Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Cuevas, Erik; Ortega-Sánchez, Noé
El algoritmo de búsqueda armónica y sus usos en el procesamiento digital de imágenes
Computación y Sistemas, vol. 17, núm. 4, octubre-diciembre, 2013, pp. 543-560
Instituto Politécnico Nacional
Distrito Federal, México

El algoritmo de búsqueda armónica y sus usos en el procesamiento digital de imágenes

Erik Cuevas y Noé Ortega-Sánchez

Departamento de Electrónica, Universidad de Guadalajara, CUCEI,
Guadalajara, Jalisco,
México

erik.cuevas@cucei.udg.mx, noah55mx@gmail.com

Resumen. Métodos tradicionales de procesamiento de imagen presentan diferentes dificultades al momento de ser usados en imágenes que poseen ruido considerable y distorsiones. Bajo tales condiciones, el uso de técnicas de optimización se ha extendido en los últimos años. En este artículo se explora el uso del algoritmo de Búsqueda Armónica (BA) para el procesamiento digital de imágenes. BA es un algoritmo metaheurístico inspirado en la manera en que músicos buscan la armonía óptima en la composición musical, el cual ha sido empleado exitosamente para resolver problemas complejos de optimización. En este artículo se presenta dos problemas representativos del área de procesamiento digital de imágenes, como lo son: la detección de círculos y la estimación de movimiento, los cuales son planteados desde el punto de vista de optimización. Considerando este enfoque, en la detección de círculos se utiliza una combinación de tres puntos borde para codificar círculos candidatos. Utilizando las evaluaciones de una función objetivo (que determina si tales círculos están presentes en la imagen) el algoritmo de BA realiza una exploración eficiente hasta encontrar el circulo que mejor se aproxime a aquel contenido en la imagen (armonía óptima). Por otro lado, en la estimación de movimiento se utiliza el algoritmo de BA para encontrar el vector de movimiento que minimice la suma de diferencias absolutas entre bloques de dos imágenes consecutivas. Resultados experimentales muestran que las soluciones generadas son capaces de resolver adecuadamente los problemas planteados.

Palabras clave. Búsqueda armónica, detección de círculos, comparación de bloques, algoritmos meta-heurísticos, procesamiento digital de imágenes.

Harmony Search Algorithm and its Use in Digital Image Processing

Abstract. Classical methods often face big difficulties in solving image processing problems when images contain noise and distortions. For such images, the use of optimization approaches has been extended. This paper explores application of the Harmony Search (HS) algorithm to digital image processing. HS is a meta-heuristic optimization algorithm inspired by musicians improvising new harmonies while performing. In this paper, we consider two tasks as examples: circle detection and motion estimation, both issues are approached as optimization problems. In such approach, circle detection uses a combination of three edge points as parameters to construct candidate circles. A matching function determines if such candidate circles are actually present in a given image. In motion estimation, the HS algorithm is used to find a motion vector that minimizes the sum of absolute differences between two consecutive images. Experimental results show that the generated solutions are able to properly solve the problems under consideration.

Keywords. Harmony search, circle detection, block matching, meta-heuristics algorithms, digital image processing.

1 Introducción

Durante la última década se ha presentado un crecimiento sostenido en el campo de los algoritmos meta-heurísticos para la búsqueda y optimización. Tales métodos, debido a su capacidad de encontrar soluciones muy cercanas a las óptimas dentro de un periodo de tiempo razonable, han empezado a ser usados para resolver diferentes problemas de ingeniería, los cuales tradicionalmente usaban otros enfoques [1]. Un ejemplo de estos métodos es el algoritmo de Búsqueda Armónica introducido por Geem, Kimm and Loganatham [2], el cual usa como

ISSN 1405-5546
DOI: 10.13053/CyS-17-4-2013-007

metáfora el proceso de improvisación musical que ocurre cuando un músico busca la armonía óptima. De esta manera en BA una posible solución es similar a una armonía, mientras que sus operadores simulan el proceso de improvisación. En comparación con otros algoritmos meta-heurísticos reportados en la literatura, BA presenta varias ventajas como lo son la utilización de pocos parámetros de configuración [3] y su rapidez [4,5]. Considerando estas ventajas el algoritmo de BA ha sido satisfactoriamente aplicado para resolver una gran cantidad de problemas complejos de optimización. Algunos ejemplos son reportados en [6-9].

Una de las operaciones de importancia en el procesamiento digital de imágenes es la detección de primitivas geométricas circulares [10]. La detección de círculos en imágenes digitales es comúnmente resuelta a través de la transformada circular de Hough [11]. Sin embargo, en el caso particular de imágenes digitales con dimensiones significativas y un mapa de bordes densamente poblado (posiblemente ruido), el tiempo de procesamiento requerido para la transformada circular de Hough hace prohibitivo su uso en diferentes aplicaciones [12]. Para poder superar este problema algunos investigadores han propuesto nuevos enfoques siguiendo los principios de la transformada de Hough, dando como resultado la transformada probabilística de Hough [13], la transformada aleatoria de Hough (RHT) [14], la transformada difusa de Hough [15] y algunas otras discutidas en [16].

Como una alternativa a las técnicas basadas en la transformada de Hough, el problema de detección de formas ha sido también tratado por medio de métodos de optimización. En general, tales métodos han demostrado dar mejores resultados que aquellos basados en la Transformada de Hough en términos de precisión, velocidad y robustez [17]. En [18] Ayala-Ramírez et al., presentaron un detector de círculos basado en algoritmos genéticos el cual aunque es capaz de detectar múltiples círculos sobre imágenes reales, falla frecuentemente al detectar círculos imperfectos o distorsionados. Por otra parte en [19] Dasgupta et al., propusieron recientemente un detector de círculos automático usando el algoritmo de optimización de búsqueda de comida de bacterias “bacterial foraging” (BFOA).

En este artículo se presenta la manera en que el algoritmo de optimización de BA puede ser usado para la detección automática de formas circulares en imágenes ruidosas y complicadas, sin considerar los principios convencionales de la transformada de Hough. Considerando este enfoque se utiliza una combinación de tres puntos borde para codificar los círculos candidatos (posibles soluciones), mientras que una función objetivo es utilizada para medir la existencia de un circulo candidato sobre la imagen de bordes. Guiado por los valores de tal función objetivo, el conjunto de círculos candidatos es modificado por medio del algoritmo de Búsqueda Armónica (BA) de manera tal que el mejor circulo candidato pueda ser ajustado dentro de la figura más circular contenida en la imagen de bordes. Resultados experimentales muestran la efectividad para detectar círculos bajo diferentes circunstancias.

Por otro lado, el problema de estimación de movimiento tiene gran importancia dentro del análisis de imágenes, con aplicaciones en la navegación automotriz, compresión de video, cámaras de vigilancia, etc. Para el cálculo de la estimación de movimiento han sido propuestos muchos métodos [20], siendo la mayoría de ellos basados en técnicas diferenciales y de Comparación de Bloques (CB). Sin embargo, los métodos de comparación de bloques (Block Matching) son los más ampliamente usados debido a su simplicidad y fácil implementación [21]. En un algoritmo de CB, cada una de las imágenes pertenecientes a una secuencia de video se divide en bloques cuadrados de *N*x*N* pixeles denominados macro-bloques. Para cada macro-bloque de la imagen actual se determina dentro de una ventana de búsqueda su mejor similitud (función objetivo) en la imagen previa. De esta manera, la posición en la ventana de búsqueda donde se obtuvo la mejor semejanza define al vector de movimiento (estimación de movimiento). El análisis de similitud, que usa todos los puntos (búsqueda exhaustiva, FSA) de la ventana de búsqueda, representa el método más preciso y a la vez el más costoso computacionalmente. Para superar el problema

del costo computacional, varios algoritmos de CB han sido desarrollados. Entre estos algoritmos, los que usan un patrón predefinido de búsqueda son considerados los más rápidos, pero a la vez los más imprecisos. Ejemplos de estos enfoques son: El algoritmo de los tres pasos (TSS) [22] y el de búsqueda en diamante (DS) [23]. Recientemente, nuevos enfoques de CB basados en gradiente descendente han sido reportados en la literatura. Tales enfoques han demostrado ser más precisos que sus contrapartes basadas en patrones fijos a costa de un mayor tiempo de procesamiento [24]. Un algoritmo representativo de esta clase es el algoritmo rápido de comparación de bloques basado en predicción (FBMAUPR) [25]. A pesar de su velocidad, los algoritmos de patrones fijos y los basados en gradiente muestran una baja precisión debido a su tendencia a quedar atrapados en un mínimo local [26].

