New ACO Algorithm for Image Edge Detection

Cristian A. Martínez
Departamento de Informática, Universidad Nacional de Salta
Salta, Argentina
Email: cmartinez@unsa.edu.ar

María E. Buemi
Departamento de Computación, Universidad de Buenos Aires
Buenos Aires, Argentina
Email: mebuemi@dc.uba.ar

***Abstract*—Ant Colony Optimization (ACO) is a metaheuristic based on real behavior of ants to solve optimization problems. Edge detection plays an important role in image processing. It consists in detecting edges or contours in images which allows to extract information from them.**
An ACO algorithm to edge detection is proposed. Using heuristic and knowledge information and applying an improvement operator, binary images including edges detected are obtained. The algorithm was tested with several images (real and synthetic, with and without presence of noise). The results reached were competitive in terms of quality of images and required computational time.

I. Introduction

Edge detection techniques to locate objects in images are used in different situations. Mostly, they are used in order to find edge regions that allow identifying objects in images.

Edge detection includes mathematical methods to locate points or regions in images where changes in image intensity values are detected. In other words, an edge is the boundary between an object and the background. In the literature, edge detectors based on first and second derivative operations are usual. The Sobel, Prewitt and Robert operators are cases of the former while the Laplacian operator is a case of the latter [8]. Based on these operators, Canny presented a systematic method to detect edges [3].

A metaheuristic is defined as a set of algorithmic concepts that can be applied in order to obtain solutions to optimization problems. Their applications to different problems in the areas of planning, telecommunications, transport, graphs, biology, continuous optimization, among others have been successful ([5], [7], [14]).

Ant Colony Optimization (ACO) is a metaheuristic in which a colony of artificial ants cooperate in finding good solutions to optimization problems [5].

Here, a new ACO algorithm for edge detection is proposed. Given a grayscale image (without any enhancement), the colony works in a systematic and cooperative way to build a binary image which includes edges detected from the former. This is possible using heuristic and knowledge information and an operator that improves the solution reached by the colony. Compared to similar proposals ([2], [4], [6], [10], [13], [17]), our algorithm achieves good quality images in a short time.

The paper is organized as follows: Section II is dedicated to edge detection. Section III describes ACO metaheuristic. Our algorithm is explained in Section IV. Experimental results are presented and discussed in Section V. Finally, in Section VI final conclusions and future work are considered.

II. Edge Detection

A. Introduction

Edge detection is essential in image analysis, specially in object segmentation. Detecting an edge in a digital image consists of analyzing relationship between a pixel and its neighborhoods.

Difficulties in localizing edges arise from the presence of noise, shadows, low resolution and excessive local illumination, among others. Let see this situation on one-dimensional signal. Fig. 1 shows from top to bottom: a noisy signal, the impulse response of a filter that performs the first derivative, the result of applying this filter to noisy signal, the impulse response of a filter that results from the convolution of the above mentioned filter with a Gaussian smoothing filter and finally, the result of applying this second filter to noisy signal of first row.

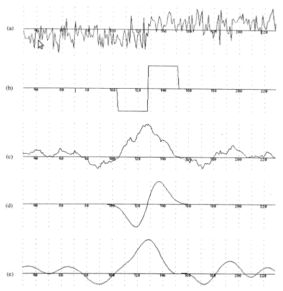


Fig. 1. One-dimensional signal extracted from [3]

B. Edge Detection Methods

Image edges are abrupt changes in intensity values. The gradient provides information on intensity and orientation of those changes which can be used for edge detection. The norm of gradient is given by

$$|G| = \sqrt{G_x^2 + G_y^2},$$

where G_x and G_y are the first derivatives of the image in the x and y directions, respectively. The orientation of the gradient is given by

$$\vec{\theta_x} = arctg \left(\begin{array}{c} G_x \\ \\ G_y \end{array} \right)$$

The Laplacian operator is based on the estimation of the second derivative followed by detection of zero crossings.

