Algoritmo ACO para Detección de Bordes

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

María E. Buemi
Departamento de Computación, Universidad de Buenos Aires

ETSI de Sistemas Informáticos
Universidad Politécnica de Madrid
16 de Febrero de 2016

Outline

1. Introducción
2. Detección de Bordes
   - Introducción
   - Métodos
3. Ant Colony Optimization (ACO)
   - Introducción
   - Ant Colony System
4. Algoritmo ACO
   - Introducción
   - Fase de Inicialización
   - Fase de Construcción
   - Fase de Mejora
   - Imagen Binaria
   - Pseudo-código
5. Pruebas Computacionales
   - Parámetro del algoritmo
   - Resultados
6. Conclusiones

Motivación

Presentar un nuevo algoritmo basado en Ant Colony Optimization (ACO) para detectar bordes en imágenes digitales.

Las técnicas de detección de bordes se utilizan en diferentes situaciones.

Mayormente, se usan para encontrar regiones que permitan identificar objetos en imágenes.

La detección de bordes incluye métodos matemáticos para localizar puntos o regiones en imágenes, en donde los cambios en los valores de intensidad, *son detectados*.

En otras palabras, un borde es el límite entre un objeto y el fondo (background) de la imagen.

Una metaheurística se define como un conjunto de conceptos algorítmicos que pueden ser aplicados para obtener soluciones a problemas de optimización.

La aplicación de metaheurísticas a diferentes problemas en las áreas de planificación, telecomunicaciones, transporte, grafos, biología, optimización continua, entre otras, ha sido exitosa (Dorigo & Stutzle [5], Glover & Laguna [7], Mills et al. [14]).

La detección de bordes es esencial en Análisis de Imágenes, especialmente en segmentación de objetos.

Detectar un borde en una imagen digital, consiste en analizar la relación entre un píxel y sus vecinos.

La dificultad en localizar bordes surge con la presencia de ruido, sombras, baja resolución, y excesiva iluminación local, entre otras.

Los bordes de imágenes son *cambios abruptos* en los valores de intensidad.

El gradiente provee información sobre intensidad y orientación de aquellos cambios, los cuales *pueden usarse* para la detección de bordes. La norma del gradiente está dada por:

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

donde G_x and G_y son las primeras derivadas de la imagen en las direcciones x e y, respectivamente. La orientación del gradiente está dada por

$$\vec{\theta_x} = arctg \begin{pmatrix} G_x \\ \\ G_y \end{pmatrix}$$

En la literatura, son usuales los detectores de bordes basados en la primera y segunda derivada.
Sobel, Prewitt y Robert son operadores basados en la primera derivada, mientras que el operador Laplaciano es basado en la segunda (González & Woods [8]).

El método desarrollado por Canny, es considerado un estándar en la industria (Nadernejad & Sharifzadeh [15]). Consiste en una secuencia de cuatro pasos: suavizado del ruido mediante un filtro gaussiano, filtro basado en gradiente para localizar cambios de intensidad, supresión de píxels que no son máximos locales y eliminación de bordes falsos mediante un umbral (threshold) con hysteresis.

En general, las técnicas clásicas de detección de bordes imponen un alto costo computacional, debido a las operaciones relacionadas sobre cada píxel.[1]

Recientemente, métodos basados en metaheurísticas han sido propuestos tales como Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization y Artificial Bee Colony (Baterina & Oppus [2], Djename & Batouche [4], Jevtić & Andina [10], Li et al. [11], Manish et al. [13], Tian et al. [17], Yigitbasi & Baykan [18]).
Las características más relevantes que se han encontrado en estas propuestas son entre otras: bajo esfuerzo computacional, resultados competitivos y paralelización de tareas.

[1] Para obtener resultados de calidad.

Las hormigas, abejas y avispas son insectos que han sido estudiados por investigadores de diferentes áreas del conocimiento.
Un comportamiento estudiado es *la búsqueda de alimento*.

Mientras caminan desde la fuente de alimento hacia el hormiguero (y viceversa), las hormigas reales *depositan* una sustancia química en el suelo llamada *feromona*. Sobre los caminos más cortos (en términos de distancia) existen altos niveles de feromona, debido a que éstos se transitan más rápido.
Esta situación hace que la colonia recorra los caminos más cortos.

Propuesta por Dorigo, Ant Colony Optimization (ACO) es una metaheurística inspirada en el comportamiento real de las hormigas respecto a la búsqueda de alimento.
Mediante una colonia de hormigas artificiales las cuales cooperan y se comunican indirectamente (por *stigmergy*[2]), los algoritmos ACO alcanzan buenas soluciones a diferentes problemas de optimización.

Ha sido exitosamente aplicada a problemas como ruteo de vehículos, planificación, procesamiento de imágenes, grafos, bioinformática.

[2]Coordinación indirecta entre agentes o acciones, donde la huella dejada en el medio ambiente por una acción, estimula el rendimiento de una acción posterior.

En los algoritmos ACO, una hormiga de manera incremental *construye* una solución agregando componentes a una solución parcial.