En este artículo, se introduce un nuevo algoritmo de comparación de bloques basado en BA. Debido a que las soluciones generadas por BA no corresponden a un patrón de búsqueda predeterminado ni se basan en el cálculo del gradiente, el enfoque propuesto ayuda a escapar de los mínimos locales, permitiendo obtener de esta manera, vectores de movimiento que correspondan al mínimo global existente.

De esta manera el algoritmo, usando los operadores definidos por BA, modifica un conjunto de soluciones iniciales durante un determinado número de iteraciones. Al final de este proceso la solución (armonía óptima) dentro de la ventana de búsqueda con mayor afinidad es considerada el vector de movimiento. Comparaciones realizadas con otros enfoques reportados en la literatura validan la eficiencia de la técnica propuesta, con respecto a precisión y velocidad.

El artículo está organizado de la siguiente manera: La sección 2 describe el funcionamiento y características del algoritmo de búsqueda armónica, mientras que la sección 3 detalla el procedimiento para la detección de círculos considerando el algoritmo de BA. En la sección 4 se describe el enfoque de estimación de movimiento y finalmente la sección 5 presenta las conclusiones obtenidas.

2 Búsqueda armónica

En el modelo básico de BA, cada solución candidata es considerada una “armonía”, y es representada por un vector *n*-dimensional. La población inicial de las soluciones candidatas es aleatoriamente inicializada y almacenada dentro de una memoria (Harmony Memory “HM”). De esta manera, una nueva solución es generada a partir de uno de los elementos contenidos en HM, a través de una operación de re-inicialización aleatoria o mediante una operación de ajuste de “tono” de un vector contenido en ella. Finalmente, la HM es actualizada mediante la comparación de la nueva solución candidata y el peor de los vectores contenidos en HM, si esta es mejor, remplazará el lugar del vector dentro de la memoria, de lo contrario no existirá cambio alguno. Este proceso se realiza hasta cumplir el criterio de paro. La forma básica del algoritmo BA consiste en tres etapas: inicialización, improvisación de nuevas armonías (generación de nuevas soluciones) y la actualización de la memoria (HM). En los siguientes párrafos se abordaran los detalles de cada etapa.

2.1 Inicialización

El algoritmo comienza inicializando un conjunto de soluciones (armonías) iniciales; cada solución es un vector *n*-dimensional que contiene los valores de los parámetros a ser optimizados, los cuales son aleatoriamente determinados entre los límites inferiores $l(j)$ y superiores $u(j)$ previamente definidos. Los parámetros de algoritmo de BA, son el tamaño de la memoria (*HMS*), la razón de exploración (*HMCR*), la razón de ajuste de tono (*PAR*), el ancho de desplazamiento (*BW*) y el número de improvisaciones (*NI*), el cual representa el número total de iteraciones. En este contexto, el tamaño de la memoria *HMS* define el número de elementos cuyo mejor desempeño en el proceso de optimización pueden ser almacenados, para su uso en el proceso de evolución. El parámetro *HMCR* representa la probabilidad de construir una nueva solución a partir de los elementos previamente almacenados en la memoria HM. La razón de ajuste de tono (*PAR*) define la

probabilidad de volver a modificar una solución previamente generada, mientras que el ancho de desplazamiento (*BW*) expresa la magnitud máxima de dicha modificación. Finalmente el número de improvisaciones (*NI*) representa la cantidad de iteraciones que el algoritmo ejecuta.

De esta manera cada vector $x_i = \{x_i(1), x_i(2), ..., x_i(n)\}$, que representa la solución numero *i*, es aleatoriamente generada usando: $x_i(j) = l(j) + (u(j) - l(j)) \cdot rand(0,1)$, de tal manera que $j = 1,2,...,n$ y $i = 1,2,...,HMS$, donde $rand(0,1)$ es un número distribuido uniformemente entre los valores 0 y 1. Por lo tanto, la memoria HM almacenará los *HMS* vectores generados inicialmente, quedando la memoria configurada de la siguiente manera:

$$\mathrm{HM} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{HMS} \end{bmatrix} \tag{1}$$

2.1.1 Improvisación de nuevas armonías

En esta etapa, un nuevo vector $\mathbf{x}_s$ es generado (improvisado) en base a los siguientes tres operadores: por examinado de memoria, re-inicialización aleatoria y ajuste de tono. En el paso de examinado de memoria, el valor de la primera variable $x_s(1)$ para el nuevo vector es elegido de cualquier de los valores ya existentes dentro de HM (del conjunto $\{x_1(1), x_2(1), ..., x_{HMS}(1)\}$). Para esta operación, un número aleatorio distribuido uniformemente r_1 es generado dentro del rango [0,1]. Sí r_1 es menor que *HMCR*, entonces la variable $x_s(1)$ es generada usando el método discutido; de otro modo, $x_s(1)$ se obtiene a partir un valor aleatorio re-inicializado entre las fronteras del problema $[l(1), u(1)]$. Los valores de otras variables de decisión $x_s(2), x_s(3), ..., x_s(n)$ son seleccionadas de forma similar. Por lo tanto, ambas operaciones, examinado de memoria y re-inicialización aleatoria, pueden ser modeladas de la siguiente forma:

$$x_s(j) = \begin{cases} \text{con probabilidad } (HMCR) \\ x_s(j) \in \{x_1(j), x_2(j), \ldots, x_{HMS}(j)\} \\ \\ \text{con probabilidad } (1\text{-}HMCR) \\ l(j) + (u(j) - l(j)) \cdot rand(0,1) \end{cases} \tag{2}$$

Para cada solución generada $x_s(j)$ se realiza además un ajuste de tono, el cual es definido por la frecuencia de ajuste (*PAR*) y por el factor de modificación (*BW*), que controla el tamaño de la perturbación aplicada a la variable. Por lo tanto, el ajuste de tono se calcula de la siguiente manera:

$$x_s(j) = \begin{cases} \text{con probabilidad } (PAR) \\ x_s(j) = x_s(j) \pm rand(0,1) \cdot BW \\ \\ \text{con probabilidad } (1\text{-}PAR) \\ x_s(j) = x_s(j) \end{cases} \tag{3}$$

El operador de ajuste de tono es responsable de generar nuevas soluciones por medio de ligeros ajustes a la posición de la variable original. Esta operación puede ser considerada similar al proceso de mutación en los algoritmos evolutivos. Con la finalidad de proteger las variables por los cambios presentados, es importante asegurar que las soluciones perturbadas se encuentren dentro de los límites permitidos $[l, u]$, en caso contrario deberán ser truncados al máximo o mínimo valor del intervalo.

2.1.2 Actualización de la memoria (HM)

Después de que un nuevo vector solución $\mathbf{x}_s$ es generado, la memoria (HM) es actualizada mediante la competencia de afinidad entre $\mathbf{x}_s$ y la peor solución $\mathbf{x}_w$ contenida en la memoria (HM). Por lo tanto, si el valor de la función objetivo de $\mathbf{x}_s$ es mejor que $\mathbf{x}_w$, entonces $\mathbf{x}_s$ lo remplazara dentro de la memoria (HM). De lo contrario el contenido de HM permanece sin cambios.

Debido a que la memoria HM contendrá únicamente elementos cuyos valores de función

objetivo representan las mejores soluciones, al paso de las iteraciones dicha memoria almacenara solo a la mejor solución (o versiones desplazadas alrededor de ella), por lo que el algoritmo convergerá inevitablemente a un solo valor [27].

2.1.4 Procedimiento computacional

El procedimiento computacional del algoritmo de BA puede ser resumido de la siguiente manera:

Paso 1: Fijar los parámetros *HMS*, *HMCR*, *PAR*, *BW* y *NI*.

Paso 2: Inicializar HM y calcula el valor de la función objetivo de cada vector solución.

Paso 3: Improvisa una nueva armonía (solución) $\mathbf{x}_s$ de la siguiente manera:

for (*j* = 1 to *n*) do
if (r_1 < *HMCR*) then
$x_s(j) = x_a(j)$
where $a \in (1,2,\ldots,HMS)$
if (r_2 < *PAR*) then
$x_s(j) = x_s(j) \pm r_3 \cdot BW$
where $r_1, r_2, r_3 \in \text{rand}(0,1)$
end if
if $x_s(j) < l(j)$
$x_s(j) = l(j)$
end if
if $x_s(j) > u(j)$
$x_s(j) = u(j)$
end if
else
$x_s(j) = l(j) + r \cdot (u(j) - l(j)),$
where $r \in \text{rand}(0,1)$
end if
end for

Paso 4: Actualizar *HM*,
$\mathbf{x}_w = \mathbf{x}_s$ if $f(\mathbf{x}_s) < f(\mathbf{x}_w)$

Paso 5: Si *NI* es finalizado, el mejor vector solución $\mathbf{x}_b$ en HM es localizado, de otras manera regresar al paso 3.

