*Ingeniería Investigación y Tecnología.* Vol. XII, Núm. 4, 2011, 469-485
ISSN 1405-7743 FI-UNAM
(artículo arbitrado)

# Detección de primitivas circulares usando un algoritmo inspirado en el electromagnetismo

## *Circle Detection Using an Electromagnetism-Inspired Algorithm*

Cuevas E.
*Departamento de Ciencias Computacionales*
*Universidad de Guadalajara, CUCEI*
*E-mail: erik.cuevas@cucei.udg.mx*

Osuna-Enciso V.
*Departamento de Ciencias Computacionales*
*Universidad de Guadalajara, CUCEI*
*E-mail: valentin.osuna@cucei.udg.mx*

Oliva D.
*Departamento de Ciencias Computacionales*
*Universidad de Guadalajara, CUCEI*
*E-mail: diego.oliva@cucei.udg.mx*

Wario F.
*Departamento de Ciencias Computacionales*
*Universidad de Guadalajara, CUCEI*
*E-mail: fernando.wario@cucei.udg.mx*

Información del artículo: recibido: abril de 2010, aceptado: octubre de 2010

**Resumen**

La computación basada en principios físicos recientemente ha ganado respeto en la comunidad científica. Esta área emergente, en poco tiempo ha logrado desarrollar un amplio rango de técnicas y métodos que han servido para resolver diversos problemas, considerados como complejos. Por otra parte, la detección automática de círculos en imágenes se considera una tarea importante, es por esto que se han realizado un gran número de trabajos tratando de encontrar el detector de círculos óptimo. Este artículo presenta un nuevo algoritmo para la detección de primitivas circulares contenidas en imágenes sin la consideración de la transformada de Hough. El algoritmo propuesto está basado en un nuevo enfoque inspirado en principios físicos llamado: *Electromagnetism-Like Optimization* (EMO), el cual es un método heurístico que emplea algunos principios de la teoría del electromagnetismo para resolver problemas complejos de optimización. En el algoritmo EMO las soluciones se construyen considerando la atracción y repulsión electromagnética entre las partículas cargadas; dicha carga representa la afinidad que tiene cada partícula con la solución. El algoritmo de detección de círculos emplea una codificación de tres puntos no colineales, dichos puntos representan los círculos candidatos sobre una imagen que sólo contiene sus bordes. Empleando una función objetivo, el conjunto de círculos candidatos considerados como partículas cargadas, son operados por medio del algoritmo EMO hasta que logren coincidir con los círculos existentes en la imagen real. Los resultados experimentales en diversas imágenes complejas validaron la eficiencia de la técnica propuesta en cuanto a su exactitud, velocidad y robustez.

**Descriptores**

- detección de círculos
- procesamiento de imágenes
- algoritmo *Electromagnetism-Like Optimization*

***Abstract***

*The Physic-inspired computation is becoming popular and has been acknowledged by the scientific community. This emerging area has developed a wide range of techniques and methods for dealing with complex problems. On the other hand, automatic circle detection in digital images has been considered as an important and complex task for the computer vision community that has devoted a tremendous amount of research seeking for an optimal circle detector. This article presents an algorithm for the automatic detection of circular shapes embedded into complicated and noisy images with no consideration of the conventional Hough transform techniques. The approach is based on a nature-inspired technique called the Electromagnetism-Like Optimization (EMO) which is a heuristic method following electromagnetism principles for solving complex optimization problems. For the EMO algorithm, solutions are built considering the electromagnetic attraction and repulsion among charged particles with a charge representing the fitness solution for each particle. The algorithm uses the encoding of three non-collinear points as candidate circles over an edge-only image. Guided by the values of the objective function, the set of encoded candidate circles (charged particles) are evolved using the EMO algorithm so that they can fit into the actual circles on the edge map of the image. Experimental results from several tests on synthetic and natural images with a varying range of complexity are included to validate the efficiency of the proposed technique regarding accuracy, speed, and robustness.*

***Keywords***

- *circle detection*
- *image processing*
- *Electromagnetism-Like Optimization*

## Introducción

La computación basada en principios físicos es parte de la computación inteligente que se inspira en correlaciones, características y efectos que experimentan los sistemas físicos para el desarrollo de nuevos algoritmos. En años recientes, varios científicos han introducido nuevos algoritmos inspirados en leyes físicas, algunos de ellos son la optimización basada en estructuras físicas (Fen *et al*., 2010; Contet *et al*., 2007; Li *et al*., 2009), la optimización usando el concepto de fuerza central (Richard, 2008) y los algoritmos basados en mecánica quántica (Wang, 2010).

Por otro lado, el problema de detección de primitivas circulares tiene una gran importancia en el análisis de imágenes, en particular para aplicaciones industriales y médicas, tales como la inspección automática de productos y componentes manufacturados, vectorización de planos, detección de células, etcétera (Davies, 1990). De forma general, en procesamiento de imágenes, el problema de la detección de formas circulares suele llevarse a cabo por medio de la Transformada Circular de Hough (Muammar *et al*., 1989). Sin embargo, la exactitud de los parámetros de los círculos detectados es pobre en presencia de ruido (Atherton *et al*., 1993). Además, el tiempo de procesamiento requerido por la Transformada Circular de Hough hace prohibitivo su uso por algunas aplicaciones, en particular para imágenes digitales grandes y áreas densamente pobladas de pixeles borde. Para tratar de superar tales problemas, se han propuesto varios algoritmos basados en la transformada de Hough (TH), tales como la TH probabilística (Fischer *et al*., 1981; Shaked *et al*., 1996), la TH aleatoria (THA) (Xu *et al*., 1990) y la TH difusa (THD) (Han *et al*., 1993). En Lu & Tan (2008) propusieron una aplicación basada en la THA llamada THA iterativa (THAI), que logra mejores resultados en imágenes complejas y ambientes ruidosos. El algoritmo aplica iterativamente la THA a regiones de interés en la imagen, las cuales son determinadas a partir de la última estimación de los parámetros del círculo/elipse detectados. La detección de formas puede también realizarse usando métodos de búsqueda estocástica, tal como los algoritmos genéticos (AG), los cuales recientemente se han aplicado a importantes tareas de detección de formas. Por ejemplo, Roth y Levine propusieron el uso de AG para extraer primitivas geométricas en imágenes (Roth *et al*., 1994). Lutton *et al*. (1994) realizaron mejoras al método anterior, mientras que Yao *et al*. (2004) usaron un AG multipoblación para detectar elipses. En Lu *et al*. (2008) se usaron AG para buscar similitudes cuando el patrón a detectar ha estado sujeto a una transformación de afinidad desconocida. En Ayala-Ramírez *et al*. (2006) se presentó un detector de círculos basado en AG, que es capaz de detectar múltiples círculos en imágenes reales; sin embargo, falla frecuentemente al detectar círculos imperfectos o en condiciones difíciles. Recientemente, Dasgupta *et al*. (2009) propusieron otro excelente trabajo de detec-

ción automática de círculos, usando un algoritmo de alimentación de bacterias como método de optimización. Considerando el caso de detección elipsoidal, Rosin propuso en (2000 y 1997) un algoritmo de ajuste de elipses que usa cinco puntos.

En este artículo se presenta un nuevo algoritmo para la detección círculos en imágenes basado en la técnica inspirada en principios físicos *Electromagnetism-Like Optimization* EMO (Íker *et al*., 2003). El algoritmo EMO es un método estocástico poblacional basado en la teoría del electromagnetismo, cuyas propiedades de convergencia ya han sido probadas en (Íker *et al*., 2004), (Rocha *et al*., 2009). La forma en que las partículas se calculan dentro del algoritmo corresponde al grupo de algoritmos de enjambre de partículas (PSO por sus siglas en inglés) (Ying *et al*., 2010) y al grupo de algoritmos basados en colonias de hormigas (ACO por sus siglas en inglés) (Blum, 2005).

El primer paso del algoritmo EMO es producir un grupo de soluciones aleatorias a partir de un dominio, en el cual se encuentren las soluciones posibles, suponiendo que cada solución es una partícula cargada. La carga de cada partícula se determina por la función de afinidad (función a optimizar), modificando la posición de cada partícula de acuerdo a su carga, dentro de un campo de atracción o repulsión existente dentro de la población de partículas.

Desde el punto de vista de que EMO es un algoritmo que opera con múltiples soluciones a la vez, su accionar puede relacionarse con los algoritmos genéticos, siendo su mecanismo de atracción-repulsión parecido al de mutación y *crossover* utilizado en los algoritmos genéticos (Rocha *et al*., 2009). Al igual que otros algoritmos heurísticos como Temple Simulado (SA por sus siglas en inglés), EMO tiene la capacidad de minimizar globalmente una función objetivo determinada, ya que ambos, mediante mecanismos diferentes (inspirados en principios físicos) permiten evitar mínimos locales. Sin embargo, debido a que el algoritmo EMO a diferencia de SA explora una función objetivo en múltiples puntos a la vez, es posible obtener en diferentes casos mejores soluciones en un menor tiempo (Vasan *et al*., 2009).

El algoritmo EMO calcula la fuerza resultante de desplazamiento de la población por medio de la ley de Coulomb y el principio de superposición. Esta fuerza resultante se deduce por medio de las cargas y la distancia existente entre cada una de ellas. En este sentido, una carga con un valor elevado producirá una mayor atracción o repulsión. La fuerza resultante tiende a ser pequeña cuando la distancia entre las partículas es grande. En la última iteración del algoritmo los movimientos de las partículas serán lentos siguiendo el caso del SA (Naderi *et al*., 2010; Rocha *et al*., 2009). Por otra parte, el algoritmo EMO puede mejorar la solución óptima en cada iteración, por medio de la búsqueda local, aumentando la posibilidad de salir de determinados mínimos locales.

