DOI: http://dx.doi.org/10.7551/978-0-262-33027-5-ch071

An Efficient Ant Colony System for Edge Detection in Image Processing

Yara Khaluf[1] and Syam Gullipalli[2]

University of Paderborn, Paderborn, Germany
[1]yara.khaluf@uni-paderborn.de
[2]syam@mail.uni-paderborn.de

Abstract

Edge detection is a fundamental procedure in image processing, machine vision, and computer vision. Its application area ranges from astronomy to medicine in which isolating the objects of interest in the image is of a significant importance. However, performing edge detection is a non-trivial task for which a large number of techniques have been proposed to solve it. This paper investigates the use of Ant Colony Optimization — a prominent set of optimization heuristics — to solve the edge detection problem. We propose two modified versions of the algorithm Ant Colony System (ACS) for an efficient and a noise-free edge detection.

1 Introduction

Edge detection is a fundamental process in analyzing images. It attempts to find points at which the image brightness has discontinuities. These discontinuities allow changes in pixels intensities which may define the boundaries of an object. Applying edge detection may reduce significantly the amount of data to be processed by filtering out the information that is less relevant and preserving the important structural properties of an image. Therefore, edge detection is involved as an essential stage in a wide range of applications. Examples include medicine applications, pattern recognition, machine vision, image analysis, automotive applications, and others. Furthermore, edge detection should be performed in a reliable way as the validation and the efficient completion of the following stages in the image processing rely on it. At the same time, obtaining ideal edges from real life images with a moderate complexity is a challenging task.

In general, complex mathematical functions such as the first and the second order derivatives of the image are used for performing edge detection. Moreover, smoothing (filtering) functions are required to remove the noise obtained from the detection process. Applying such techniques increases the complexity of the detecting process and they may fail in maximizing the number of detected edges. Ant Colony Optimization Colorni et al. (1991) is a set of heuristics for optimization that has proven its efficiency in a large number of areas including scheduling problems Martens et al. (2007); Blum (2005), vehicle routing problem Toth and Vigo (2002); Secomandi (2000), assignment problems Stützle and Hoos (2000), and others. ACO is inspired by the foraging behavior of ants in which ants leave pheromone trails on the ground while searching for food in order to guid the search of other ants to the best food sources. There are several ACO algorithms presented in the literature, see Dorigo and Stützle (2004) and Christian (2005). They vary either in the way they select their next choice in the solution space or in their way of updating the pheromone trails. In this paper, we use the particular variant Ant Colony System (ACS) Dorigo and Gambardella (1997) to solve the problem of edge detection. ACS applies two rules for the pheromone update: local and global, in order to achieve a better search of the problem space. Additionally, its probabilistic decision rule focuses on both exploring the solution space and exploiting previous solutions. We propose two modified algorithms of ACS, which are developed to obtain an efficient edge detection with a minimized complexity.

The rest of the paper is organized as follows: section 2 reviews a list of related works in the field of edge detection. Section 3 illustrates the Ant Colony System used in solving the edge detection problem. In Section 4 we present the first algorithm (FACS) proposed to solve the edge detection based on ACS. Experimental results of FACS are reported in Section 5. The Extended FACS (the second algorithm proposed) is presented in Section 6 and its experimental results are reported in Section 7. A comparison between the performance of the algorithms is given in Section 8 and the paper is concluded in section 9.

2 Related Work

Several approaches have been introduced in the literature for edge detection and the solutions proposed were mainly based on complex mathematical techniques. Two of the wide-used methodologies are the gradient and the Laplacian. In the gradient, detecting edges is performed by searching for the maximum and minimum in the first derivative of the image. Whereas in the Laplacian, it is done by searching for the zero crossings in the second derivative of the im-

Yara Khaluf, Syam Gullipalli (2015) An Efficient Ant Colony System for Edge Detection in Image Processing. Proceedings of the European Conference on Artificial Life 2015, pp. 398-405