3 Detección usando el algoritmo de BA

En este artículo, los círculos son representados por la conocida ecuación de segundo grado que se muestra en la Ec. (4). Como pre-procesamiento se aplica a la imagen original un detector de bordes Canny que permite marcar el contorno en un solo píxel. Todos los puntos borde son guardados dentro de la lista de vectores $P = \{p_1, p_2, ..., p_{E_p}\}$ donde E_p representa el número total de píxeles contenidos en la imagen de bordes. Cada vector p_i almacena las coordenadas (x_i, y_i) de cada píxel borde.

3.1. Representación de las armonías

Con la finalidad de construir un círculo candidato (armonías en el contexto de BA), se toman tres diferentes píxeles no-colineales i, j y k de la imagen de bordes, asumiendo que el contorno del círculo creado pasa a través de los puntos p_i, p_j, p_k. Como cada solución candidata $\mathbf{C} = \{p_i, p_j, p_k\}$ usa tres puntos del mapa de bordes, es necesario transformar a partir de estos tres puntos, los parámetros del círculo x_0, y_0 y r al cual representan, siendo (x_0, y_0) las coordenadas del centro y r el radio. Dicha transformación se realiza considerando, que la ecuación que describe un círculo se modela como:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{4}$$

de esta manera, se definen las matrices:

$$\mathbf{A} = \begin{bmatrix} x_j^2 + y_j^2 - (x_j^2 + y_i^2) & 2 \cdot (y_j - y_i) \\ x_k^2 + y_k^2 - (x_i^2 + y_i^2) & 2 \cdot (y_k - y_i) \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 2 \cdot (x_j - x_i) & x_j^2 + y_j^2 - (x_i^2 + y_i^2) \\ 2 \cdot (x_k - x_i) & x_k^2 + y_k^2 - (x_i^2 + y_i^2) \end{bmatrix} \tag{5}$$

A partir de **A** y **B** se calculan los parámetros del círculo siguiendo las siguientes ecuaciones:

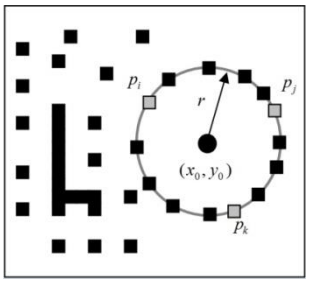
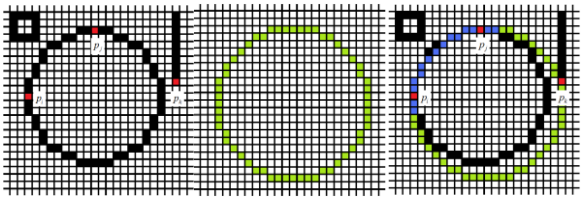


Fig. 1. Formación de circulo candidato sobre una imagen de bordes ejemplo. La combinación de los puntos p_i, p_j, p_k representa una solución candidata al problema de detección

$$x_0 = \frac{\det(\mathbf{A})}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))}$$
$$y_0 = \frac{\det(\mathbf{B})}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \quad (6)$$
$$r = \sqrt{(x_0 - x_d)^2 + (y_0 - y_d)^2}$$

Siendo $\det(\cdot)$ el determinante y el subíndice $d \in \{i, j, k\}$. La Figura 1 ilustra los parámetros definidos por las ecuaciones (4)-(6) sobre una imagen de bordes ejemplo. De la Figura puede apreciarse los pixeles elegidos del conjunto *P* (todos los pixeles de la imagen de bordes) para formar una solución candidata al problema de detección. Dicha solución candidata constituida por la combinación de los puntos p_i, p_j, p_k representa un elemento (armonía) que será operado por el algoritmo de BA durante su evolución.

Considerando a la combinación de puntos como los parámetros de cada armonía, es factible aplicar el algoritmo de BA para detectar los parámetros apropiados que describen al círculo. Este enfoque reduce el espacio de búsqueda eliminando las soluciones improbables. Por lo tanto, el número de soluciones candidatas generadas inicialmente son almacenadas en la memoria (HM). Estas soluciones son operadas mediante el algoritmo de BA hasta que la armonía óptima sea alcanzada. La mejor armonía de HM es por lo tanto considerada como la solución al problema de detección de círculos.



(a) (b) (c)

Fig. 2. Evaluación de la función objetivo $J(\mathbf{C})$. (a) Imagen original, (b) la figura virtual considerando los puntos p_i, p_j y p_k, (c) puntos coincidencia entre las dos imágenes marcados por azul y rojo; la figura virtual del círculo es descrita en verde

3.2. Función objetivo

Con la finalidad de calcular el error producido por una solución candidata **C**, la existencia de los píxeles que describen su circunferencia deberán de ser validados con la imagen de bordes. Esto es, si el círculo realmente existe en la imagen de bordes. De esta manera, el conjunto de píxeles de prueba es definido por $S = \{s_1, s_2, ..., s_N\}$, donde N representa el número de puntos de **C**, donde se verificara la existencia en la imagen de bordes. El conjunto de puntos *S* es generado para cada circulo candidato **C** por el algoritmo del punto central (MCA) [28]. EL MCA es un método de dibujo que usa el mínimo número de puntos requeridos para dibujar un círculo. MCA, evita el cálculo de raíces cuadradas de la Ec. 4 mediante la comparación de distancias entre los píxeles.

La función objetivo $J(\mathbf{C})$ representa el error que se produce al comparar los píxeles del círculo candidato **C** (almacenados en *S*) y la imagen de bordes. De esta manera la función objetivo se define como:

$$J(\mathbf{C}) = 1 - \frac{\sum_{v=1}^{N_s} E(s_v)}{N_s}, \quad (7)$$

donde $E(s_v)$ es una función que verifica si el píxel (x_v, y_v), existe en la imagen de bordes

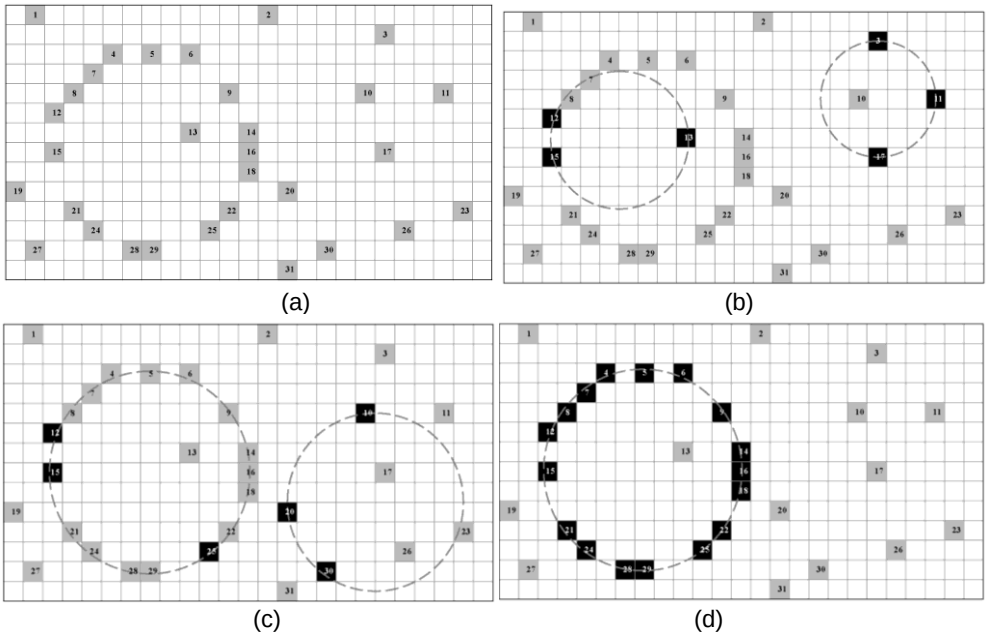


Fig. 3. Ejemplo de la evolución de las soluciones candidato operadas por el algoritmo de BA. (a) Mapa de bordes ejemplo. (b) Dos diferentes soluciones generadas aleatoriamente. (c) Configuraciones finales de ambas soluciones después de una iteración del algoritmo de BA. (d) Solución final obtenida después de haber ejecutado totalmente el algoritmo de BA

