TESIS KI142502

MODIFIKASI DISTRIBUSI SEMUT PADA *ANT COLONY OPTIMIZATION* BERDASARKAN *GRADIENT* UNTUK DETEKSI TEPI CITRA

FEBRI LIANTONI
NRP. 5113201022

DOSEN PEMBIMBING
Dr. Eng. Nanik Suciati, S.Kom, M.Kom
Dr. Eng. Chastine Fatichah, S.Kom, M.Kom

PROGRAM MAGISTER
BIDANG KEAHLIAN KOMPUTASI CERDAS DAN VISUALISASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

THESIS KI142502

Ants Distribution Modification in Ant Colony Optimization Based on Gradient For Image Edge Detection

FEBRI LIANTONI
NRP. 5113201022

SUPERVISOR
Dr. Eng. Nanik Suciati, S.Kom, M.Kom
Dr. Eng. Chastine Fatichah, S.Kom, M.Kom

MASTER PROGRAM
INFORMATICS ENGINEERING
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Febri Liantoni
Nrp. 5113201022

Dengan judul :
Modifikasi Distribusi Semut Pada Ant Colony Optimization Berdasarkan Gradient Untuk Deteksi Tepi Citra

Tanggal Ujian : 13-1-2015
Periode Wisuda : 2014 Gasal

Disetujui oleh:

Dr. Eng. Nanik Suciati, S.Kom, M.Kom (Pembimbing 1)
NIP. 197104281994122001

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom (Pembimbing 2)
NIP. 197512202001122002

Prof.Ir.Handayani Tjandrasa ,M.Sc, Ph.D (Penguji 1)
NIP. 194908231976032001

Anny Yuniarti, S.Kom.,M.Comp.Sc (Penguji 2)
NIP. 198106222005012002

Wijayanti Nurul Khotimah, S.Kom., M.Sc (Penguji 3)
NIP. 198603122012122004

Direktur Program Pascasarjana

Prof.Dr.Ir.Adi Soeprijanto, MT
NIP. 196404051990021001

Modifikasi Distribusi Semut Pada *Ant Colony Optimization* Berdasarkan *Gradient* Untuk Deteksi Tepi Citra

Nama Mahasiswa : Febri Liantoni

NRP : 5113 201 022

Pembimbing : 1. Dr. Eng. Nanik Suciati, S.Kom, M.Kom

2. Dr. Eng. Chastine Fatichah, S.Kom, M.Kom

ABSTRAK

Ant Colony Optimization (ACO) merupakan algoritma optimasi yang terinspirasi oleh tingkah laku semut dalam mencari makan. Karena keunggulan yang dimilikinya, ACO banyak digunakan untuk menyelesaikan permasalahan non-polinomial yang sulit, salah satunya adalah deteksi tepi pada citra. Penerapan ACO pada deteksi tepi telah terbukti berhasil dengan baik, akan tetapi metode penyebaran semut pada ACO sangat mempengaruhi tingkat akurasi.

Pada ACO tradisional semut awal disebarkan secara acak. Kondisi ini dapat menyebabkan ketidakseimbangan distribusi semut yang kemudian mempengaruhi proses penemuan jalur pada deteksi tepi. Berdasarkan permasalahan tersebut, modifikasi distribusi semut pada ACO diusulkan untuk mengoptimalkan penyebaran semut berdasarkan *gradient*. Nilai *gradient* digunakan untuk menentukan penempatan semut. Metode yang diusulkan pada penelitian ini adalah dengan melakukan penyebaran semut berdasarkan nilai *gradient*. Semut tidak disebar secara acak, akan tetapi ditempatkan di *gradient* tertinggi. Cara ini diharapkan dapat digunakan untuk optimasi penemuan jalur. Berdasarkan hasil uji coba, dengan menggunakan ACO modifikasi yang diusulkan dapat diperoleh nilai rata-rata *Peak Signal to Noise Ratio* (PSNR) 12,884. Sedangkan, menggunakan ACO tradisional diperoleh nilai rata-rata PSNR 11,665. Hasil ini menunjukkan bahwa ACO modifikasi mampu menghasilkan citra keluaran yang lebih baik dibandingkan ACO tradisional yang sebaran semut awalnya dilakukan secara acak.

Kata kunci*: Ant Colony Optimization, gradient,* deteksi tepi, *peak signal to noise ratio*

Ants Distribution Modification in Ant Colony Optimization Based on Gradient For Image Edge Detection

Student Name : Febri Liantoni
NRP : 5113 201 022
Supervisor : 1. Dr. Eng. Nanik Suciati, S.Kom, M.Kom
2. Dr. Eng. Chastine Fatichah, S.Kom, M.Kom

ABSTRACT

Ant Colony Optimization (ACO) is an optimization algorithm which is inspired by the behavior of ants when finding foods. Because of its advantages, ACO is widely used to solve the difficult non-polynomial problems, one of them is image edge detection. Application of ACO on edge detection has been proven to work well, but the method of spreading ants in ACO greatly affects the accuracy.

In traditional ACO, the initial ant randomly distributed. This condition can cause an imbalance ants distribution which can affect the path discovery process on edge detection. Based on this problem, a modified ant distribution in ACO is proposed to optimize the deployment of ant based gradient. Gradient value is used to determine the placement of the ants. Ants are not distributed randomly, but placed in the highest gradient. This method is expected to be used for optimization path discovery. Based on the test results, using the proposed ACO modification can be obtained average values of Peak Signal to Noise Ratio (PSNR) 12.884. Meanwhile, using traditional ACO obtained an average value of PSNR 11.665. These results indicate that the ACO modification capable of generating output image better than traditional ACO which ants are initially distributed randomly.

Keywords: Ant Colony Optimization, gradient, edge detection, peak signal to noise ratio

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur alhamdulilah, saya panjatkan kepada Allah SWT yang senantiasa melimpahkan rahmat dan hidayah-Nya sehingga dapat terselesaikannya buku tesis yang berjudul.

"MODIFIKASI DISTRIBUSI SEMUT PADA *ANT COLONY OPTIMIZATION* BERDASARKAN *GRADIENT* UNTUK DETEKSI TEPI CITRA "

Buku tesis ini ditulis dengan maksud untuk memenuhi salah satu guna menyelesaikan program pendidikan S2 Teknik Informatika di Institut Teknologi Sepuluh Nopember.

Dengan selesainya buku tesis ini penyusun berharap semoga buku ini dapat membawa manfaat bagi pembaca pada umumnya dan juga bagi penulis pada khususnya serta semua pihak yang berkepentingan. Penulis juga berharap agar penelitian ini dapat dikembangkan lebih lanjut sehingga dapat digunakan sebaik-baiknya untuk mendukung perkembangan ilmu pengetahuan.

Penulis sadar bahwasanya masih banyak kesalahan dan kekurangan dalam penulisan buku tesis ini, untuk itu penulis mohon maaf dan mengharapkan kritik dan saran yang bersifat membangun kami harapkan untuk perbaikan selanjutnya.

Wassalamu'alaikum Wr. Wb.

Surabaya, Januari 2015

Febri Liantoni

UCAPAN TERIMA KASIH

Segala puji bagi Allah SWT sehingga buku tesis ini dapat diselesaikan dengan baik. Meski dalam menyelesaikan buku ini banyak ditemui kesulitan, namun berkat bantuan dan bimbingan berbagai pihak, akhirnya Penulis berhasil menyelesaikan buku ini. Pada kesempatan ini Penulis ingin mengucapkan terima kasih kepada pihak-pihak yang membantu penulis dalam penulisan buku tesis ini.

1. Allah SWT atas semunya. Hanya kepadaMu lah aku memohon dan meminta pertolongan. Hamba bersyukur atas segala karuniaMu.
2. Nabi besar kita, Muhammad SAW. Shalawat dan Salam akan selalu tercurah untukmu.
3. Bapak dan Ibuku yang sudah merawat dan selalu mendoakan, serta mendukungku untuk melanjutkan pendidikan yang lebih tinggi sehingga aku bisa mencapai ini semua. Terima kasih atas kasih sayang kalian, semoga aku bisa menjadi anak yang bisa membahagiakan dan membanggakan kalian.
4. Adik-adikku yang memberi dukungan dan menjadi sumber penyemangat.
5. Bapak Prof.Dr.Ir.Adi Soeprijanto, MT selaku Direktur Program Pascasarjana ITS.
6. Ibu Dr. Eng. Nanik Suciati, S.Kom, M.Kom. selaku pembimbing pertama yang telah meluangkan waktu untuk memberikan saran berharga dan semangat serta menjadi sumber inspirasi Penulis dalam setiap melakukan penelitian.
7. Ibu Dr. Eng. Chastine Fatichah, S.Kom, M.Kom. selaku pembimbing kedua yang dengan sabar membimbing Penulis dan mendorong Penulis untuk terus maju.
8. Bapak Waskitho Wibisono, S.Kom, M.Eng, Ph.D. selaku Kepala Program Studi Pascasarjana Teknik Informatika ITS yang memberi arahan dan masukan kepada Penulis.
9. Kepada para Dosen Penguji, Prof.Ir.Handayani Tjandrasa ,M.Sc, Ph.D., Anny Yuniarti, S.Kom., M.Comp.Sc., Wijayanti Nurul Khotimah, S.Kom, M.Sc. yang telah memberikan masukan berharga.

10. Seluruh Bapak Ibu dosen Teknik Informatika - Institut Teknologi Sepuluh Nopember yang telah membagikan ilmunya, mohon maaf penyusun tidak dapat menyebutkan satu persatu.
11. Tri Hadiah Muliawati, Kartika Candra Kirana & Fidi Wincoko Putro, terima kasih sudah memeperkenalkan metode ACO sehingga bisa menjadi topik penelitian Penulis.
12. Kepada teman seperjuangan dan seangkatan, yang tidak dapat disebutkan satu persatu disini, yang telah menjadi teman diskusi yang baik sekaligus sebagai tempat untuk mencari motivasi. Selamat meraih cita-cita teman.
13. Teman-teman kos keputih gg 1 no 5, terima kasih atas semangat kalian.
14. Arief, Leo, Andika, Anas, Swastiko, pertahankan kekompakan Calpico Flaser Band. Terima kasih atas bantuan kalian dalam upaya menghilangkan kepenatan.
15. Semua pihak yang sudah membantu namun tak cukup saya sebutkan satu persatu disini. Terima kasih banyak, semoga Allah SWT membalas kebaikan kalian.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi untuk teknik pengolahan citra telah berkembang pesat. Berbagai teknik dikembangkan untuk mempermudah pekerjaan manusia, baik sebagai pengolah citra, analis citra maupun penggguna citra untuk berbagai tujuan dan keperluan. Seringkali citra yang digunakan tidak dalam kondisi yang ideal. Kondisi ini berupa gangguan bayangan, foto dan gambar kabur. Berbagai masalah tentang analisa dan perencanaan dapat berpengaruh pada hasil interpelasi, oleh karena itu diperlukan teknik pengolahan citra untuk memperoleh citra yang ideal.

Salah satu teknik pengolahan citra yang biasa dilakukan adalah deteksi tepi. Deteksi tepi merupakan proses penggalian informasi tepi dari sebuah gambar. Hal ini dianggap sebagai langkah dasar yang digunakan dalam sebagian besar aplikasi pengolahan citra (Verma. dkk, 2011). Tepi dalam sebuah gambar dapat dianggap sebagai batas antara dua daerah yang berbeda.

Banyak pendekatan telah digunakan untuk melakukan deteksi tepi pada sebuah gambar. Beberapa metode yang umum digunakan adalah sobel, prewitt, dan canny (Verma. dkk, 2010),), (Charu & Sunanda, 2013). Penelitian terbaru menggunakan *Ant Colony Optimization* (ACO) untuk melakukan deteksi tepi pada gambar. ACO merupakan metode heuristik yang meniru perilaku semut untuk memecahkan masalah optimasi diskrit (Dorigo. dkk, 2004). Semut menggunakan senyawa kimia khusus yang disebut feromon untuk menandai jalur antara sumber makanan dan koloni mereka. Jalur feromon digunakan oleh semut berikutnya sebagai referensi untuk mencari makanan karena feromon meningkatkan kemungkinan jalan untuk dipilih.

Ada beberapa keuntungan dari ACO yang bisa digunakan untuk memecahkan berbagai masalah Non Polinomial (NP) (Dorigo. dkk, 2004), seperti t*raveling salesmanp problem* (TSP), deteksi tepi, *network packet routing, vehicular routing*,

quadratic assignment problem, dan sebagainya. Pada penelitian ini ACO digunakan untuk tujuan deteksi tepi pada sebuah gambar.