The method created by Canny is considered the standard edge detection algorithm in the industry [15]. It consists of a sequence of four steps: noise smoothing using a Gaussian filter, filtering based on gradient to localize intensity changes, suppression of pixels that are not local maxima, and removal of false edges applying a threshold with hysteresis ([3], [8], [9], [12]).

In general, the classic edge detection techniques impose a high computational cost due to the operations performed on each pixel. Recently, methods based on metaheuristics have been proposed such as Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization and Artificial Bee Colony ([2], [4], [10], [11], [13], [17], [18]). Low computational effort, competitive results and task paralellization are relevant features found in these proposals.

The next section is dedicated to the ACO metaheuristic.

III. Ant Colony Optimization (ACO)

A. Introduction

An optimization problem consists in finding values for a set of constrained discrete variables so that its objective function is optimized [5]. The motivation for finding solutions to these problems is related to a better theoretical study and real world application.

Several exact and heuristic algorithms to these problems have been presented. Heuristic algorithms find good solutions in a reasonable time (unlike exact ones, that find optimal solutions). Since the 90s, a new class of heuristic algorithms called metaheuristics has been proposed. Glover and Laguna [7] define them as a master strategy that guides and controls other heuristics to obtain better solutions. Some of well-known metaheuristics are Genetic Algorithms, Tabu Search, GRASP and Ant Colony Optimization [14].

Ants, bees and wasps are social insects which are studied by researchers from different areas. One behavior studied is the search of food. While walking from food sources to their nest and vice-versa, real ants deposit a chemical substance called pheromone on the ground. High levels of pheromone remain on shorter paths because pheromone is laid down faster. This situation causes colony traverses shorter paths.

Proposed by Dorigo, ACO is a metaheuristic inspired by foraging behavior of real ants. By means of a colony of artificial ants which cooperate and communicate indirectly by stigmergy, ACO algorithms reach good solutions to different optimization problems. It has been successfully applied to problems such as vehicle routing, scheduling, image processing, bioinformatics, graphs ([5], [10]). In ACO algorithms, an ant incrementally builds a solution by adding solution components to a partial solution. To achieve this, ant moves on vertexes of a weighted and connected graph, applying a state transition rule. This rule indicates to the ant which vertex must move until path is built. Next, ant can update pheromone trails over its path (used connections of the graph).

In general, an ACO algorithm includes the procedures:

- ConstructAntsSolutions(): controls that all ants construct their solutions incrementally.

```
while $ScheduleActivities()$ do
  $ConstructAntsSolutions()$
  $UpdatePheromones()$
  $DaemonActions()$
end while
```

Fig. 2. ACO Pseudo-code

- UpdatePheromones(): increases pheromone values on paths recently built by the colony and decreases all pheromone values on graph.
- DaemonActions(): can be used to implement centralized actions not individually performed by ants. These actions are related to improve solutions or bias the search process of solutions.

They are managed by ScheduleActivities(). Interaction among them depend on the problem and ACO algorithm used. In Fig. 2, an ACO pseudo-code is shown. For more details, see [5].

B. Ant Colony System

Ant System (AS) was the first ACO algorithm. Later, Max-Min Ant System (MMAS) and Ant Colony System (ACS) were presented as its extensions. ACS differs from AS in the decision rule and pheromone update [5].

Ant *k* located on vertex *i* moves towards vertex *j* according to a pseudo-random proportional rule (Eq. 1):

$$j = \begin{cases} argmax_{l \in \aleph_i^k} \{\tau_{il}[\eta_{il}]^\beta\} & \text{if q} \leq q_0 \\ \frac{\tau_{ij}[\eta_{ij}]^\beta}{\sum_{l \in \aleph_i^k} \tau_{il}[\eta_{il}]^\beta} & \text{otherwise} \end{cases} \tag{1}$$

where q is a random variable distributed uniformly in [0,1), q_0 is an algorithm parameter ($0 \leq q_0 \leq 1$) and β is another parameter which adjusts the relative importance of heuristic information.

So, with probability q_0 ant *k* chooses best neighbor vertex (according to a greedy criteria) and with probability $(1 - q_0)$, chooses probabilistically with the usual decision rule.