(siendo $s_v \in S$). De este modo, $E(s_v)$ es definida como:

$$E(s_v) = \begin{cases} 1 & \text{si el pixel en } (x_v, y_v) \text{ existe} \\ 0 & \text{de otro modo} \end{cases} \quad (8)$$

Un valor cerca de cero de la función objetivo $J(\mathbf{C})$ implica una mejor calidad de la solución **C**. La Figura 2 muestra el procedimiento de evaluación de la función objetivo $J(\mathbf{C})$. Primero, tres puntos seleccionados de la imagen de bordes (Figura 2a) codifican un círculo candidato **C**. Después, usando MCA, se construye una figura virtual (Figura 2b). Dicha figura virtual corresponde a los puntos almacenados en *S,* que de acuerdo a la Figura 2b estará integrada por $N_s = 56$. Finalmente, la figura virtual del círculo candidato es comparada con la imagen de bordes punto a punto con la finalidad de encontrar las coincidencias entre los puntos virtuales y los del mapa de bordes (Figura 2c). Como resultado, solo 18 puntos encuentran correspondencia entre ambas imágenes produciendo $\sum_{v=1}^{N_s} E(s_v) = 18$ y $J(\mathbf{C}) = 0.67$.

La Fig. 3 presenta un ejemplo de la evolución de las soluciones candidato operadas por el algoritmo de BA. La Fig. 3a muestra el mapa de bordes ejemplo. En dicho mapa de bordes, la lista de vectores *P* contiene 32 elementos. De tal lista *P*, se obtienen tres puntos aleatoriamente para formar cada uno de los elementos, que el algoritmo BA operara evolutivamente. La Fig. 3b presenta 2 diferentes elementos construidos de acuerdo a este procedimiento. El elemento $\mathbf{C}_1 = \{12,15,13\}$ está constituido por la

Tabla 1. Configuración de los parámetros del algoritmo BA como detector de círculos

HMS	HMCR	PAR	BW	NI
100	0.7	0.3	5	200

combinación de los puntos 12, 15 y 13, mientras que el elemento $\mathbf{C}_2 = \{3,11,17\}$ por la combinación de los puntos 3, 11 y 17. Ambas soluciones al ser operadas por el algoritmo de BA, modificaran sus valores conforme al procedimiento descrito en las secciones 2.1.2 y 2.1.3. La Fig. 3c muestra la nueva configuración de $\mathbf{C}_1$ y $\mathbf{C}_2$ después de ser modificadas. En el caso de $\mathbf{C}_1$ la solución es modificada atendiendo a1 caso de examinado de memoria, donde solo uno de los elementos (13) es modificado (23) mientras que los otros permanecen sin cambios. En el caso de la solución $\mathbf{C}_2$ es modificada por re-inicialización aleatoria, de tal forma que sus elementos son totalmente diferentes a los originales. La Fig. 3d presenta el mejor elemento obtenido (solución) después de haber finalizado el proceso de evolución.

3.3 Resultados experimentales

3.3.1 Configuración de parámetros

Para lograr un ajuste correcto de los parámetros del algoritmo de BA, es necesario pesar dos factores importantes uno contra el otro. El primero es la habilidad del algoritmo para converger reduciendo el número de iteraciones, mientras que el segundo factor es la capacidad de encontrar estimaciones más precisas de las figuras circulares. Los parámetros principales que controlan dichos factores son *HMS*, *HMCR*, *PAR* y *BW*. Tales parámetros fueron identificados experimentalmente después de procesar varias imágenes, obteniendo los valores mostrados en la Tabla 1. Estos valores fueron usados en todos los experimentos presentados en este artículo.

3.3.2 Detección de círculos

Se realizaron diversas pruebas para evaluar el desempeño del detector de círculos propuesto. Tales experimentos refieren a las siguientes tareas:

1. Localización de círculos
2. Discriminación de formas
3. Aproximación circular: círculos ocluidos y detección de arcos

Localización de círculos

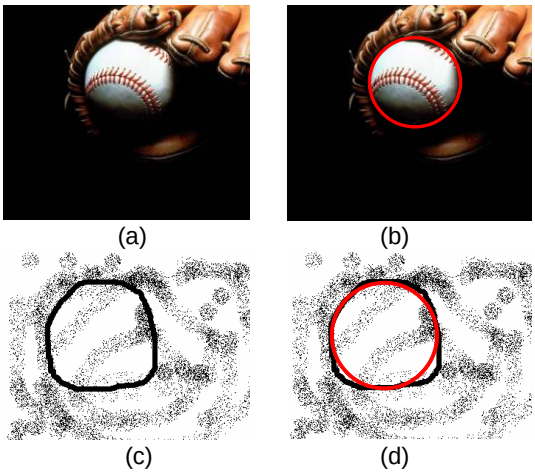


Fig. 4. Localización de círculos sobre imágenes. (a) Muestra la imagen original, mientras que (b) presenta el círculo detectado superpuesto. (c) Imagen contaminada con ruido y (d) resultado obtenido

Para el desarrollo experimental de esta prueba se usaron varias imágenes sintéticas y naturales de 352x256 píxeles, todas ellas contienen una figura circular y algunas incluso fueron contaminadas agregando ruido a manera de incrementar la complejidad de la tarea de localización. El algoritmo fue ejecutado 50 veces para cada imagen de prueba, identificando exitosamente todos los círculos en la imagen. La detección probó ser robusta, requiriendo menos de un segundo en ser completada. La Figura 3 muestra los resultados después de aplicar el algoritmo sobre dos imágenes tomadas del conjunto experimental de imágenes.

Las Fig. 4a y 4b presentan el resultado de aplicar el algoritmo de detección sobre una imagen natural común extraída del conjunto de prueba. Las Fig. 4c y 4d muestran el caso de una imagen sintética en la que el algoritmo de BA presenta la solución que mejor se acopla al problema de detección. Debido a que los métodos de detección operan directamente sobre el mapa de bordes, la Fig. 4c fue contaminado con varios puntos (ruido impulsivo) cerca de los bordes del círculo dibujado a mano con el objeto de dificultar su detección.

Discriminación de formas

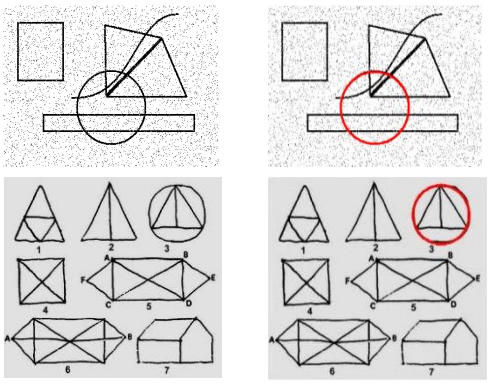


Fig. 5. Discriminación de círculos en imágenes. (a) Imagen contaminada con ruido, mientras que (b) presenta el círculo detectado superpuesto. (c) Imagen original dibujada a mano y (d) circulo obtenido como resultado

En esta prueba se discute la habilidad del detector para diferenciar patrones circulares, los cuales puedan estar presentes en la imagen, entre otras figuras. La Figura 5 muestra dos diferentes figuras y su resultado, después de haber aplicado el detector de BA.

Aproximación circular: círculos ocluidos y detección de arcos

Una importante particularidad del detector basado en BA, es que puede detectar círculos ocluidos o imperfectos, así mismo formas parcialmente definidas como segmentos de arco. La importancia de tal funcionalidad proviene del hecho de que este tipo de problemas son comunes en aplicaciones de procesamiento de imágenes. Debido a que la detección de círculos ha sido considerada como un problema de optimización, el algoritmo basado en BA permite encontrar círculos que puedan asemejar una forma dada de acuerdo al valor de afinidad para cada circulo candidato. De la misma forma, el algoritmo propuesto es capaz de identificar los arcos o círculos ocluidos. La Figura 6 muestra algunos ejemplos de esta funcionalidad. Un valor pequeño para $J(\mathbf{C})$, (cercano a cero) refiere a un círculo mientras que un valor ligeramente mayor implica un arco o una figura circular ocluida.