Pada ACO tradisional, proses penyebaran semut awal dilakukan secara acak. Kondisi ini dapat menyebabkan ketidakseimbangan distribusi semut yang kemudian mempengaruhi proses penemuan jalur. Ada beberapa penelitian yang membahas mengenai penyebaran semut awal. Seperti yang telah dilakukan Rahebi. dkk. (2010) melakukan penelitian dengan menggabungkan ACO dan algoritma genetika dalam memperbaiki penyebaran semut untuk meningkatkan konvergensi. Liantoni. dkk. (2014) melakukan pembagian gambar dan menghitung jumlah semut berdasarkan *gradient* tiap area pembagian. Akan tetapi, pada proses penyebaran semut awal masih dilakukan secara acak seperti ACO tradisional sehingga memungkinan semut terjebak pada kondisi lokal optimal. Metode yang dilakukan disebut *adaptive* ACO. Verma. dkk. (2011) melakukan pendekatan Fuzzy-ACO. Jumlah semut dihitung dan ditempatkan di titik akhir pada gambar menggunakan detektor sobel. Teknik *Derivatif Fuzzy* mengimplementasikan faktor probabilitas fuzzy untuk menentukan piksel berikutnya yang paling mungkin untuk menjadi tepi. Ya-Ping. dkk (2008) melakukan perbaikan deteksi tepi dengan cara menggabungkan metode canny dengan ACO.

Berdasarkan permasalahan tersebut, modifikasi distribusi semut pada ACO diusulkan untuk mengoptimalkan penempatan semut berdasarkan *gradient*. Nilai *gradient* pada suatu gambar menunjukkan adanya keberadaan tepi pada gambar tersebut. Pada penelitian ini, nilai *gradient* digunakan untuk menentukan penempatan semut awal. Semut tidak disebar secara acak, akan tetapi ditempatkan di *gradient* yang tinggi. Pada *gradient* tertinggi, perjalanan semut dilakukan dahulu untuk menentukan jalur. Cara ini diharapkan dapat digunakan untuk optimasi penemuan jalur.

1.2. Perumusan masalah

Berdasarkan latar belakang yang sudah diuraikan sebelumnya, perumusan masalah yang terdapat pada penelitian ini yaitu

1. Bagaimana menempatkan semut berdasarkan nilai *gradient*?
2. Bagaimana menentukan suatu piksel dapat dikatakan sebagai tepi berdasarkan *ACO* modifikasi?

1.3. Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah untuk mengetahui apakah ACO modifikasi dapat digunakan untuk penyebaran semut dan optimalisasi penentuan jalur dalam deteksi tepi.

Manfaat dari penelitian ini adalah mengetahui hasil deteksi tepi menggunakan ACO modifikasi.

1.4. Kontribusi

Kontribusi penelitian yang dilakukan adalah ACO modifikasi yang berguna untuk perbaikan proses distribusi semut pada ACO berdasarkan nilai *gradient*.

1.5. Batasan Masalah

Batasan masalah pada pembuatan penelitian ini adalah sebagai berikut:

1. Penelitian ini terbatas sampai deteksi tepi citra.
2. Citra uji coba dengan ukuran 128x128 piksel.
3. Uji coba dilakukan dengan membandingkan nilai PSNR antara ACO modifikasi dengan ACO tradisional.

[Halaman ini sengaja dikosongkan]

BAB II

DASAR TEORI DAN KAJIAN PUSTAKA

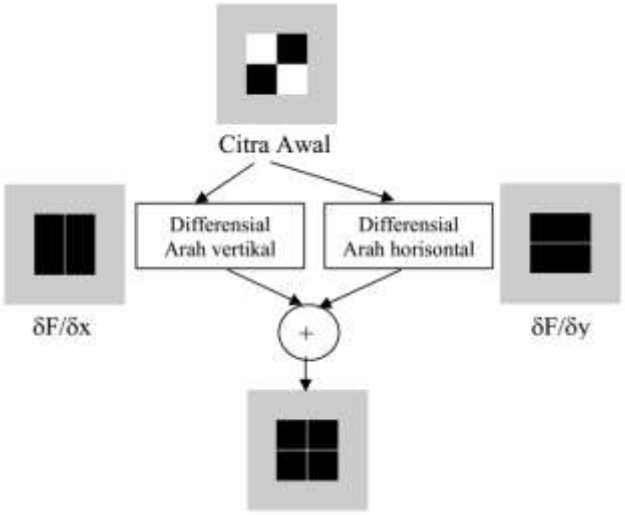
Bab ini berisi pembahasan tentang dasar teori dan kajian pustaka yang digunakan sebagai teori pendukung penelitian ini. Adapun pembahasannya meliputi deteksi tepi, *Ant Colony Optimization* (ACO), *gradient*.

2.1. Deteksi Tepi

Deteksi tepi adalah suatu proses penggalian informasi tepi dari sebuah gambar. Hal ini dianggap sebagai langkah dasar yang digunakan dalam sebagian besar aplikasi pengolahan citra (Verma. dkk, 2011). Deteksi tepi memiliki tujuan antara lain digunakan sebagai :

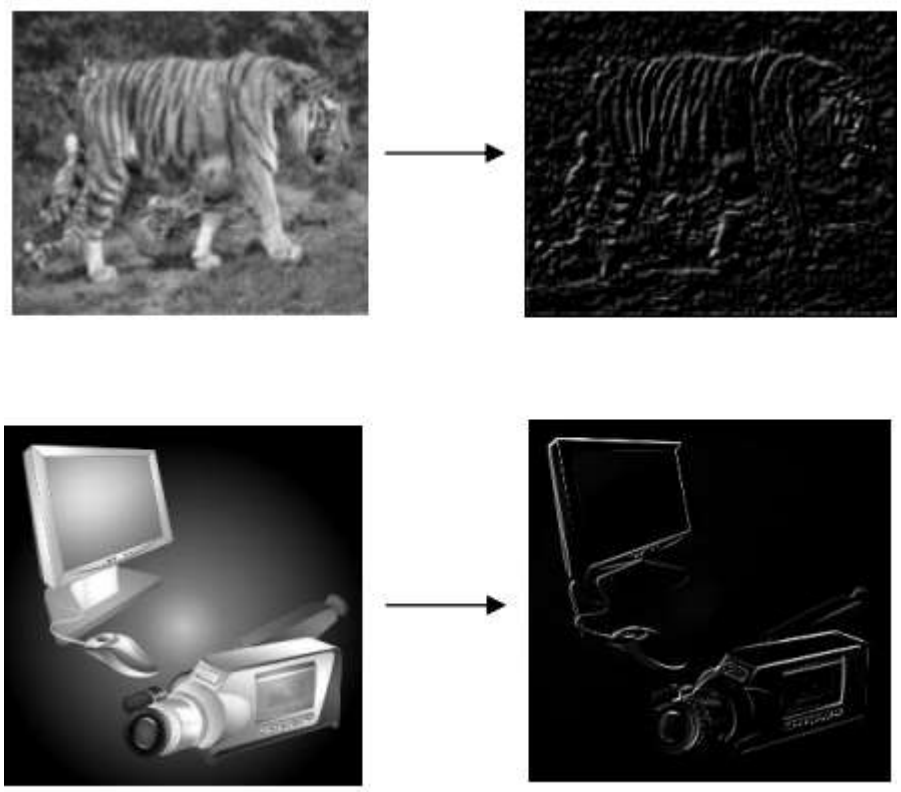
- Untuk menandai bagian yang menjadi detail dari sebuah gambar.
- Untuk memperbaiki detail dari gambar yang kabur, yang terjadi karena error atau adanya efek dari proses akuisisi gambar.

Suatu titik (x,y) dikatakan sebagai tepi dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar 2.1 berikut ini menggambarkan bagaimana tepi suatu gambar diperoleh.



Gambar 2.1 Proses deteksi tepi citra

Gambar 2.1 menjelaskan bagaimana sebuah citra dilakukan perhitungan differensial terhadap arah vertikal dan differensial terhadapt arah horisontal. Pada Gambar 2.2 berikut ini merupakan contoh yang dihasilkan dari proses deteksi tepi citra menggunakan model differensial.



Gambar 2.2 Hasil deteksi tepi citra

Pada Gambar 2.2 terlihat bahwa hasil deteksi tepi berupa tepi-tepi dari suatu gambar. Bila diperhatikan tepi suatu gambar yang dihasilkan terletak pada titik-titik yang memiliki perbedaan tinggi. Ada beberapa metode konvensional yang sering digunakan untuk melakukan deteksi tepi, seperti Robert, Prewitt dan Sobel (Gonzalez. dkk, 1992).

Metode Robert adalah nama lain dari teknik differensial pada arah horisontal 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 ditunjukkan pada Persamaan (2.1).

$$H = [-1 \quad 1] \; dan \; V = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{2.1}$$

Metode Prewitt merupakan pengembangan metode robert dengan menggunakan filter *High Pass Filter* (HPF) yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF.

Kernel filter yang digunakan dalam metode Prewitt ditunjukkan pada Persamaan (2.2).

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \; dan \;\; V = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.2}$$

Metode Sobel 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 sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi.

Kernel filter yang digunakan dalam metode sobel ditunjukkan pada Persamaan (2.3).

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} ; \;\; V = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.3}$$

Metode lain yang juga dilakukan untuk melakukan deteksi tepi sebuah gambar adalah menggunakan ACO. Algoritma ACO merupakan algoritma optimasi terinspirasi alam yang dimotivasi oleh perilaku mencari makan semut. Semut akan menggunakan senyawa kimia khusus yang disebut feromon untuk menandai jalur antara sumber makanan dan koloni mereka. Jalur feromon digunakan oleh semut berikutnya sebagai referensi untuk mencari makanan karena feromon meningkatkan kemungkinan jalan untuk dipilih.

Karena manfaat yang menguntungkan, ACO telah banyak digunakan untuk memecahkan beberapa masalah Non Polinomial (NP), t*raveling salesman problem* (TSP), deteksi tepi, *network packet routing, vehicular routing, quadratic assignment problem,* dan sebagainya. Dalam penelitian ini kami menggunakan ACO untuk deteksi tepi sebuah gambar.

2.2. *Ant Colony Optimization* (ACO)

ACO merupakan metode heuristik yang meniru perilaku semut untuk memecahkan masalah optimasi diskrit (Dorigo. dkk, 2004). Semut menggunakan senyawa kimia khusus yang disebut feromon untuk menandai jalur antara sumber makanan dan koloni mereka. Jalur feromon digunakan oleh semut berikutnya sebagai referensi untuk mencari makanan karena feromon meningkatkan kemungkinan jalan untuk dipilih.

Ada Ada beberapa jenis ACO yaitu Ant System (AS), Ant Colony System (ACS), Min-Max Ant System (MMAS), Elitist Ant System (EAS), Rank-Based Ant System (ASRank), Approximate Nondeterministic Tree Search (ANTS). (Etemad dkk, 2011) Dari jenis-jenis tersebut AS dan ACS merupakan jenis ACO yang paling populer digunakan. (Charu. dkk, 2013). AS merupakan algoritma versi pertama ACO yang diusulkan pada tahun 1992. Kemudian metode ini berkembang menjadi ACS. (Dorigo. dkk, 2004).

Jenis ACS sering dipilih karena algoritma ini telah menerapkan penurunan konsentrasi feromon, sehingga kemungkinan semut terjebak dalam satu daerah eksplorasi tertentu menjadi lebih kecil. (Verma. dkk, 2011). Secara umum, *pseudocode* dalam ACO *metaheuristic* ditunjukkan pada Gambar 2.3.

```
Initialize
SCHEDULE_ACTIVITIES
      ConstructAntSolutions
      DoDaemonActions (optional)
      UpdateFeromons
END_SCHEDULE_ACTIVITIES
```

Gambar 2.3 *Pseudocode* ACO *metaheuristic*

Initialize merupakan langkah yang dilakukan di awal proses. Dalam langkah ini dilakukan prosedur inisialisasi, seperti pengaturan parameter dan penempatan nilai feromon awal. Adapun parameter yang digunakan yaitu jumlah semut *(K),* jumlah step konstruksi *(L),* iterasi *(N), feromon evaporation rate (ρ), feromon decay (φ),* faktor pembobot feromon *(α),* dan faktor bobot informasi heuristik *(β).*

ConstructAntSolutions merupakan kegiatan perjalanan semut. Proses konstruksi berisi sejumlah langkah-langkah konstruksi. Semut akan bergerak dalam suatu gambar sampai jumlah target langkah konstruksi terbentuk. Pada proses konstruksi ke (n^{th}) jumlah semut (k^{th}) akan berpindah dari node *(i)* ke node *(j)* yang mengikuti perpindahan probabilitas $(P_{i,j}^{(n)})$. Aturan *proportional pseudorandom* ini ditunjukkan pada Persamaan 2.4.

$$P_{i,j}^{(n)} = \frac{\left(\tau_{i,j}^{n-1}\right)^{\alpha}\left(\eta_{i,j}{}^{\beta}\right)}{\sum_{j\in\Omega_i}\left(\tau_{i,j}^{n-1}\right)^{\alpha}\left(\eta_{i,j}{}^{\beta}\right)} \; if \; j \in \Omega_i \qquad (2.4)$$

dengan :

$P_{i,j}^{(n)}$ = perpindahan probabilitas,

α = faktor pembobot feromon,

β = faktor bobot informasi heuristik,

Ω_i = node ketetanggaan dari semut yang diberikan pada node ke *(i)*,

τ = *update* feromon,

η = informasi heuristik.