De forma general, el algoritmo EMO se puede considerar como un algoritmo rápido y robusto que representa una alternativa real para resolver problemas de optimización complejos, no-lineales, no-diferenciables y no-convexos (Naderi *et al*., 2010; Yurtkuran *et al*., 2010; Jhen *et al*., 2009). Las principales ventajas del algoritmo EM radican principalmente en las siguientes características: no tiene operaciones de gradiente, se puede emplear directamente en el sistema decimal (a diferencia de AG), necesita pocas partículas para converger y se garantiza su convergencia (Íker *et al*., 2004; Rocha *et al*., 2009).

En este artículo se presenta un nuevo algoritmo para la detección círculos en imágenes basado en EMO. En este enfoque la detección de primitivas circulares se considera como un problema de optimización. Para la detección se usa la codificación de tres puntos no colineales extraídos del mapa de bordes de la imagen. Estos tres puntos constituyen círculos, los cuales se consideran soluciones candidatas al problema de detección. Una vez que se evalúan estos círculos (partículas cargadas) por una función objetivo (que verifica su existencia en la imagen real), los valores obtenidos sirven de guía para la modificación de las partículas por parte del algoritmo EMO. De esta manera, el algoritmo opera hasta que las soluciones candidatas coincidan con los círculos existentes en la imagen real. Este enfoque genera un detector de círculos, el cual puede identificar círculos eficientemente en imágenes reales, inclusive si los objetos circulares se encuentran parcialmente ocluidos, traslapados o en ambientes ruidosos. Los resultados experimentales muestran evidencia del desempeño en la detección de círculos considerando diferentes tipos de condiciones.

En este artículo se ofrece un panorama general del algoritmo EMO. Después se formula el enfoque propuesto, considerando la aplicación del algoritmo EMO para detectar círculos y presenta los resultados experimentales y pruebas de robustez. Finalmente, se discuten las conclusiones y se establece el trabajo futuro.

## Algoritmo: *Electromagnetism – Like Optimization* (EMO)

El algoritmo EMO es un algoritmo de optimización, el cual está inspirado en principios físicos. El algoritmo se

basa en una población que permite la optimización global de funciones multimodales. En comparación con los AG, no emplea operadores de mutación o *crossover* para explorar el espacio de búsqueda, ya que está basado en el fenómeno físico del electromagnetismo.

El algoritmo EMO permite resolver problemas de optimización, definidos de la siguiente forma:

$$\begin{aligned} &\min f(x) \\ &x \in [l,u] \end{aligned} \tag{1}$$

Donde $[l,u]=\{x \in \Re^n \mid l_d \le x_d \le u_d, d=1,2...n\}$, $n$ es la dimensión de la variable $x$, $[l,u] \subset \Re^n$ es un subconjunto no vacio, y $f:[l,u] \to \Re$ es una función de valores reales. Por lo tanto, se conocen las siguientes características del problema:

$n$ es la dimensión del problema,
$u_d$ es el límite superior de la dimensión,
$l_d$ es el límite inferior de la dimensión,
$f(x)$ es la función que será minimizada.

Con estas condiciones, el algoritmo EMO emplea dos procesos básicos para la optimización, primero explora el espacio de búsqueda de forma aleatoria. El segundo proceso consiste en la explotación local de los puntos elegidos, para este propósito el algoritmo EMO usa los principios de la teoría del electromagnetismo. Utilizando ambos procesos se garantiza que el algoritmo converge en los mínimos de la función que son altamente atrayente, y se aleja de los valores máximos.

Partiendo de estos dos procesos fundamentales, el algoritmo EMO tiene cuatro fases para lograr la optimización global (Íker *et al*., 2003). Cada etapa se describe a continuación.

Inicialización**:** *m* partículas se toman aleatoriamente considerando el limite superior (*u*) y el limite inferior (*l*).

Búsqueda local**:** se busca un mínimo en la vecindad de un punto $\mathbf{x}^p$, donde $p \in (1,\ldots,m)$.

Cálculo del vector de fuerza total**:** las cargas y fuerzas se calculan para cada partícula.

Movimiento**:** cada partícula se desplaza de acuerdo con el vector de fuerza total calculada.

## Inicialización

Se produce de manera aleatoria un grupo ***G*** de *m* soluciones *n*-dimensionales, el cual se considera como la población de soluciones iniciales. Cada solución se considera como una partícula cargada y se supone que todas las partículas están distribuidas uniformemente entre el límite superior (*u*) y el límite inferior (*l*). La mejor partícula (mejor solución) se encuentra mediante la evaluación de la función objetivo, la cual depende de cada problema de optimización. Este procedimiento termina cuando todas las *m* partículas están evaluadas, eligiendo la partícula que en relación a la función objetivo tiene un mejor resultado. Esta fase corresponde al proceso de explorar las regiones atrayentes en un espacio factible.

## Búsqueda local

En esta etapa se intenta mejorar la solución ya encontrada. Sin embargo, para algunos problemas puede resultar innecesaria. Considerando esto, es posible formular una clasificación de los algoritmos EMO: EMO sin búsqueda local, EMO con búsqueda local aplicada sólo a la mejor partícula actual, y EMO con búsqueda local aplicada a todas las partículas, siendo este último el caso de estudio de este artículo.

Considerando un determinado número de iteraciones, llamado *LSITER* y un parámetro de vecindad de búsqueda $\delta$, el procedimiento para encontrar el valor óptimo local se lleva a cabo de la siguiente manera: el punto $\mathbf{x}^p$ se asigna a una variable temporal $\mathbf{y}$ para almacenar la información inicial. A continuación, para una coordenada dada *d* de una partícula, se selecciona un número aleatorio ($\lambda_1$) y se combina con $\delta$ obteniendo la longitud de paso de búsqueda. El punto $\mathbf{y}$ entonces se desplaza en la dirección que la longitud de paso indique, el signo que esta dirección tenga, también se calcula de manera aleatoria ($\lambda_2$). Si el valor obtenido tras evaluar $\mathbf{y}$ en la función a optimizar es mejor después de haber realizado *LSITER* iteraciones, el punto $\mathbf{x}^p$ se reemplaza por $\mathbf{y}$, terminando así la búsqueda en la vecindad *p*, de otra forma $\mathbf{x}^p$ conserva su valor. Por último, el mejor punto actual se actualiza en la partícula. El seudocódigo de este algoritmo se describe en la figura 1.

De forma general, la búsqueda local aplicada a todas las partículas, puede reducir el riesgo de caer en un mínimo local, pero la desventaja es que tiende a aumentar el tiempo de cómputo. Mantener la búsqueda local centrada en la mejor partícula actual resulta más conveniente, ya que así es posible mantener la eficiencia y precisión computacional. En la búsqueda local la longitud de paso representa un importante factor que depende de los límites de cada dimensión y determina el desempeño del método de búsqueda local.

1: $contador \leftarrow 1$
2: $longitud \leftarrow \delta\left(\max\left\{u_d - l_d\right\}\right)$
3: **for** $p = 1$ to $m$ **do**
4: **for** $d = 1$ to $n$ **do**
5: $\lambda_1 \leftarrow \mathrm{U}(0,1)$
6: **while** $contador < LSITER$ **do**
7: $\mathbf{y} \leftarrow \mathbf{x}^p$
8: $\lambda_2 \leftarrow \mathrm{U}(0,1)$
9: **if** $\lambda_1 < 0.5$ **then**
10: $\mathbf{y}_d \leftarrow \mathbf{y}_d + \lambda_2\left(longitud\right)$
11: **else**
12: $\mathbf{y}_d \leftarrow \mathbf{y}_d - \lambda_2\left(longitud\right)$
13: **end if**
14: **if** $f(\mathbf{y}) < f(\mathbf{x}^p)$ **then**
15: $\mathbf{x}^p \leftarrow \mathbf{y}$
16: $contador \leftarrow LSITER - 1$
17: **end if**
18: $contador \leftarrow contador + 1$
19: **end while**
20: **end for**
21: **end for**
22: $\mathbf{x}^{mejor} \leftarrow \arg\min\left\{f(\mathbf{x}^p), \forall p\right\}$

Figura 1. Seudocódigo del método de búsqueda local

## Cálculo del vector de fuerza total

El cálculo del vector de fuerza total se basa en el *principio de superposición* (figura 2) de la teoría del electromagnetismo, el cual establece que: "la fuerza ejercida en una partícula por medio de otra partícula es inversamente proporcional a la distancia entre los puntos y directamente proporcional al producto de sus cargas" (Cowan, 1968). Cada partícula se desplaza de acuerdo con la ley de Coulomb (figura 3), la cual emplea la fuerza producida entre las partículas que depende del valor de carga que posee cada una de ellas. Dicha carga se determina por su valor de desempeño de la función objetivo y se calcula de la siguiente manera:

$$q^p = \exp\left(-n\frac{f\left(\mathbf{x}^p\right) - f\left(\mathbf{x}^{mejor}\right)}{\sum_{h=1}^{m}\left(f\left(\mathbf{x}^h\right) - f\left(\mathbf{x}^{mejor}\right)\right)}\right), \forall p \quad (2)$$

Donde $n$ denota la dimensión y $m$ representa el tamaño de la población de partículas del algoritmo EMO. Un problema de optimización que se defina con un número de dimensiones elevado, generalmente requiere una mayor población. En la ecuación (2) la partícula con el mejor valor en la función objetivo $\mathbf{x}^{mejor}$ se llama "la mejor partícula", y tiene una mayor carga. La fuerza de atracción que ejerce la mejor partícula sobre otra partícula dada es inversamente proporcional a la distancia existente entre ellas. Por lo tanto, esta partícula atrae otras partículas que tengan peores resultados de afinidad, y repele a las que tienen mejores valores de afinidad.

