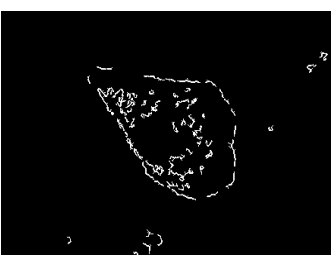
No	Citra Masukan	Citra Keluaran
1		
2		
3		
4		





No	Citra Masukan	Citra Keluaran
5		
6		
7		
8		



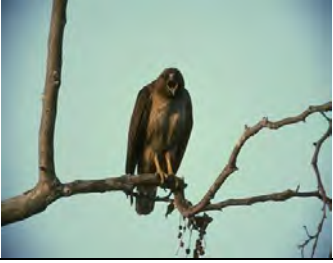



No	Citra Masukan	Citra Keluaran
9		
10		
11		
12		


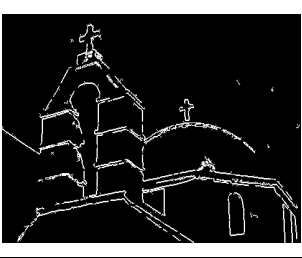





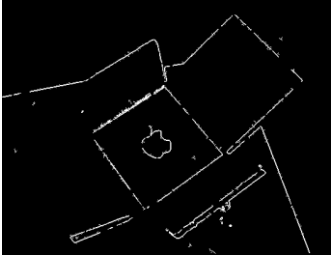
No	Citra Masukan	Citra Keluaran
13		
14		
15		


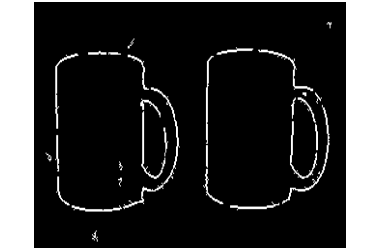
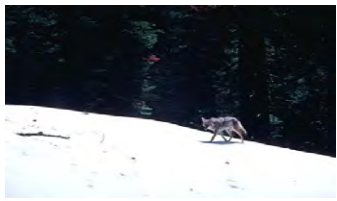




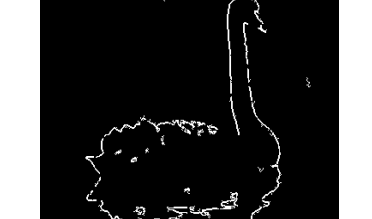
No	Citra Masukan	Citra Keluaran
16	 An aerial photograph of a forested island with a dark background. The island is roughly oval-shaped with a jagged coastline. The forest is a mix of green and brown, with a prominent white area in the center.	 A binary edge detection image of the island from the input image. The edges are highlighted in white against a black background, showing the coastline and internal features.
17	 An aerial photograph of a large island with a blue background. The island is irregularly shaped with a jagged coastline. The terrain is a mix of green and brown, with a prominent white area in the center.	 A binary edge detection image of the island from the input image. The edges are highlighted in white against a black background, showing the coastline and internal features.
18	 An aerial photograph of a small island with a dark background. The island is roughly triangular with a jagged coastline. The terrain is a mix of green and brown, with a prominent white area in the center.	 A binary edge detection image of the island from the input image. The edges are highlighted in white against a black background, showing the coastline and internal features.