age Ziou and Tabbone (1998). One of the prominent algorithms that has applied the first order derivative for edge detection is the Canny's algorithm Canny (1986). This algorithm has introduced an edge detection operator using the calculus of variations, Gelfand et al. (2000). Canny's algorithm applies a set of mathematical functions for detecting the edges and requires image filtering for removing the noise. Despite being one of the most applied edge detection techniques, Canny's algorithms is associated with a set of preconditions for enabling its application in addition to the high complexity associated with the execution of this algorithm. Prewitt has introduced in Prewitt (1970) another edge detection operator that uses the first order derivative of the image by computing an approximation of the gradient of the image intensity function. Another edge detection operation that computes an approximation of the gradient of an image through discrete differentiation is the one introduced by Roberts cross in Davis (1975). The main disadvantage of the first order derivative operators is their sensitivity to the noise while detecting both the edges and their orientations. On the other hand, detecting the image edges using the second order derivative of the image was first presented in Haralick (1984). This technique captures the rate of change in the intensity gradient and looks for the zero crossing of the second derivative of the image. The Marr-Hildreth operator Marr and Hildreth (1980) is a well-known operator that uses the second order derivative by convolving the image with the Laplacian of the Gaussian function. The Marr-Hildreth operator suffers from the possibility of generating responses that do not correspond to edges and are referred to as false edges. The second order derivative operators suffer, as the first order derivative ones, from their sensitivity to noise. Therefore using filtering techniques becomes a necessity for both types of operators, which increases the complexity of the edge detection process.

Different from the complex mathematical approaches used for the sake of edge detection, Ant Colony Optimization has offered efficient heuristics to deal with the problem. For example, the authors in Rezaee (2008) have investigated the use of a particular ACO variant, referred to as the ant system (AS) to detect edges. In Nezamabadi-pour et al. (2006) a modified ant colony system has been presented to solve the edge detection problem. The authors have derived an experimental relationship between the size of the image and the algorithm parameters. The quality of the detection was further improved in the ant colony algorithm presented in Tian et al. (2008). Another work in which ant colony system was involved in detecting edges was in Zhuang (2004), in which a perceptual graph was used to represent the processed image. A hybrid edge detection using Canny edge detector and an ant colony algorithm was presented in Manish et al. (2013). Even quantum computing was proposed to be used in combination with ant colony system to detect edges in Jian et al. (2012). In this work, the authors have applied matrix multiplications and complex mathematical functions such as trigonometric functions in order to detect edges. Many experiments have proven that Ant Colony System (ACS) — a variant of ant colony algorithms — dominates Ant system (AS) for edge detection problems, as we can see in Baterina and Oppus (2010). In this work, the authors investigated the ACS algorithm for solving edge detection problems. However, noise filtering was still a necessary step for obtaining a feasible result.

The work presented in this paper proposes the use of two modified versions of the ACS algorithm for solving edge detection problems. Moreover, it offers a comparison between the performance of the proposed algorithms and the performance of the ACS system presented in Baterina and Oppus (2010) (without noise filtering). A remarkable improvement, in terms of both quality and complexity, is achieved.

3 Ant Colony System (ACS)

Ant colony algorithms are population based meta-heuristics which share two main components: the probabilistic decision rule and the pheromone update rule. The probabilistic decision rule is the rule used by each ant to decide concerning its next step in the solution space. This decision is performed probabilistically based on both the problem heuristic information and the exploitation of the experiences of other ants that have explored the space before. Updating the pheromone trails includes two operations: pheromone evaporation to forget about the previous bad solutions and pheromone deposit to reinforce the good solutions. The pheromone trails, left by the ants are exploited by other ants to improve the quality of their search.

Ant Colony System (ACS) is a particular variant of the ant colony algorithms which has its own probabilistic decision and pheromone update rules. The decision rule applied in ACS exploits, with a particular probability, the search experience accumulated by other ants. Thus, the probability that the ant moves to position j is defined as in the following:

$$j = \begin{cases} \arg\max_{j \in N_i^k} (\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}) & \text{if } q \leq q_0 \quad \text{(Exploitation)} \\ J & \text{otherwise} \quad \text{(Exploration)} \end{cases} \tag{1}$$

where q is a random variable uniformly distributed in $[0, 1]$, q_0 ($0 \leq q_0 \leq 1$) is an algorithm parameter, N_i^k is the set of unvisited neighbors, and J is defined using the probabilistic decision rule of the ant colony algorithms, given by:

$$p_{ij}^{(k)}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \tag{2}$$

where τ_{ij} is the pheromone value deposited on the edge (i,j). η_{ij} is the heuristic information assigned to the edge (i,j). α and β are two parameters that determine the relative influence of both the pheromone trails and the heuristic information.