DoDaemonActions merupakan solusi konstruksi yang dilakukan untuk tambahan langkah sebelum pembaruan nilai feromon. Kegiatan ini sebagai tindakan tambahan sebelum memperbarui nilai-nilai feromon. Proses ini tidak bisa dilakukan jika hanya dengan *single ant.*

UpdateFeromons merupakan kegiatan pembaruan feromon setelah proses kontruksi dan *daemon actions* dilakukan. Terdapat dua kali *update*, yaitu *update* global feromon dan *update* lokal feromon. Pembaruan lokal feromon dilakukan setiap kali langkah konstruksi. Pada tahap ini feromon akan mengalami kerusakan

(*feromon decay*). Hal ini bertujuan untuk menurunkan konsentrasi feromon di tepi yang dilalui. *Update* local feromon ditunjukkan pada Persamaan 2.5.

$$\tau_{i,j} = (1 - \varphi)\tau_{i,j} + \varphi\tau_0 \quad (2.5)$$

dengan :

φ = kerusakan feromon,

τ = *update* feromon.

Update global feromon dilakukan setelah step konstruksi maksimal dalam satu iterasi dilalui. Pada tahap ini terjadi penguapan feromon. *Update* global feromon ditunjukkan pada Persamaan (2.6).

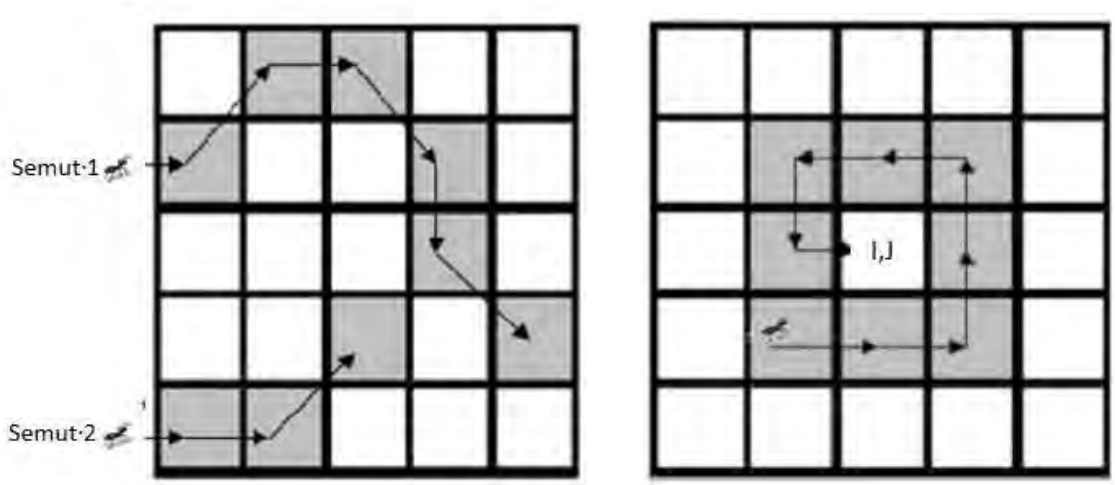
$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \rho\Delta\tau_{i,j} \quad (2.6)$$

dengan:

ρ = feromon tingkat penguapan,

$\Delta\tau_{i,j}$ = total feromon untuk pembaruan feromon global.

Pada ACO terdapat aturan perpindahan dengan faktor probabilitas pada delapan piksel ketetanggaan. Proses ini dihitung dengan menggunakan matrik perpindahan probabilitas. Piksel dengan faktor probabilitas maksimum dalam mendeteksi ketetanggaan memiliki piksel tepi. Untuk mengurangi perpindahan yang berulang pada semut dilakukan aturan *stopping criteria,* yaitu perpindahan semut akan berhenti jika melewati jalur yang sudah dilewati semut yang lain dan ketika semua piksel ketetanggaan (8 piksel) sudah dilewati semua oleh semut maka perpindahan akan berhenti. (Verma dan Singhal, 2011). Gambar 2.4 menunjukkan aturan perpindahan pada semut.



Gambar 2.4 Aturan perpindahan semut

2.3. *Gradient*

Implementasi penggunaan *magnitude* dari *gradient* merupakan langkah awal pada pengolahan citra digital. (Gonzalez. dkk, 1992). Nilai *gradient* pada suatu area menunjukkan adanya keberadaan tepi pada area tersebut. *Gradient* terbentuk atas transisi atau perubahan warna secara gradual. Untuk fungsi *f(x,y)* pada koordinat *gradient (x,y)* didefinisikan sebagai *two-dimensional column vector* dengan Persamaan (2.7).

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.7}$$

Untuk *magnitude* dari vektor ditunjukkan oleh Persamaan (2.8).

$$\nabla f = mag(\nabla f)$$

$$\nabla f = \left[G_x^2 + G_y^2\right]^{1/2}$$

$$\nabla f = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \tag{2.8}$$

Komponen dari *gradient* vektor itu sendiri adalah operator linear, tetapi magnitude dari vektor adalah bukan merupakan operator linear karena merupakan operasi kuadrat dan akar kuadrat (Gonzalez. dkk, 1992). Dalam pengolahan citra, operator berbasis turunan menjadi dasar dalam penghitungan. Beberapa contoh yaitu menggunakan operator sobel, prewit, robert. (Gonzalez. dkk, 1992). Salah satu operator yang digunakan adalah operator sobel.

Gambar 2.5 merupakan *pseudo-convolution kernels* yang digunakan untuk menghitung *gradient magnitude* dengan cepat.

a_1	a_2	a_3
a_4	a_5	a_6
a_7	a_8	a_9

Gambar 2.5 *Pseudo-convolution kernels* ukuran 3 x 3

Untuk menghitung *kernel* pada proses penjumlahan *magnitude* ditunjukkan pada Persamaan (2.10).

$$|G| = |(a_1 + 2 * a_2 + a_3) - (a_7 + 2 * a_8 + a_9)| + |(a_3 + 2 * a_6 + a_9) - (a_1 + 2 * a_4 + a_7)| \quad (2.10)$$

Salah satu metode penentuan *gradient* citra yaitu operator sobel. Langkah pertama yang dilakukan operator sobel adalah memperkirakan nilai *gradient* horisontal (*Gx*) dan *gradient* vertikal (*Gy*). Cara ini dilakukan dengan operasi *kernel* terhadap matrik citra seperti yang ditunjukkan pada Gambar 2.5. *Matrik kernel* untuk *Gx,* didefinisikan sebagai operator *Kx*, seperti ditunjukkan pada Persamaan (2.11). Sedangkan matrik *kernel* untuk *Gy* didefinisikan sebagai *Ky*, seperti ditunjukkan pada Persamaan (2.12).

$$Kx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2.11)$$

$$Ky = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (2.12)$$

(Gonzalez. dkk, 1992)

Setelah nilai *gradient* pada arah vertikal dan horisontal didapatkan, kemudian dicari nilai penghitungan matrik yang menghasilkan nilai *gradient* citra. Nilai gradient pada arah vertikal dan horisontal juga digunakan untuk menentukan arah sudut dari tepi (θ). Nilai dari θ akan mempengaruhi perlakukan suatu piksel terhadap piksel tetangganya. Penghitungan nilai arah tepi dihitung dengan Persamaan (2.9).

$$\theta = tan^{-1}\left(\frac{Gy}{Gx}\right) \quad (2.9)$$

dengan :

θ = arah sudut dari tepi,
Gx = gradient horisontal,
Gy = .*gradient* vertikal.

[Halaman ini sengaja dikosongkan]

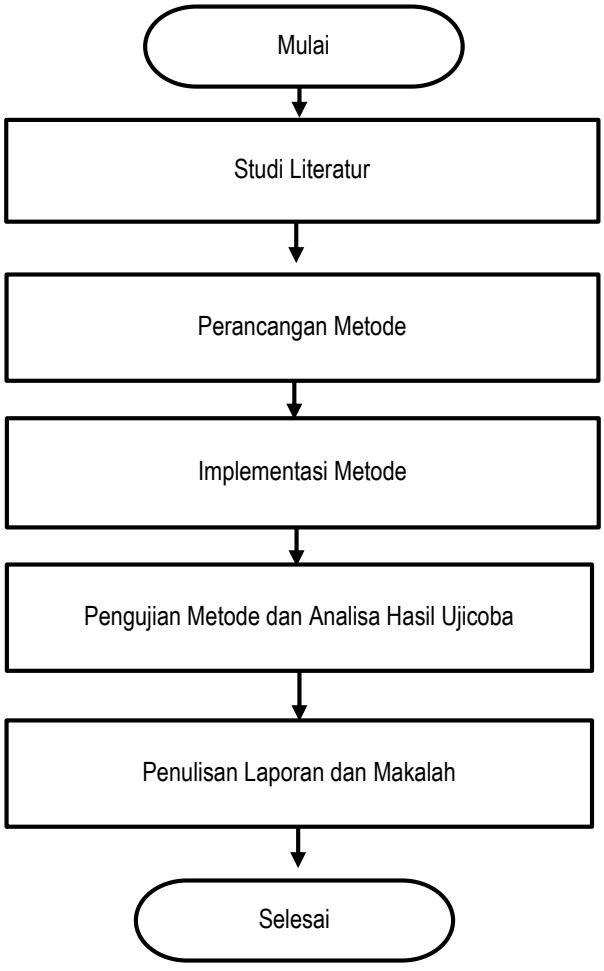
BAB III

METODOLOGI

Bab ini akan memaparkan tentang metodologi penelitian yang akan digunakan, terdiri dari studi literatur, perancangan metode, implementasi metode, pengujian metode dan analisa hasil ujicoba, penulisan laporan dan makalah. Selanjutnya jadwal kegiatan penelitian memuat garis waktu dari semua langkah penelitian.

3.1 Tahapan Penelitian

Berikut ini merupakan tahapan yang akan dilakukan dalam pengerjaan penelitian ini, Gambar 3.1 adalah diagram metodologi penelitian.



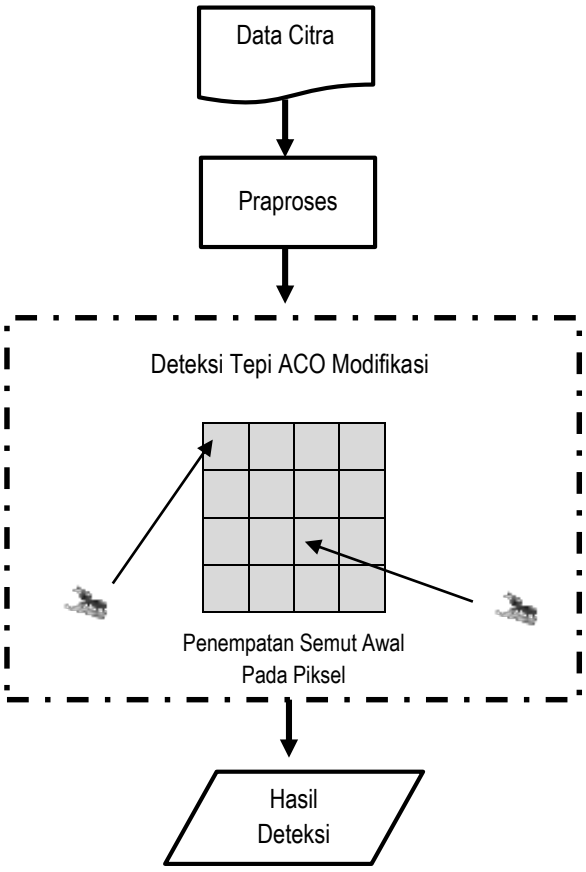
Gambar 3.1 Diagram metodologi penelitian

3.2 Studi Literatur

Penelitian selalu diawali dengan proses pengkajian yang berkaitan dengan topik penelitian yang diambil. Proses ini meliputi pencarian referensi pendukung yang sesuai dan dapat dipertanggungjawabakan. Referensi yang digunakan dapat berupa buku jurnal mengenai maupun artikel. Seperti, mengenai penggunaan *gradient*, metode deteksi tepi citra, metode *Ant Colony Optimization* (ACO). Informasi yang didapat nantinya berguna untuk mendesain algoritma untuk memecahkan permasalahan yang sedang diteliti. Setelah studi literatur dilakukan, pendefinisian masalah dapat diketahui lebih detail.

3.3 Perancangan Metode

Dalam penelitian ini akan dilakukan beberapa tahapan meliputi pemilihan data citra, praproses dan deteksi tepi. Gambar 3.2 berikut menggambarkan ilustrasi tahapan yang akan dikerjakan.