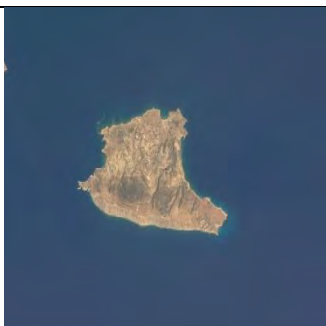

No	Citra Masukan	Citra Keluaran
19		
20		


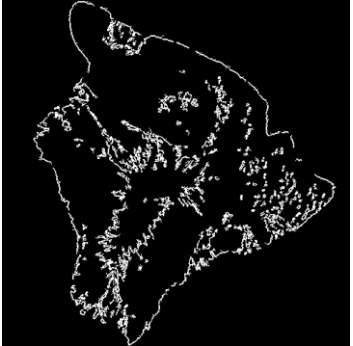

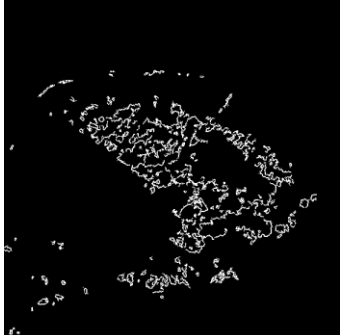


**Tabel A.4 Hasil Uji Coba Menggunakan 100 Iterasi**

No	Citra Masukan	Citra Keluaran
1		
2		







No	Citra Masukan	Citra Keluaran
3		
4		
5		
6		

No	Citra Masukan	Citra Keluaran
7		
8		
9		
10		


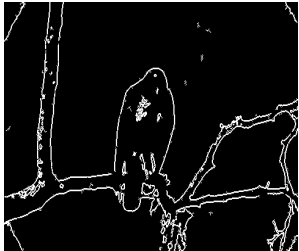
No	Citra Masukan	Citra Keluaran
11		
12		
13		
14		


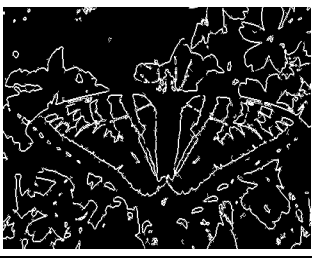


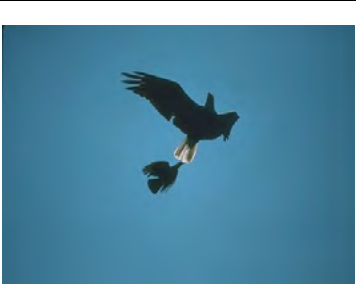
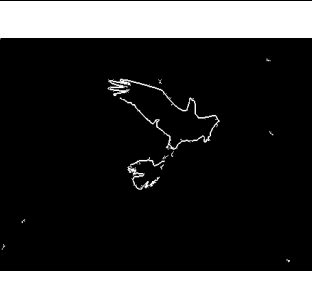

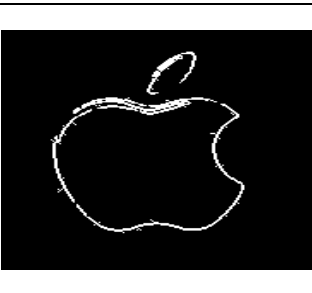
No	Citra Masukan	Citra Keluaran
15		
16		
17		


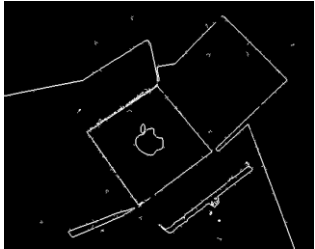

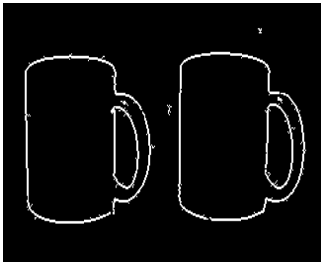
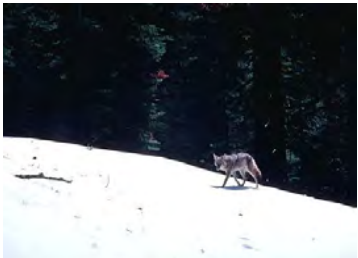
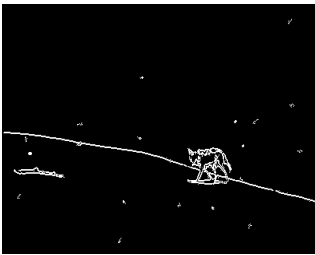





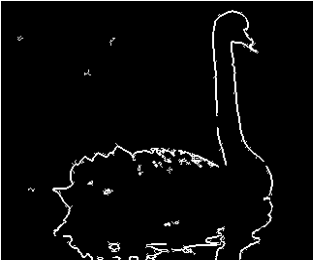

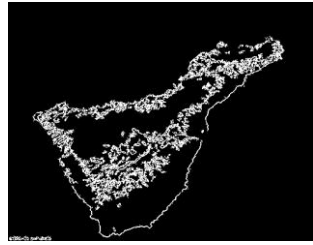




No	Citra Masukan	Citra Keluaran
18		
19		
20		








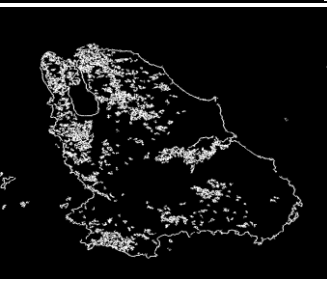
**Tabel A.5 Hasil Uji Coba Menggunakan 500 Iterasi**