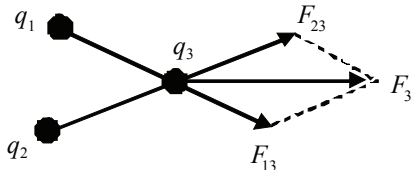


Figura 2. Principio de superposición

La fuerza resultante que existe entre las partículas determina la modificación del valor de las partículas en el proceso de optimización. La fuerza de cada partícula se calcula por la ley de Coulomb y el principio de superposición, por medio de la ecuación (3):

$$\mathbf{F}^p = \sum_{h \neq p}^{m}\left\{\begin{array}{ll}\left(\mathbf{x}^h - \mathbf{x}^p\right)\dfrac{q^p q^h}{\left\|\mathbf{x}^h - \mathbf{x}^p\right\|^2} & si \quad f(\mathbf{x}^h) < f(\mathbf{x}^p) \\ \left(\mathbf{x}^p - \mathbf{x}^h\right)\dfrac{q^p q^h}{\left\|\mathbf{x}^h - \mathbf{x}^p\right\|^2} & si \quad f(\mathbf{x}^h) \geq f(\mathbf{x}^p)\end{array}\right\}, \forall p \quad (3)$$

Donde $f(\mathbf{x}^h) < f(\mathbf{x}^p)$ representa la atracción y $f(\mathbf{x}^h) \geq f(\mathbf{x}^p)$ representa la repulsión (figura 3). La fuerza resultante

de cada partícula es proporcional al producto de las cargas e inversamente proporcional a la distancia entre las partículas. Para que el proceso sea numéricamente consistente, la ecuación (3) debe ser normalizada como se muestra a continuación:

$$\mathbf{F}^p = \frac{\mathbf{F}^p}{\left\|\mathbf{F}^p\right\|}, \quad \forall p. \tag{4}$$

### Movimiento

De acuerdo con la fuerza resultante cada partícula se desplaza como indica la siguiente ecuación:

$$\mathbf{x}^p = \begin{Bmatrix} \mathbf{x}^p + \lambda \cdot \mathbf{F}^p \cdot \left(u_d - \mathbf{x}_d^p\right) & si \quad \mathbf{F}^p > 0 \\ \mathbf{x}^p + \lambda \cdot \mathbf{F}^p \cdot \left(\mathbf{x}_d^p - l_d\right) & si \quad \mathbf{F}^p \le 0 \end{Bmatrix}, \forall p \neq mejor \tag{5}$$

En la ecuación (5), $\lambda$ es un paso de búsqueda aleatorio, el cual es uniformemente distribuido entre cero y uno, por otro lado $u_d$ y $l_d$ representan los límites superior e inferior de la dimensión *d*, respectivamente. Si la fuerza es positiva, la partícula se desplaza hacia el límite superior, en caso contrario se desplaza hacia el límite inferior, en ambos casos utilizando una longitud de paso aleatoria. La mejor de las partículas no se mueve, debido a que es la que tiene una atracción absoluta y repele o atrae a los demás elementos de la población.

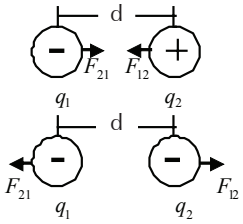


Figura 3. La ley de Coulomb

El proceso termina cuando se alcanza un número máximo de iteraciones, o cuando un valor $f(\mathbf{x}^{mejor})$ es optimo en algún sentido. Estas tres fases de EMO (búsqueda local, cálculo del vector de fuerza total, y movimiento), representan el proceso de explotación para encontrar el valor óptimo.

## Detección de círculos usando EMO

En este artículo los círculos se representan por la ecuación de segundo grado que se muestra en la ecuación (6), la cual considera tres puntos (Fischer *et al*., 1981), que se toman del mapa de bordes de la imagen. Para obtener los contornos de los objetos en la imagen, es necesario aplicar un método de detección de bordes. Para el empleo del enfoque propuesto en este artículo, esta tarea se lleva a cabo por medio del algoritmo de Canny. Las coordenadas de cada punto del borde se almacenan en un vector $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{Np}\}$ donde *Np* es la cantidad total de pixeles de borde que existe en la imagen. Donde a cada punto $\mathbf{e}_v$ del vector de bordes le corresponden las coordenadas $(x_v, y_v)$. Estos puntos, al tomarse en triadas, definirán un círculo único.

Para construir cada círculo candidato (una partícula cargada, según el enfoque de EMO), los índices $v_1$, $v_2$ y $v_3$ de tres puntos borde no colineales deben combinarse, suponiendo que la circunferencia del círculo pasa por los puntos correspondientes a estos índices $\mathbf{e}_{v_1}$ ; $\mathbf{e}_{v_2}$ ; $\mathbf{e}_{v_3}$. De esta manera, un conjunto de soluciones candidatas se genera de manera aleatoria para formar la población inicial de partículas. Estas soluciones serán operadas por medio del algoritmo EMO, hasta que se alcanza un mínimo aceptable, correspondiendo la mejor partícula al círculo real contenido en la imagen.

Como el proceso de desplazamiento modifica las partículas, la función objetivo va mejorando en cada generación por medio de la discriminación de los círculos con un mayor valor de error y eligiendo aquellos con un menor error. El siguiente análisis, explica los pasos requeridos para llevar a cabo la tarea de detección de círculos considerando el enfoque propuesto.

### Representación de las partículas

Cada partícula $\mathbf{C}$ agrupa tres puntos de borde. En esta representación, los puntos de borde se almacenan de acuerdo con el índice relativo a su posición en el arreglo de bordes $\mathbf{E}$. A su vez, una partícula se codifica como un círculo que pasa a través de los puntos $\mathbf{e}_i$, $\mathbf{e}_j$ y $\mathbf{e}_k$ de forma que cada partícula sea representada de la siguiente manera ($\mathbf{C} = \{\mathbf{e}_i, \mathbf{e}_j, \mathbf{e}_k\}$). Cada círculo es representado por tres parámetros: $x_0$, $y_0$ y *r*, siendo $(x_0, y_0)$ las coordenadas del centro del círculo y *r* su radio. La ecuación del círculo que pasa a través de los tres puntos de borde puede calcularse como sigue:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{6}$$

Considerando:

$$\mathbf{A} = \begin{bmatrix} x_j^2 + y_j^2 - (x_j^2 + y_i^2) & 2 \cdot (y_j - y_i) \\ x_k^2 + y_k^2 - (x_i^2 + y_i^2) & 2 \cdot (y_k - y_i) \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 2 \cdot (x_j - x_i) & x_j^2 + y_j^2 - (x_i^2 + y_i^2) \\ 2 \cdot (x_k - x_i) & x_k^2 + y_k^2 - (x_i^2 + y_i^2) \end{bmatrix} \tag{7}$$

$$x_0 = \frac{\det(\mathbf{A})}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))}$$

$$y_0 = \frac{\det(\mathbf{B})}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \quad (8)$$

mientras que el radio, se calcula usando:

$$r = \sqrt{(x_0 - x_b)^2 + (y_0 - y_b)^2} \quad (9)$$

Donde det(.) indica el determinante, y $b \in \{i, j, k\}$. La figura 4 muestra los parámetros definidos en las ecuaciones 7 a 9.

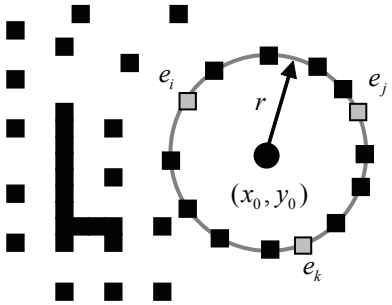


Figura 4. Círculo candidato (partícula cargada) construido a partir de la combinación de los puntos $e_i$, $e_j$ y $e_k$

Por lo tanto, es posible representar los parámetros [ $x_0$, $y_0$, $r$], que definen a cada círculo como una transformación $T$ de los índices $i$, $j$ y $k$ del vector de bordes, esta relación puede expresarse de la siguiente manera:

$$[x_0, y_0, r] = T(i, j, k) \quad (10)$$

Mediante la exploración de los índices agrupados en una partícula, es posible barrer el espacio de búsqueda, donde se encuentran los círculos a detectar por medio del algoritmo de EMO. Este enfoque reduce el espacio de búsqueda eliminando las soluciones que no son probables.

## Función objetivo

La existencia de una posible circunferencia en una imagen puede verificarse como la manera en que una forma virtual circular concuerda con los puntos borde de un círculo real en la imagen. La función objetivo validará la relación existente entre el círculo candidato $\mathbf{C}$ (partícula) y el contenido presente en la imagen (círculo real). Para realizar tal prueba se considera un vector de puntos $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_{N_s}\}$, donde $N_s$ es el número de puntos de prueba sobre los cuales se verificará la existencia de un punto borde.