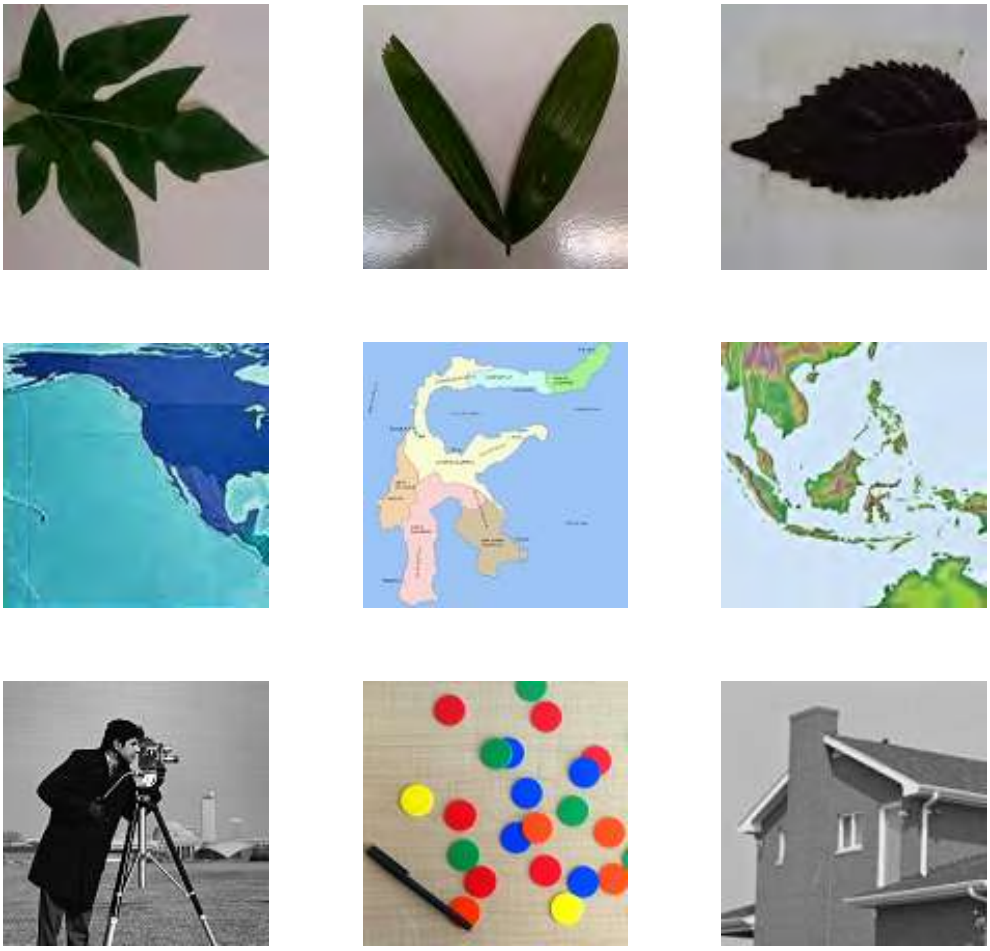


Gambar 3.2 Diagram metode penelitian

3.3.1 Data Citra

Terdapat dua jenis data yang digunakan pada penelitian ini, yaitu data dengan objek yang memiliki nilai kontras yang berbeda dengan *background* dan data dengan objek komplek. Gambar dengan kontras berbeda digunakan sebagai uji coba pengaruh penyebaran jumlah semut berdasarkan gradient. Sedangkan gambar komplek digunakan untuk melihat hasil akurasi deteksi tepi.

Data yang digunakan berupa gambar daun hasil foto, gambar peta dunia dan berbagai macam gambar komplek. Gambar 3.3 berikut ini contoh gambar dengan kontras yang berbeda dengan *background* dan gambar dengan objek komplek.



Gambar 3.3 Contoh data citra

3.3.2 Praproses

Data masukan yang digunakan dalam sistem adalah citra RGB. Dari data citra RGB tersebut diubah menjadi citra keabuan. Proses perubahan citra warna menjadi citra keabuan menggunakan metode penghitungan *lightness* dan *average*. Selain metode itu ada juga metode luminosity yang merupakan pengembangan dari metode *lightness* dan *average*. Metode ini menghitung nilai setiap elemen warna dari citra, yaitu *red*, *green* dan *blue*. Penghitungan ini dengan cara menambahkan bobot sesuai persepsi penglihatan manusia. Proses penghitungan untuk konvensi citra warna menjadi citra keabuan dapat dirumuskan secara matematis seperti berikut ini:

a). Lightness

Metode *lightness* dilakukan dengan menghitung rata-rata dari nilai maksimum dan mninimum *R, G*, dan *B*.

$$Lightness = \frac{\max(R,G,B)+\min(R,G,B)}{2} \quad (3.1)$$

b). Average

Metode *average* dilakukan dengan menghitung rata-rata setiap elemen warna *R, G*, dan *B*.

$$average = \frac{R+G+B}{3} \quad (3.2)$$

c). Luminosity

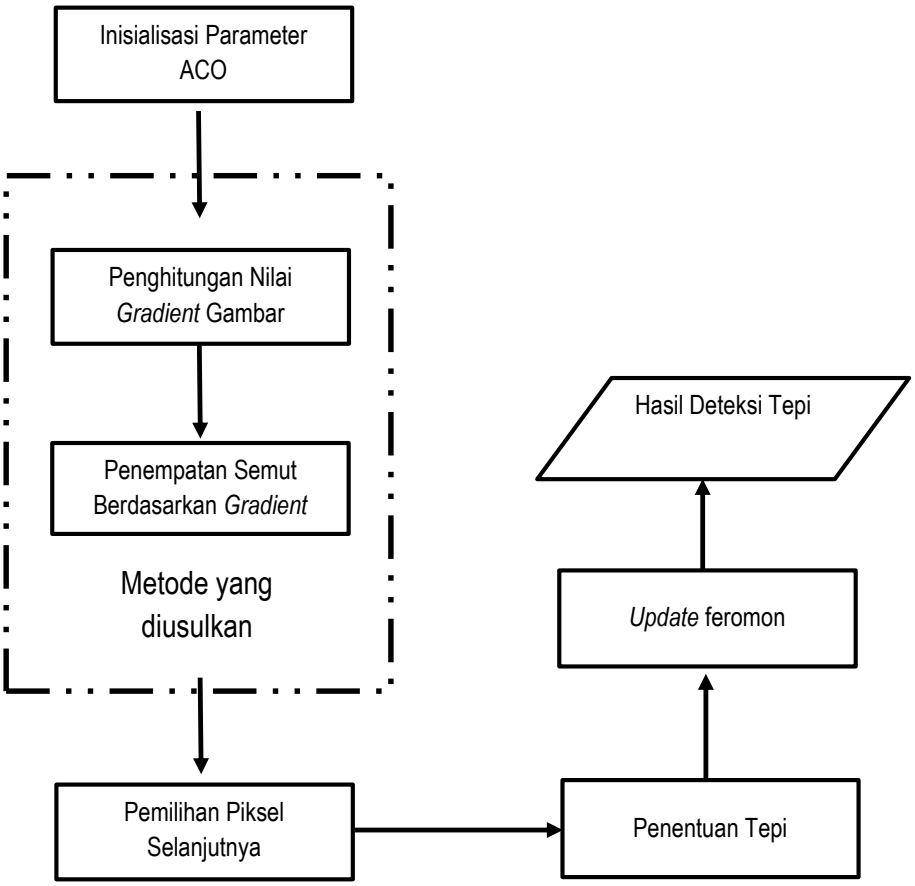
Metode luminosity dilakukan dengan menghitung nilai setiap elemen warna *R, G*, dan *B* dengan menambahkan bobot sesuai persepsi penglihatan manusia. Penglihatan manusia lebih sensitif terhadap warna hijau, oleh karena itu elemen G memiliki bobot yang paling tinggi dari elemen warna lainnya (Mathworks, 2014).

$$Luminosity = \frac{R*299+G*587+B*114}{1000} \quad (3.3)$$

Pada penelitian ini menggunakan metode *luminosity* untuk melakukan konversi citra warna menjadi citra keabuan karena sifat metode *luminosity* yang lebih dekat dengan persepsi penglihatan manusia. Setelah diperoleh citra keabuaan kemudian dilakukan deteksi tepi menggunakan metode ACO modifikasi.

3.3.3 Deteksi Tepi Modifikasi ACO

Pada penelitian ini diusulkan ACO modifikasi pada tahapan proses penyebaran semut awal berdasarkan jumlah *gradient*. Gambar 3.4 adalah diagram proses deteksi tepi yang akan dilakukan pada ACO modifikasi.



Gambar 3.4 Blok diagram metode ACO modifikasi

a). Inisialisasi Parameter ACO

Tahapan awal dari metode ACO modifikasi adalah inisialisasi parameter masukkan. Parameter terdiri dari jumlah semut *(K),* jumlah step konstruksi *(L),* iterasi *(N), feromon evaporation rate (ρ), feromon decay (φ),* faktor bobot feromon *(α),* dan faktor pembobot informasi heuristik (*β*). Matrik informasi heuristik berupa matrik intensitas warna setiap piksel. Matrik feromon awal diinisialisasi 1 atau 0.

b). Penghitungan Nilai *Gradient*

Proses ini bertujuan untuk mengetahui potensi nilai tepi dari sebuah gambar. Operator *gradient* dapat direpresentasikan oleh dua buah kernel konvolusi *(Gx)* dan *(Gy)*, yang masing-masing mendefinisikan operasi penghitungan *gradient* dalam arah sumbu *(x)* dan sumbu *(y)* yang saling tegak lurus.

Setelah proses perhitungan nilai *gradient* piksel, selanjutnya memilih *gradient* yang nilainya lebih dari atau sama dengan *threshold*. Jumlah *gradient* yang berdasarkan *threshold* ini kemudian menjadi jumlah semut *(K)*. *Pseudocode* untuk menghitung jumlah *gradient* yang nilainya lebih dari atau sama dengan *threshold* ditunjukkan pada Pseudocode 3.1.

Pseudocode 3.1 Penghitungan nilai *gradient*

```
Start
    num_gradient ← 0
    For i ← 0 to image_row do
        For j ← 0 to image_col do
            If image_gradient[i][j] >= Threshold
                num_gradient ← num_gradient + 1
                im_gradient[i][x] ← piksel_gradient[x]
                im_gradient[i][y] ← piksel_gradient[y]
            End If
        End For
    End For
End
```

c). Penempatan Semut

Berdasarkan *gradient* yang nilainya lebih dari atau sama dengan *threshold* akan menjadi parameter jumlah semut masukkan. Setiap posisi piksel *gradient* yang nilainya lebih dari atau sama dengan *threshold* akan digunakan sebagi posisi dari semut. Semut tidak disebar secara acak, akan tetapi ditempatkan pada piksel yang memiliki *gradient* yang nilainya lebih dari atau sama dengan *threshold*.

Semut yang menempati posisi piksel berdasarkan *gradient* yang nilainya lebih dari atau sama dengan *threshold* kemudian akan bergerak ke piksel ketetanggaan untuk menentukan tepi. *Pseudocode* penempatan semut ditunjukkan pada Pseudocode 3.2.

Pseudocode 3.2 Penempatan semut

```
Start
    For a ← 1 to K do
        nilaiX ← im_gradient[a][x]
        nilaiY ← im_gradient[a][y]
        semut [a][x] ← nilaiX
        semut [a][y] ← nilaiY
    End For
End
```

d). Pemilihan Piksel

Pada deteksi tepi ACO, semut akan berpindah ke piksel tetangga. Piksel ketetanggaan yang digunakan adalah 8 ketetanggaan. Gambar 3.7 berikut ini menunjukkan hubungan ketetanggaan pada tiap piksel.

$I_{i-1,j-1}$	$I_{i-1,j}$	$I_{i-1,j+1}$
$I_{i,j-1}$	I	$I_{i,j+1}$
$I_{i+1,j-1}$	$I_{i+1,j}$	$I_{i+1,j+1}$

Gambar 3.5 Hubungan ketetanggaan

Proses perpindahan piksel ke piksel tetangganya berdasarkan derajat eksplorasi *(q)*. Algoritma ACS digunakan karena memiliki aturan *proportional pseudorandom* untuk mengoptimalkan gerakan semut. Aturan *proportional pseudorandom* menggunakan ambang batas yang ditetapkan *(q_0),* yang nilainya antara 0 hingga 1.

Hubungan ketetanggan menunjukkan nilai intensitas yang mewakili tiap piksel dari gambar. Dari nilai intensitas tersebut akan diperoleh informasi heuristik *(η_{ij})* yang ditunjukkan pada Persamaan (3.4).

$$\eta_{ij} = \frac{max_{ij}\begin{pmatrix} |I(i-1,j-1)-I(i+1,j+1)|, \\ |I(i-1,j+1)-I(i+1,j-1|, \\ |I(i,j-1)-I(i,j+1)|, \\ |(i-1,j)-I(i+1,j)| \end{pmatrix}}{\eta_{max}} \tag{3.4}$$

dengan:

η_{max} = nilai heuristik maksimum.