ACS applies two rules for updating the pheromone trails: the global update and the local update. The global pheromone update is applied after each iteration by only one ant (the best-so-far), as in the following:

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

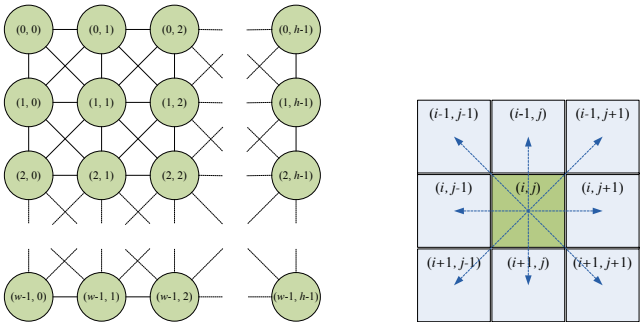
where ρ is the evaporation parameter and $\Delta\tau_{ij}^{bs}$ is the amount of pheromone deposited by the best-so-far ant.
One of the main difference between ACS and other ant colony algorithms is the use of local pheromone. The purpose of using this pheromone is to reduce the probability for ants to select an edge that was selected before. The updating of local pheromone is done by each ant immediately after crossing a particular edge during the phase of the solution construction. The rule for updating the local pheromone is given by:

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0 \tag{4}$$

where ξ, $0 < \xi < 1$, and τ_0 are two parameters of the algorithm.

4 Focused ACS Algorithm for Edge Detection (FACS)

For enabling the application of any ant colony algorithm on edge detection problems, the image should be transformed into a graph. This graph is generated by introducing a matrix $I_{w\times h}$ that represents the intensity at each pixel of the image, as in Figure 1(a). Each pixel is considered as a node and is connected to its neighbors (horizontal, vertical, and diagonal) to form the graph edges.



(a) *Graph representation of $w \times h$ image.*

(b) *Feasible neighborhood of pixel (i, j).*

Figure 1 *– Graph representation of the edge detection problem.*

As illustrated in Figure 1(b), the neighborhood of each pixel is defined so that each ant can move in 8 directions if applicable. Repeated visits to the same node are restricted by maintaining a piece of memory by each ant. However, it is allowed in deadlock situations. In general, the heuristic information defined for most of the problems, which were solved with ant colony algorithms, is encoded on the edges of the graph that represents the problem. Differently, for edge detection problems, the heuristic information is encoded at the nodes of the graph (the pixels) and is defined based on the intensity value of the pixel, as in the following:

$$\eta_{ij} = \frac{V_c(I_{ij})}{V_{max}} \tag{5}$$

I_{ij} is the intensity of pixel (i, j). V_{max} is the maximum intensity variation in the given image and $V_c(I_{ij})$ is the function of intensity variation around the pixel (i, j) which is given by:

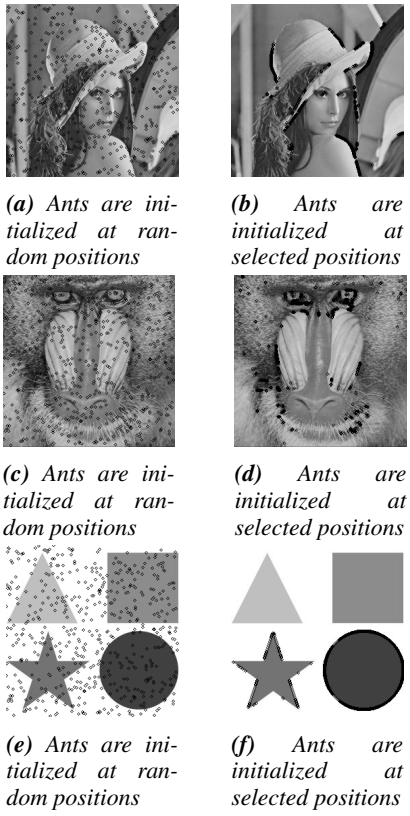
$$\begin{aligned} V_c(I_{(i,j)}) = |I_{(i-1,j-1)} - I_{(i+1,j+1)}| + |I_{(i,j-1)} - I_{(i,j+1)}| \\ + |I_{(i+1,j-1)} - I_{(i-1,j+1)}| + |I_{(i+1,j)} - I_{(i-1,j)}| \end{aligned} \tag{6}$$

Consequently, the pixel with a higher intensity variation (its heuristic information value is higher) has a higher probability of being selected in the next step of the ant tour.
In ACS, the global pheromone update is applied only by the best-so-far ant. However for the edge detection problem, we modify this rule to allow all ants to update the pheromone trails at the end of each iteration. Since all the solutions found by ants represent potential edges in the image. Therefore, Equation Eq.(3) becomes:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{k} \tag{7}$$