After moving towards vertex *j*, ant *k* updates pheromone value related to last connection traversed (Eq. 2):

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{2}$$

where ξ $(0 < \xi < 1)$ and τ_0 (initial pheromone trial value) are algorithm parameters as well. Local pheromone update (Eq. 2) tries to diversify the search process of solutions.

At the end of each iteration, the best ant bs only updates pheromone on its tour T^{bs}. The global pheromone update follows the equation (Eq. 3):

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs} \; \forall (i, j) \in T^{bs} \tag{3}$$

where $\Delta\tau_{ij}^{bs} = \frac{1}{C^{bs}}$ is a value associated to tour length traversed by the best ant and ρ is a parameter related to pheromone evaporation rate $(0 < \rho \leq 1)$.

It is important to note that previous equation (Eq. 3) makes pheromone evaporation and deposit at the same time.

Next, our algorithm for Edge Detection is presented.

IV. ACO Algorithm

Using heuristic and knowledge information, our colony of ants traverses a grayscale image in search of edges. By means of an intensity matrix I ($0 \leq I(i,j) \leq 255,\ i = 1, \ldots, m; j = 1, \ldots, n$), the ants move over the image pixels (that represent vertexes of the graph) in order to detect edges and finally obtain a binary image as output.

A. Initialization Phase

At the beginning, pheromone and heuristic matrices (m×n both) must be initialized. The first is initialized to small positive value τ_0 ($\tau_{ij} = \tau_0\ \forall i, j$). The second matrix, is initialized according to image local information (Eq. 4):

$$\eta_{ij} = \mu_{ij} * \sigma_{ij} \tag{4}$$

where μ_{ij} and σ_{ij} are mean and variance intensity values for every pixel. These values are calculated considering the Moore neighborhood (8-neighborhood) around every pixel at position(i,j). Other proposals only use expressions related to intensity mean as heuristic information ([2], [17]). In our work, variance is also included because a pixel with high variance is candidate to be and edge [16].

Also, vertical and horizontal gradient matrices G_x and G_y are needed. They are obtained applying the Sobel operator on the input image [8]. Both will be used when ants apply the transition rule (Eq. 5) to construct their tours.

Next, in each iteration, artificial ants will start their tours on pixels with high variance. This will allow us to analyze image regions with possible edges. When all of them have been visited, they will start in different positions of the image to diversify the edge search process.

B. Construction Phase

In every iteration, ants move on non-visited pixels in search of edges. In each move, each ant applies the next rule based on ACS (Eq. 1):

$$(i,j) = \begin{cases} argmax_{(l,m)\in\aleph_{(i_0,j_0)}}\{\tau_{lm}[\eta_{lm}]^{\beta}\} & \text{if q} \leq q_0 \\ \frac{\tau_{ij}[\eta_{ij}]^{\beta}}{\sum_{(l,m)\in\aleph_{(i_0,j_0)}}\tau_{lm}[\eta_{lm}]^{\beta}} & \text{otherwise,} \end{cases} \tag{5}$$

where $\aleph_{(i_0,j_0)}$ is its restricted neighborhood. Besides excluding visited pixels, this neighborhood rejects non-maximum pixels. The latter is based on Canny [3]. So, the rule proposed (Eq. 5) encourages selection of local maximals or possible candidates (pixels with high variance respect to its neighborhood). Each ant will make up to L moves (length of its tour).

Based on the literature ([2], [17]), after moving to a new pixel each ant locally updates pheromone levels on its location (i,j)(Eq. 6):

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\eta_{ij} \tag{6}$$

Unlike ACS (Eq. 2), the rule used tries that other ants (in current iteration) consider the previous one as an edge.

After all ants make their tours and prior starting a new iteration, global pheromone update is performed (Eq. 7):

$$\tau \leftarrow (1-\psi)\tau + \psi\tau_0 \tag{7}$$

This rule is different regarding ACS (Eq. 3). Based on Tian et al. [17], it updates pheromone levels over the whole image in order to avoid stagnation at certain regions of image.