Pada gerakan setiap semut yang dilakukan secara acak menghasilkan *(q)* yang nilainya disebarkan antara 0 dan 1. Jika *(q)* lebih besar dari *(q_0)*, maka aturan *proportional pseudorandom* digunakan untuk menentukan gerakan semut. Namun, jika *(q)* lebih kecil dari *(q_0)*, semut harus bergerak sesuai dengan transisi yang memaksimalkan nilai *update* feromon *(τ_{ij}^{α})* dan informasi heuristik *(η_{ij}^{β})*. Aturan *proportional pseudorandom* diberikan dalam Persamaan (3.5).

$$P_{(l,m),(i,j)}^{(n)} = \frac{\left(\tau_{i,j}^{(n-1)}\right)^{\alpha}\left(\eta_{i,j}\right)^{\beta}}{\sum_{(i,j)\in\Omega_{(l,m)}}\left(\tau_{i,j}^{n-1}\right)^{\alpha}\left(\eta_{i,j}{}^{\beta}\right)} \quad (3.5)$$

dengan:

$P_{i,j}^{(n)}$ = perpindahan probabilitas,

α = faktor pembobot feromon,

β = faktor bobot informasi heuristik,

$\Omega_{(l,m)}$ = node ketetanggaan dari semut yang diberikan pada node ke *(l,m)*,

$\tau_{i,j}^{n-1}$ = feromon pada piksel *(i,j)*,

η = informasi heuristik.

Pergerakan atau perpindahan semut pada pixel baris *i* dan kolom *j* ($P_{i,j}$) sama dengan perkalian feromon (τ) dan informasi heuristik (η) yang dibagi dengan total perkalian lingkungan 8-ketetanggaan. Kedua feromon dan informasi heuristik yang digunakan dilengkapi dengan feromon bobot faktor (α) dan informasi heuristik bobot faktor (β).

e). Update Process

Feromon intensitas sebagai informasi heuristik akan menarik semut untuk mengikuti jalur perjalanan semut lainnya. Karena itu, feromon akan diperbarui dua kali, pertama setelah perpindahan yang dilakukan tiap semut dan kedua setelah perpindahan yang dilakukan semua semut. Terdapat dua mekanisme pembaruan feromon (τ), yaitu pembaruan feromon lokal dan pembaruan feromon global. Setiap kali langkah konstruksi, matrik feromon local akan diperbarui karena feromon akan mengalami kerusakan (*pheromone decay)*. Pembaruan feromon lokal bertujuan untuk memverifikasi pencarian yang dilakukan oleh semut berikutnya selama iterasi (Dorigo. dkk, 2006). Persamaan untuk pembaruan feromon lokal ditunjukkan pada Persamaan (3.6).

$$\tau_{i,j}^{(n-1)} = \begin{Bmatrix} (1-\rho).\tau_{ij}{}^{(n-1)} + \rho\,.\Delta_{i,j}^{(k)}, jika\ (i,j) jalur\ terbaik \\ \tau_{ij}{}^{(n-1)},\ lainnya \end{Bmatrix} \quad (3.6)$$

dengan:

$\tau_{i,j}^{(n-1)}$ = pembaruan feromon lokal,

ρ = feromon penguapan,

$\Delta_{i,j}^{(k)}$ = total deposit feromon.

Nilai feromon penguapan *(ρ)* akan menurunkan nilai feromon. Sedangkan total deposit feromon ($\Delta_{i,j}^{(k)}$) dari semut yang nilainya ditunjukkan pada Persamaan (3.7).

$$\Delta_{i,j}^{(k)} = \eta_{i,j} \tag{3.7}$$

dengan:

$\eta_{i,j}$ = informasi heuristik.

Pseudocode pembaruan feromon lokal ditunjukkan pada Pseudocode 3.3.

Pseudocode 3.3 Pembaruan feromon lokal

```
Start
    Parameter step
    Parameter k
    row ← semut[k][x]
    col ← semut[k][y]
    If step != 0
        result ← (1 – pheromoneEva) * pheromone[row][col] +
                 pheromoneEva * informasiHeuristik[row][col]
    Else
        result ← (1 – pheromoneEva) * pheromone[row][col] +
                 pheromoneEva
    End If
End
```

Implementasi dengan menggunakan feromon peluruhan dan feromon init bertujuan untuk menurunkan konsentrasi feromon di tepi yang dilalui. Oleh karena itu semut berikutnya dapat menghasilkan solusi yang berbeda.

Setelah perjalanan semua semut selama langkah kontruksi selesai, akan dilakukan pembaruan feromon global. Persamaan untuk pembaruan feromon global ditunjukkan pada Persamaan (3.8)

$$\tau^{(n)} = (1 - \varphi).\tau^{(n-1)} + \varphi.\tau^{(0)} \quad (3.8)$$

dengan:

$\tau^{(n)}$ = pembaruan feromon global,

φ = feromon peluruhan (*pheromone decay coefficient)*,

τ_0 = feromon init.

Pseudocode pembaruan feromon global ditunjukkan pada Pseudocode 3.4.

Pseudocode 3.4 Pembaruan feromon global

```
Start
    For count ← 0 to total_visited_pixel do
        row ←  visited_pixel[count][0]
        col ←  visited_pixel[count][1]
        value ← (1 - pheromoneDecay) * pheromone[row][col] +
                pheromoneDecay * pheromoneof_pixel[count]
    End for
End
```

f). Penentuan Tepi

Pada langkah ini, sebuah keputusan diambil pada setiap piksel untuk menentukan tepi atau bukan. Cara ini dilakukan dengan mengimplementasikan sebuah *threshold (T)* pada matrik feromon akhir ($\tau^{(N)}$). (Tian dkk, 2008). Pada penelitian yang diusulkan ini nilai *threshold (T)* diadaptasi berdasarkan metode otsu. (Otsu, 1979). *Otsu thresholding* diimplementasikan untuk menentukan solusi terbaik berdasarkan jumlah feromon yang disimpan dalam setiap piksel. Pendekatan otsu dapat menentukan suatu variabel yang berguna membedakan antara dua atau lebih kelompok yang muncul secara alami. Penentuan tepi berdasarkan nilai matrik feromon akhir. Teknik otsu *threshold* akan mengurangi hasil citra keabuan menjadi citra *binary* dengan dua kemungkinan nilai untuk tiap

piksel. Dari nilai matrik feromon akhir akan menunjukkan sebuah piksel dikatakan tepi atau bukan tepi.

Inisialisasi threshold $T^{(0)}$ dipilih berdasarkan rata-rata nilai feromon matrik. Selanjutnya, feromon matrik diklasifikasinya kedalam dua kategori dengan kriteria nilai kurang dari $T^{(0)}$ atau lebih dari $T^{(0)}$. Inisialisasi nilai $T^{(0)}$ dihitung berdasarkan Persamaan 3.9.

$$T^{(0)} = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} \tau_{i,j}^{(N)}}{M_1 M_2} \tag{3.9}$$

dengan :

M_1 = lebar gambar,

M_2 = tinggi gambar.

Nilai *threshold (T)* akan dibandingkan dengan hasil feromon akhir yang digunakan sebagai nilai *fitness* dalam menentukan tepi atau bukan. Penentuan tepi dengan nilai *fitness* yang menentukan sebuah piksel sebagai tepi (E_{ij} = 1) atau bukan (E_{ij} = 1) berdasarkan kriteria seperti ditunjukkan pada Persamaan 3.10.

$$E_{i,j} = \begin{Bmatrix} 1, & jika\ \tau_{i,j}^{(N)} \geq T^{(l)} \\ 0 & lainnya \end{Bmatrix} \tag{3.10}$$

dengan :

T = *threshold*,

N = iterasi

l = indek iterasi

Pseudocode penentuan tepi ditunjukkan pada Pseudocode 3.5.

Pseudocode 3.5 Penentuan tepi

```
Start
    binarized ← BufferedImage
    For i ← 0 to image[height] do
        For j ← 0 to image[width] do
            rgb ← image[getRGB(i,j)]
            alpha ← rgb[alpha]
            if pheromone[i][j] >= threshold
                newPixel ← 255
            else
                newPixel ← 0
            end if
            newPixel ← colorToRGB (alpha, newPixel)
        End for
    End for
    binarized ← setRGB(i, j, newPixel)
End
```

3.4 Implementasi Metode

Tahapan ini digunakan untuk mengimplementasikan metode yang diusulkan ke dalam perangkat lunak. Sebelum implementasi perlu dilakukan perencanaan sistem, hal ini bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan mempertimbangkan berbagai faktor permasalahan dan kebutuhan yang ada pada sistem. Upaya yang dilakukan adalah dengan berusaha mencari kombinasi penggunaan teknologi dan perangkat lunak (*software*) yang tepat sehingga diperoleh hasil yang optimal dan mudah untuk diimplementasikan.

3.5 Pengujian Metode dan Analisis Hasil Uji Coba

Penilaian secara numerik dapat digunakan untuk mengetahui akurasi hasil uji coba deteksi tepi yang dilakukan. Pengujian hasil deteksi tepi citra dilakukan berdasarkan metode *Peak Signal to Noise Ratio* (PSNR) dan M*ean Square Error* (MSE). PSNR merupakan sebuah hubungan perbandingan antara nilai maksimum dan nilai kerusakan (*noise*) yang menunjukkan ketelitian dari hasil proses. Nilainya akan tinggi pada citra berkualitas baik. (Agrawal dkk, 2012). Penghitungan nilai MSE ditunjukkan pada Persamaan (3.11).

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (3.11)$$

dengan:

MSE = Nilai *Mean Square Error*,
m = tinggi citra,
n = lebar citra,
I(i,j) = nilai piksel frame masukan,
K(i,j) = nilai piksel frame rekonfigurasi.

Sedangkan penghitungan nilai PSNR ditunjukkan pada Persamaan (3.12).

$$PSNR = 10\ log_{10}\left(\frac{{MAX_I}^2}{MSE}\right) \quad (3.12)$$

dengan:

$PSNR$ = *Peak Signal to Noise Ratio,*
MAX_I = nilai maksimum piksel,

Pada uji coba akan membandingkan nilai PSNR antara hasil deteksi tepi menggunakan metode ACO tradisional dan metode ACO modifikasi yang diusulkan. Selain itu metode ACO modifikasi yang diusulkan juga akan dibandingkan dengan *Adaptive* ACO. Metode yang diajukan dikatakan berhasil memperbaiki ACO tradisional jika nilai PSNR lebih besar dari ACO tradisional.

Proses pengujian dilakukan berdasarkan *ground truth* deteksi tepi dari gambar uji. *Ground truth* didapatkan secara manual berdasarkan fungsi aplikasi photoshop. Kemudian akan dibandingkan dengan hasil uji coba yang diperoleh dengan metode ACO modifikasi, metode ACO tradisional dan *Adaptive* ACO.

Metode ACO tradisional yang digunakan sebagai pembanding pada tahap skenario uji coba berdasarkan penelitian yang dilakukan oleh Tian dkk tahun (2008). Sedangkan metode *Adaptive* ACO yang juga digunakan sebagai pembanding pada skenario uji coba merupakan penelitian yang dilakukan oleh Liantoni. dkk tahun (2014).

Pada penelitian ini digunakan beberapa parameter masukkan sebagai nilai penghitungan metode ACO. Tabel 3.1 berikut merupakan parameter uji coba yang digunakan pada metode yang diusulkan.

Tabel 3.1 Parameter uji coba

Parameter	Nilai
K (jumlah semut)	K = Jumlah *gradient* >= threshold
L (jumlah konstruksi step)	300
α (faktor pembobot feromon)	1
β (faktor pembobot informasi heuristik)	1
ρ (feromon penguapan)	0,1
φ (feromon peluruhan)	0,05
τ_{init} (feromon init)	0,0001
Ω (piksel ketetanggaan)	8
N (jumlah iterasi)	4

[Halaman ini sengaja dikosongkan]

BAB 4

HASIL UJI COBA DAN PEMBAHASAN

Pada bab ini dijelaskan mengenai skenario pengujian beserta hasil pengujian yang dilakukan dan analisis hasil uji yang diperoleh. Pengujian dilakukan untuk mengetahui pengaruh nilai *gradient* terhadap penyebaran semut, nilai M*ean Square Error* (MSE) dan *Peak Signal to Noise Ratio* (PSNR) yang dihasilkan.

4.1 Lingkungan Uji Coba

Spesifikasi dari perangkat keras yang digunakan untuk implementasi dan uji coba yang dilakukan adalah *processor* 2.5 Ghz Intel Core i5 4200M dengan Turbo 2.6Ghz, *memory* berkapasitas 4GB. 12800 Mhz DDR3. Sedangkan untuk perangkat lunak untuk implementasi digunakan untuk implementasi dan uji coba pada penilitian ini menggunakan pemrograman bahasa java dengan library pada aplikasi java netbean.

4.2 Pelaksanaan Uji Coba

Pengujian akan dilakukan dengan nilai MSE dan PSNR. Pengujian ini digunakan untuk mengetahui performa sistem dalam mendeteksi tepi suatu gambar. Pengujian dilakukan dengan membandingaan nilai PSNR dan MSE yang dihasilkan ACO tradisional dengan metode yang diusulkan. ACO tradisional yang digunakan sebagai pembanding berdasarkan penelitian Tian dkk tahun 2008. Selain itu, pengujian juga dilakukan dengan membandingan hasil deteksi tepi *adaptive* ACO berdasarkan penelitian Liantoni dkk tahun 2014.