Para lograr esto, la hormiga se mueve sobre vértices de un grafo conexo y ponderado, *aplicando* una regla de transición. Esta regla indica a la hormiga a *cuál vértice debe moverse*, hasta que el camino (solución del problema de estudio) esté construído. Luego, la hormiga *puede actualizar* rastros de feromona sobre su camino (conexiones usadas del grafo asociado).

En general, un algoritmo ACO incluye los procedimientos:

- ConstructAntsSolutions(): controla que las hormigas construyan incrementalmente sus soluciones.
- UpdatePheromones(): incrementa los niveles de feromona sobre los caminos recientemente construídos por la colonia, y decrementa sobre el grafo.
- DaemonActions(): pueden usarse para implementar acciones centralizadas que no sean realizadas individualmente por las hormigas. Estas acciones están relacionadas con la mejora y el control del proceso de búsqueda de soluciones.

Table: Algoritmo ACO

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

Revisión histórica

- En 1991, Dorigo propone Ant System en su tesis doctoral.
- En 1996, se publica el primer artículo sobre Ant System.
- En el mismo año, Hoos and Stutzle presentan MAX-MIN Ant System.
- En 1997, Dorigo y Gambardella publicaron Ant Colony System.
- En 1998, Stutzle propone implementaciones en paralelo.
- En 2001, Iredi et al. publicaron el primer algoritmo multi-objetivo.

Introducción

Ant System (AS) fue el primer algoritmo ACO.
Luego, Max-Min Ant System (MMAS) y Ant Colony System (ACS) fueron presentados como extensiones.

ACS difiere de AS en la regla de decisión y en la actualización de feromona (Dorigo & Stutzle [5]).

Regla de decisión

La hormiga k ubicada en el vértice i, se mueve hacia el vértice j, de acuerdo a una regla proporcional pseudo-aleatoria:

$$j = \begin{cases} argmax_{l \in \aleph_i^k} \{\tau_{il}[\eta_{il}]^\beta\} & \text{si q} \leq q_0 \\ \frac{\tau_{ij}[\eta_{ij}]^\beta}{\sum_{l \in \aleph_i^k} \tau_{il}[\eta_{il}]^\beta} & \text{c.o.c.} \end{cases} \quad (1)$$

Donde q es una variable aleatoria uniformemente distribuida en [0,1), q_0 es un parámetro de algoritmo ($0 \leq q_0 \leq 1$) y β es otro parámetro que ajusta la importancia relativa de la información heurística (o a priori).

Por lo tanto, con probabilidad q_0 la hormiga k elige al mejor vértice vecino (según criterio greedy) y con probabilidad $(1 - q_0)$, aplica la regla decisión conocida (Ant System).

Actualización local de feromona

Luego de moverse hacia el vértice j, la hormiga k actualiza el nivel de feromona relacionado con la última conexión recorrida:

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

Donde ξ $(0 < \xi < 1)$ y τ_0 son parámetros del algoritmo.
La actualización local de feromona *intenta diversificar* el proceso de búsqueda de soluciones.

Actualización global de feromona

Al final de cada iteración, la mejor hormiga *bs* es la única que actualiza feromona sobre su recorrido T^{bs}. La actualización global de feromona es mediante la siguiente expresión:

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

Donde $\Delta\tau_{ij}^{bs} = \frac{1}{C^{bs}}$ es un valor asociado a la longitud de lo recorrido por la mejor hormiga, y ρ es un parámetro relacionado por la tasa de evaporación $(0 < \rho \leq 1)$.

Es importante destacar que la expresión anterior realiza *evaporación* y *depósito* de feromona al mismo tiempo.

Para más detalles sobre ACO, se recomienda el libro de Dorigo & Stutzle [5].

Nuestra propuesta

Usando información heurística y de aprendizaje, nuestra colonia de hormigas atraviesa una imagen en escala de grises, en busca de bordes.

Mediante una matriz de intensidad I
$(0 \leq I(i,j) \leq 255,\ i = 1,\ldots,m; j = 1,\ldots,n)$, las hormigas se mueven sobre los píxels de la imagen (que representan vértices del grafo) de manera de detectar bordes y finalmente, obtener una imagen binaria como salida.

Al comienzo, las matrices de feromona y heurística (ambas m×n) deben ser inicializadas.
La primera, es inicializada con valor real positivo τ_0 ($\tau_{ij} = \tau_0 \ \forall i, j$).
La segunda, de acuerdo a información local de la imagen:

$$\eta_{ij} = \frac{\sigma_{ij}}{\mu_{ij} + 1} \tag{4}$$

Donde μ_{ij} y σ_{ij} son la media y la varianza (de intensidad) para cada píxel. Estos valores se calculan considerando el vecindario de Moore (8-neighborhood) alrededor de cada píxel en posición (i,j).

Además, las matrices gradiente horizontal y vertical G_x y G_y son requeridas. Se obtienen aplicando el operador de Sobel sobre la imagen original. Ambas, serán usadas cuando las hormigas apliquen las regla de transición (Ec. 5) para construir sus tours.