If ant k has visited pixel (i, j), then $\Delta\tau_{ij}^{k}$ is computed as the average of the heuristic values associated with pixel (i, j). Otherwise, it is set to zero. Local pheromone update is applied by the ants after each move as defined in Eq.(4).
The ant colony algorithms presented in the literature for solving edge detection problem, such as in Baterina and Oppus (2010), require mostly a post-processing step of thresholding and filtering for removing the noise. Such techniques remove often edges apart from the noise and increases significantly the complexity of the detection process. In this paper, we propose an efficient algorithm (FACS) to overcome loosing the details of edges on resultant images and to reduce the complexity of the process. The core idea of FACS is to perform a focused distribution of the initial positions of ants on the image graph. The way the ants are places initially on the image graph affects significantly both the quality of the results and the complexity of the detection process. Most of the used ant colony algorithms apply a random distribution of the initial positions of ants. One of the main disadvantages of the random distribution is the noise emerges in the resultant images. Since the heuristic matrix of the image could be computed offline before starting the algorithm, FACS proposes to guide all ants to start at selected positions, at which the values of the heuristic information are at their maximum. FACS computes first the heuristic information matrix associated with the image graph. Afterwards, this heuristic information is ranked and the initial positions are

selected based on the obtained ranking. This leads to concentrate the search by making it starts at positions related to the best solutions. Consequently, this will reduce the complexity of the search algorithm and increase the quality of the results. Figure 2 shows two standard test images (lena and mandril), in addition to one image of simple shapes that are drawn with an increasing intensity in order to clarify the applicability of the proposed algorithms. The Figure shows the initial positions of ants. In Figure 2(a), (c), and (e) the positions are generated randomly using a uniform distribution as done by most of the ant colony algorithms. Whereas, in Figure 2(b), (d), and (f) the positions are generated based on the ranked heuristic information, as described above.



(a) *Ants are initialized at random positions*

(b) *Ants are initialized at selected positions*

(c) *Ants are initialized at random positions*

(d) *Ants are initialized at selected positions*

(e) *Ants are initialized at random positions*

(f) *Ants are initialized at selected positions*

Figure 2 *– Initial positions of ants.*

The pseudocode of the FACS algorithm is shown in Algorithm 1, where initially K artificial ants are placed at particular positions that are selected based on the ranked heuristic information. The algorithm runs for N iterations and in each iteration each ant constructs its specific solution through choosing its steps probabilistically using Eq.(1) and Eq.(2). After that, the ant updates the local pheromone trails using Eq.(4). At the end of each iteration the pheromone trails are updated globally using the solutions found by all ants as in Eq.(7).

Algorithm 1: Pseudocode of FACS for edge detection in images.

```
Initialization at focused positions;   { Compute the heuristic information
                                       { Rank the heuristic information
                                       { Select initial positions
forall the iterations n in 1:N do
    forall the construction steps l in 1:L do
        forall the ants k in 1:K do
            Choose and move to the next pixel;
            Update local pheromone;
    Update global pheromone values on visited pixels;
```

5 Experimental Results with FACS

We perform a set of experiments on the images shown in Figure 2 using two edge detection algorithms: the one presented in Baterina and Oppus (2010) and the FACS algorithm. The results obtained from applying the Baterina and Oppus (2010) algorithm are reported without performing any thresholding or filtering in order to compare them with the results obtained using FACS, which does not undergo any filtering process.

We have performed a parameter sensitivity analysis for FACS, in which we have varied the parameters over the following ranges: $q_0 \in \{0.1, 0.3, 0.5, 1\}$, $\alpha, \beta \in \{0.1, 0.3, 0.5, 0.7, 1, 2, 5\}$, $\tau_0 \in \{0.5, 1, 10, 100\}$, and $K \in \{128, 256, 512, 1024\}$. We have noticed that while changing the values of β or q_0 doesn't have a significant influence on the results, high values of α or τ_0 allow to preserve the edges found in pervious iterations. On the other hand, the time it takes to find good result is a tradeoff between the number of ants and the number of iterations. ρ and ξ were tested pairwise $(\rho, \xi) \in \{(0.01, 0.01), (0.1, 0.05), (1, 0.01), (1, 0.1), (1, 1)\}$ and experiments showed that better results were obtained for a high ρ and a relatively low ξ. For the implementation, we adopt the best parameter settings found in Baterina and Oppus (2010) and those are given in the following:

Initial pheromone value (τ_0) =	0.1
Number of ants (K) =	512
Number of iterations (N) =	10
Number of constructions (L) =	40
Parameter influencing pheromone trail (α) =	1
Parameter influencing heuristic information (β) =	1
Pheromone decay coefficient (ξ) =	0.05
Pheromone evaporation coefficient (ρ) =	0.1
Degree of exploration (q_0) =	0.7

We have $K = 512$ ants initialized on K positions that are selected in a decreasing order of the heuristic values. Those

ants are initialized randomly in the algorithm presented in Baterina and Oppus (2010). Figures 3 and 4 show the binary images resulted over different iterations of both ACS and FACS algorithms, where neither thresholding nor filtering techniques were applied. The results show that all the edges found by ants in FACS are potential edges and there was a significant improvement in reducing the detection complexity and in the quality of the results (no noise) compared to the ACS algorithm presented in Baterina and Oppus (2010) (without filtering). After a particular number of iterations, the results from FACS may stop to improve (stagnation) as we can see in Figure 3 for the iterations 8 and 10.

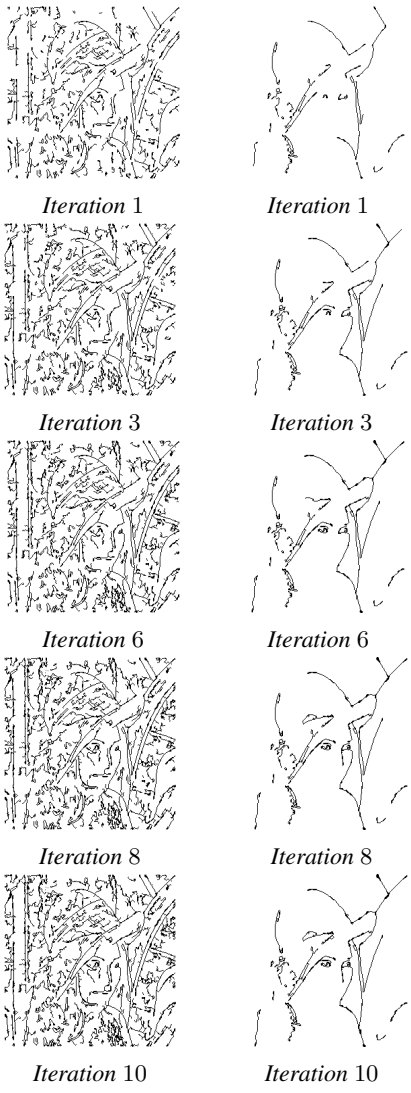


Figure 3 *– The binary images of the image "Lena" obtained over different iterations for both Baterina and Oppus (2010) algorithm and FACS.*

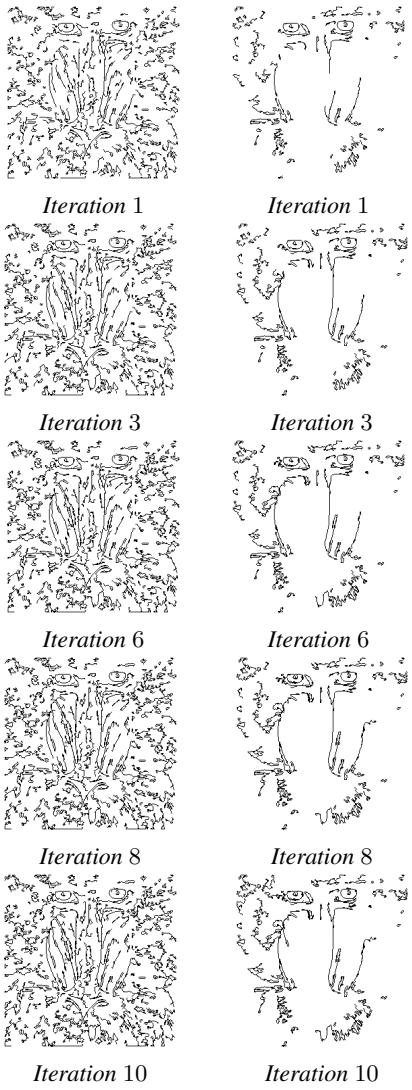


Figure 4 *– The binary images of the image "Mandril" obtained over different iterations for both Baterina and Oppus (2010) algorithm and FACS.*

6 The Extended FACS

Initializing ants at the best heuristic positions has shown a significant improvement over methods that initialize ants at random positions. However, after certain number of iterations the results may stagnate. The reason for this potential stagnation is associated with the reduced probability for the ants to detect edges in the next iterations, while the initial positions selected for those ants did not lead to detect edges in pervious iterations. In cases where the initial positions selected for the ants are adequate for detecting the edges in the image, FACS performs at its optimal. However when it is not the case, potential edges may not be found by the solutions generated over the different iterations.