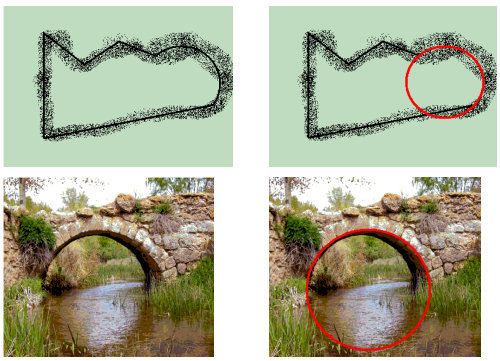


Fig. 6. Aproximación circular. (a) Imagen contaminada con ruido local, mientras que (b) presenta el círculo detectado superpuesto. (c) Imagen natural y (d) circulo obtenido como resultado

3.3.3 Comparación del desempeño

Con el objetivo de medir el desempeño del algoritmo propuesto, en esta sección se presenta la comparación del detector basado en BA con respecto a otros tres algoritmos, tal como lo son el método basado en GA [18], el detector BFAO [19] y la técnica RHT [14] sobre un conjunto de imágenes.

El algoritmo basado en GA propuesto por Ayala-Ramírez et al. [18], se configuro con una población de tamaño igual a 70, la probabilidad de crossover de 0.55, la probabilidad de mutación de 0.10 y el número de individuos-elite de 2. La selección se baso en el método de la ruleta y se aplico el crossover en el primer punto. Para la configuración del algoritmo BFAO se consideraron los siguientes valores: S=50, $N_c = 4$, $N_c = 350$, $N_{ed} = 1$, $P_{ed} = 0.25$, $d_{attract} = 0.1$, $w_{attract} = 0.2$, $h_{repellant} = 0.1$, $\lambda = 400$ y $\psi = 6$. Dichos valores fueron reportados como la mejor configuración posible de acuerdo a [19]. De la misma manera, el método RHT fue implementado para la comparación utilizando la configuración sugerida en [14].

Las imágenes difícilmente contienen formas circulares perfectas. Por lo tanto, con el propósito de probar la precisión en la detección circular, los parámetros del círculo detectado P_D (por alguno de los algoritmos usados en la comparación) son cotejados con los parámetros de la mejor aproximación circular producida por un observador humano P_H. De esta manera, considerando que los parámetros (x_D, y_D, r_D) representan al círculo automáticamente

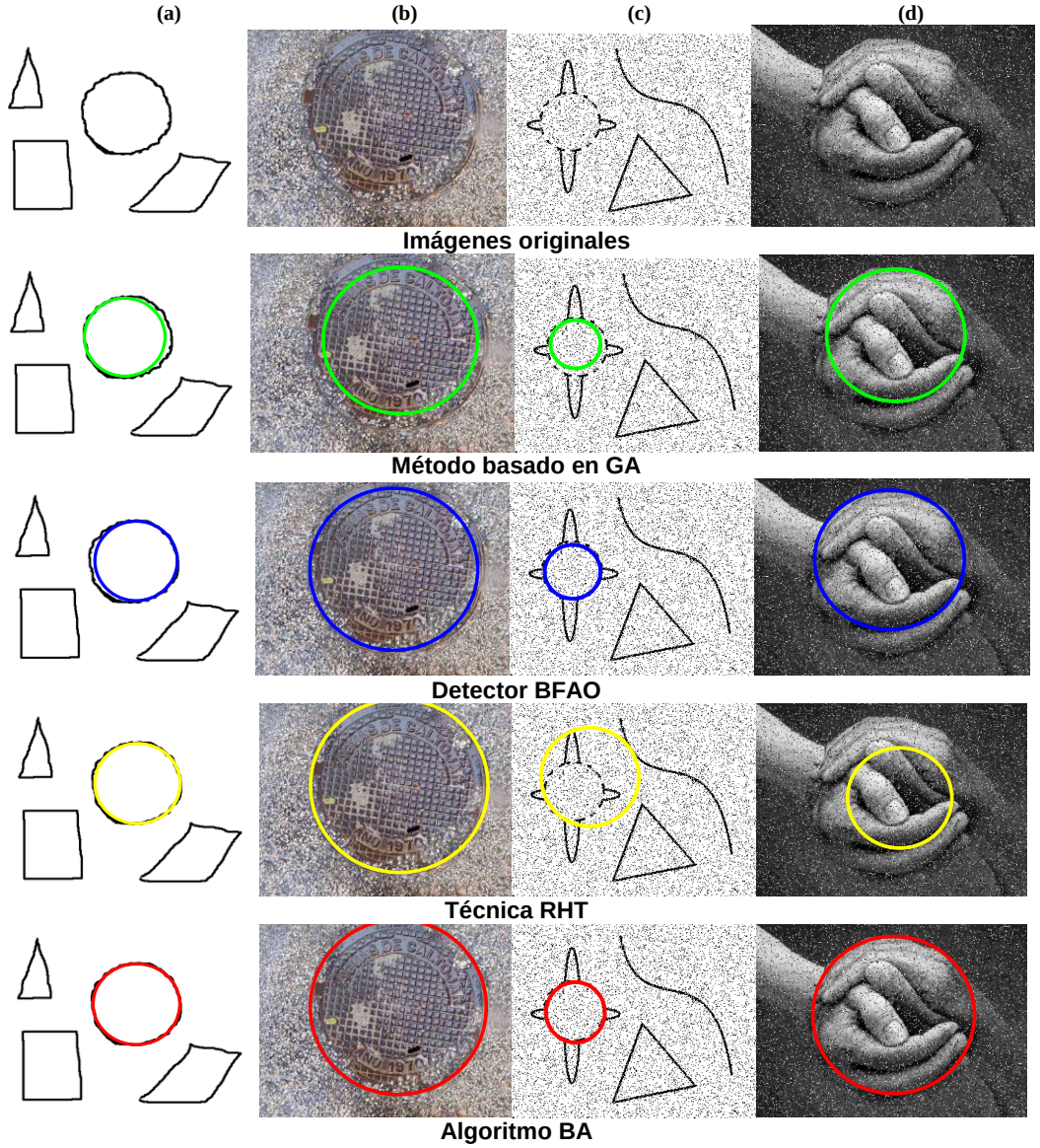


Fig. 7. Conjunto de imágenes usadas en la comparación, con la detección del círculo obtenido por el algoritmo de BA

detectado y (x_H, y_H, r_H) los parámetros obtenidos por el observador humano, el error resultante (Er) puede ser calculado de acuerdo al siguiente modelo:

$$Er = \eta(|x_H - x_D| + |y_H - y_D|) + \mu \cdot |r_H - r_D| \quad (9)$$

donde $(|x_{true} - x_D| + |y_{true} - y_D|)$ representa la diferencia del punto central entre el circulo detectado P_D y el considerado como optimo P_H, mientras que $(|r_{true} - r_D|)$ corresponde a la diferencia entre los radios. η y μ representan factores que ponderan la importancia relativa de cada diferencia en el

Tabla 2. Comparación de los índices Relación de éxito (RE) y promedio del error resultante (*Er*)

Imagen	**Relación de éxito (RE) (%)**			
	GA	**BFOA**	**RHT**	**BA**
(a)	75	90	**100**	**100**
(b)	90	98	**100**	**100**
(c)	95	98	91	**100**
(d)	91	95	85	**100**
	Promedio *Er* ± desviación estándar			
	GA	**BFOA**	**RHT**	**BA**
(a)	0.56±(0.029)	0.46±(0.051)	0.48±(0.077)	**0.21±(0.012)**
(b)	0.62±(0.021)	0.53±(0.018)	0.47±(0.081)	**0.28±(0.018)**
(c)	0.87±(0.056)	0.81±(0.021)	0.92±(0.053)	**0.39±(0.027)**
(d)	0.56±(0.017)	0.48±(0.017)	0.76±(0.037)	**0.24±(0.031)**

Tabla 3. Comparación de los índices número de iteraciones (NI) y la memoria usada

Imagen	**Número de iteraciones (*NI*) ± Desviación estándar**			
	GA	**BFOA**	**RHT**	**BA**
(a)	250±(39)	213±(14)	6,218±(0)	**141±(19)**
(b)	311±(11)	204±(21)	7,124±(0)	**145±(16)**
(c)	373±(21)	302±(27)	15,228±(0)	**171±(12)**
(d)	274±(25)	228±(38)	14,238±(0)	**150±(19)**
	Uso de la memoria (*MU*)			
	GA	**BFOA**	**RHT**	**BA**
(a)	**1.8Kb**	3.8Kb	3.6Mb	2.4Kb
(b)	**1.8Kb**	3.8Kb	5.8Mb	2.4Kb
(c)	**1.8Kb**	3.8Kb	6.8Mb	2.4Kb
(d)	**1.8Kb**	3.8Kb	4.8Mb	2.4Kb

error final *Er*. En este artículo, con el objetivo de asegurar que la diferencia de radios tenga una mayor relevancia en *Er*, los valores de η y μ son elegidos como 0.05 y 0.1 respectivamente. Con estos valores, $Er < 1$ cuando la diferencia máxima tolerada en la distancia entre radios es de 10 píxeles, mientras que la diferencia máxima en la ubicación del centro debe ser de 20 píxeles. Tomando en consideración estos límites, se define a la relación de éxito (RE) como el porcentaje de veces en las cuales un detector circular permite detectar el círculo correspondiente, con un *Er* menor a uno.