Luego, en cada iteración, las hormigas artificiales empiezan sus respectivos tours sobre píxels con alta varianza. Esto permite analizar regiones con posibles bordes. Cuando todos hayan sido visitados, comenzarán en diferentes posiciones de la imagen *para diversificar* el proceso de búsqueda.

En cada iteración, las hormigas se mueven sobre píxels (no visitados aún en el actual recorrido) en busca de bordes. En cada movimiento, cada hormiga aplica la siguiente regla basada en Ant Colony System:

$$(i,j) = \begin{cases} argmax_{(l,m)\in\aleph_{(i_0,j_0)}}\{\tau_{lm}[\eta_{lm}]^{\beta}\} & \text{si } q \leq q_0 \\ \dfrac{\tau_{ij}[\eta_{ij}]^{\beta}}{\sum_{(l,m)\in\aleph_{(i_0,j_0)}}\tau_{lm}[\eta_{lm}]^{\beta}} & \text{c.o.c.} \end{cases} \quad (5)$$

Donde $\aleph_{(i_0,j_0)}$ es su vecindario restringido. Es decir, se descartan píxels que no tienen chances de ser bordes.

Cada hormiga realiza L movimientos (longitud del tour).

Actualización local

A diferencia de otras propuestas (Baterina & Oppus [2], Tian et al. [17] que intentan que otras hormigas consideran al píxel recientemente visitado como futuro borde, se propone usar la propuesta por ACS (Ec. 2).

En nuestro caso, se intenta que las hormigas exploren toda la imagen en busca de bordes.

Actualización global

Luego que todas las hormigas han construído sus caminos, y previo a una nueva iteración del algoritmo, se realiza la actualización global de feromona (Ec. 6):

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

Esta regla, similar a ACS (Ec. 3) y diferente a la propuesta por Tian et al. [17], lo que intenta es actualizar niveles de feromona sobre toda la imagen, *pero con información relevante*.

A diferencia de otros problemas (como el TSP) donde las hormigas construyen (cada una) soluciones completas, en nuestro caso las hormigas cooperan para obtener una única solución.

Por lo tanto, cuando se cumple la condición de parada, la matriz final de feromona es obtenida. Esta será usada para generar la imagen binaria de salida.

Vista como solución, la matriz de feromona *puede ser mejorada*.

La fase de mejora (o fase de pre-binarización de la imagen) consiste en la detección y corrección de píxels detectados incorrectamente.

Los píxels detectados incorrectamente como bordes o no-bordes son analizados y corregidos, de ser necesario.

Basados en la propuesta de *eliminación de bordes falsos* de Ahmad & Choi [1], las matrices de feromona y de varianza son analizadas.

Sin embargo, existen dos aspectos distintivos en nuestra propuesta: mejora parametrizada y análisis local.
Esto resulta útil para imágenes de baja resolución, ruidosas y con efectos de luces y sombras.

Si el valor de feromona *del píxel (i,j)* es mayor que AT y la varianza es menor que GT (párametro del algoritmo), *entonces* el valor de feromona asociado es actualizado a 0. Significa que fue corregido a *no-borde*.

Por otra parte, si el valor de feromona *del píxel (i,j)* es menor que AT y la varianza es mayor que $step * GT$[3], *entonces* el valor de feromona asociado es actualizado a 255. Significa que fue corregido a *borde*.

[3] Esta expresión asegura el porcentaje de corrección.

Obtención de Matriz E

Como se mencionó anteriormente, usando la matriz final de feromona, la imagen binaria de salida es generada, conteniendo los bordes detectados.

Basado en Tian et al. [17], todos los valores de feromona τ_{ij} deben compararse con AT, para obtener la matriz binaria E que permite generar la imagen de salida.

Para cada píxel (i,j), si el valor de feromona asociado es mayor o igual que AT, *entonces E(i,j)=1* (borde detectado). *Sino E(i,j)=0* (no-borde detectado).

Obtención de AT

Luego de la primera iteración del algoritmo, se obtiene un AT inicial (umbral de algoritmo):

$$AT = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \tau_{ij}}{m \times n} \tag{7}$$

En próximas iteraciones, AT es el promedio de dos promedios: promedio de valores de feromona menores $\bar{\tau}_L$ y de valores mayores $\bar{\tau}_L$ que AT previo:

$$AT = \frac{\bar{\tau}_L + \bar{\tau}_U}{2} \tag{8}$$

La condición de parada de nuestro algoritmo está basado en los valores AT. Cuando dos valores consecutivos son similares, entonces se cumple la condición. *Esto indica que el algoritmo no continuará buscando bordes*.

Luego que los valores de parámetros de algoritmo y la matriz de intensidad asociada a la imagen son obtenidos, se ejecuta la fase de inicialización. Esta consiste en obtener gradiente, matrices de varianza y heurística e inicializar matrices de visita y feromona.

Luego, en cada iteración, las hormigas construyen sus recorridos en busca de bordes, aplicando una regla de decisión y actualizando niveles de feromona.
Antes de comenzar una nueva iteración, se realizan la actualización global de feromona y cálculo de AT.