C. Improvement Phase

Unlike another problems (such as TSP described in [5]) where ants build complete solutions each, in our case all ants cooperate to build one solution. So, after stopping condition is met, final pheromone matrix is obtained. This matrix will be used to generate the output binary image. Seen as a solution, the pheromone matrix can be improved to achieve a better one.

Improvement phase (or image pre-binarization phase) consists in detection and correction of pixels labeled incorrectly. Pixels detected inaccurately as edges or non-edges are analyzed and then corrected if necessary. Based on the false edge removal method proposed by Ahmad and Choi [1], final pheromone and variance matrices are locally analyzed using 3×3 windows. For each window, the pixel with higher variance is determined. If this value is less than global threshold GT (algorithm parameter) and its pheromone value associated is greater than algorithm threshold AT (explained in next subsection), then the pheromone value is updated to 0. That means the pixel is correctly labeled as non-edge. On the other hand, if the value is greater than global threshold and the pheromone value is less than algorithm threshold, then the pheromone value will be 255. This last case occurs when the pixel was wrongly labeled as non-edge.

D. Binary Image

As aforementioned, using the final pheromone matrix the output binary image (including edges detected) will be generated.

Based on Tian et al. [17], all final pheromone values τ_{ij} must be compared to an algorithm threshold AT in order to obtain binary matrix E used to generate binary image. On each pixel position (i,j) if the pheromone value associated is greater than AT then E(i,j)=1 (edge detected). Otherwise E(i,j)=0 (non-edge detected).

Following the first iteration, an initial algorithm threshold AT is met (Eq. 8):

$$AT = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\tau_{ij}}{m \times n} \tag{8}$$

Subsequently, AT is the average of two averages: average of pheromone values less $\bar{\tau}_L$ and greater $\bar{\tau}_U$ than previous AT obtained (Eq. 9):

$$AT = \frac{\bar{\tau}_L + \bar{\tau}_U}{2} \tag{9}$$

The stopping condition of our algorithm is mainly based on AT values. When two consecutive AT values are similar, the stopping condition is met. This indicates that the algorithm will not obtain new edges.

```
Require: $imageM, K, L, \tau_0, \rho, \beta, \psi, GT, \epsilon, maxT$
  $G_x, G_y \leftarrow applyFilter(imageM)$
  $initializePheromoneM(pheromoneM, \tau_0)$
  $heuristicM, varianceM \leftarrow generateM(imageM)$
  $initializeVisitM(visitM, varianceM)$
  $initializeAditionalInformation()$
  $flag \leftarrow False$
  while $not flag$ do
    for $l = 1$ to $L$ do
      for $k = 1$ to $K$ do
        if $l = 1$ then
          $restartLocalInformation()$
          $i, j \leftarrow generateStartPosition(visitM)$
        end if
        $i, j \leftarrow applyDRule(pheromoneM, i, j, heuristicM, G_x, G_y, \beta)$
        $updateLocalPheromone(pheromoneM, i, j, \rho)$
      end for
    end for
    $updateGlobalPheromone(pheromoneM, \psi)$
    $oldAT \leftarrow AT$
    $AT \leftarrow calculateThreshold(oldAT, pheromoneM)$
    $flag \leftarrow stoppingConditionIsMet(AT, oldAT, \epsilon, maxT)$
  end while
  $improveSolution(pheromoneM, varianceM, GT)$
  $edgeM \leftarrow generateEdgeM(pheromoneM, AT)$
  return $edgeM$
```

Fig. 3. Pseudo-code of ACO algorithm for edge detection

E. Pseudo-code

Here, it will be briefly explained how image edge detection was solved using a modified version of ACS.

After algorithm parameter values (explained in next section) and intensity matrix associated to input image are obtained, initialization phase is executed. This consists in getting gradient, heuristic and variance matrices and initializing visit and pheromone ones. Then in each iteration ants construct their tours in search of edges, applying a decision rule and locally updating pheromone levels. Before starting a new iteration, a global pheromone update is performed and an algorithm threshold is calculated. When the stopping condition is fulfilled (similar AT values or maximum CPU time), the final pheromone matrix is improved. Using this matrix, a binary image with edges detected is finally reached.