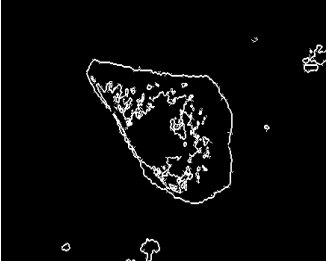

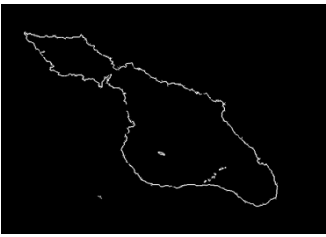

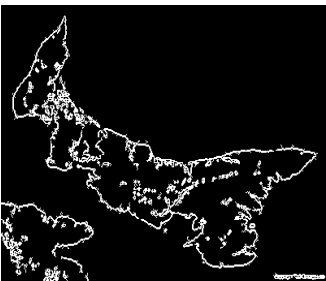
No	Citra Masukan	Citra Keluaran
1		

No	Citra Masukan	Citra Keluaran
2		
3		
4		
5		



No	Citra Masukan	Citra Keluaran
6		
7		
8		
9		




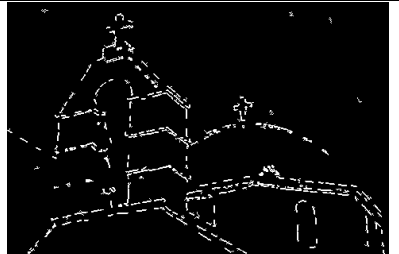
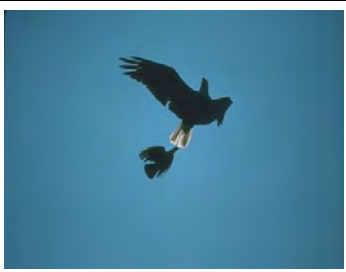
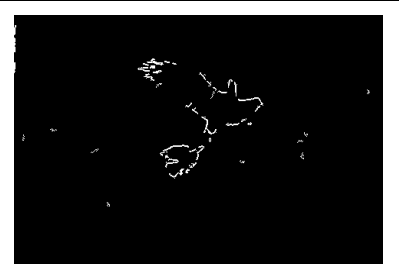


No	Citra Masukan	Citra Keluaran
10		
11		
12		
13		




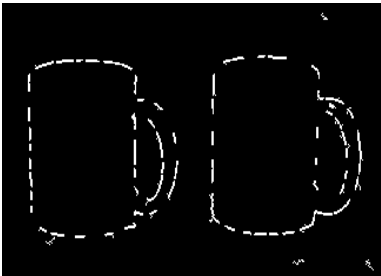

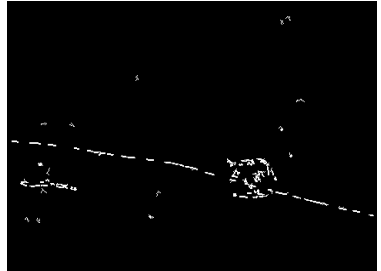
No	Citra Masukan	Citra Keluaran
14		
15		
16		
17		