Cuando se cumple la condición de parada (AT similares o tiempo de CPU), la matriz de feromona es corregida.
Usando ésta, se obtiene la matriz binaria con bordes detectados.

Require: $imageM, K, L, \tau_0, \rho, \beta, \xi, GT, improvPct, \epsilon, maxT$
$G_x, G_y \leftarrow applyFilter(imageM)$
$initializePheromoneM(pheromoneM, \tau_0)$
$heuristicM, varianceM \leftarrow generateM(imageM)$
$initializeVisitM(visitM, varianceM)$
$initializeAditionalInformation()$
$flag \leftarrow False$
while $notflag$ **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
      $i, j \leftarrow applyDRule(pheromoneM, i, j, heuristicM, G_x, G_y, \beta)$
      $updateLocalPheromone(pheromoneM, i, j, \xi, \tau_0)$
    end for
  end for
  $updateGlobalPheromone(pheromoneM, \rho, heuristicM)$
  $oldAT \leftarrow AT$
  $AT \leftarrow calculateThreshold(oldAT, pheromoneM)$
  $flag \leftarrow stoppingConditionIsMet(AT, oldAT, \epsilon, maxT)$
end while
$improveSolution(pheromoneM, varianceM, GT, improvPct)$
$edgeM \leftarrow generateEdgeM(pheromoneM, AT)$
return $edgeM$

Los parámetros y sus valores son:

- K $= \lfloor \sqrt{m \times n} \rfloor$: número de hormigas.
- L $= 40$: número de movimientos por hormiga por iteración.
- $\tau_0 = 0.0001$: valor inicial para los elementos de la matriz de feromona.
- $\beta = 0.1$: factor de influencia de información heurística usada en regla de transición (Ec. 5).
- $\xi = 0.05$: decrecimiento de feromona aplicado en actualización local (Ec. 2).
- $\rho = 0.1$: tasa de evaporación usada en actualización global (Ec. 6).
- GT $= 70$: umbral global usado en fase de mejora.
- $\epsilon = 0.01$: valor de tolerancia usado para determinar si se cumple condición de parada.
- maxT $= 120$ segs.: tiempo máximo de CPU, en caso que los valores de AT no sean similares.
- improvPct: porcentaje de mejora de la matriz de feromona.

Todos los valores excepto ϵ, *maxT* y *improvPct* han sido tomados de la literatura (Ahmad & Choi [1], Baterina & Oppus [2], Dorigo & Stutzle [5], Jevtić & Andina [10], Tian et al. [17]) y **se mantuvieron fijos durante todas las pruebas**.

El algoritmo fue implementado en JAVA, y las pruebas se realizaron en una notebook con procesador CORE I5 2.4 Ghz con 4Gb de Ram sobre Windows 7.

Sobre las pruebas

Fueron realizadas sobre imágenes con y sin ruido, tomando imágenes de la literatura y sobre diferentes datasets.

Las primeras pruebas se realizaron sobre imágenes conocidas (Peppers, Cameraman, Baboon y Lena)(Figs. 14-10) y los resultados se compararon con los de Baterina & Oppus [2], Etemad & White [6], Jevtić & Andina [10] y Tian et al. [17]. Li et al. [11] y Yigitbasi & Baykan [18] no fueron considerados, porque sus imágenes no estaban disponibles para las pruebas.