En este artículo, también fueron usados en la comparación otros índices de desempeño los cuales reflejan los requerimientos computacionales de los algoritmos cotejados, tales como la cantidad de memoria usada (*MU*) y el número de iteraciones (*IN*).

La figura 7 presenta el conjunto de imágenes que fueron usadas en este análisis y la detección obtenida por los diferentes algoritmos usados en la comparación. La Tabla 2 muestra la relación de éxito (RE) en porcentaje y el promedio de error del resultado (*Er*) considerando las imágenes de prueba que se muestran en la Figura 5. Igualmente, la Tabla 3 representa el número de

ISSN 1405-5546
DOI: 10.13053/CyS-17-4-2013-007

Tabla 4. Valores-*p* producido por la prueba de Wilcoxon en la comparación del índice *Er*

Imagen	Valores-*p*		
	BA vs. GA	**BA vs. BFOA**	**BA vs. RHT**
(a)	1.46E-04	1.86E-04	1.97E-04
(b)	1.13E-04	1.22E-04	2.51E-04
(c)	8.92E-05	1.17E-04	9.98E-05
(d)	8.92E-05	1.77E-04	1.56E-04

iteraciones (IN) y la memoria usada (MU), sobre el mismo conjunto de imágenes. Los índices de desempeño representan los promedios tomando en cuenta 35 ejecuciones diferentes por algoritmo. Los mejores resultados presentados en las Tablas 2 y 3 se muestran de manera resaltada usando otra tipografía. Una inspección de los resultados muestra que el método propuesto es capaz de lograr un alto porcentaje de RE manteniendo un pequeño *Er* así como también requiriendo un bajo número de iteraciones para la mayoría de los casos.

Una prueba estadística no paramétrica llamada Wilcoxon [29-31] se ha llevado a cabo con el 5% de significación sobre los valores de Er reportados en la Tabla 2. En la Tabla 4 se reporta los valores-p producidos por la prueba Wilcoxon con respecto a la comparación entre pares formado tres grupos. Un grupo corresponde al algoritmo BA contra GA, otro corresponde con el algoritmo BA contra BFOA el ultimo corresponde al algoritmo BA contra RHT. En la prueba, la hipótesis nula asume que no existe diferencia significante entre los valores analizados de los tres grupos. Todos los valores-p reportados en la Tabla 4 (son menores que 0.05 (5%)) muestran evidencia que la hipótesis nula no se cumple, indicando estadísticamente el mejor desempeño promedio del algoritmo de BA.

4 Estimación de movimiento

Para estimar movimiento considerando un enfoque de comparación de bloques (CB), la imagen actual I_t de una secuencia de imágenes es dividida en varios macro-bloques no sobrepuestos de *N*x*N* píxeles. Para cada marco-bloque en la imagen actual, se determina dentro

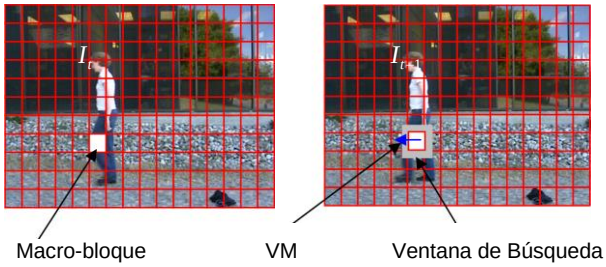


Fig. 8. Método de comparación de bloques (CB)

de una ventana de búsqueda de tamaño $(2W+1)\text{x}(2W+1)$ su mejor similitud en la imagen anterior I_{t-1}, donde *W* es el desplazamiento máximo permitido. Las diferencia de la posición entre el macro-bloque en la imagen actual y la posición de mejor similitud en la imagen anterior es llamado el vector de movimiento (VM) (ver Fig 8).

El criterio de semejanza (función objetivo) comúnmente usado en algoritmos de CB es la Suma de las Diferencias Absolutas (SAD), la cual está definida como la diferencia entre los elementos del macro-bloque de la imagen actual y los del bloque candidato en la imagen anterior I_{t-1}.

$$\text{SAD}(\hat{u},\hat{v}) = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1} |I_t(x+i,y+j) - I_{t-1}(x+\hat{u}+i,y+\hat{v}+j)| \quad (10)$$

De esta manera el VM (*u*,*v*) es definido como sigue:

$$(u,v) = \arg \min_{(u,v)\in S} \text{SAD}(\hat{u},\hat{v}) \quad (11)$$

donde $S = \{(\hat{u},\hat{v}) \,|\, -W \le \hat{u},\hat{v} \le W$ y $(x+\hat{u}, y+\hat{v})$ es una posición valida de un píxel en $I_{t-1}\}$.

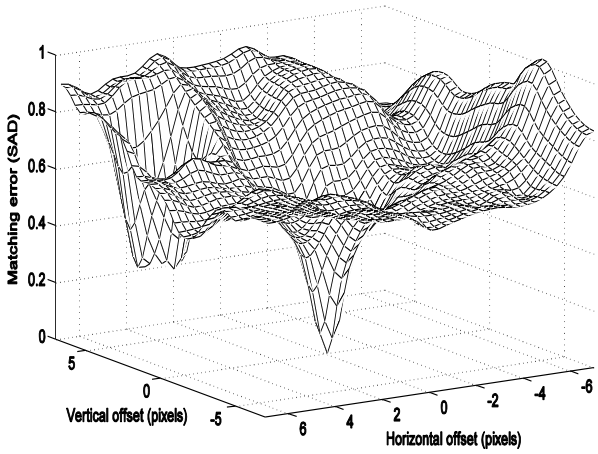


Fig. 9. Superficie común de error multi-modal con múltiples mínimos

Fig. 10. Procedimiento de búsqueda del algoritmo de BA. Un conjunto de siete armonías son generadas aleatoriamente en el espacio de búsqueda *S*. Una de ellas se encuentra en el punto central [0,0]

La búsqueda exhaustiva (FSA) es la forma más robusta y precisa para encontrar el VM. En FSA se prueban todos los puntos dentro del área de búsqueda para encontrar la posición con el mínimo SAD. Considerando el máximo desplazamiento de *W*, el FSA requiere $(2W+1)^2$ puntos de búsqueda. De esta manera los requerimientos computacionales hacen difícil su implementación en aplicaciones prácticas. Para superar el problema del costo computacional, varios algoritmos de CB han sido desarrollados [22,23,25]. La mayoría de estos algoritmos se basan en el supuesto de que la función de error (valor SAD de todos los puntos de la ventana de búsqueda) es uni-modal, además de comportarse monótonamente. Sin embargo muchos movimientos (especialmente los rápidos) generan superficies multi-modales que hacen que los errores producidos por tales enfoques sean considerables. La Figura 9 muestra una superficie de error típica de un movimiento rápido.

4.1 BA en comparación de bloques

Con el objetivo de estimar movimiento (mínimo global de la función de error), el algoritmo de BA realiza la búsqueda de manera no uniforme en puntos localizados en la ventana de búsqueda. La manera en que se generan nuevos puntos de búsqueda depende del desempeño (valor SAD) de los puntos ya calculados y al mapeo producido por los operadores definidos por BA.