In order to solve this stagnation problem, we introduce the Extended FACS algorithm, in which the positions of the ants

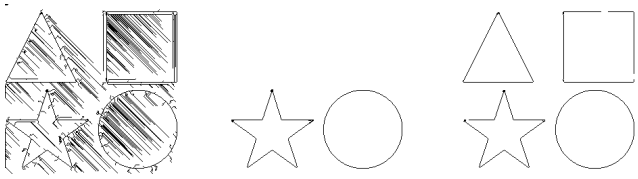
might be re-initialized after each iteration. The algorithm works as in the following: After the global pheromone is deposited at the end of the i-th iteration, we compute the average amount of pheromone deposited by all ants and which is denoted by $\tau_{avg}(i)$. The ants are ranked based on the quality of their solutions in the i-th iteration using the average $\tau_{avg}(i)$. Ants which have deposited a larger amount of pheromone than $\tau_{avg}(i)$ will start from the positions (pixels) they have finished at in the latest iteration. Whereas, ants which have deposited less pheromone than $\tau_{avg}(i)$ are place on new initial positions in the next iteration $i+1$. The new positions assigned to those ants are selected based on the heuristic information computed offline. Thus, They will be placed at the pixels with the next highest heuristic values. The memories of the ants which are placed at new positions are reset to allow them to re-visit the pixels they may have visited in the previous iterations. This extension allows the ants which have performed above the average to continue the search they have started and to detect connected edges. Whereas, the ants which have performed below the average, they have most likely detect no connected edges. Therefore, they should be placed at new positions for increasing their probability of finding new potential edges. The pseudocode of the Extended FACS algorithm is shown in Algorithm 2.

Algorithm 2: Pseudocode of the Extended FACS for edge detection in images.

```
1 Initialization at focused positions;
    { Compute the heuristic information
      Rank the heuristic information
      Select initial positions
2 forall the iterations n in 1:N do
3     forall the construction steps l in 1:L do
4         forall the ants k in 1:K do
5             Choose and move to the next pixel;
6             Update local pheromone;
7     Update global pheromone values on visited pixels;
8     Assign low ranked ants new initial positions;
        { Compute the average pheromone
          Rank the ants
          Re-assign initial positions
```

7 Experimental Results with the Extended FACS

In this section, we apply the Extended FACS to validate the algorithm performance in terms of the solution quality and the ability to deal with the stagnation problem. We adopt the same parameter settings as in Section 5. First, Figure 5 shows the results of applying all of ACS, FACS and the extended FACS on the shapes image. It depicts the results obtained after several iterations of each of the algorithms. In Figure 5(a), we can notice the noise generated when using ASC without filtering. On the contrary, we obtain free-of-noise binary images, when using any of FACS or the extended FACS algorithms, as we can see in Figures 5(b) and (c). Because of initializing the ants at the same positions with the highest heuristics (highest intensity), the ants used by FACS were able to detect edges only on the circle and the star shapes. This is not the case when applying the extended FACS, in which ants are re-intialized at new positions when their previous positions were not promising, see Figure 5(c). Additionally, having a larger number of ants will lead to decrease the number of iterations required by both FACS and the extended FACS to obtained the optimal results and to allow FACS to detect more edges (on the square and the triangle).



(a) *Iteration* 10 ***(b)*** *Iteration* 10 ***(c)*** *Iteration* 23

Figure 5 *– The binary images of the shapes image obtained over different iterations of Baterina and Oppus (2010), FACS, and the extended FACS.*

The results for the lena and the mandril images that are obtained by both FACS and the extended FACS are shown in Figure 6 and Figure 7. Since the results generated by both FACS and the Extended FACS are noise-free, we can combine the binary images obtained over several iterations in order to achieve a better edge detection.