Comparación con propuestas de la literatura

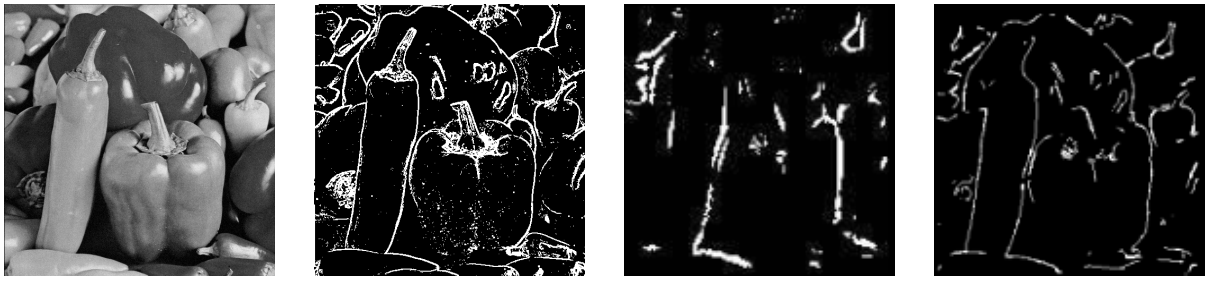

Figure: 1.a) Imagen original 1.b) **ACO** 1.c) Tian et al. 1.d) Baterina & Oppus

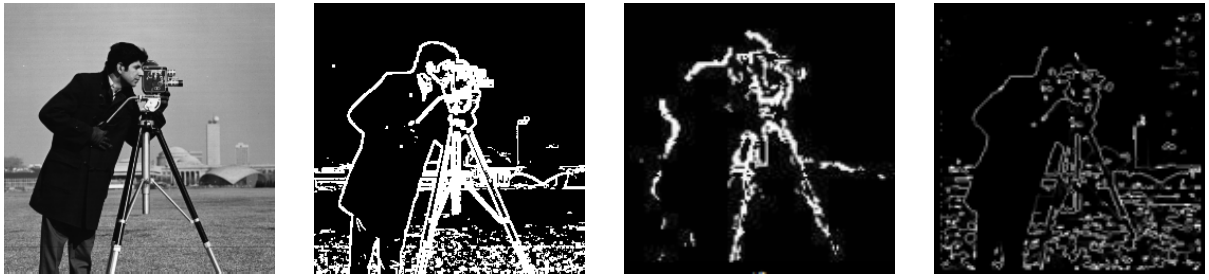

Figure: 2.a) Imagen original 2.b) **ACO** 2.c) Tian et al. 2.d) Jevtić & Andina

Figure: 3.a) Imagen original 3.b) **ACO** 3.c) Baterina and Oppus

Figure: 4.a) Imagen original 4.b) **ACO** c) Tian et al. 4.d) Etemad & White