El conjunto de prueba $\mathbf{S}$ es generado por el algoritmo de círculo de punto central (Midpoint Circle Algorithm, MCA) (Bresenham, 1987). El MCA calcula considerando un radio $r$ y las coordenadas del centro los $N_s$ puntos requeridos para representar el círculo, produciendo el vector de prueba $S$. El algoritmo emplea la ecuación del círculo $x^2 + y^2 = r^2$ sólo en el primer octante. Dibuja una curva iniciando en el punto ($r$, 0) y continúa arriba y hacia la izquierda usando sumas y restas de enteros. Vea detalles completos en (Van, 1984). Aunque el algoritmo se considera el más rápido y que provee una precisión de subpíxel, es importante asegurar que los puntos que caigan fuera del plano de la imagen no sean considerados ni incluidos en $\mathbf{S}$.

La función de costo o función objetivo $J(\mathbf{C})$, representa la correspondencia (o error) resultante de los pixeles $S$ del círculo candidato y los pixeles realmente existentes en la imagen de bordes, resultando:

$$J(\mathbf{C}) = 1 - \frac{\sum_{v=1}^{Ns} Error(x_v, y_v)}{Ns} \quad (11)$$

donde $Error(x_v, y_v)$ es una función que comprueba la existencia del píxel $(x_v, y_v)$ esto es:

$$Error\ (x_v, y_v) = \begin{cases} 1 & \text{si el pixel en la posicion } (x_v, y_v) \text{ existe} \\ 0 & \text{cualquier otro caso} \end{cases} \quad (12)$$

La función objetivo en la ecuación (11) acumula el número de puntos que de acuerdo con $Error(x_i, y_i)$ existen en la imagen de bordes. Los pixeles $\mathbf{S}$ bajo prueba, constituyen el perímetro del círculo que corresponde a $\mathbf{C}$. Por tanto, el algoritmo busca minimizar $J(\mathbf{C})$, dado que un valor pequeño implica una mejor respuesta (o correspondencia) de la "circularidad". El proceso de optimización puede entonces detenerse después de un número máximo de generaciones o cuando los individuos presenten un error mínimo definido como umbral.

## Implementación del algoritmo EMO

La implementación del algoritmo propuesto se puede resumir en los siguientes pasos:

*Paso 1*

Se aplica el filtro de Canny para encontrar los bordes y almacenarlos en el vector $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{Np}\}$. El contador de iteraciones se inicializa en $n = 0$.

*Paso 2*

Se generan *m* partículas iniciales (cada una con $e_i$, $e_j$ y $e_k$ elementos, donde $e_i$, $e_j$ y $e_k \in \mathbf{E}$). Las partículas que presentan radios demasiado pequeños o grandes, se eliminan (los puntos colineales se descartan). Se evalúa la función objetivo $J(\mathbf{C}^p)$ para determinar la mejor partícula $\mathbf{C}^{mejor}$ donde $\mathbf{C}^{mejor} \leftarrow \arg\min\{J(\mathbf{C}^p), \forall p\}$.

*Paso 3*

Se asigna la partícula $\mathbf{C}^p$ y se almacena temporalmente en $\mathbf{y}$. Se selecciona un número aleatorio y se combina con $\delta$ para obtener la longitud de paso para una coordenada dada *i*, *j* o *k*. Por lo tanto, la partícula $\mathbf{C}^p$ se desplaza a lo largo de esa dirección. Si se minimiza $J(\mathbf{C}^p)$, la partícula $\mathbf{C}^p$ se reemplaza por su nuevo valor, de lo contrario, se mantendrá el valor almacenado temporalmente.

*Paso 4*

La carga entre las partículas se calcula usando la expresión (2), y su vector de fuerza se calcula con la ecuación (3). La partícula $\mathbf{C}^{mejor}$ con un mejor valor en la función objetivo, mantiene una carga grande y, por lo tanto, una mayor fuerza de atracción o repulsión.

*Paso 5*

Las partículas se desplazan de acuerdo con la magnitud de su fuerza. La nueva posición de la partícula se calcula por la expresión (4). $\mathbf{C}^{mejor}$, no se desplaza porque tiene la fuerza más grande y ésta atrae las otras partículas hacia ella.

*Paso 6*

El índice *n* se incrementa. Si *n=MAXITER* o si el valor de $J(\mathbf{C})$ es menor que un valor de umbral predefinido, entonces el algoritmo se detiene y el flujo continua en el paso 7. De lo contrario, se regresa al paso 3.

*Paso 7*

Se selecciona la mejor partícula $\mathbf{C}^{mejor}$ de la última iteración.

*Paso 8*

En el mapa original de bordes, el algoritmo marca lo puntos correspondientes a $\mathbf{C}^{mejor}$. En caso de la detección de múltiples círculos, se regresa al paso 2.

*Paso 9*

Finalmente, la mejor partícula $C_{Nc}^{mejor}$ de cada círculo se usa para dibujar (sobre la imagen original) los círculos detectados. Considerando *Nc* como el número de círculos encontrados.

La figura 5 muestra una representación de la ley de Coulomb en el problema de detección de círculos. El círculo original que será detectado se representa por la línea sólida mientras que la línea discontinua representa los círculos que mantienen la mayor fuerza de atracción, es decir, que tienen el menor valor de error. Los círculos que se repelen debido a un mayor valor de error, están representados por la línea punteada.

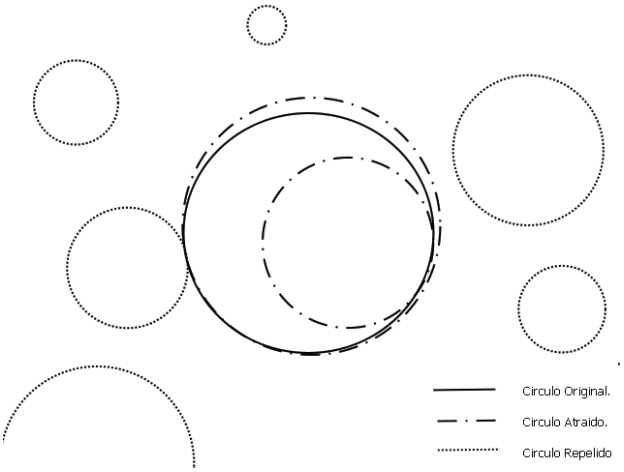


Figura 5. Analogía de los círculos con la ley de Coulomb

## Resultados experimentales

Para evaluar el desempeño del detector de círculos propuesto en este artículo, se implementaron las siguientes pruebas:

- Detección de círculos
- Discriminación de formas
- Detección de múltiples círculos
- Aproximación circular
- Aproximación de círculos imperfectos, ocluidos y detección de arcos.

### Precisión y tiempo computacional

Todas las pruebas se desarrollaron sobre imágenes naturales y sintéticas consideradas como complejas en cada análisis. Cada prueba se llevó a cabo con un conjunto de *m=10* partículas, un máximo de iteración para la búsqueda local de $LSITER = 2$, la longitud de paso para la búsqueda local es $\delta = 3$, y un valor máximo de iteraciones *n=20*. Finalmente, el espacio de búsqueda (pixeles de borde) tiene las siguientes implicaciones en los bordes $u = 1$, $l = Np$ para cada variable $\mathbf{e}_i$, $\mathbf{e}_j$ y $\mathbf{e}_k$.

## Detección de círculos

### Detección de círculos en imágenes sintéticas

Para las pruebas se usaron imágenes sintéticas de 256 × 256 pixeles. Cada imagen fue generada dibujando sólo un círculo imperfecto (elipse), localizado aleatoriamente. Las imágenes se contaminaron añadiendo ruido para incrementar la complejidad del proceso de detección. Los parámetros a detectar son la posición del centro del círculo ($x$, $y$) y su radio ($r$).

El algoritmo se configuró para realizar 20 iteraciones en cada imagen de prueba. En todos los casos, el algoritmo fue capaz de detectar los parámetros del círculo a pesar de la presencia del ruido. La detección es robusta y es posible manejar imágenes a mayor escala, manteniendo un tiempo de cómputo razonablemente bajo (típicamente bajo 10 ms). La figura 6 muestra el resultado de la detección de círculos para una imagen sintética.

### Detección de círculos en imágenes naturales

Este experimento pone a prueba la detección de círculos sobre imágenes reales. Se usaron veinticinco imágenes de 640 × 480 pixeles en esta prueba. Todas las imágenes se capturaron usando una cámara digital a color de 8-bits. Las imágenes fueron procesadas usando un algoritmo de detección de bordes antes de aplicar el detector de círculos EMO. La figura 7 muestra dos casos tomados de las 25 imágenes probadas.

Las imágenes reales difícilmente contienen círculos perfectos. Sin embargo, el algoritmo de detección propuesto aproxima circularmente al cuasi-círculo encontrado en la imagen, esto es, el círculo candidato que corresponde al de menor error obtenido en la función objetivo $J(\mathbf{C})$ es el que corresponderá al círculo de la imagen real. El tiempo de detección para la imagen mostrada en la figura 7a fue 13.540807 segundos mientras que para la figura 7b fue 27.020633 segundos. Los resultados obtenidos fueron analizados estadísticamente ejecutándose 20 veces sobre las mismas imágenes, generando los mismos valores para los parámetros $x_0$, $y_0$ y $r$. De esta manera, el algoritmo EMO propuesto fue capaz de converger a la solución mínima referida por la función objetivo $J(\mathbf{C})$, utilizando sólo 20 iteraciones.

## Prueba de discriminación de formas

En esta sección se discute la habilidad del algoritmo para detectar círculos en presencia de otras figuras que fungen como distractores. Cinco imágenes sintéticas de 540 × 300 pixeles se consideraron en este experimento. Además, se agregó ruido a todas las imágenes y se usó un máximo de 20 iteraciones para la detección. Dos ejemplos de la detección de círculos en este tipo de imágenes se muestran en la figura 8. El mismo experimento fue repetido sobre imágenes reales (figura 9).