No	Citra Masukan	Citra Keluaran
18		
19		
20		




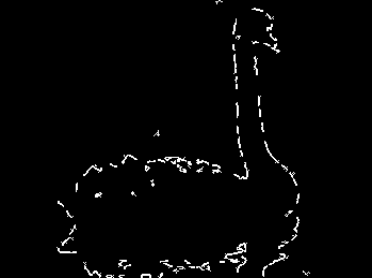


**Tabel A.6 Hasil Uji Coba Menggunakan  $\alpha$  0,5 dan  $\beta$  0,5**









No	Citra Masukan	Citra Keluaran
1		




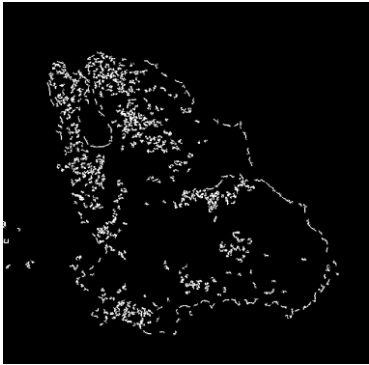


No	Citra Masukan	Citra Keluaran
2		
3		
4		
5		





No	Citra Masukan	Citra Keluaran
6		
7		
8		







No	Citra Masukan	Citra Keluaran
9		
10		
11		


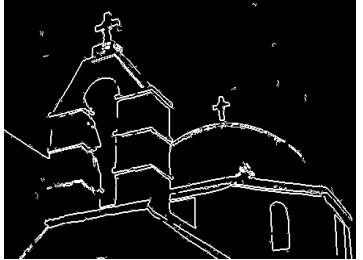

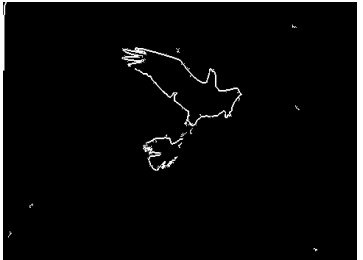



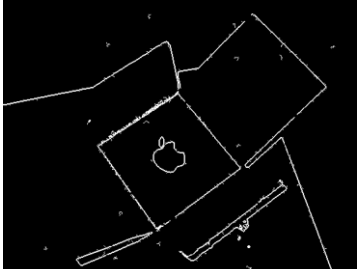
No	Citra Masukan	Citra Keluaran
12		
13		
14		
15		


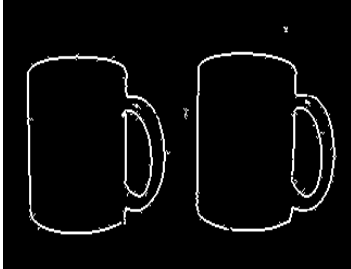




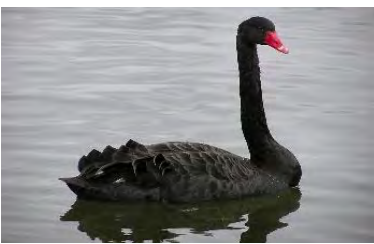
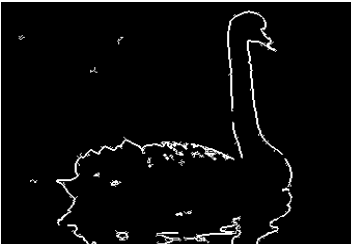
No	Citra Masukan	Citra Keluaran
16		
17		
18		







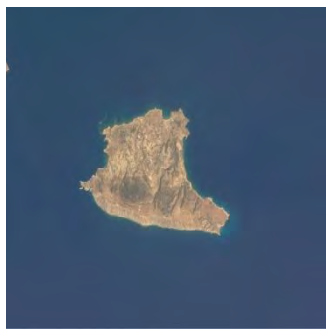

No	Citra Masukan	Citra Keluaran
19		
20		






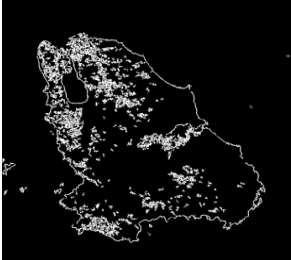

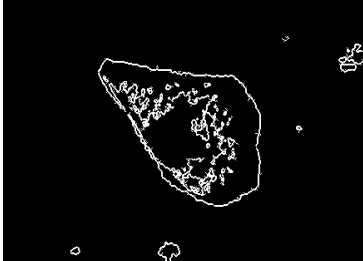
**Tabel A.7 Hasil Uji Coba Menggunakan  $\alpha$  1 dan  $\beta$  0**

No	Citra Masukan	Citra Keluaran
1		
2		

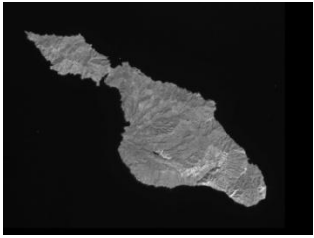


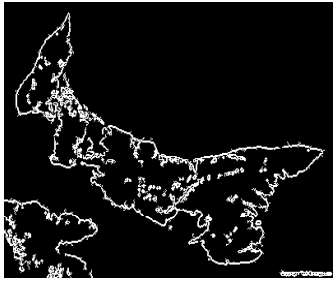
No	Citra Masukan	Citra Keluaran
3		
4		
5		
6		