Comparación con métodos clásicos

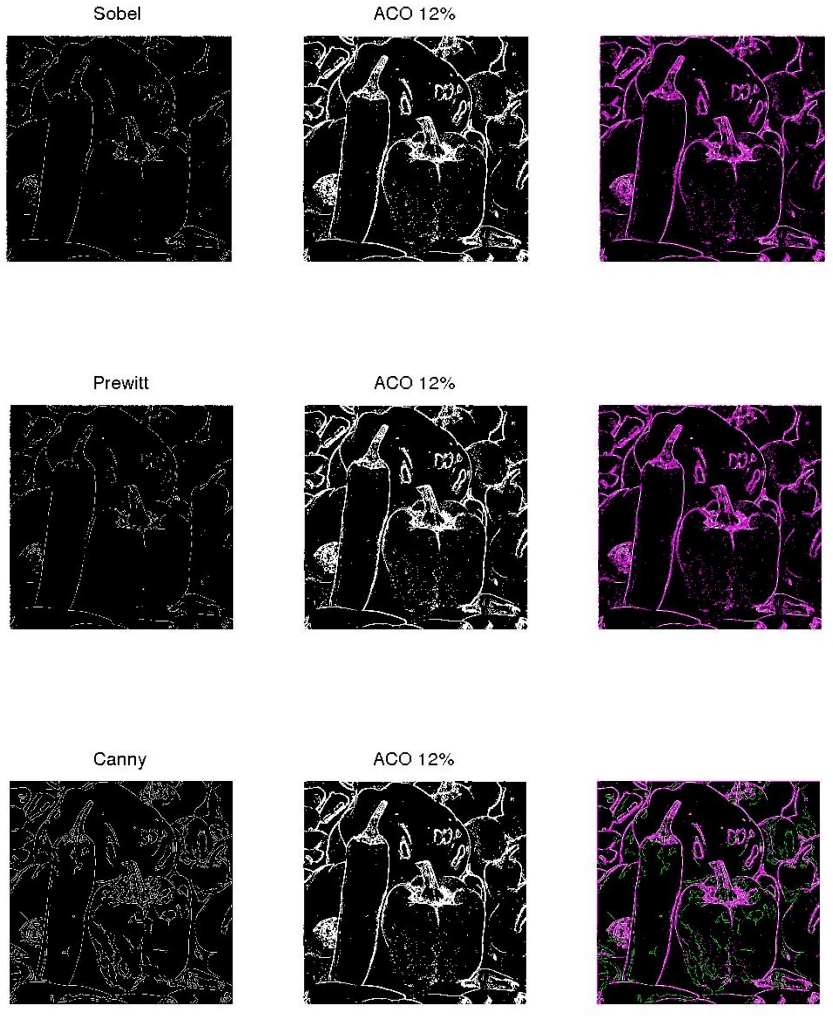

Figure: 5.a) vs Sobel 5.b) vs Prewitt 5.c) vs Canny

Resultados

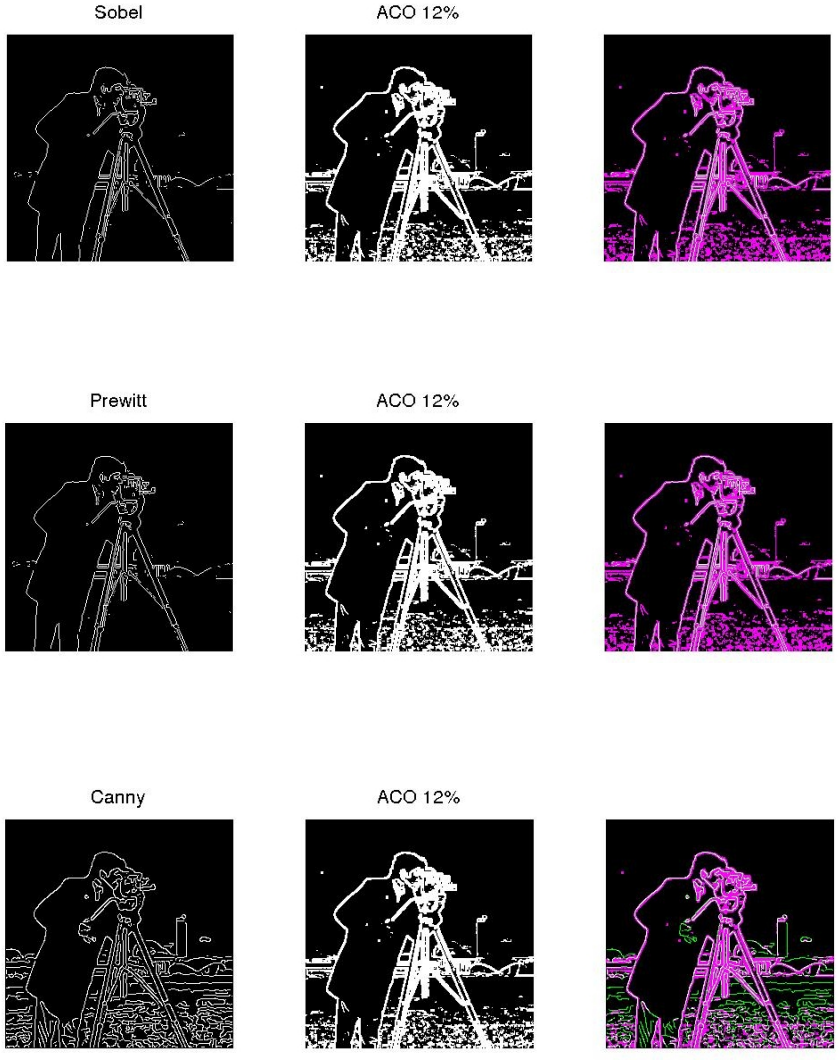

Figure: 6.a) vs Sobel 6.b) vs Prewitt 6.c) vs Canny

Resultados

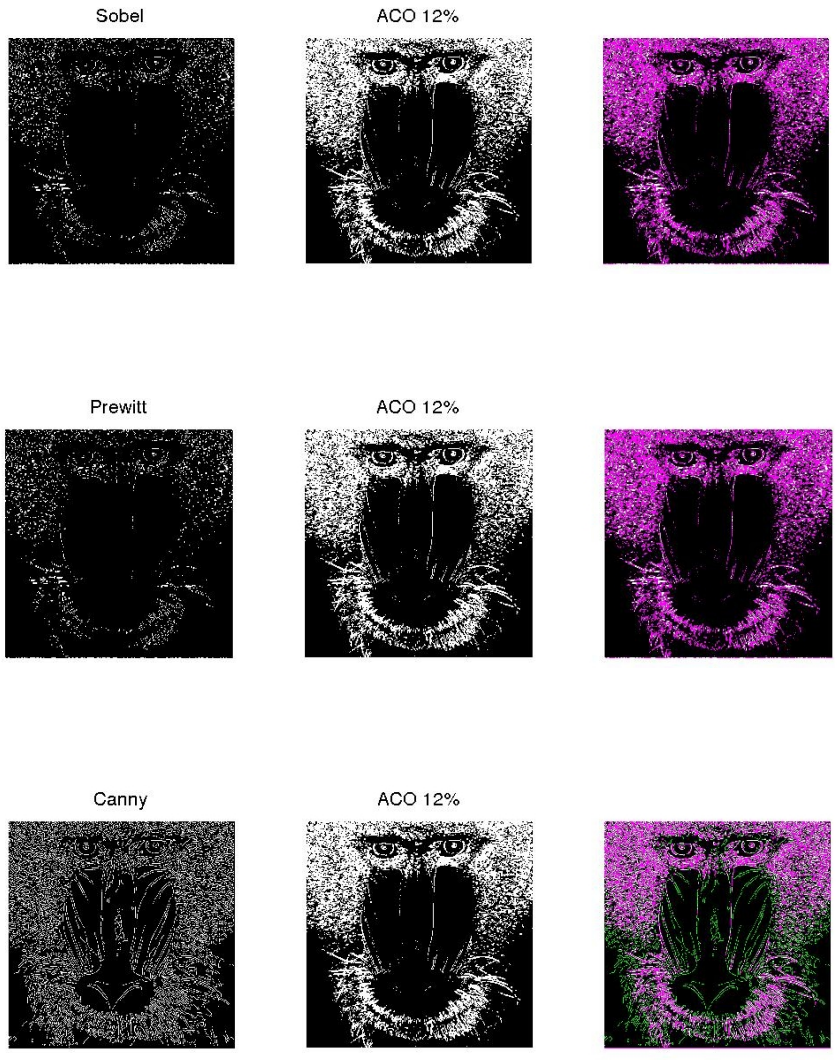

Figure: 7.a) vs Sobel 7.b) vs Prewitt 7.c) vs Canny

Resultados

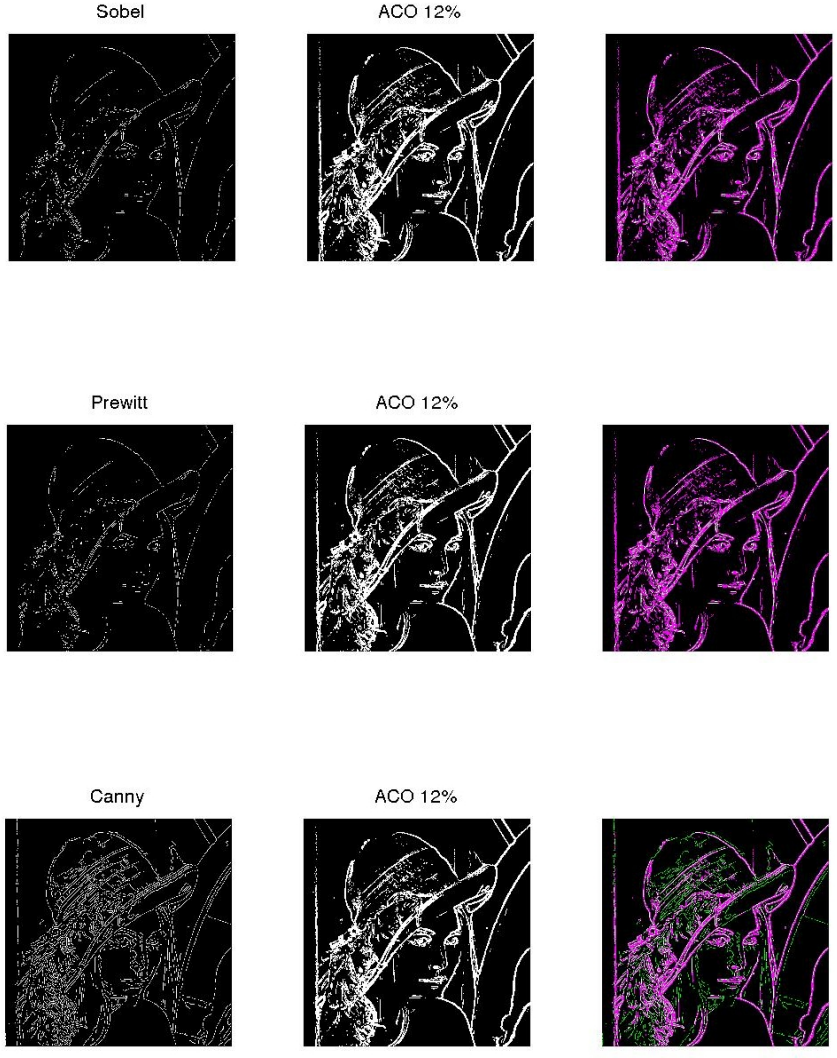