De esta manera, cada armonía *I* (solución) es representada como una posición $(\hat{u}_i, \hat{v}_i)$ dentro del

Tabla 5. Parámetros del algoritmo BA para CB

HMS	HMCR	PAR	*BW*	NI
7	0.75	0.35	2	6

espacio de búsqueda *S* (ventana de búsqueda) definido como:

$$S = \left\{ \hat{u}_l, \hat{v}_l \middle| -W \le \hat{u}_l, \hat{v}_l \le W \right\} \quad (12)$$

El primer paso en el algoritmo es inicializar un numero *HMS* aleatorio de posiciones iniciales. Considerando que no todos los macro-bloques experimentan movimiento de una imagen a otra, algunos de ellos tienen como vector de movimiento [0,0]. Debido a esto una de las *HMS* posiciones iniciales debe de ser [0,0]. En el segundo paso, usando los operadores definidos por BA, se modifica el conjunto de soluciones iniciales durante un determinado número de iteraciones (*NI*). Al final de este proceso la solución (armonía óptima) dentro de la ventana de búsqueda con mejor desempeño SAD es considerada como el vector de movimiento. La Figura 10 muestra el procedimiento de búsqueda del algoritmo de BA para la estimación de movimiento.

Debido a que las soluciones generadas por BA no corresponden a un patrón fijo ni considera ninguna suposición de la función error, se tiene una mayor posibilidad de encontrar el vector de movimiento adecuado.

4.2 Resultados experimentales

4.2.1 Configuración de parámetros

Los parámetros *HMS*, *HMCR*, *PAR* y *BW* que controlan el funcionamiento de BA son mostrados en la Tabla 5, los cuales han sido usados en todos los experimentos presentados en esta sección. Dichos valores fueron identificados experimentalmente después de procesar varias secuencias de imágenes.

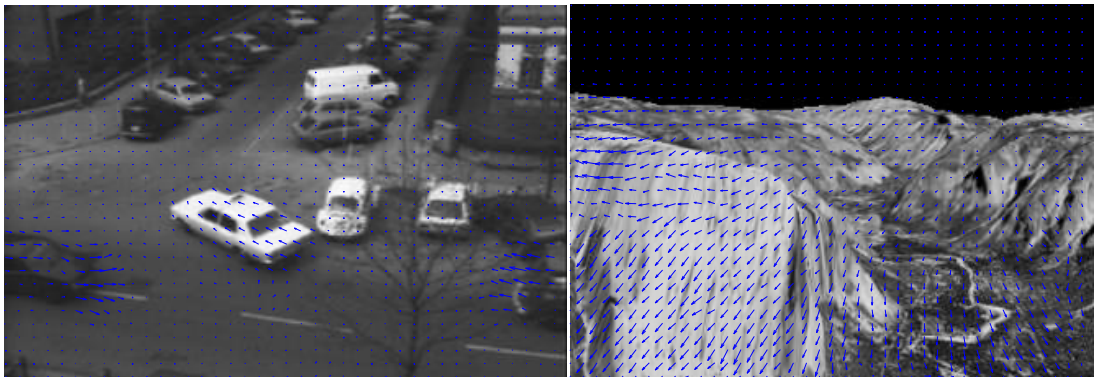
4.2.2 Estimación de movimiento

Para ilustrar la fiabilidad del enfoque de BA en la estimación de movimiento, consideramos dos secuencias de video: el taxista de Hamburgo y la Yossemite. La Figura 11 muestra los vectores de movimiento obtenidos por el algoritmo de BA en dos cuadros ejemplo. Tales resultados consideraron el uso de una ventana de búsqueda de 8x8.

4.2.3 Comparación del desempeño

Con la finalidad de analizar el rendimiento del algoritmo de BA en estimación de movimiento, se comparó su desempeño con respecto a otros algoritmos que desempeñan la misma tarea, tales como: TSS [22], DS [23] y FBMAUPR [25]. El conjunto experimental de secuencias usadas en la comparación son mostradas en la Figura 12. Tales secuencias experimentan movimientos de diferentes grados de dificultad, conteniendo imágenes de 352x240 píxeles de tamaño.

La Tabla 5 muestra los parámetros con los



(a) (b)

Fig. 11. Vectores de movimiento producidos por BA. (a) secuencia del taxista de Hamburgo, (b) secuencia Yossemite

Fig 12. Secuencias usadas en la comparación: (a) “salesman”, (b)”foreman”, (c) “akiyo” y (d) “susie”

que se configuró el algoritmo de BA para la comparación. Para el resto de los algoritmos, los valores de los parámetros fueron configurados tal y como son sugeridos en sus respectivas referencias. En el caso particular de FBMAUPR, el parámetro más importante *r*, fue considerado como 0.98. Dicha configuración es considerada de acuerdo a [25] como la mejor posible. Para todos los experimentos fue considerado macro-bloques de dimensión 15x15.

En la comparación fueron considerados dos índices de desempeño como lo son: la relación de señal a ruido (peak-signal-to-noise-ratio, PSNR) y el tiempo de cómputo invertido, los cuales son considerados índices comúnmente usados en la comparación de los algoritmos basados en CB [32,33].

El índice PSNR permite medir la calidad de los vectores de movimiento obtenidos como consecuencia del algoritmo de búsqueda usado, el cual es definido como:

$$\mathrm{PSNR}=10\cdot\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \qquad (13)$$

donde MSE es el error cuadrático medio entre la imagen original y la imagen estimada (macro-bloques con los vectores de movimiento). Debido a que el algoritmo FSA prueba todos los puntos dentro del área de búsqueda para encontrar la posición con el mínimo SAD, su PSNR representa el mejor valor posible (el más grande). Todos los demás métodos al utilizar técnicas que aceleran el proceso de búsqueda, no siempre encontrarán la posición con el mínimo SAD, por lo que sus valores PSNR serán menores al obtenido por FSA. De esta manera, entre mas pequeño sea el valor del PSNR, implicara un peor desempeño por parte del algoritmo en la determinación del VM correcto.

Tabla 6. Comparación del PSNR de los diferentes métodos

	Salesman	**Foreman**	**Akiyo**	**Susie**
FSA	35.40	23.34	29.07	35.96
TSS	35.14	23.05	27.58	34.90
DS	35.21	23.28	28.18	35.14
FBMAUPR	**35.31**	**3.30**	**28.91**	**35.80**
BA	35.25	23.31	28.14	33.16

Tabla 7. Tiempo computacional (en segundos) de los diferentes métodos

	Salesman	**Foreman**	**Akiyo**	**Susie**
FSA	16.33	16.29	16.27	16.32
TSS	0.50	0.50	0.50	0.50
DS	**0.21**	**0.22**	**0.26**	**0.25**
FBMAUPR	0.57	0.72	1.26	0.75
BA	0.32	0.36	0.40	0.38

Las tablas 6 y 7 presentan los resultados de PSNR y tiempo computacional obtenidos por los diferentes algoritmos al ser aplicados sobre las secuencias de prueba. En ambas tablas, fueron añadidos los resultados obtenidos por el método de búsqueda exhaustiva (FSA) a manera de referencia. Con el objetivo de facilitar su análisis los resultados de los algoritmos con el mejor desempeño fueron puestos en negritas.

De las tablas 6 y 7, puede notarse que el método de BA fue capaz de reducir el tiempo computacional de FBMAUPR, obteniendo un menor PSNR. Comparado con TSS el método de BA tiene mejor desempeño en términos de tiempo computacional y PSNR. Comparado con DS, el método de BA mejora en PSNR, sin

embargo presenta un incremento de tiempo computacional para las secuencias de video.

5 Conclusiones

En este artículo se explora el uso del algoritmo de Búsqueda Armónica (BA) para el procesamiento digital de imágenes. Para ello se presentan dos problemas representativos del área de procesamiento digital de imágenes, como lo son: la detección de círculos y la estimación de movimiento, los cuales son planteados desde el punto de vista de optimización.

En la detección de círculos se introdujo un algoritmo para la detección automática de formas circulares en imágenes ruidosas y complicadas, el cual no toma en consideración los principios convencionales de la transformada de Hough. Considerando este enfoque se utilizo una combinación de tres puntos borde para codificar los círculos candidatos (posibles soluciones), mientras que una función objetivo es utilizada para medir la existencia de un circulo candidato sobre la imagen de bordes. Guiado por los valores de tal función objetivo, el conjunto de círculos candidatos es modificado por medio del algoritmo de Búsqueda Armónica (BA) de manera tal que el mejor circulo candidato pueda ser ajustado dentro de la figura más circular contenida en la imagen de bordes.

Con el objetivo de medir la precisión en la detección, se propuso una función (Ec. 9) la cual permite evaluar objetivamente la disparidad entre el círculo automáticamente detectado y el aproximado por un observador humano. Evidencia experimental obtenida de tales evaluaciones demostró que el algoritmo basado en BA posee un mejor desempeño en comparación a otros algoritmos reportados en la literatura [14, 18, 19].