No	Citra Masukan	Citra Keluaran
7		
8		
9		
10		





No	Citra Masukan	Citra Keluaran
11		
12		
13		
14		



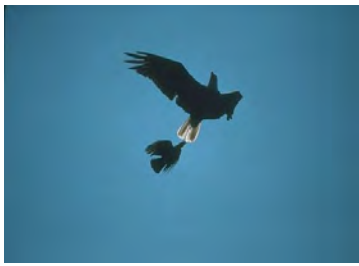


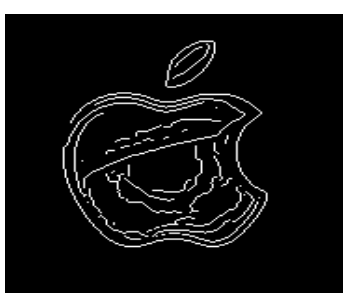


No	Citra Masukan	Citra Keluaran
15		
16		
17		
18		


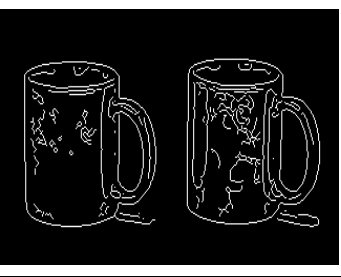







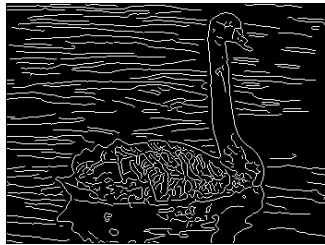

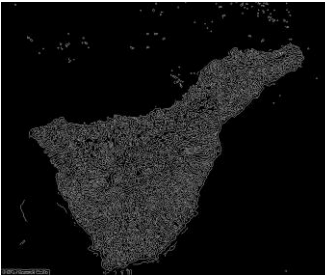

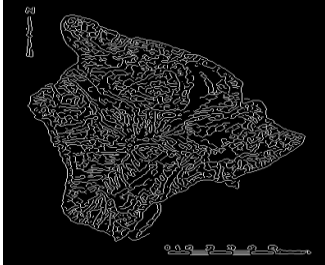

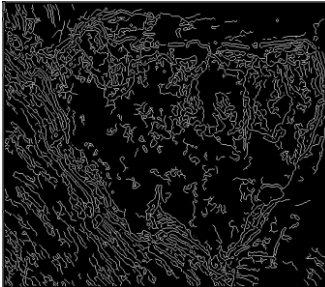
No	Citra Masukan	Citra Keluaran
19		
20		

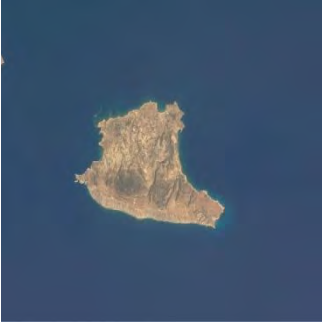
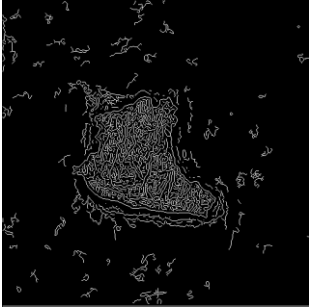



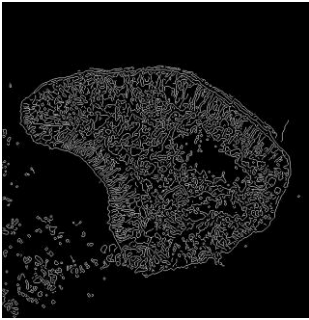
Tabel A.8 Hasil Uji Coba Menggunakan Metode *Canny*


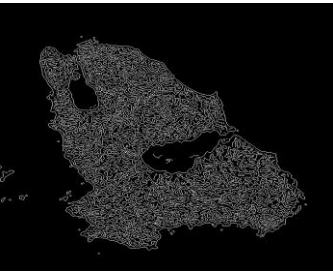

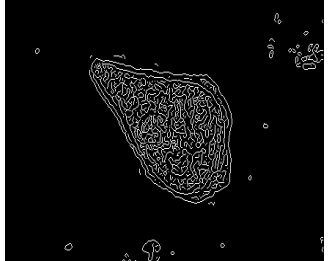


No	Citra Masukan	Citra Keluaran
1		
2		

No	Citra Masukan	Citra Keluaran
3		
4		
5		
6		

No	Citra Masukan	Citra Keluaran
7		
8		
9		

No	Citra Masukan	Citra Keluaran
10		
11		
12		
13		

No	Citra Masukan	Citra Keluaran
14		
15		
16		

No	Citra Masukan	Citra Keluaran
17		
18		
19		
20	