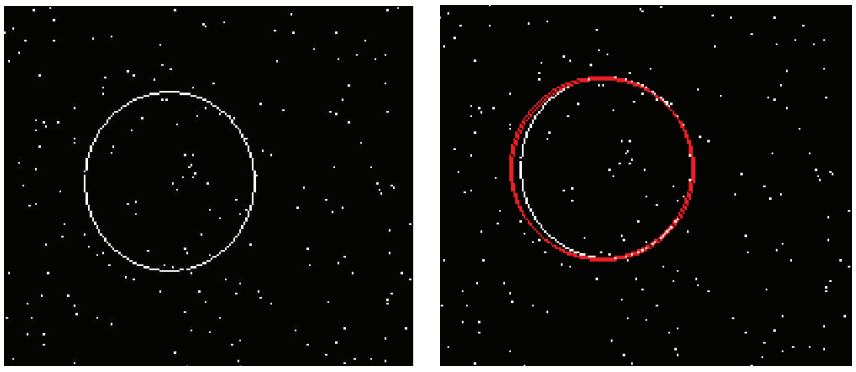

Figura 6. Detección de círculos en imágenes sintéticas: a) imagen de círculo original con ruido, b) círculo detectado

Figura 7. Detección de círculos aplicada a dos imágenes reales: a) el círculo detectado se muestra cerca de la periferia del ring y b) el círculo detectado se muestra en la circunferencia de la pelota

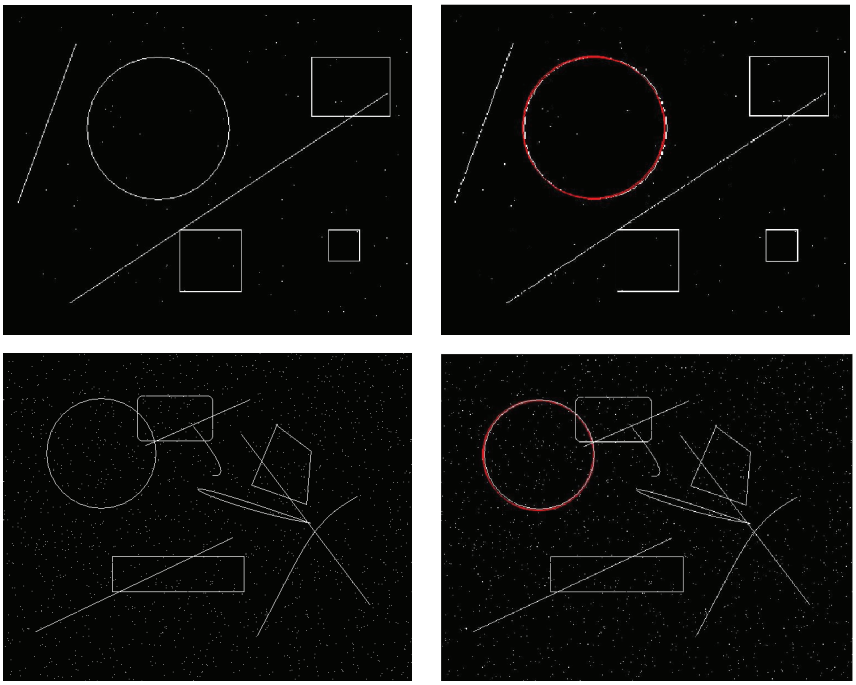

Figura 8. Detección de círculos en imágenes con otras primitivas geométricas; la imágenes a) y b) son las originales, mientras que b) y d) muestran sus respectivos círculos detectados

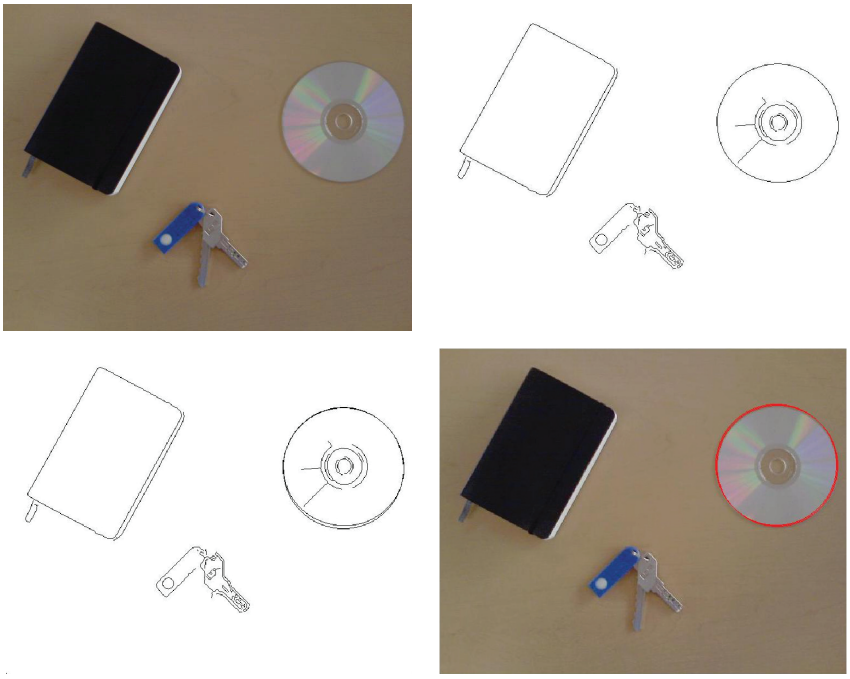

Figura 9. Diferentes formas incrustadas dentro de imágenes de la vida real: a) imagen de prueba, b) el correspondiente mapa de bordes, c) el círculo detectado y d) el círculo detectado sobre la imagen original

## Detección de múltiples círculos

El algoritmo propuesto es también capaz de detectar varios círculos presentes en imágenes. Primero, se define un número posible de formas circulares. El algoritmo entonces trabaja en la imagen de bordes original, hasta que se detecta el primer círculo. Este primer círculo representa la partícula (o círculo candidato) con el mínimo valor obtenido en la función objetivo $J(\mathbf{C})$ durante la búsqueda. Entonces se elimina esta forma circular, mientras que el detector de círculos EMO opera sobre la imagen de bordes con el círculo eliminado.

Este procedimiento se repite hasta alcanzar el máximo número de formas circulares. Finalmente, se lleva a cabo una validación de todos los círculos detectados por medio de un análisis de la continuidad de la circunferencia, como se propone en Roth *et al*. (1994). Este procedimiento se vuelve necesario en caso de que se requieran más formas circulares en el futuro, superando el número de círculos detectados en la imagen. En tal caso, el sistema puede mostrar una declaración falsa "no se detectan nuevos círculos en la imagen". Por otro lado, el algoritmo también identifica cualquier otra forma circular en la imagen por medio de la selección de las mejores formas, realizando esto hasta que se alcanza un número máximo de formas circulares.

Figura 10. Detección de múltiples círculos en imágenes reales: a) y c) muestran las imágenes de bordes que son obtenidas aplicando el algoritmo de Canny; b) y d) presentan las imágenes originales con los círculos encontrados superpuestos

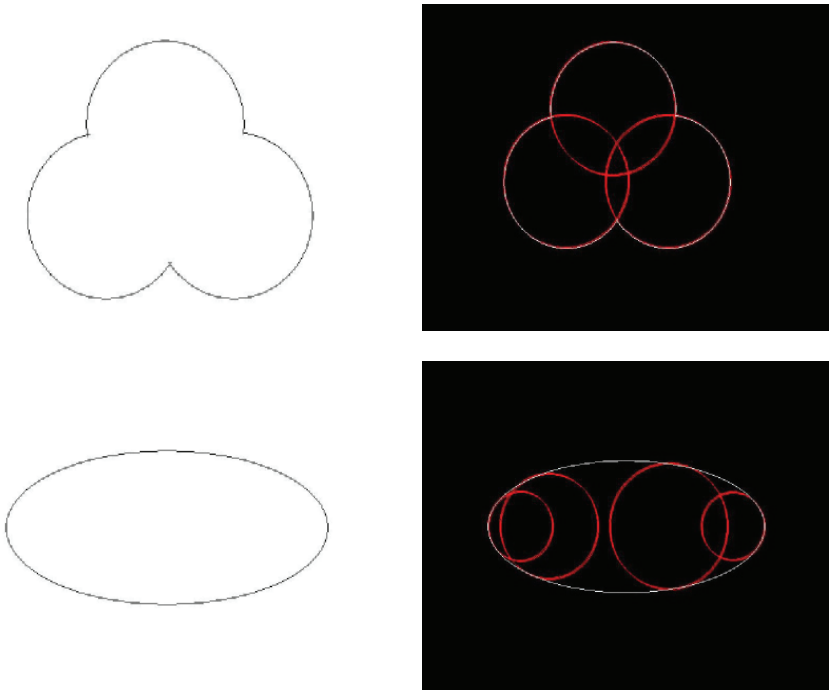

Figura 11. Aproximación circular: a) imagen original, b) su aproximación circular considerando 3 círculos, c) imagen original y d) su aproximación circular considerando 4 círculos

La figura 10a muestra el mapa de bordes después de aplicar el algoritmo de Canny, y la figura 10b presenta la imagen actual incluyendo varios círculos detectados, los cuales se sobrepusieron. Lo mismo se hizo para otros casos mostrados en las figuras 10c y 10d. El algoritmo EMO toma la imagen con el círculo eliminado proveniente de un paso anterior como la nueva imagen de entrada. La última imagen no incluye ningún círculo, porque todos ya han sido detectados y eliminados. Así, el algoritmo se enfoca en la detección de otros círculos potenciales. Se consideró un máximo de 20 iteraciones.