In Fig. 3, a pseudo-code of our algorithm is shown.

V. Computational Tests

This section is devoted to tests performed using different images in order to measure the quality of our proposal.

A. Algorithm Parameters

The algorithm parameters and their values are:

- K = $\lfloor\sqrt{m \times n}\rfloor$: number of ants.
- L = 40: maximum number of moves per ant in each iteration.
- τ_0 = 0.0001: initial value of each element in pheromone matrix.
- ρ = 0.1: pheromone evaporation rate adopted in Eq. 6.
- β = 0.1: influence factor of heuristic information used in Eq. 5.
- ψ = 0.05: pheromone decay value applied in Eq. 7.
- GT: global threshold used in improvement phase.
- ϵ: tolerance value used to determine if stopping condition is met. In all tests was adopted 0.00001.
- maxT: maximum cpu time applied in case of AT values are not similar. In all tests 120 seconds was used.

All values except ϵ and $maxT$ have been taken from the literature ([1], [2], [5], [10], [17]) and were kept during tests.

The algorithm was implemented in JAVA and the tests were performed in Eclipse on a notebook with a CORE I5 2.4 GHz CPU and 4 GB RAM running on Windows 7.

B. Results

The tests were carried out on images with and without noise. Four typical test images (Peppers, Cameraman, Baboon and Lena) and an artificial one were used. The first tests (Figs. 4-7) were performed using noiseless images and our results were compared to Tian et al. [17], Baterina and Oppus [2], Jevtić and Andina [10], and Etemad and White [6]. Li et al. [11] and Yigitbasi and Baykan [18] were not considered because they used lesser-known and not available images for tests. The second tests were focused on achieving quantitative measures for edge detection. A ground-truth image was used (Fig. 8) and the MSE and SSIM values on binary images obtained by our algorithm and the Sobel and Canny methods were respectively calculated. Given a reference image f and a test image g, both of size $m \times n$, the MSE between f and g is defined by (Eq. 10):

$$MSE(f, g) = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} (f_{ij} - g_{ij})^2 \qquad (10)$$

and the SSIM is defined as (Eq. 11):

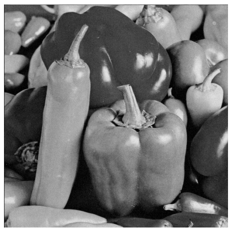
$$SSIM(f, g) = l(f, g)c(f, g)s(f, g) \qquad (11)$$

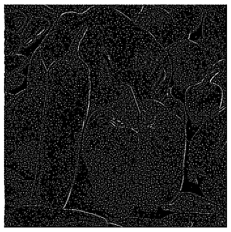
where

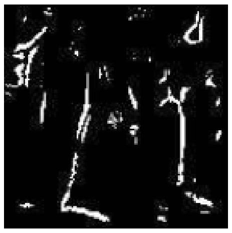
$$\begin{cases} l(f,g) = \frac{2\mu_f\mu_g + C_1}{\mu_f^2 + \mu_g^2 + C_1} \\ c(f,g) = \frac{2\sigma_f\sigma_g + C_2}{\sigma_f^2 + \sigma_g^2 + C_2} \\ s(f,g) = \frac{\sigma_{fg} + C_3}{\sigma_f\sigma_g + C_3} \end{cases}$$

Finally, the last tests were performed on contaminated images with Gaussian noise ($\sigma = 0.02$) (Fig. 9-11) and our results were compared to obtained by the two aforementioned methods.