Skenario uji coba dilakukan untuk mengetahui pengaruh penyebaran semut terhadap hasil deteksi tepi berdasarkan nilai *gradient* dibandingkan dengan penyebaran secara acak. Data uji coba yang digunakan sebanyak 30 gambar. Data gambar terdiri dari 20 gambar komplek dan 10 gambar dengan kontras yang berbeda dengan *background*.

Pada penelitian ini dilakukan pengujian awal terhadap jumlah iterasi. Pengujian ini dilakukan untuk mendapatkan jumlah iterasi yang optimal yang akan digunakan pada skenario uji coba. Jumlah iterasi yang digunakan antara lain sebanyak 1 iterasi, 2 iterasi, dan 4 iterasi.

Pengujian jumlah iterasi dilakukan pada metode ACO modifikasi. Contoh hasil pengujian untuk jumlah iterasi pada metode ACO modifikasi ditunjukkan pada Tabel 4.1.

Tabel 4.1 Iterasi deteksi tepi ACO modifikasi

ACO Modifikasi Iterasi 1	ACO Modifikasi Iterasi 2	ACO Modifikasi Iterasi 4
		
		
		
		

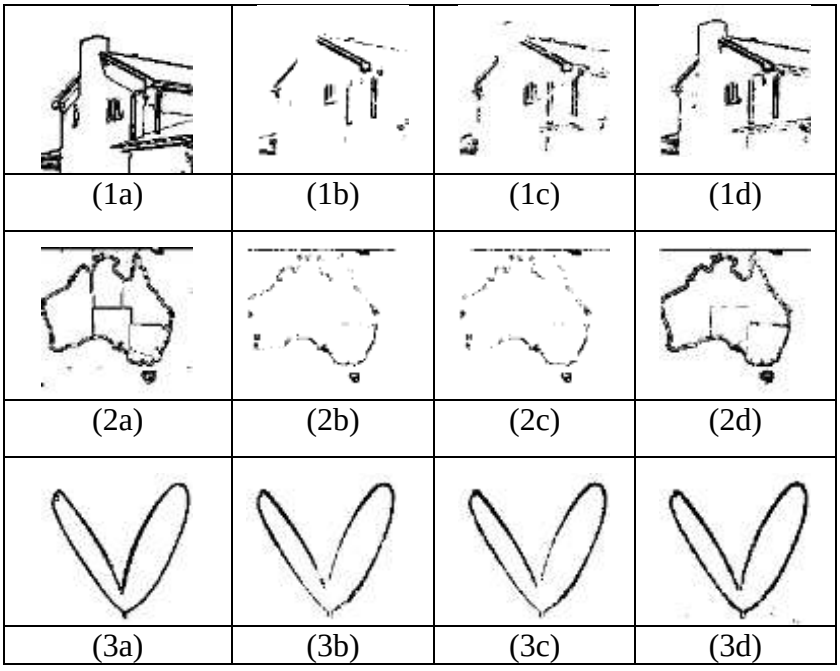
Dari hasil uji coba terhadap jumlah iterasi yang dilakukan pada metode ACO modifikasi menunjukkan semakin banyak jumlah iterasi maka hasil deteksi tepi juga semakin baik. Hasil deteksi tepi yang dihasilkan menjadi semakin tajam dan tebal.

4.3 Uji Coba Skenario 1

Uji coba skenario 1 dilakukan untuk menunjukkan pengaruh dari penyebaran semut pada deteksi tepi ACO tradisional, ACO modifikasi dan *adaptive* ACO. Pada pengujian ACO tradisional dan *adaptive* ACO akan dilakukan proses deteksi tepi untuk setiap gambar, kemudian hasil dari deteksi tepi yang telah dilakukan tadi digunakan sebagai pembanding dalam proses pengujian. Pengujian dilakukan pada gambar komplek dan gambar dengan *background* kontras. Hal ini bertujuan untuk menguji akurasi hasil deteksi tepi dengan berbagai macam gambar tersebut. Uji coba skenario 1 menggunakan jumlah parameter semut sebanyak K = 512.

4.3.1 Hasil Uji Coba Skenario 1

Pada skenario 1 dilakukan pengujian menggunakan metode ACO tradisional, ACO modifikasi dan *adaptive* ACO. Pengujian dilakukan pada gambar komplek dan gambar dengan *background* kontras. Contoh hasil pengujian dari deteksi tepi yang telah dilakukan pada ACO tradisional, ACO modifikasi dan *adaptive* ACO ditujukkan pada Gambar 4.2.



Gambar 4.1 *Ground truth* (1a) (2a) (3a); ACO tradisional (1b) (2b) (3b); *adaptive* ACO (1c) (2c) (3c); ACO modifikasi (1d) (2d) (3d)

Pada Gambar 4.1 menunjukkan hasil metode ACO modifikasi memiliki garis tepi yang lebih detail dibandingkan metode ACO tradisional dan *adaptive* ACO berdasarkan gambar *ground truth*. Pada gambar daun *(3d)* dari hasil deteksi tepi dengan ACO modifikasi menunjukkan tepi yang tajam dibandingkan hasil ACO tradisional daun *(3b)* dan *adaptive* ACO daun *(3c)*. Pada gambar *house (1d)*, peta *(2d)* dari hasil ACO modifikasi menghasilkan tepi yang tajam dan lebih detail dibandingkan gambar *house (1b)*, peta *(2b)* dari hasil ACO tradisional dan gambar *house* (*1c*) dan peta *(2c)* dari hasil *adaptive* ACO. Berdasarkan uji coba skenario 1 hasil deteksi tepi ACO modifikasi mampu menghubungkan tepi yang terputus dan menghasilkan tepi yang tajam dan lebih detail.

Pengujian yang dilakukan dengan menghitung nilai MSE dan PSNR dari hasil deteksi tepi. PSNR menunjukan sebuah hubungan perbandingan antara nilai maksimum dan nilai kerusakan (*noise*) yang menunjukkan ketelitian dari hasil pengujian. Nilai PSNR yang tinggi menunjukkan bahwa deteksi tepi yang dihasilkan lebih baik. Tabel 4.2 merupakan contoh hasil deteksi tepi yang diperoleh dengan metode ACO tradisional, *adaptive* ACO dan ACO modifikasi.

Tabel 4.2 Contoh hasil uji coba ACO tradisional, *adaptive* ACO dan ACO modifikasi.

Hasil	MSE			PSNR		
	Tradisional	*Adaptive*	Modifikasi	Tradisional	*Adaptive*	Modifikasi
Leaf1	0,012	0,013	0,007	19,311	18,985	21,352
Leaf2	0,030	0,026	0,020	15,242	15,789	16,946
Leaf8	0,013	0,012	0,011	18,740	19,112	19,764
Leaf9	0,017	0,017	0,013	17,581	17,719	18,881
Peta6	0,063	0,062	0,028	12,029	12,071	15,517
Peta7	0,128	0,127	0,113	8,937	8,957	9,454
Peta8	0,108	0,106	0,102	9,669	9,759	9,920
Data2	0,213	0,201	0,167	6,725	6,976	7,771
Data3	0,134	0,121	0,103	8,738	9,180	9,873
Data9	0,140	0,130	0,113	8,544	8,877	9,470
Data10	0,198	0,201	0,162	7,031	6,974	7,908

Tabel 4.2 menunjukkan nilai PSNR deteksi tepi yang dilakukan pada proses uji coba menghasilkan nilai tinggi untuk gambar dengan *background* kontras seperti pada kelompok gambar daun. Sedangkan untuk gambar komplek menghasilkan nilai PSNR yang lebih kecil.

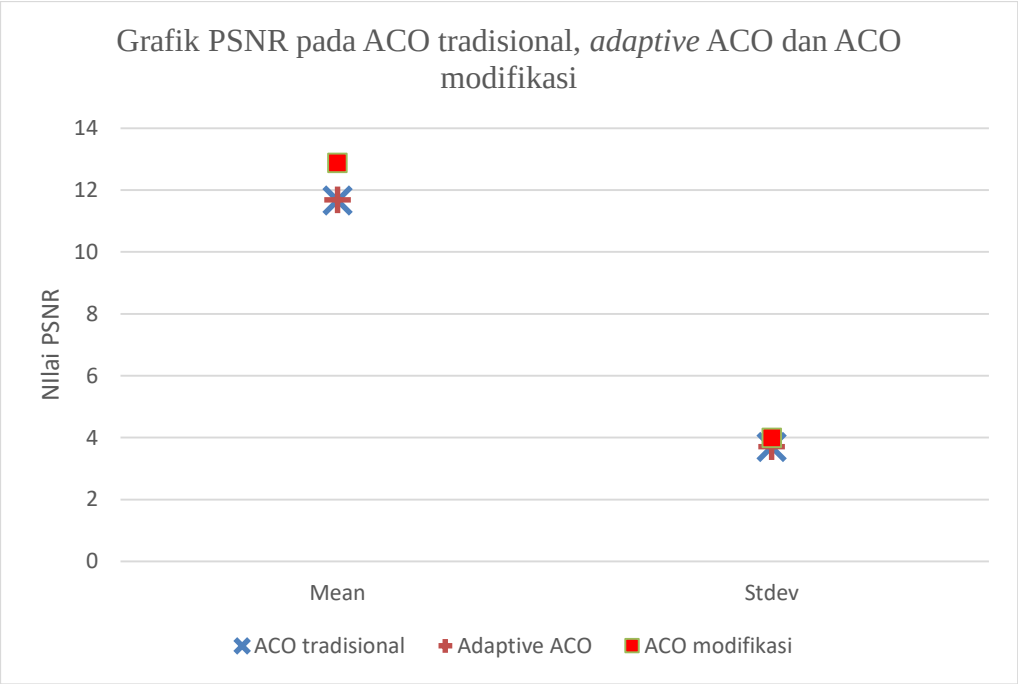
Setelah melakukan penghitungan nilai MSE dan PSNR terhadap semua gambar maka langkah selanjutnya dari hasil MSE dan PSNR tersebut akan dihitung nilai rata-rata, nilai standar deviasi, nilai minimum dan nilai maksimum. Tabel 4.3 merupakan nilai rata-rata, nilai standar deviasi, nilai minimum dan nilai maksimum dari hasil uji coba dengan ACO tradisional, *adaptive* ACO dan ACO modifikasi.

Tabel 4.3 Hasil uji coba ACO tradisional, *adaptive* ACO dan ACO modifikasi

Hasil	MSE			PSNR		
	Tradisional	*Adaptive*	Modifikasi	Tradisional	*Adaptive*	Modifikasi
Mean	0,091	0,089	0,072	11,665	11,713	12,884
Stdev	0,060	0,057	0,050	3,697	3,638	3,990
Min	0,012	0,012	0,007	6,725	6,974	7,771
Max	0,213	0,201	0,167	19,311	19,112	21,352

Pada Tabel 4.3 menunjukkan rata-rata PSNR ACO tradisional 11.665, rata-rata PSNR *adaptive* ACO 11,713 sedangkan ACO modifikasi mencapai 12,884. Nilai rata-rata PSNR ACO modifikasi memiliki nilai tertinggi dibandingkan metode ACO tradisional dan *adaptive* ACO. Kondisi ini dikarenakan ACO modifikasi telah melakukan penempatan semut sehingga lebih optimal dalam menentukan tepi. Nilai rata-rata tradisional dan *adaptive* ACO memiliki selisih kecil. Hal ini disebabkan kedua metode masih melakukan penyebaran semut secara acak.

Grafik nilai PSNR dari hasil uji coba yang dilakukan pada metode ACO tradisional, *adaptive* ACO dan ACO modifikasi ditunjukkan pada Gambar 4.2.



Gambar 4.2 Grafik nilai PSNR pada ACO tradisional, adaptive ACO, ACO modifikasi

Gambar 4.2 merupakan grafik hasil uji coba yang telah dilakukan pada ACO modifikasi, ACO tradisional dan *adaptive* ACO. Nilai rata-rata PSNR pada ACO modifikasi menunjukkan nilai yang paling tinggi dibandingkan rata-rata PSNR ACO tradisional dan *adaptive* ACO.

Hasil uji skenario 1 menunjukkan bahwa terdapat pengaruh penempatan semut berdasarkan nilai *gradient*. Hasil rata-rata nilai MSE untuk ACO modifikasi lebih kecil dibandingkan ACO tradisional dan *adaptive* ACO, sedangkan rata-rata PSNR untuk ACO modifikasi lebih besar dibandingkan ACO tradisional dan *adaptive* ACO. Hal ini menunjukkan bahwa ACO modifikasi lebih baik dibandingkan ACO tradisional dan *adaptive* ACO.

4.3.2 Analisis Hasil Uji Coba Skenario 1

Hasil uji coba skenario 1 menunjukkan hasil dengan akurasi tinggi ketika gambar uji yang digunakan memiliki warna *background* berbeda jauh dengan warna gambar utama seperti pada gambar daun, hal ini ditunjukkan dari nilai PSNR yang besar dibandingkan gambar komplek yang menghasilkan nilai PSNR lebih kecil.