Figure: 8.a) vs Sobel 8.b) vs Prewitt 8.c) vs Canny

Pruebas con imágenes ruidosas

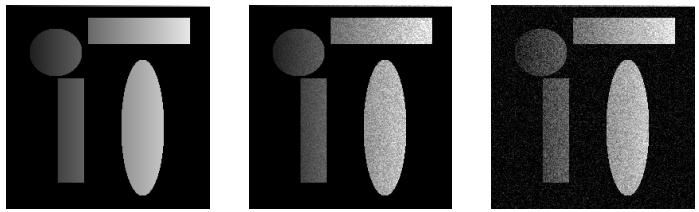

Figure: 9.a) Imagen original 9.b) Imagen con ruido mult. 9.c) Imagen con ruido gaussiano

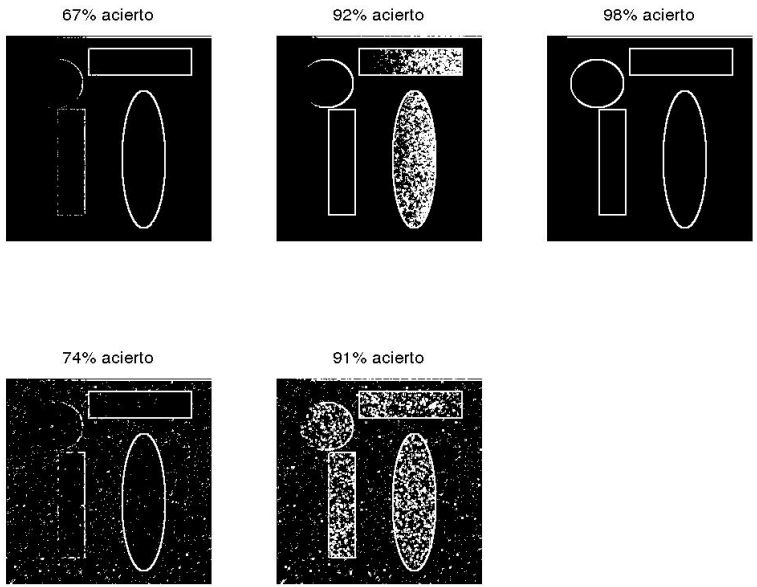

Figure: 9.d) Binaria 1% mejora s/ruido mult. 9.e) Binaria 10% mejora s/ruido mult. 9.f) Binaria 2% mejora s/original 9.g) Binaria 1% mejora s/ruido gauss. 9.h) Binaria 10% mejora s/ruido gauss.