## Aproximación circular

Ya que en este enfoque la detección es considerada como un problema de optimización, es posible hacer la aproximación de una forma desconocida por medio de una concatenación de círculos. Esto se puede lograr usando la característica de detección de múltiples círculos que posee el algoritmo EMO (esto se explicó en la sección anterior), de acuerdo con los valores de la función objetivo $J(\mathbf{C})$ encontrados.

La figura 11 muestra algunos ejemplos de la aproximación circular. En la figura 11a se muestra una forma que se ha construido por medio de la superposición de varios círculos. La figura 11b muestra su aproximación circular de acuerdo con el detector EMO usando 3 círculos; finalmente, la figura 11c presenta un elipse que se ha obtenido por medio de la concatenación de cuatro círculos mostrados en la figura 11d.

Es posible también aproximar formas por medio de múltiples círculos en imágenes reales. La figura 12 muestra un ejemplo de esta capacidad, en el cual una de las tres figuras se aproximaron por dos círculos, ya que por la naturaleza de la misma imagen resulta impreciso para el algoritmo EMO emplear un solo círculo.

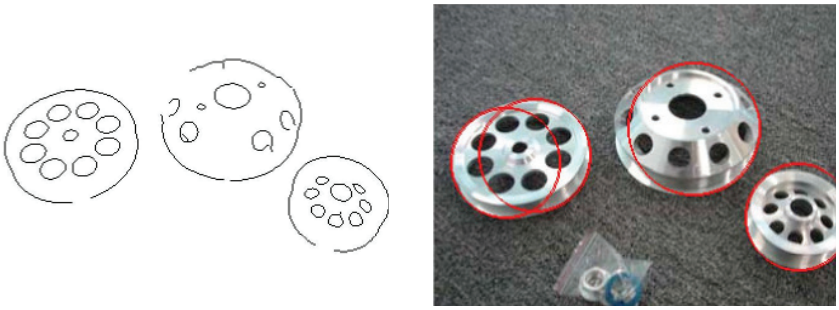

Figura 12. Aproximación circular en imágenes de la vida real: a) bordes de la imagen obtenidos por el algoritmo de Canny, b) círculos detectados, uno de los objetos se aproximó empleando dos círculos

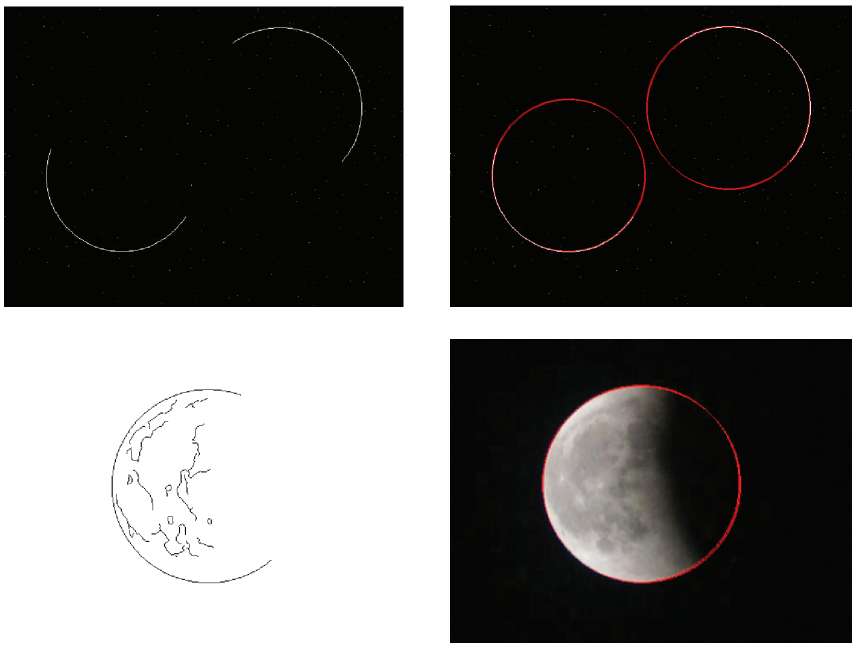

Figura 13. Aproximación circular de formas ocluidas, círculos imperfectos y detección de arcos: a) imagen original con dos arcos, b) aproximación circular de la imagen a), c) imagen natural de la luna ocluida y d) forma circular detectada de la figura c)

## Aproximación de círculos imperfectos, ocluidos y detección de arcos

La detección de círculos puede usarse también para aproximar formas circulares que provienen de segmentos de arcos, segmentos circulares ocluidos o círculos imperfectos, los cuales son desafíos comunes dentro del campo de la visión por computadora. El algoritmo EMO puede encontrar el círculo que corresponde a un arco de acuerdo con los valores de la función objetivo $J(\mathbf{C})$. La figura 13 muestra algunos ejemplos de esta funcionalidad.

## Precisión y tiempo computacional

Esta sección ofrece evidencia de la precisión que tiene el algoritmo para la detección de círculos. El experimento consiste en diez imágenes de 256 × 256 pixeles contaminadas con ruido, las cuales contienen un solo círculo centrado en la posición $x$ =128, $y$ =128, y un radio $r$ = 64. Se consideran dos tipos de ruido: el ruido Sal y Pimienta (ruido impulsivo) y el ruido Gaussiano.

El algoritmo EMO se hace iterar 20 veces por cada imagen y la partícula que muestra el mejor desempeño segúun $J(\mathbf{C})$ se considera como el círculo que mejor coincide con el que tiene la imagen. Este proceso se repite 35 veces por imagen para obtener consistencia en la prueba. Para evaluar la exactitud se emplea la suma del error (*Es*), que mide la diferencia entre el círculo verdadero (círculo actual) y el detectado (Cheng *et al*., 2009). La suma del error se define a continuación:

$$Es = |x_d - x_v| + |y_d - x_v| + |r_d - r_v|, \quad (13)$$

Donde $x_v, y_v, r_v$ son las coordenadas del centro y el valor del radio del círculo real en la imagen. Por otra parte $x_d, y_d, r_d$ corresponden a los valores de centro y radio de los círculos detectados.

El primer experimento considera imágenes contaminadas dopadas con ruido Sal y Pimienta. Los parámetros del algoritmo EMO son: máximo de iteraciones $MAXITER = 35$; para la búsqueda local, $\delta = 4$ e $LSITER = 4$. El ruido añadido se produce usando MatLab©, considerando niveles de ruido entre 1% y 10%. Los resultados de *Es* y el tiempo de cómputo transcurrido se reportan en la tabla 1. La figura 14 muestra tres diferentes imágenes como ejemplos, incluyendo los 35 círculos detectados durante la prueba, que están sobrepuestos en cada imagen original. De la figura 14 resulta evidente que entre mayor sea la cantidad de ruido agregado, mayor será la dispersión en las formas detectadas.

El segundo experimento considera imágenes contaminadas con ruido Gaussiano. En el ruido Gaussiano se requiere un valor de umbral para convertir a pixeles binarios, creando así una imagen que presenta una cantidad de ruido mayor a la que se añade por contaminación Sal y Pimienta. La cantidad de ruido Gaussiano añadido, se encuentra entre 1% y 10%. Los valores resultantes de *Es* y el tiempo computacional consumido se reportan en la tabla 2. La figura 15 muestra tres diferentes imágenes como ejemplo, incluyendo los 35 círculos sobrepuestos. De nuevo es evidente que la dispersión de los círculos encontrados incrementa proporcionalmente a la cantidad de ruido añadido.

Tabla 1. Resultados de precisión y tiempo consumido en la detección de círculos en imágenes que fueron contaminadas con ruido Sal y Pimienta

| Propiedades de la imagen | | | | Resultados | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Suma del Error (Es) | | | | Tiempo computacional (Segundos) | |
| Tamaño | Centro del círculo | Radio | Nivel de ruido Sal y Pimienta | Total | Media | Desviación Estándar | Moda | Media | Desviación Estándar |
| 256x256 | (128,128) | 64 | 0.01 | 0 | 0 | 0 | 0 | 12.5505 | 0.5426 |
| 256x256 | (128,128) | 64 | 0.02 | 0 | 0 | 0 | 0 | 13.0491 | 0.7455 |
| 256x256 | (128,128) | 64 | 0.03 | 0 | 0 | 0 | 0 | 13.3401 | 0.7365 |
| 256x256 | (128,128) | 64 | 0.04 | 0 | 0 | 0 | 0 | 14.3228 | 0.655 |
| 256x256 | (128,128) | 64 | 0.05 | 2 | 0.0571 | 0.2355 | 0 | 14.2141 | 0.7982 |
| 256x256 | (128,128) | 64 | 0.06 | 0 | 0 | 0 | 0 | 14.8669 | 0.9604 |
| 256x256 | (128,128) | 64 | 0.07 | 3 | 0.0857 | 0.5071 | 0 | 15.3725 | 0.5147 |
| 256x256 | (128,128) | 64 | 0.08 | 7 | 0.2 | 0.6774 | 0 | 14.5373 | 0.5147 |
| 256x256 | (128,128) | 64 | 0.09 | 6 | 0.1714 | 0.5137 | 0 | 14.5747 | 0.8907 |
| 256x256 | (128,128) | 64 | 0.10 | 28 | 0.8 | 2.1666 | 0 | 14.7288 | 0.8116 |