Regarding the first tests, our algorithm obtained a competitive result (Fig. 4.b regarding Fig. 4.d that achieved the best binary image) and the other results were better (in terms of edge detection and clarity) compared to other works (Fig. 5.b respect to 5.d), (Fig. 6.b as regards as Fig. 6.c) and (Fig. 7.b regarding Fig. 7.d). In relation to MSE and SSIM comparison, our results (Fig. 8.b, MSE=0.08 SSIM=0.31) turned out superior as regard as Canny (Fig. 8.c, MSE=0.12 SSIM=0.22) and Sobel (Fig. 8.d, MSE=0.10 SSIM=0.25). This indicates our algorithm makes less errors on edge detection


(a) Input image


(b) ACO


(c) Tian et al. [17]

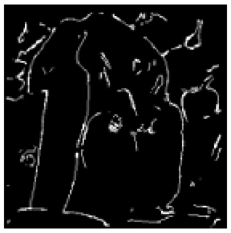
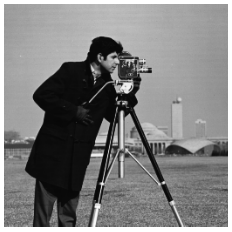
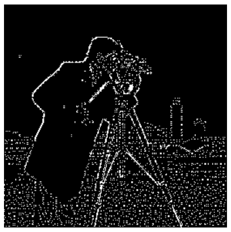
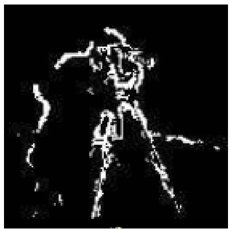

(d) Baterina and Oppus [2]

Fig. 4. Peppers and output binary images


(a) Input image


(b) ACO


(c) Tian et al. [17]

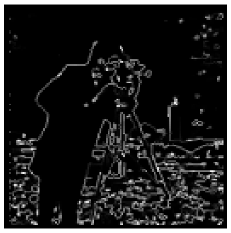

(d) Jevtić and Andina [10]

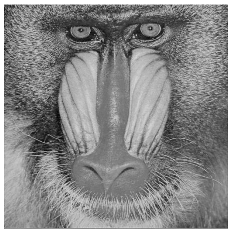
Fig. 5. Cameraman and output binary images

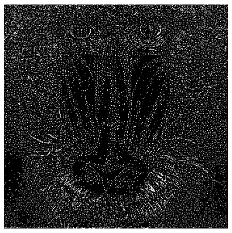
and the binary image associated is more similar to Fig. 8.a. Finally, our algorithm achieved good results on noisy images (Figs. 9.b, 10.b and 11.b). On the contrary, results of Canny (9.c, 10.c, 11.c) and Sobel (9.d, 10.d, 11.d) show high levels of noise and lower edge detection.

Our binary edges were obtained using GT=50. Although other values were tested, their results were dismissed. Finally, the computational time to obtain binary images (Fig. 4.b=7.9, Fig. 5.b=3.7, Fig. 6.b=8.1, Fig. 7.b=8.7, Fig. 9.b=4.2, Fig. 10.b=3.8, Fig. 11.b=6.9) is around 3 to 9 seconds. This time (best of 5 runs per instance) is considered competitive for images used during these tests.

VI. Conclusions and Future Work

A new ACO algorithm for image edge detection has been proposed. Based on tests performed on images with and without noise, our proposal is robust and competitive compared to other works from literature. This was achieved by means of


(a) Input image


(b) ACO

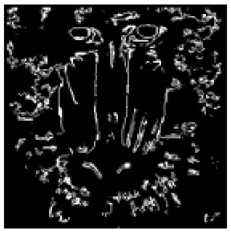
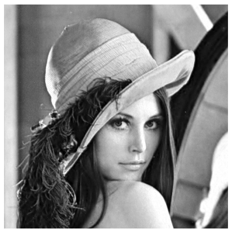
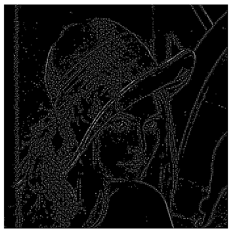
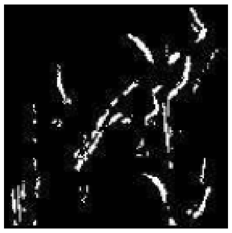

(c) Baterina and Oppus [2]