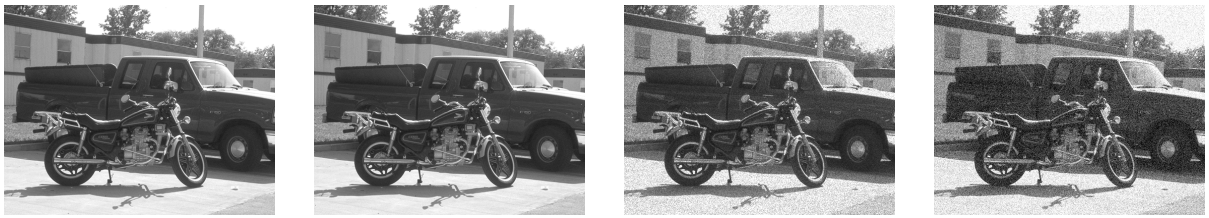

Figure: 10.a) Imagen original 10.b) con ruido Sal y Pimienta 10.c) con ruido multiplicativo 10.d) con ruido gaussiano

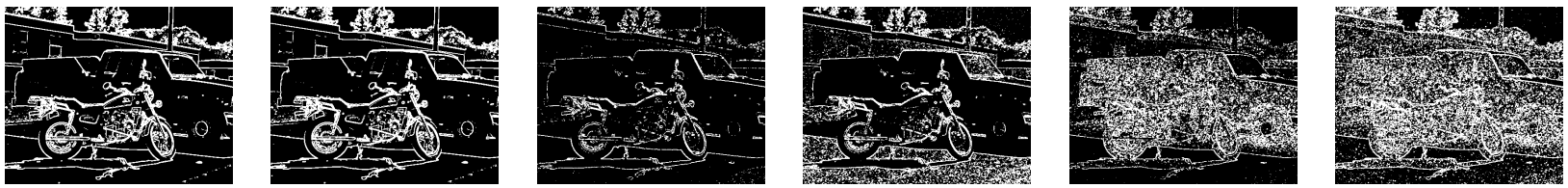

Figure: 10.d) Binaria 10% mejora s/original 10.e) Binaria 10% mejora s/ruido SP. 10.f) Binaria 1% mejora s/ruido mult. 10.g) Binaria 10% mejora s/ruido mult. 10.h) Binaria 1% mejora s/ruido gauss. 10.i) Binaria 10% mejora s/ruido gauss.

Figure: 11.a) Imagen original 11.b) con ruido Sal y Pimienta 11.c) con ruido multiplicativo 11.d) con ruido gaussiano

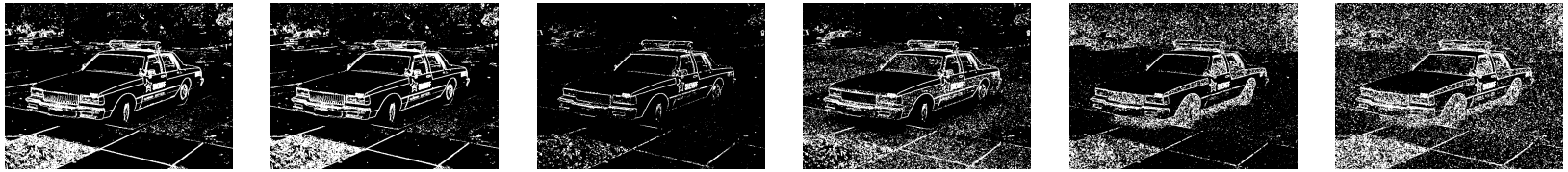

Figure: 11.d) Binaria 10% mejora s/original 11.e) Binaria 10% mejora s/ruido SP. 11.f) Binaria 1% mejora s/ruido mult. 11.g) Binaria 10% mejora s/ruido mult. 11.h) Binaria 1% mejora s/ruido gauss. 11.i) Binaria 10% mejora s/ruido gauss.

Conclusiones

Se propuso un nuevo algoritmo ACO para detección de bordes. Respecto a una versión anterior, la nueva realiza mejores detecciones, incluso sobre imágenes con baja resolución, y también sobre ruidosas.

Respecto a otras propuestas de la literatura, nuestro algoritmo alcanza mejores detecciones.

Sobre imágenes ruidosas (gaussianas, sal y pimienta, multiplicativo), tiene un comportamiento robusto.

El tiempo computacional para todas las pruebas *no supera los 10 segs*.

Conclusiones

Esto fue alcanzado a través de un análisis eficiente de la imagen en busca de bordes, mediante:

- Información heurística relevante.
- Vecindarios restringidos.
- Exploración amplia de la imagen.
- Mejora parametrizada de la solución.

Gracias!


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.


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.


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.


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.


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


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.


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


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


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


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.


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.


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


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.

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.

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

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.

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.

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.