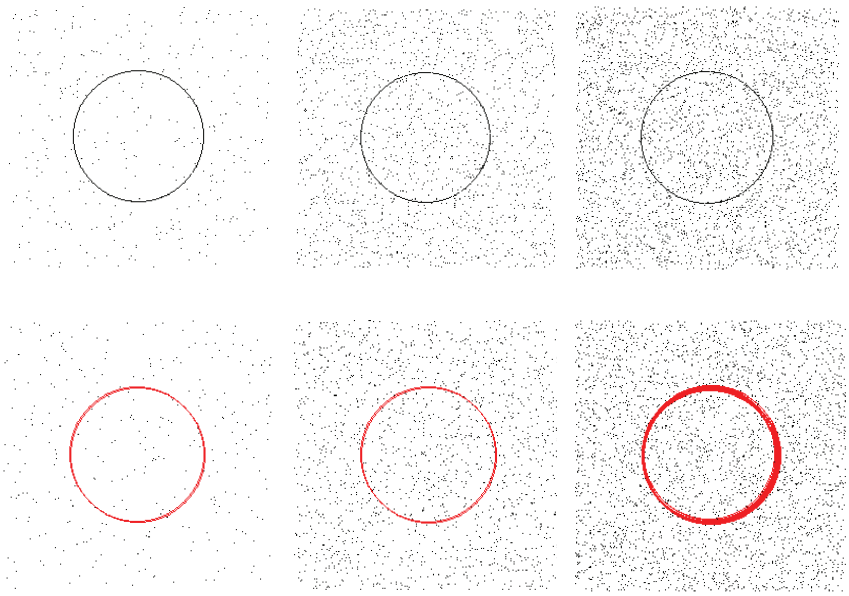

Figura 14. Detección de círculos en imágenes con ruido Sal y Pimienta: a) la imagen presenta un nivel de ruido añadido de 0.01, b) la imagen contiene 0.05 de ruido añadido, c) la imagen tiene un nivel de ruido añadido de 0.1, d), e) y f) son las imágenes que muestran los 35 círculos marcados para cada imagen de prueba. Todas las imágenes resultan después de aplicar el algoritmo EMO

Tabla 2. Resultados de precisión y tiempo consumido en la detección de círculos en imágenes que fueron contaminadas con Gaussiano

| Propiedades de la imagen | | | | | Resultados | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Suma del error (Es) | | | | Tiempo computacional (Segundos) | |
| | | | Ruido Gaussiano | | | | | | | |
| Tamaño | Centro del círculo | Radio | Media | Desviación Estándar | Total | Media | Desviación Estándar | Moda | Media | Desviación Estándar |
| 256x256 | (128,128) | 64 | 0 | 0.01 | 0 | 0 | 0 | 0 | 11.7008 | 0.6566 |
| 256x256 | (128,128) | 64 | 0 | 0.02 | 0 | 0 | 0 | 0 | 12.8182 | 0.9257 |
| 256x256 | (128,128) | 64 | 0 | 0.03 | 0 | 0 | 0 | 0 | 13.8073 | 0.6438 |
| 256x256 | (128,128) | 64 | 0 | 0.04 | 6 | 0.1714 | 0.5681 | 0 | 13.7866 | 0.8309 |
| 256x256 | (128,128) | 64 | 0 | 0.05 | 10 | 0.2857 | 0.825 | 0 | 14.0476 | 1.4691 |
| 256x256 | (128,128) | 64 | 0 | 0.06 | 26 | 0.7429 | 0.95 | 0 | 14.0622 | 0.5463 |
| 256x256 | (128,128) | 64 | 0 | 0.07 | 32 | 0.9143 | 1.961 | 0 | 14.6079 | 0.4346 |
| 256x256 | (128,128) | 64 | 0 | 0.08 | 158 | 4.5143 | 11.688 | 0 | 14.1732 | 0.8198 |
| 256x256 | (128,128) | 64 | 0 | 0.09 | 106 | 3.0286 | 6.5462 | 0 | 13.9074 | 0.8165 |
| 256x256 | (128,128) | 64 | 0 | 0.1 | 175 | 5 | 8.7447 | 0 | 15.1276 | 0.8606 |

Un experimento similar a los anteriores fue realizado sobre diferentes imágenes reales. Sin embargo, la suma del error (*Es*) no se usa porque las coordenadas del centro y el valor del radio de cada círculo son desconocidos. Por lo tanto, la ecuación 11 se usa para calcular el error de coincidencia. La tabla 3 muestra los resultados obtenidos después de aplicar el detector de círculos EMO, considerando las imágenes reales presentadas en la figura 16. Los 25 círculos detectados también están superpuestos en las imágenes originales.

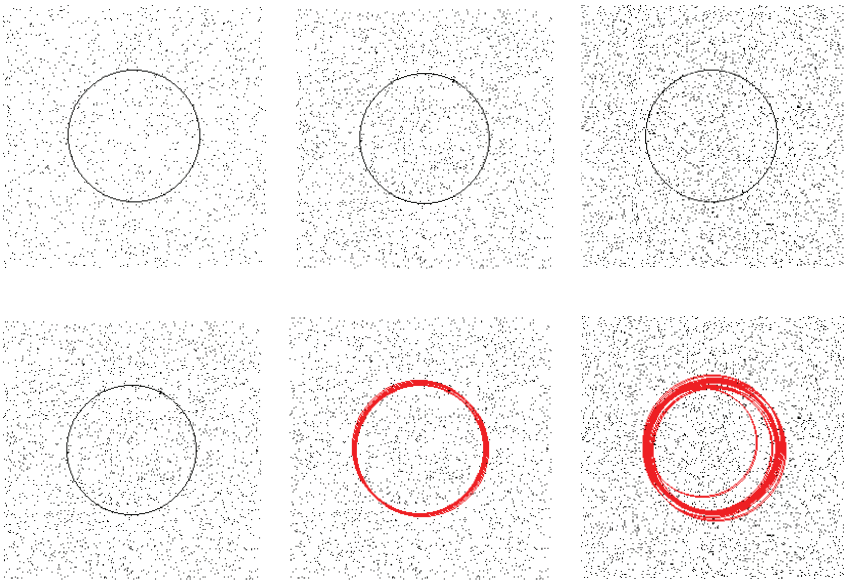

Figura 15. Detección de círculos en imágenes con ruido Gaussiano: a) la imagen presenta un nivel de ruido añadido de 0.01, b) la imagen contiene 0.05 de ruido añadido, c) la imagen tiene un nivel de ruido añadido de 0.1, d), e) y f) son las imágenes que muestran los 35 círculos marcados para cada imagen de prueba. Todas las imágenes resultan después de aplicar el algoritmo EMO

Tabla 3. Resultados obtenidos después de aplicar en detector de círculos EMO a imágenes reales

| Propiedades de la imagen | | Resultados | | | | | |
|---|---|---|---|---|---|---|---|
| | | Error de coincidencia (%) | | | | Tiempo computacional (Segundos) | |
| Nombre de la imagen | Tamaño | Total | Media | Desviación Estándar | Moda | Media | Desviación Estándar |
| Cue Ball | 430 x 473 | 28.1865 | 0.805 | 0.0061 | 0.8035 | 26.6641 | 1.4018 |
| Street Lamp | 474 x 442 | 19.0638 | 0.545 | 0.0408 | 0.5326 | 23.4763 | 1.7857 |
| Wheel | 640 x 480 | 22.265 | 0.636 | 0.0062 | 0.6336 | 31.356 | 1.4351 |

Figura 16. Imágenes reales usadas en los experimentos: a) *Cue ball*, b) *Wheel*, c) *Street lamp*; incluyendo los círculos detectados superpuestos

## Conclusiones

En este artículo se presenta un algoritmo para la detección automática de formas circulares, sin el uso del método tradicional de la transformada de Hough. El algoritmo está basado en una técnica inspirada en principios físicos, llamada *Electromagnetism-Like Optimization* (EMO), la cual permite resolver problemas de complejos de optimización, utilizando para ello principios del electromagnetismo. Hasta donde se conoce por parte de los autores, el algoritmo EMO no se ha aplicado en ninguna tarea de procesamiento de imágenes. Para la detección se usa la codificación de tres puntos no colineales extraídos del mapa de bordes de la imagen. Estos tres puntos constituyen círculos, los cuales se consideran soluciones candidatas al problema de detección. Una vez que estos círculos (partículas cargadas) se evalúan mediante una función objetivo (que verifica su existencia en la imagen real), los valores obtenidos sirven de guía para la modificación de las partículas por parte del algoritmo EMO. De esta manera, el algoritmo opera hasta que las soluciones candidatas coincidan con los círculos existentes en la imagen real. Este enfoque genera un detector de círculos, el cual puede iden-

tificar círculos eficientemente en imágenes reales, inclusive si los objetos circulares se encuentran parcialmente ocluidos, traslapados o en ambientes ruidosos. Los resultados experimentales muestran evidencia del desempeño en la detección de círculos considerando diferentes tipos de condiciones.

Una importante característica de este trabajo es considerar la detección de círculos como un problema de optimización. Tal enfoque permite al algoritmo EMO encontrar los parámetros de los círculos de acuerdo con el desempeño de la partícula (círculo candidato) con respecto a una función objetivo $J(\mathbf{C})$, en vez de usar todas las posibilidades que pueden suscribirse de acuerdo con la imagen de bordes, como la mayoría de los métodos lo hacen.

Aunque los métodos para la detección de círculos basados en la transformada de Hough, también usan tres puntos de borde para realizar un voto a favor de la forma circular potencial en el espacio de parámetros, ellos requieren grandes cantidades de memoria y tiempo computacional para obtener una solución subpíxel. Esto se debe a que los métodos que emplean la HT, el espacio de parámetros es cuantizado, lo que supone una pérdida de precisión en la determinación de los parámetros del círculo. Sin embargo, el método EMO propuesto, no emplea ninguna cuantizacion del espacio de parámetros. En este enfoque los círculos detectados se obtienen directamente por las ecuaciones (6) y (9), detectando efectivamente los círculos con precisión subpíxel.