Fig. 6. Baboon and output binary images


(a) Input image


(b) ACO


(c) Tian et al. [17]

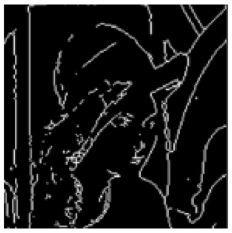
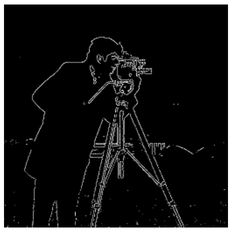
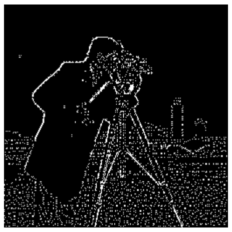
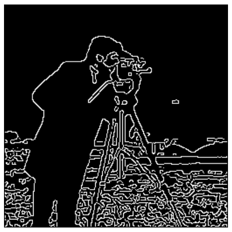

(d) Etemad and White [6]

Fig. 7. Lena and output binary images


(a) Input image


(b) ACO


(c) Canny

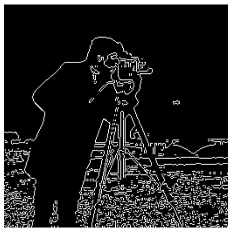
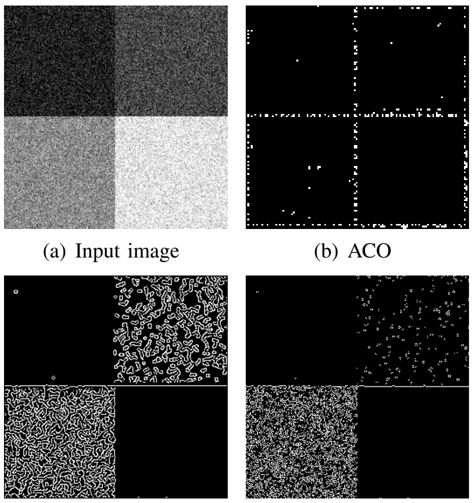
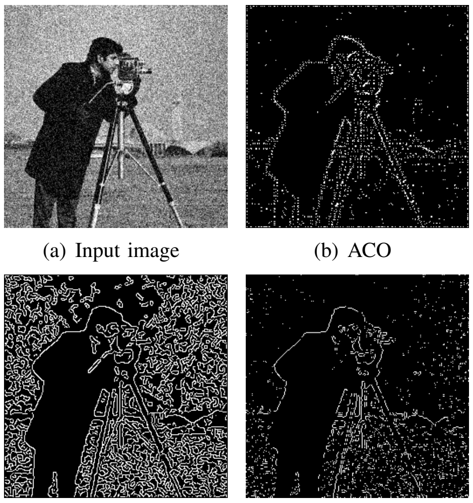

(d) Sobel

Fig. 8. Ground-truth Cameraman and output binary images


(a) Input image (b) ACO

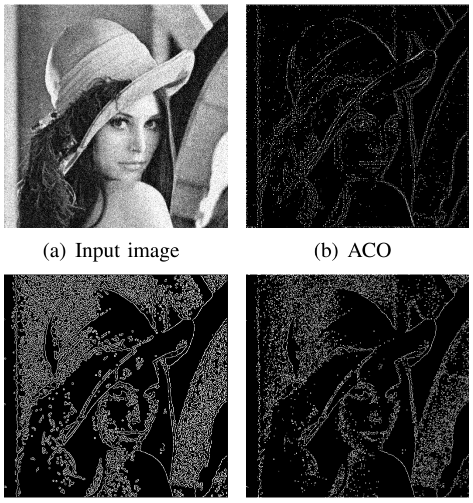
(c) Canny (d) Sobel

Fig. 9. Noisy 4 Regions and output binary images


(a) Input image (b) ACO

(c) Canny (d) Sobel

Fig. 10. Noisy Cameraman and output binary images


(a) Input image (b) ACO

(c) Canny (d) Sobel

Fig. 11. Noisy Lena and output binary images

efficient analysis of image in search of edges using suitable heuristic information and restricted neighborhoods which discarded pixels with low chances of becoming edges. Also, the improvement operator corrected edge detections made by the colony.