Saat menggunakan metode ACO tradisional untuk proses deteksi tepi didapatkan hasil rata-rata PSNR 11,665. Saat menggunakan metode *adaptive* ACO didapatkan hasil rata-rata PSNR 11,682. Saat menggunakan metode ACO modifikasi yang diusulkan didipatkan hasil rata-rata PSNR 12,884. Sedangkan nilai standar deviasi PSNR ACO modifikasi mencapai 3,709 lebih besar dari ACO tradisional dan adaptive ACO. Kondisi ini disebabkan metode ACO tradisional dan *adaptive* ACO masih melakukan penyebaran semut secara acak, sedangkan ACO modifikasi yang diusulkan sudah menempatkan semut berdasarkan nilai *gradient* sehingga lebih optimal.

4.4 Uji Coba Skenario 2

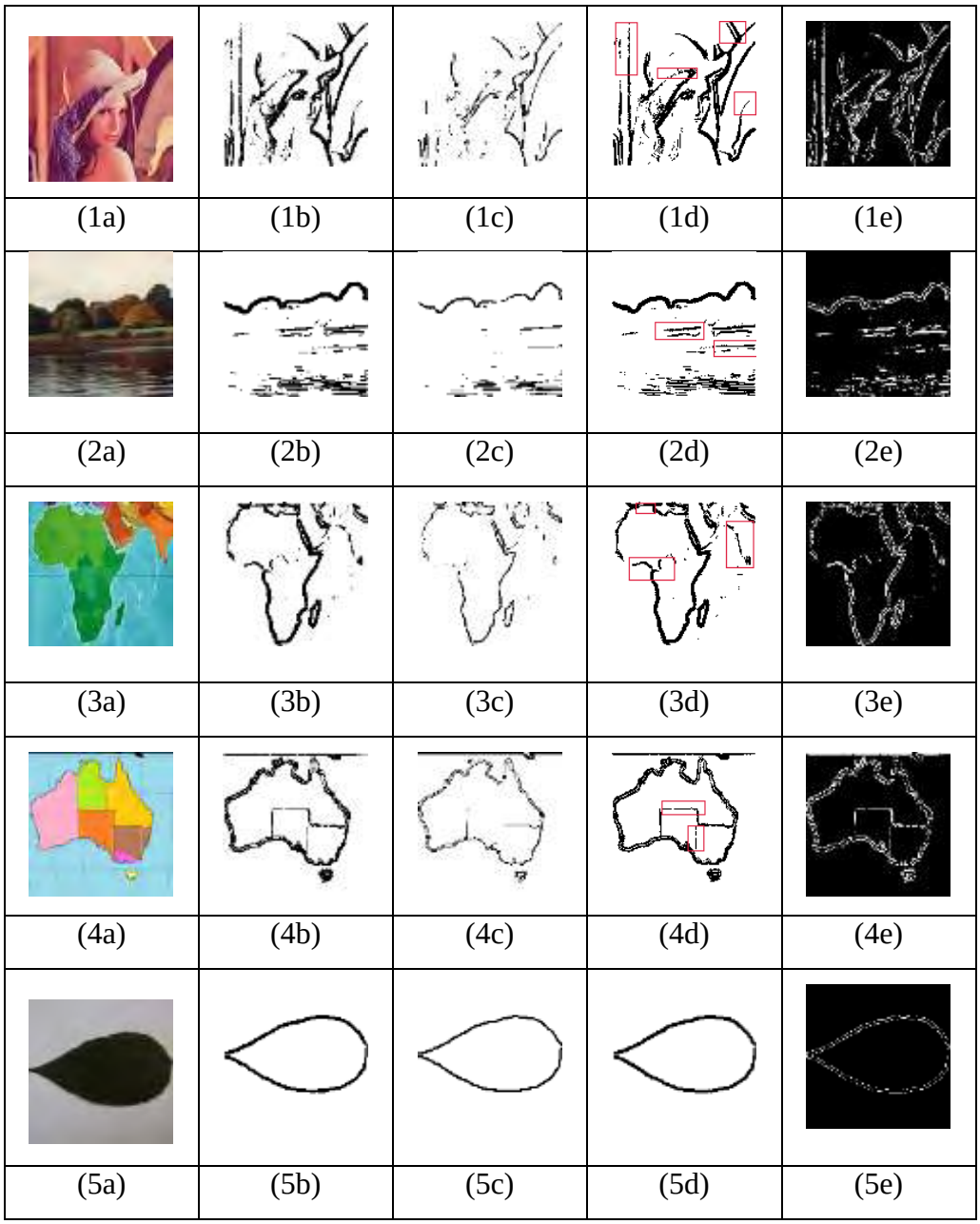
Pada uji coba skenario 2 dilakukan untuk menunjukkan apakah ACO modifikasi mampu memperbaiki deteksi tepi sobel. Skenario ini dilakukan untuk gambar dengan kontras yang berbeda dengan *background* dan untuk gambar komplek. Pada skenario 2 akan membandingkan hasil deteksti tepi dari metode ACO modifikasi yang diusulkan dengan deteksi tepi sobel.

4.4.1 Hasil Uji Coba Skenario 2

Pengujian yang dilakukan dengan menggunakan 30 gambar uji menghasilkan deteksi tepi yang bervariasi. Pada uji coba skenario 2 dilakukan deteksi tepi menggunakan operator sobel. Hasil deteksi tepi sobel ini kemudian dibandingkan dengan hasil deteksi tepi ACO modifikasi untuk mengetahui perbedaan kedua metode.

Pada gambar uji dilakukan penghitungan nilai *gradient* kemudian memilih nilai *gradient* yang lebih dari atau sama dengan *threshold*. Jumlah piksel *gradient*

yang lebih dari atau sama dengan *threshold* akan digunakan sebagai parameter jumlah semut pada metode ACO modifikasi. Contoh hasil pengujian dari deteksi tepi yang telah dilakukan pada ACO modifikasi dan deteksi tepi sobel ditujukkan pada Gambar 4.3.



Gambar 4.3 Gambar uji (1a) (2a) (3a) (4a) (5a); ACO modifikasi (1b) (2b) (3b) (4b) (5b); deteksi tepi sobel (1c) (2c) (3c) (4c) (5c); contoh perbedaan ACO modifikasi dan sobel (1d) (2d) (3d) (4d) (5d); selisih tepi ACO modifikasi dan sobel (1e) (2e) (3e) (4e) (5e)

Pada Gambar 4.3 menunjukkan hasil deteksi tepi dari metode ACO modifikasi gambar lenna *(1b)* memiliki garis tepi yang lebih detail dibandingkan dengan deteksi tepi sobel gambar lenna *(1c).* Perbedaan hasil deteksi antara metode ACO modifikasi dan deteksi sobel ditunjukkan pada gambar lenna *(1d)* yang berupa kotak merah. Selisih hasil uji coba gambar lenna ditunjukkan pada lenna *(1e).* Pada hasil uji gambar peta dengan menggunakan ACO modifikasi mampu menutup tepi yang terputus atau hilang seperti ditunjukkan pada gambar peta *(3b).* Pada gambar peta *(4b)* hasil deteksi tepi mampu menghubungkan tepi yang terputus yang dihasilkan deteksi tepi sobel peta *(4c).* Selisih hasil uji coba gambar peta ditunjukkan pada peta *(4d).* Beberapa contoh ini menunjukkan hasil deteksi tepi ACO modifikasi mampu menutup tepi yang terputus atau tepi yang hilang dari hasil deteksi tepi sobel untuk gambar komplek. Sedangkan untuk gambar dengan kontras yang berbeda dengan *background* hasil deteksi tepi menggunakan metode ACO modifikasi dan deteksi tepi sobel menghasilkan tepi yang mirip. Seperti ditunjukkana pada hasil deteksi tepi ACO modifikasi daun *(5b)* yang mirip dengan hasil deteksi tepi sobel daun *(5c).* Perbedaan yang terjadi yaitu hasil deteksi tepi menggunakan ACO modifikasi menghasilkan tepi yang lebih tajam dan tebal. Hal ini disebabkan pada proses metode ACO modifikasi semakin banyak iterasi yang dilakukan menghasilkan tepi yang semakin tajam dan tebal seperti ditunjukkan pada Tabel 4.1.

4.4.2 Analisis Hasil Uji Coba Skenario 2

Pada uji coba skenario 2 dilakukan penghitungan nilai *gradient* yang lebih dari atau sama dengan nilai *threshold.* Jumlah piksel *gradient* yang lebih dari atau sama dengan *threshold* digunakan sebagai parameter jumlah semut. Berdasarkan hasil pengujian ACO modifikasi mampu menutup hasil deteksi tepi yang terputus atau hilang dari hasil deteksi tepi sobel. Deteksi tepi menggunakan ACO modifikasi menghasilkan tepi yang lebih tajam dan tebal. Hal ini disebabkan pada proses metode ACO modifikasi semakin banyak iterasi yang dilakukan menghasilkan tepi yang semakin tajam dan tebal seperti ditunjukkan pada Tabel 4.1.

Hasil deteksi tepi pada uji coba skenario 2 dengan menggunakan metode ACO modifikasi dan deteksi tepi sobel pada gambar dengan kontras yang berbeda

dengan *background* menghasilkan tepi yang mirip . Hal ini disebabkan gambar uji memiliki warna kontras yang berbeda dengan *background* sehingga mudah mencari tepi dari sebuah gambar.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan aplikasi yang telah dibuat dan hasil yang didapat dari serangkaian uji coba yang telah dilakukan, maka kesimpulan yang dapat diambil dari penelitian ini sebagai berikut :

1. Hasil deteksi tepi menggunakan ACO modifikasi lebih baik dari ACO tradisional dan *adaptive* ACO untuk berbagai macam gambar. Kondisi ini ditunjukkan oleh nilai PSNR yang lebih tinggi.

2. Hasil rata-rata PSNR ACO modifikasi sebesar 12,884, rata-rata PSNR ACO tradisional sebesar 11,665 dan rata-rata PSNR *adaptive* ACO sebesar 11,713. Hal ini menunjukkan metode ACO modifikasi dengan penempatan semut berdasarkan *gradient* dapat meningkatkan akurasi dari hasil deteksi tepi ACO tradisional dan *adaptive* ACO.

3. Metode ACO modifikasi mampu meningkatkan akurasi metode deteksi tepi sobel. Kondisi ini ditunjukkan oleh hasil deteksi ACO modifikasi yang mampu menutup tepi yang terputus atau hilang dari hasil deteksi tepi sobel.

5.2 Saran

1. Pergerakan semut untuk menentukan piksel ketetanggaan secara acak perlu dilakukan perbaikan misalnya dengan menyesuaikan tingkat *gradient* ketetanggaan.

[Halaman ini sengaja dikosongkan]

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan aplikasi yang telah dibuat dan hasil yang didapat dari serangkaian uji coba yang telah dilakukan, maka kesimpulan yang dapat diambil dari penelitian ini sebagai berikut :

1. Hasil deteksi tepi menggunakan ACO modifikasi lebih baik dari ACO tradisional dan *adaptive* ACO untuk berbagai macam gambar. Kondisi ini ditunjukkan oleh nilai PSNR yang lebih tinggi.

2. Hasil rata-rata PSNR ACO modifikasi sebesar 12,884, rata-rata PSNR ACO tradisional sebesar 11,665 dan rata-rata PSNR *adaptive* ACO sebesar 11,713. Hal ini menunjukkan metode ACO modifikasi dengan penempatan semut berdasarkan *gradient* dapat meningkatkan akurasi dari hasil deteksi tepi ACO tradisional dan *adaptive* ACO.

3. Metode ACO modifikasi mampu meningkatkan akurasi metode deteksi tepi sobel. Kondisi ini ditunjukkan oleh hasil deteksi ACO modifikasi yang mampu menutup tepi yang terputus atau hilang dari hasil deteksi tepi sobel.

5.2 Saran

1. Pergerakan semut untuk menentukan piksel ketetanggaan secara acak perlu dilakukan perbaikan misalnya dengan menyesuaikan tingkat *gradient* ketetanggaan.