Aunque los resultados experimentales ofrecen evidencia y demuestran que el algoritmo EMO puede obtener buenas aproximaciones en imágenes complicadas y con grandes cantidades de ruido, el objetivo de este artículo no es presentar un algoritmo que desmerite a los detectores de círculos que actualmente están disponibles, pero sí tiene como propósito mostrar que los sistemas basados en principios físicos como el *Electromagnetism-Like*, e pueden ser una alternativa sumamente atractiva para detectar formas paramétricas.

## Referencias


Fen X., Lau F.C.M. Parallel Physics-Inspired Waterflow Particle Mechanics Algorithm for Load Rebalancing, Comput. Netw. 2010, doi:10.1016/j.comnet.2010.02.002.

Contet J., Franck G., Pablo G., Abder K. Physics Inspired Multiagent System for Vehicle Platooning. AAMAS'07, Honolulu, Hawaii, USA, 2007.

Li-Ping X., Zeng J.C. A Global Optimization Based on Physicomimetics Framework. GEC'09, June, 2009, Shanghai, China.

Richard A. Formato JD. Central Force Optimization: A New Nature Inspired Computational Framework for Multidimensional Search and Optimization. Nature Inspired Cooperative Strategies for Optimization (NICSO), 2008. 129, pp. 221-238.

Wang L., Li L.P. An Effective Hybrid Quantum-Inspired Evolutionary Algorithm for Parameter Estimation of Chaotic Systems. *Expert Systems with Applications*, 37:1279–1285, 2010.

Davies E.R. *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press, London, 1990.

Muammar H., Nixon M. Approaches to Extending the Hough Transform., Proc. Int. *Conference on Acoustics, Speech and Signal Processing* ICASSP-89, 1989, 3, pp. 1556-1559.

Atherton T.J., Kerbyson D.J. Using Phase to Represent Radius in the Coherent Circle Hough Transform, Proc. *IEE Colloquium on the Hough Transform*, IEE, London, 1993.

Fischer M., Bolles R. Random Sample Consensus: A Paradigm to Model Fitting with Applications to Image Analysis and Automated Cartography. *CACM*, 24(6):381-395, 1981.

Shaked D., Yaron O. , Kiryati N. Deriving Stopping Rules for the Probabilistic Hough Transform by Sequential Analysis, Comput. *Vision Image Understanding*, 63:512-526, 1996.

Xu L., Oja E., Kultanen P. A New Curve Detection Method: Randomized Hough Transform (RHT). *Pattern Recognition Lett*, 11(5):331-338, 1990.

Han J.H., Koczy L.T., Poston T. Fuzzy Hough Transform, *Proc. 2nd International Conference on Fuzzy Systems*, 1993, 2, pp. 803-808.

Lu W., Tan J.L. Detection of Incomplete Ellipse in Images with Strong Noise by Iterative Randomized Hough Transform (IRHT). *Pattern Recognition*, 41(4):1268–1279, 2008.

Roth G., Levine M.D. Geometric Primitive Extraction Using a Genetic Algorithm, IEEE Trans. *Pattern Anal. Machine Intell*, 16(9):901-905, 1994.

Lutton E., Martinez P. A Genetic Algorithm for the Detection 2-D Geometric Primitives on Images, *Proc. of the 12th Internarional Conference on Pattern Recognition*, 1994, 1, pp. 526-528.

Yao J., Kharma N., Grogono P. Fast Robust GA-Based Ellipse Detection, proc. 17th International Conference on Pattern Recognition ICPR-04, 2004, 2, Cambridge, UK, pp. 859-862.

Ayala-Ramírez V., García-Capulin C., Pérez-García A., Sánchez-Yáñez E. Circle Detection on Images Using Genetic Algorithms. *Pattern Recognition Letters*, 27(6):652–657, 2006.

Dasgupta S., Das S., Biswas A. Abraham A. Automatic Circle Detection on Digital Images whit an Adaptive Bacterial Forganging Algorithm. Soft Computing, 2009, DOI 10.1007/s00500-009-0508-z.

Rosin P.L., Nyongesa H.O. Combining Evolutionary, Connectionist and Fuzzy Classification Algorithms for Shape Analysis, Cagnoni S. *et al.* (Eds.), Proc. EvoIASP, Real-World Applications of Evolutionary Computing, 2000, pp. 87-96.

Rosin P.L. Further Five Point Fit Ellipse Fitting, Proc. *8th British Machine Vision Conference*, Cochester, UK, 1997, pp. 290-299.

İlker-Birbil S., Shu-Cherng F. An Electromagnetism-Like Mechanism for Global Optimization. *Journal of Global Optimization*, 25:263–282, 2003.

İlker-Birbil S., Shu-Cherng F., Sheu R.L. On the Convergence of a Population-Based Global Optimization Algorithm. *Journal of Global Optimization*, 30(2):301–318, 2004.

Rocha A., Fernández E. Hybridizing the Electromagnetism-Like Algorithm with Descent Search for Solving Engineering Design Problems. *International Journal of Computer Mathematics*, 86(10):1932–1946, 2009.

Rocha A., Fernández E. Modified Movement Force Vector in an Electromagnetism-Like Mechanism for Global Optimization. *Optimization Methods & Software*, 24(2):253–270, 2009.

Naderi B., Tavakkoli-Moghaddam R., Khalili M. Electromagnetism-Like Mechanism and Simulated Annealing Algorithms for Flowshop Scheduling Problems Minimizing the Total Weighted Tardiness and Makespan. *Knowledge-Based Systems*, 23(2):77-85, 2010.

Tsou C.S., Kao C.H. Multi-Objective Inventory Control Using Electromagnetism-Like Meta-Heuristic. *International Journal of Production Research*, 46(14):3859–3874, 2008.

Yurtkuran A., Emel E. A New Hybrid Electromagnetism-Like Algorithm for Capacitated Vehicle Routing Problems. *Expert Systems with Applications*, 37(4):3427-3433, 2010.

Jhen-Yan J., Kun-Chou L. Array Pattern Optimization Using Electromagnetism-Like Algorithm. *AEU-International Journal of Electronics and Communications*, 63(6):491-496, 2009.

Cowan E.W. *Basic Electromagnetism*, Academic Press, New York, 1968.

Fischer M., Bolles R. Random Sample Consensus: A Paradigm to Model Fitting with Applications to Image Analysis and Automated Cartography. *CACM*, 24(6):381-395, 1981.

Bresenham J.E. A *Linear Algorithm for Incremental Digital Display of Circular Arcs*. Communications of the ACM 1987, 20, pp. 100-106.

Van Aken J.R. An Efficient Ellipse Drawing Algorithm. *CG&A*, 4(9):24-35. 1984.

Roth G., Levine M.D. Geometric Primitive Extraction Using a Genetic Algorithm. *IEEE Trans.Pattern Anal. Machine Intell*, 16(9):901–905, 1994.

Cheng H.D., Guo Y., Zhang Y. A Novel Hough Transform Based on Eliminating Particle Swarm Optimization and its Applications. *Pattern Recogn*., 42(9):959-1969, 2009.

Ying-ping C., Pei J. Analysis of Particle Interaction in Particle Swarm Optimization. *Theoretical Computer Science*, 411(21):2101-2115, 2010.

Blum C. Ant Colony Optimization: Introduction and Recent Trends. *Physics of Life Reviews*, 2(4):353-373, 2005.

Vasan A., Raju K.S. Comparative Analysis of Simulated Annealing, Simulated Quenching and Genetic Algorithms for Optimal Reservoir Operation. Applied Soft Computing 9, 2009, pp. 274–281.

## Semblanza de los autores

*Erik Cuevas*. Obtuvo en 1995 su título como ingeniero en comunicaciones y electrónica en la Universidad de Guadalajara, México. En el 2000, recibió el grado de maestría en electrónica industrial en ITESO y finalmente en el año 2006, el grado de doctor en ciencias en la Universidad Libre de Berlín, Alemania. Desde el 2007, trabaja como investigador en el Departamento de Electrónica en la Universidad de Guadalajara, sus áreas de investigación incluyen la visión computacional e inteligencia artificial.

*Diego Oliva*. Recibió en 2010 el grado de maestría en ciencias en ingeniería en electrónica y computación por la Universidad de Guadalajara. Desde el 2007, trabaja como profesor de tiempo parcial en el Instituto Politécnico de Tlajomulco y la Universidad de Guadalajara. Sus áreas de interés son la visión por computadora y los algoritmos metaheurísticos.

*Valentin Osuna-Enciso*. Recibió el título como ingeniero en electrónica y sistemas digitales por el Instituto Tecnológico del Mar en Mazatlán, Sinaloa, en 1999. En 2010, recibió el grado de maestro en ciencias en ingeniería en electrónica y computación por la Universidad de Guadalajara. Trabaja como profesor de tiempo parcial en la Universidad de Guadalajara desde 2008. Sus áreas de investigación son los algoritmos bio-inspirados y visión por computadora.

*Fernando Wario*. Recibió en 2010 el grado de maestría en ciencias en ingeniería en electrónica y computación por la Universidad de Guadalajara. Desde el 2007, trabaja como profesor de tiempo parcial en la Universidad de Guadalajara. Sus áreas de interés son la visión por computadora y los algoritmos metaheurísticos.