En la estimación de movimiento, se presentó un estimador de movimiento de comparación de bloques basado en BA. El algoritmo usando los operadores definidos por BA, modifica un conjunto de posiciones iniciales durante un determinado número de iteraciones. Al final de este proceso la posición (armonía óptima) dentro de la ventana de búsqueda con mejor valor de SAD es considerada como el vector de movimiento.

El algoritmo fue comparado con otros enfoques reportados en la literatura [22, 23, 25], considerando dos índices de desempeño: la relación de señal a ruido (peak-signal-to-noise-ratio, PSNR) y el tiempo de computo invertido. Resultados arrojados de este análisis mostraron que el método de BA presenta un desempeño competitivo en relación a los otros algoritmos, al estar solo debajo del algoritmo más preciso FBMAUPR y del más rápido DS.

Referencias

1. **Shilane, D., Martikainen, J., Dudoit, S., & Ovaska, S.J. (2008).** A general framework for statistical performance comparison of evolutionary computation algorithm. *Information Sciences*, 178(14), 2870–2879.
2. **Geem, Z.W., Kim, J.H., & Loganathan, G.V. (2001).** A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2), 60–68.
3. **Mahdavi, M., Fesanghary, M., & Damangir, E. (2007).** An improved harmony search algorithm for solving optimization problems. *Applied Mathematics Computation*, 188(2), 1567–1579.
4. **Lee, K.S. & Geem, Z.W. (2005).** A new meta-heuristic algorithm for continuous engineering optimization, harmony search theory and practice. *Computer Methods in Applied Mechanics Engineering,* 194(36-38), 3902–3933.
5. **Lee, K.S., Geem, Z.W., Lee, S.H., & Bae, K.-W. (2005).** The harmony search heuristic algorithm for discrete structural optimization. *Engineering Optimization,* 37(7), 663–684.
6. **Geem, Z.W. (2006).** Optimal cost design of water distribution networks using harmony search. *Engineering Optimization,* 38(3), 259–280.
7. **Lee, K.S. & Geem, Z.W. (2004).** A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9-10), 781–798.
8. **Ayvaz, M.T. (2007).** Simultaneous determination of aquifer parameters and zone structures with fuzzy c-means clustering and meta-heuristic harmony search algorithm. *Advances in Water Resources,* 30(11), 2326–2338.

9. **Geem, Z.W., Lee, K.S., & Park, Y. (2005).** Application of harmony search to vehicle routing. *American Journal of Applied Sciences,* 2(12), 1552–1557.
10. **Costa, L.F. & Cesar, R.M. (2001).** *Shape analysis and classification: theory and practice*. Boca Raton, FL.: CRC Press.
11. **Muammar, H. & Nixon, M. (1989).** Approaches to extending the Hough transform. *International Conference on Acoustics, Speech and Signal Processing (ICASSP-89),* Glasgow, Scotland, 3, 1556–1559.
12. **Atherton, T.J. & Kerbyson, D.J. (1993).** Using phase to represent radius in the coherent circle Hough transform. *IEE colloquium on Hough transforms.* London, England, 5/1–5/4.
13. **Shaked, D., Yaron, O., & Kiryati, N. (1994).** Deriving stopping rules for the probabilistic Hough transform by sequential analysis. *12th IAPR International Conference on Pattern Recognition*, Jerusalem, Israel, 2, 229–234.
14. **Xu, L., Oja, E., & Kultanen, P. (1990).** A new curve detection method: randomized Hough transform (RHT). *Pattern Recognition Letters,* 11(5), 331–338.
15. **Han, J.H., Koczy, L.T., & Poston, T. (1993).** Fuzzy Hough transform. *2nd IEEE international Conference on Fuzzy Systems*, San Francisco, CA, 2, 803–808.
16. **Becker, J.M., Grousson, S., & Coltuc, D. (2002).** From Hough Transforms to Integral Geometry. *2002 IEEE international geoscience and remote sensing symposium (IGARSS'02)*, Toronto, Canada, 3, 1444–1446.
17. **Cuevas, E., Oliva, D., Zaldivar, D., Pérez, M., & Rojas, R. (2013).** Circle Detection Algorithm Based on Electromagnetism-Like Optimization. *Handbook of Optimization, Intelligent Systems Reference Library*, 38, 907–934. DOI: 10.1007/978-3-642-30504-7.
18. **Ayala-Ramirez, V., Garcia-Capulin, C.H., Perez-Garcia, A., & Sanchez-Yanez, R.E. (2006).** Circle detection on images using genetic algorithms. *Pattern Recognition Letters,* 27(6), 652–657.
19. **Dasgupta, S., Das, S., Biswas, A., & Abraham, A. (2010).** Automatic circle detection on digital images whit an adaptive bacterial foraging algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 14(11), 1151–1164. doi:10.1007/s00500-009-0508-z.
20. **Barron, J.L., Fleet, D.J., & Beauchemin, S.S. (1994).** Systems and experiment performance of optical flow techniques, *International Journal of Computer Vision,* 12(1), 43–77.
21. **Saha, A., Mukherjee, J., & Sural, S. (2008).** New pixel-decimation patterns for block matching in motion estimation. *Signal Processing: Image Communication*, 23(10), 725–738.
22. **Jong, H.-M., Chen, L.-G., & Chiueh, T.-D. (1994).** Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding. *IEEE Transactions on Circuits and Systems for Video Technology,* 4(1), 88–90.
23. **Zhu, S. & Ma, K.-K. (2000).** A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. *IEEE Transactions on Image Processing*, 9(2), 287–290.
24. **Nie, Y., & Ma, K.K. (2002).** Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation. *IEEE Transactions on Image Processing*, 11(12), 1442–1449.
25. **Yi-Ching, L., Jim, Z.C.L., & Zuu-Chang, H. (2009).** Fast block matching using prediction and rejection criteria. *Signal Processing*, 89(6), 1115–1120.
26. **Saha, A., Mukherjee, J., & Sural, S. (2011).** A neighborhood elimination approach for block matching in motion estimation. *Signal Processing: Image Communication,* 26(8-9), 438–454. doi:10.1016/j.image.2011.06.002.
27. **Yadav, P., Kumar, R., Panda, S.K., & Chang, C.S. (2012).** An Intelligent Tuned Harmony Search Algorithm for Optimisation. *Information Sciences*, 196, 47–72.
28. **Bresenham, J.E. (1987).** A Linear Algorithm for Incremental Digital Display of Circular Arcs. *Communications of the ACM*, 20(2), 100–106.
29. **Wilcoxon, F. (1945).** Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6), 80–83.
30. **Garcia, S., Molina, D., Lozano, M., & Herrera, F. (2008).** A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special session on real parameter optimization. *Journal of*

Heuristic, 15(6), 617–644. doi:10.1007/s10732-008-9080-4.

31. **Santamaría, J., Cordón, O., Damas, S., García-Torres, J.M., & Quirin, A. (2008).** Performance Evaluation of Memetic Approaches in 3D Reconstruction of Forensic Objects. *Soft Computing*, 13(8), 883–904. DOI: 10.1007/s00500-008-0351-7.
32. **Saha, A., Mukherjee, J., & Sural, S. (2008).** New pixel-decimation patterns for block matching in motion estimation. *Signal Processing: Image Communication*, 23(10), 725–738.
33. **Yi-Ching, L., Jim, Z.C.L., & Zuu-Chang, H. (2009).** Fast block matching using prediction and rejection criteria. *Signal Processing*, 89(6), 1115–1120.

Erik Cuevas obtuvo en 1995 su título de Ingeniero en Comunicaciones y Electrónica en la Universidad de Guadalajara, México. Después en el 2000 recibió en grado de Maestría en Electrónica Industrial en ITESO y finalmente en el 2006 el grado de doctor en Ciencias en la Universidad Libre de Berlín en Alemania. Desde el 2007, trabaja como investigador en el Departamento de Electrónica en la Universidad de Guadalajara en México, sus áreas de investigación incluyen la visión computacional e inteligencia artificial.

Noé Ortega Sánchez recibió en 2011 el grado de Maestría en Ciencias en Ingeniería en Electrónica y Computación por la Universidad de Guadalajara. Desde el 2010, es profesor asistente en el Departamento de Ciencias computacionales de la Universidad de Guadalajara. Sus áreas de interés son la visión por computadora y los algoritmos meta-heurísticos.

Article received on 01/10/2011, accepted on 26/11/2012.