LAMPIRAN 4

Potongan Program Deteksi Tepi ACO Modifikasi

Potongan program penempatan semut.

```
public void penempatanAnt() {
        int i, j;
        int id_ant = 0;
        int a;
        for (a = 0; a < this.K; a++) {

            i = this.box.get(a).getX();
            j = this.box.get(a).getY();

            semut.setCurrPosition(id_ant, 0, i);
            semut.setCurrPosition(id_ant, 1, j);
            id_ant++;
        }
    }
```

Potongan program penghitungan informasi heutistik.

```
public void heuristicCount() {
     int i, j;
     double h_kanan, h_kiri, v_atas, v_bawah, dkr_atas,
dkr_bawah, dkn_atas, dkn_bawah;
     for (i = 0; i < im_row; i++) {
         for (j = 0; j < im_col; j++) {
             if (i == 0) {
                 v_atas = 0;
                 dkr_atas = 0;
                 dkn_atas = 0;
                 if (j == 0) {
                     dkn_bawah = 0;
                     h_kiri = 0;
                     h_kanan = this.intensity[i][j + 1];
                     v_bawah = this.intensity[i + 1][j];
                    dkr_bawah = this.intensity[i + 1][j +
1];
                } else if (j == im_col - 1) {
                   dkr_bawah = 0;
                .  h_kanan = 0;
                   v_bawah = this.intensity[i + 1][j];
                   h_kiri = this.intensity[i][j - 1];
                  dkn_bawah = this.intensity[i + 1][j -
1];
                 } else {
                     h_kiri = this.intensity[i][j - 1];
                     h_kanan = this.intensity[i][j + 1];
```

```
                    v_bawah = this.intensity[i + 1][j];
                    dkr_bawah = this.intensity[i + 1][j +
1];
                    dkn_bawah = this.intensity[i + 1][j -
1];
                }
            } else if (i == im_row - 1) {
                v_bawah = 0;
                dkr_bawah = 0;
                dkn_bawah = 0;
                if (j == 0) {
                    dkr_atas = 0;
                    h_kiri = 0;
                    v_atas = this.intensity[i - 1][j];
                    h_kanan = this.intensity[i][j + 1];
                    dkn_atas = this.intensity[i - 1][j +
1];              } else if (j == im_col - 1) {
                    h_kanan = 0;
                    dkn_atas = 0;
                    v_atas = this.intensity[i - 1][j];
                    h_kiri = this.intensity[i][j - 1];
                   dkr_atas = this.intensity[i - 1][j -
1];
                } else {
                    h_kiri = this.intensity[i][j - 1];
                    h_kanan = this.intensity[i][j + 1];
                    v_atas = this.intensity[i - 1][j];
                    dkr_atas = this.intensity[i - 1][j -
1];
                    dkn_atas = this.intensity[i - 1][j +
1];
                }
            } else {
                if (j == 0) {
                    h_kiri = 0;
                    dkn_bawah = 0;
                    dkr_atas = 0;
                    h_kanan = this.intensity[i][j + 1];
                    v_atas = this.intensity[i - 1][j];
                    v_bawah = this.intensity[i + 1][j];
                    dkr_bawah = this.intensity[i + 1][j +
1];
                   dkn_atas = this.intensity[i - 1][j +
1];
                } else if (j == im_col - 1) {
                    h_kanan = 0;
                    dkn_atas = 0;
                    dkr_bawah = 0;
                    h_kiri = this.intensity[i][j - 1];
                    v_atas = this.intensity[i - 1][j];
                    v_bawah = this.intensity[i + 1][j];
```

```
                    dkr_atas = this.intensity[i - 1][j -
1];
                    dkn_bawah = this.intensity[i + 1][j -
1];
                } else {
                    h_kiri = this.intensity[i][j - 1];
                    h_kanan = this.intensity[i][j + 1];
                    v_atas = this.intensity[i - 1][j];
                    v_bawah = this.intensity[i + 1][j];
                    dkr_atas = this.intensity[i - 1][j -
1];                  dkr_bawah = this.intensity[i + 1][j +
1];
                   dkn_atas = this.intensity[i - 1][j +
1];
                   dkn_bawah = this.intensity[i + 1][j -
1];
                }
            }
            heuristic[i][j] = (Math.abs(h_kiri - h_kanan)
+ Math.abs(v_atas - v_bawah) + Math.abs(dkn_atas -
dkn_bawah)+ Math.abs(dkr_atas - dkr_bawah));
        }
    }

//note
/*
* h = horizontal
* v = vertical
* dkn = diagonal kanan, artinya dari kanan atas ke kiri
bawah
* dkr = diagonal kiri, artinya dari kiri atas ke kanan
bawah
*/
    }
```

Potongan program posisi piksel *gradient*.

```
public void pikselPosisi() {
        double reg_gradient[] = new double[4];
        int i, j;
        double reg_total = 0;

        for (i = 0; i < im_row; i++) {
            for (j = 0; j < im_col; j++) {
                this.box.add(new
GradientData(this.im_gradient[i][j], i, j));

            }
        }
    }
```

Potongan program proses kontruksi step.

```
public void constructAndUpdateMod() {
        int iteration, step, indexof_ant;
        int row, col;
        int next[];
        double temp_heuristic;
        for (iteration = 0; iteration < N; iteration++) {
            this.tabuListInitialization();
            for (step = 0; step < L; step++) {
                for (indexof_ant = 0; indexof_ant < K;
indexof_ant++) {
                    row =
semut.getCurrPosition(indexof_ant, 0);
                    col =
semut.getCurrPosition(indexof_ant, 1);

                    next = this.probabilityTransition(row,
col, indexof_ant);

                    semut.setOldPosition(indexof_ant, 0,
row);
                    semut.setOldPosition(indexof_ant, 1,
col);

                    semut.setCurrPosition(indexof_ant, 0,
next[0]);
                    semut.setCurrPosition(indexof_ant, 1,
next[1]);

                    this.localPheromoneUpdate(indexof_ant,
step);

                    temp_heuristic =
(heuristic[next[0]][next[1]] +
semut.getDepositPheromone(indexof_ant)) / L;
                    semut.setDepositPheromone(indexof_ant,
temp_heuristic);

                    semut.setTabuList(indexof_ant, 0 + (2
* step), next[0]);
                    semut.setTabuList(indexof_ant, 1 + (2
* step), next[1]);

                }
            }
            globalPheromoneUpdateMod();
        }
        int reg1 = 0, reg2 = 0, reg3 = 0, reg4 = 0;
        int ant_count;
        for (ant_count = 0; ant_count < this.K;
ant_count++) {
```

```
//            semut.getCurrPosition(ant_count, 0);
            if (semut.getCurrPosition(ant_count, 0) <=
midx) {
                if (semut.getCurrPosition(ant_count, 1) <=
midy) {
                    reg1++;
                } else {
                    reg2++;
                }
            } else {
                if (semut.getCurrPosition(ant_count, 1) <=
midy) {
                    reg3++;
                } else {
                    reg4++;
                }
            }
        }
    }
```

Potongan program kontruksi step.

```
public void constructAndUpdateMod() {
        int iteration, step, indexof_ant;
        int row, col;
        int next[];
        double temp_heuristic;
        for (iteration = 0; iteration < N; iteration++) {
            this.tabuListInitialization();
            for (step = 0; step < L; step++) {
                for (indexof_ant = 0; indexof_ant < K;
indexof_ant++) {
                    row =
semut.getCurrPosition(indexof_ant, 0);
                    col =
semut.getCurrPosition(indexof_ant, 1);

                    next = this.probabilityTransition(row,
col, indexof_ant);

                    semut.setOldPosition(indexof_ant, 0,
row);
                    semut.setOldPosition(indexof_ant, 1,
col);

                    semut.setCurrPosition(indexof_ant, 0,
next[0]);
                    semut.setCurrPosition(indexof_ant, 1,
next[1]);
```

```
                    this.localPheromoneUpdate(indexof_ant,
step);

                    temp_heuristic =
(heuristic[next[0]][next[1]] +
semut.getDepositPheromone(indexof_ant)) / L;
                    semut.setDepositPheromone(indexof_ant,
temp_heuristic);

                    semut.setTabuList(indexof_ant, 0 + (2
* step), next[0]);
                    semut.setTabuList(indexof_ant, 1 + (2
* step), next[1]);

                }
            }
            globalPheromoneUpdateMod();
        }

    }
```

Potongan program pengambilan keputusan tepi atau bukan.

```
public void decisionProcessMod() {
        int count;

        double temp_pheromone[][] = new double[im_row *
im_col][2];
        for (count = 0; count < im_row * im_col; count++)
{
            temp_pheromone[count][0] = -1;
        }

        int num_pheromone = 0;
        int i, j, exist;

        count = 0;
        for (i = 0; i < im_row; i++) {
            for (j = 0; j < im_col; j++) {
                exist =
this.havePheromoneExist(temp_pheromone, pheromone[i][j]);
                if (exist == -1) {
                    temp_pheromone[count][0] =
pheromone[i][j];
                    temp_pheromone[count][1] = 1;
                    num_pheromone++;
                    count++;
                } else {
                    temp_pheromone[exist][1]++;
                }
            }
```

```
        }

        //sort pheromone ascending
        Arrays.sort(distinct_pheromone, new
Comparator<double[]>() {
            @Override
            public int compare(double[] entry1, double[]
entry2) {
                double row1 = entry1[0];
                double row2 = entry2[0];
                if (row1 < row2) {
                    return -1;
                } else if (row1 > row2) {
                    return 1;
                } else {
                    return 0;
                }
            }
        });

        double pheromone_times_number[] = new
double[distinct_pheromone.length];
        double pheromone_times_number_total = 0;
        for (count = 0; count < distinct_pheromone.length;
count++) {
            pheromone_times_number[count] =
distinct_pheromone[count][0] *
distinct_pheromone[count][1];
            pheromone_times_number_total +=
pheromone_times_number[count];
        }

        //hitung threshold pheromone
        int indexof_threshold;
        indexof_threshold =
this.otsuThreshold(distinct_pheromone,
pheromone_times_number, pheromone_times_number_total);
//        System.out.println("\nThreshold : " +
distinct_pheromone[indexof_threshold][0]);
        this.threshold =
distinct_pheromone[indexof_threshold][0];

    }
```

DAFTAR PUSTAKA

Agrawal, Prateek, Kaur, Simranjeet, Dhiman, Amita. (2012), "*Analysis and Synthesis of an Ant Colony Optimization Technique for ImageEdge Detection*", IEEE International Conference on Computing Sciences, hal. 127:131.

Anna. B. V, Oppus. C. (2010), "*Image Edge Detection Using Ant Colony Optimization*", International Journal of Circuits, Systems and Signal Processing, Vol 4. Issue 2, hal. 24-33

Charu. G, Sunanda. G. (2013), "*Edge Detection of an Image based on Ant Colony Optimization Technique*", International Journal of Science and Research, Vol.2, Issue 6, hal 114-120

Dorigo. M,Birattari. M, Stutzle. T. (2006), "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique", IEEE Computational Intelligence Magazine, hal. 28-39.

Etemad, S.Ali., dan White, Tony., (2011), "*An Ant Inspired Algorithm for detection of Image Edge Feature*", Elsevier, Vol.11, hal.4883-4893.

Gonzalez. R., Woods. R. (1992), "Digital Image Processing", Addison Wesley, hal 414–428.

Liantoni, F., Kirana, K.C., Muliawati, T.H. (2014), "*Adaptive Ant Colony Optimization based Gradient for Edge Detection*", Journal of Computer Science, Vol.7. Issue.2, hal.78-84.

Otsu. N (1979), *"A threshold selection method from gray level histograms*", IEEE Trans. Syst., Man, Cybern.,vol. 9, hal. 62–66.

Rahebi, Javad., Elmi, Zahra., Nia, A.F., Shayan, Kamran. (2010), "*Digital Image Processing using an Ant Colony Optimization based on Genetic Algorithm*", IEEE, hal.145-149.

Tian, Jiang, Yu, Weiyu, Xie Shengli. (2008), "*An Ant Colony Optimization Algorithm For Image Edge Detection*", IEEE Congress on Evolutionary Computation, hal. 751:756.

Verma, Om P., Hanmandlu. M., Sultania. A Kumar, Dhruv.(2010) "*A Novel Fuzzy Ant System For Edge Detection*", IEEE/ACIS International Conference on Computer and Information Science, hal. 228-233.

Verma, Om P., Singhal, P., Garg, S., Deepti, S.C.(2012), *"Edge Detection Using Adaptive Thresholding and Ant Colony Optimization",* IEEE. hal. 113-118.

Ya-Ping Wong, Victor Chien-Ming Soh, Kar-Weng Ban, Yoon-Teck Bau. (2008), "*Improved Canny Edges Using Ant Colony Optimization"*, IEEE Computer Graphics, Imaging and Visualization, hal 197-202.

Zhang, Jian., He, Kun., Zheng, Xiuqing., Zhou, Jiliu. (2010), "*An Ant Colony Optimization Algorithm for Image Edge Detection*", IEEE, hal.215-219.

BIODATA PENULIS

Febri Liantoni, lahir di Magetan, 07 Februari 1988, anak pertama dari tiga bersaudara. Penulis menempuh pendidikan mulai SDN Balegondo I (1994 – 2000), SLTP 1, Magetan (2000 – 2003), SMA 1, Magetan (2003 – 2006), Penulis melanjutkan studi D4 di Politeknik Elektronika Negeri Surabaya, Institut Teknologi Sepuluh Nopember Surabaya. (2006 - 2010). Penulis dengan Motto "*Orang hebat bukanlah yang berani mati, tapi berani hidup dalam segala situasi*" sempat bekerja sebagai staff IT di Universitas Gadjah Mada Yogyakarta tahun 2011 sampai 2012. Penulis melanjutkan studi S2 di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2013. Bidang minat penelitian yang ditekuni penulis diantaranya adalah Sistem Cerdas, Pengolahan Citra Digital, dan Visi Komputer. Penulis dapat dihubungi melalui email febri.liantoni@gmail.com