8 Comparisons and Analysis

In this section, we perform a comparison between the three algorithms: the ACS system presented in Baterina and Oppus (2010) (without filtering), the FACS, and the Extended FACS. We perform our comparison from two points of view: the amount of pheromone deposited by ants over the different iterations (a measure of quality) and the time required by the algorithm over the different iterations (a measure of complexity). FACS deposits the highest amount of pheromone as we can see in Figure 8(a). This is due to the fact that FACS initiates the ants at the positions of highest heuristic values, thus, a large number of intersections exists between the solutions found by the different ants. Therefore, the selected pixels in the previous iterations have a higher probability to be re-selected and assigned additional amounts of pheromone. The amount of pheromone assigned by the Extended FACS is, in general, less than the amount assigned by FACS since ants could be placed at different initial positions. In cases when FACS is able to generate different

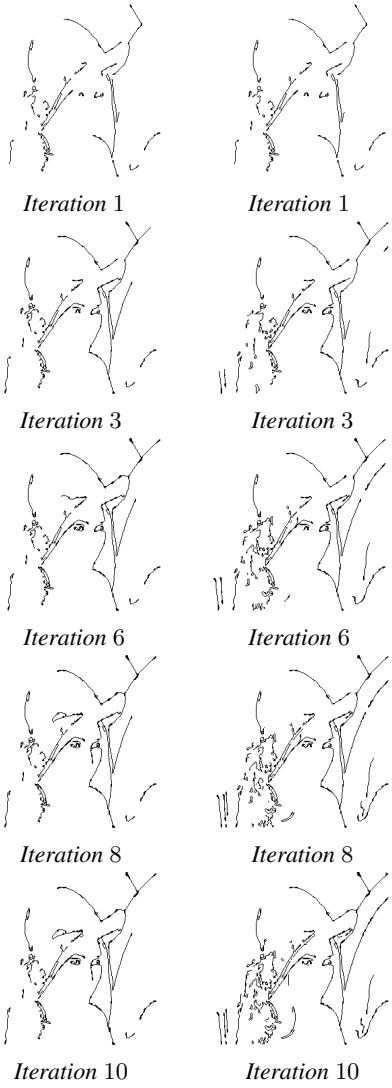


Figure 6 *– The binary images of the image "Lena" obtained over different iterations for both FACS and the Extended FACS.*

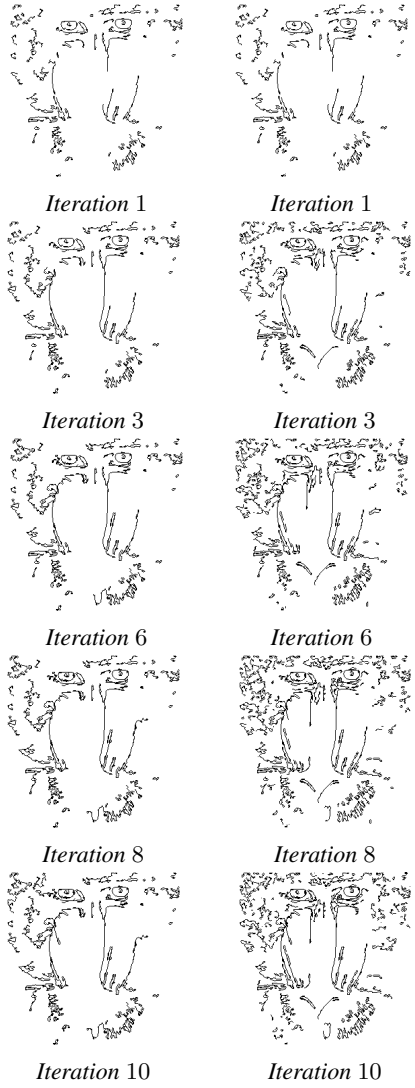
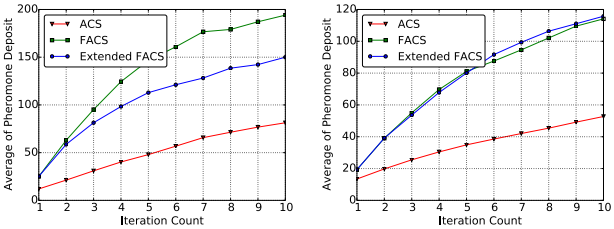


Figure 7 *– The binary images of the image "Mandril" obtained over different iterations for both FACS and the Extended FACS.*

solutions over the considered number of iterations, the average of the deposited pheromone becomes similar to the one of the Extended FACS as we can see in Figure 8(b). The average amount of deposited pheromone is at its minimum when the ants are initialized at random positions and that is what we can see in both Figure 8(a) and (b).