In the future, an extended version of the algorithm will be proposed in order to analyze images of large size and resolution such as satellite images.

Acknowledgment

The authors would like to thank the anonymous referees whose comments have improved this paper.

References

[1] M. Ahmad and T. Choi, "Local Threshold and Boolean Function Based Edge Detection", *IEEE Transactions on Consumer Electronics*, Vol. 45, No. 3, pp. 674-679, Aug. 1999.

[2] A. Baterina and C. Oppus, "Image Edge Detection using Ant Colony Optimization", *WSEAS Transactions on Signal Processing*, Vol. 6, No. 2, pp. 58-67, Apr. 2010.

[3] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679-698, Jun. 1986.

[4] S. Djemame and M. Batouche, "Combining Cellular Automata and Particle Swarm Optimization for Edge Detection", *International Journal of Computer Applications*, Vol. 57, No. 14, pp. 16-22, Nov. 2012.

[5] M. Dorigo and T. Stützle, *Ant Colony Optimization*, London, England, MIT Press, 2004.

[6] S. Etemad and T. White, "Ant ant-inspired algorithm for detection of image edge features",*Applied Soft Computing*, Vol. 11, pp.4883-4893, Jun. 2011.

[7] F. Glover and M. Laguna, *Tabu Search*, Boston, USA, Kluwer Academic Publishers, 1998.

[8] R. Gonzalez and R. Woods, *Digital Image Processing (3rd Edition)*, NJ, USA, Prentice-Hall, 2008.

[9] A. Jain, *Fundamentals of Digital Image Processing*, NJ, USA, Prentice-Hall, 1989.

[10] A. Jevtić and D. Andina, "Adaptive Artificial Ant Colonies for Edge Detection in Digital Images", *36th Annual Conference on IEEE Industrial Electronics Society*, Glendale, USA, pp. 2813-2816, Nov. 2010.

[11] Y. Li, B. Bai and Y. Zhang, "An Adaptive Inmune Genetic Algorithm for Edge Detecion", *Proceedings of the 3rd International Conference on Intelligent Computing*, Qingdao, China, pp. 565-571, Aug. 2007.

[12] J. Lim, *Two-dimensional Signal and Image Processing*, NJ, USA, Prentice-Hall, 1990.

[13] T. Manish, D. Murugan and T. Ganesh Kumar, "Edge Detection by combined Canny Filter with Scale Multiplication & Ant Colony Optimization", *Proceedings of the 2nd International Conference on Computational Science, Engineering and Information Technology*, Coimbatore, India, pp. 497-500, Oct. 2012.

[14] P. Mills, E. Tsang, Q. Zhang and J. Ford, "A survey of AI-based meta-heuristics for dealing with local optima in local search", Technical Report, CSM-416, University of Essex, pp. 1-47, 2004.

[15] E. Nadernejad and S. Sharifzadeh, "Edge Detection Techniques: Evaluations and Comparisons", *Applied Mathematical Sciences*, Vol. 2, No. 31, pp. 1507-1520, 2008.

[16] B. Neupane, Z. Aung and W. Woon, "A new Image Edge Detection Method using Quality-based Clustering", *Proceedings of the 10th LASTED International Conference on Visualization, Imaging and Image Processing*, Banff, Canada, pp. 20-26, Jul. 2012.

[17] J. Tian, W. Yu and S. Xie, "An Ant Colony Optimization Algorithm for Image Edge Detection", *IEEE Congress on Evolutionary Computation*, Hong Kong, China, pp. 751-756, Jun. 2008.

[18] E. Yigitbasi and N. Baykan, "Edge Detection using Artificial Bee Colony Algorithm (ABC)", *International Journal of Information and Electronics Engineering*, Vol. 3, No. 6, pp. 634-638, Nov. 2013.