The second criteria we use to compare the three algorithms is the time required by the algorithm to perform the edge detection over a specific number of iterations. This represents a measure of the algorithm complexity. From Figures 9(a) and (b), we can notice that FACS and the Extended FACS perform significantly better than the ACS presented in Baterina and Oppus (2010) in terms of the required time. The reason behind is that when using FACS or the Extended FACS, the probability for ants to re-visit positions is higher. Hence, the



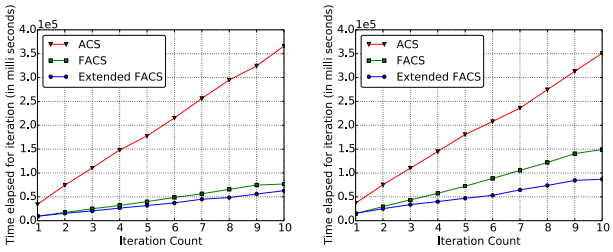
(a) *The "Lena" image.* ***(b)*** *The "Mandril" image.*

Figure 8 *– The average amount of pheromone deposited by ants over* 10 *iterations.*

update of the global pheromone deals with less number of positions (pixels), which leads to a faster update.

In summary, FACS and the Extended FACS offer solutions

with a better quality (higher pheromone concentration) and within a significantly shorter time (lower complexity) than by traditional ACS. Moreover, they don't need any additional thresholding or filtering processes.



(a) *The "Lena" image.* ***(b)*** *The "Mandril" image.*

Figure 9 *– The time required for the detection of image edges over* 10 *iterations.*

9 Conclusion

In this paper, we have presented an efficient Ant Colony System (ACS) for solving the fundamental problem of edge detection which is required in a large number of applications. Two variants of the algorithm were presented: the FACS, in which the ants are initialized at positions selected based on the ranked heuristic information, and the Extended FACS in which the ants are ranked to avoid stagnation problems and generate eventually better solutions by placing the low-ranked ants at new initial positions. The proposed algorithms were compared with the ACS system presented in Baterina and Oppus (2010) without filtering.
The two algorithms proposed in this paper have outperformed the traditional ACS (without filtering) and higher quality solutions were obtained in significantly shorter time, without the need for addition noise-filtering processes.

References

Baterina, A. and Oppus, C. (2010). Ant colony optimization for image edge detection. In *Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation*, ISPRA'10, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

Blum, C. (2005). Beam-aco–hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, 32(6):1565–1591.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698.

Christian, B. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373.

Colorni, A., Dorigo, M., and Maniezzo, V. (1991). Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142, Paris, France.

Davis, L. (1975). A survey of edge detection techniques. *Computer graphics and image processing*, 4(3):248–270.

Dorigo, M. and Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Bradford Company, MA, USA.

Gelfand, I., Fomin, S., and Silverman, R. (2000). *Calculus of Variations*. Dover Books on Mathematics. Dover Publications.

Haralick, R. (1984). Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):58–68.

Jian, Z., Jiliu, Z., Kun, H., and Huanzhou, L. (2012). Image edge detection using quantum ant colony optimization. *International Journal of Digital Content Technology and its Applications(JDCTA)*, 6(11):402–405.

Manish, T., Murugan, D., and Ganesh, K. (2013). Hybrid edge detection using canny and ant colony optimization. *Communications in Information Science and Management Engineering*, 3(8):402–405.

Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217.

Martens, D., Backer, M. D., Haesen, R., Vanthienen, J., Snoeck, M., and Baesens, B. (2007). Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, 11(5):651–665.

Nezamabadi-pour, H., Saryazdi, S., and Rashedi, E. (2006). Edge detection using ant algorithms. *Soft Computing*, 10(7):623–628.

Prewitt, J. (1970). Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19.

Rezaee, A. (2008). Extracting edge of images with ant colony. *Journal of Electrical Engineering*, 59(1):57–59.

Secomandi, N. (2000). Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers & Operations Research*, 27(11):1201–1225.

Stützle, T. and Hoos, H. (2000). Max–min ant system. *Future generation computer systems*, 16(8):889–914.

Tian, J., Yu, W., and Xie, S. (2008). An ant colony optimization algorithm for image edge detection. In *IEEE Congress on Evolutionary Computation*, pages 751–756.

Toth, P. and Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, 123(1):487–512.

Zhuang, X. (2004). Edge feature extraction in digital images with the ant colony system. In *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 133–136. IEEE.

Ziou, D. and Tabbone, S. (1998). Edge detection techniques-an overview. *Pattern Recognition and Image Analysis*, 8:537–559.

Yara Khaluf, Syam Gullipalli (2015) An Efficient Ant Colony System for Edge Detection in Image Processing. Proceedings of the European Conference on Artificial Life 2015, pp. 398-405