Comparación de Algoritmos Evolutivos Aplicados al Registro de Imágenes

Autores
Fernando Alberto Ruiz Valle
Dr. Héctor Fernando Gómez García
Agradecimientos: Ing. Otakar Molnar López

Contacto
faruizv@hotmail.com

Cancún, Quintana Roo a 23 de Noviembre de 2009

Resumen
En este trabajo, se hace una comparación experimental entre cuatro algoritmos de búsqueda al ser aplicados al problema de Registro de Imágenes. Estos algoritmos demuestran ser una muy buena opción cuando se requiere hacer búsquedas muy precisas en espacios de solución complejos. Los resultados obtenidos con las pruebas realizadas muestran cual de los cuatro algoritmos funcionan mejor para este problema y también describen el desempeño de cada uno, demostrando que los cuatro pueden ser utilizados para resolver problemas de este estilo. Tres de ellos se clasifican en el grupo de los Algoritmos Evolutivos y uno en el de los Algoritmos Bioinspirados.

Palabras Clave: Registro de Imágenes, Minimización, Métodos de Búsqueda, Algoritmos Evolutivos, Algoritmos Bioinspirados.

1 Introducción

El Registro de Imágenes es una tarea fundamental para el procesamiento de imágenes y se utiliza para emparejar, o alinear, dos o más imágenes de una misma escena que fueron tomadas a diferentes tiempos, utilizando diferentes sensores o desde distintos puntos de vista.

El campo de aplicación del problema de Registro de Imágenes es muy variado y se consideran tres grandes grupos:

- Visión por Computadora y Reconocimiento de Patrones – aplicaciones como segmentación, reconocimiento de objetos, reconstrucción de figuras, etc.
- Análisis de Imágenes Medicas – aplicaciones en diagnostico medico, en donde podemos encontrar detección de tumores, clasificación de imágenes microscópicas, etc.
- Procesamiento de Datos Sensados de manera Remota – aplicado para fines militares y de ingeniería civil, y en ramas de la geografía como la geología, oceanografía, y en actividades económicas como explotación de minerales

El problema de Registro es un problema de Optimización porque se pretende explorar un espacio de búsquedas para llegar a una solución que mejor resuelva el problema, por lo tanto es necesario utilizar algún método que haga estas búsquedas exhaustivas lo mas precisas posibles, para entregar un resultado lo mas correcto posible.

Para la optimización hay muchas opciones, como los métodos convencionales basados en gradiente que se dirigen a la solución mas próxima siguiendo la dirección en que aumenta o disminuye el gradiente, pero sin la capacidad de discernir entre una solución local o global, los

métodos basados en Algoritmos Evolutivos, que utilizan operadores de evolución y se van acercando a la solución rápidamente, pero que consumen muchos recursos computacionales y también están los métodos basados en Algoritmos Bioinspirados, que son capaces de converger a una solución de manera muy efectiva, pero que también son grandes consumidores de recursos.

Tratar el problema de Registro de Imágenes por la vía de los Algoritmos Evolutivos resulta interesante porque representa una alternativa al uso de métodos convencionales que no ha sido muy explorada, y además es una ayuda para quien esta interesado en utilizarlos como opción para optimizar.

El objetivo de esta investigación es hacer un análisis de cuatro de los Algoritmos Evolutivos más importantes y probarlos dentro del proceso de registro de imágenes, para analizar su desempeño y determinar cual brinda un mejor rendimiento.

Para probar el rendimiento de cada uno de los algoritmos y sacar conclusiones de cual funciona mejor, es necesario poner a todos los algoritmos a trabajar en ambientes similares, de modo que los resultados que entreguen puedan ser contrastados. La manera de establecer un ambiente similar es ajustando los valores de entrada de los algoritmos, como tamaño de población, numero de generaciones/iteraciones y probabilidades de mutación y recombinación en valores iguales para cada algoritmo, haciendo igual numero de pruebas para cada uno: 10, 20, 50 y 100 corridas.

2 Descripción del Problema

Para hacer un Registro de Imágenes utilizando a los Algoritmos Evolutivos como Método de Búsqueda es necesario comprender primero a que se refiere el Registro de Imágenes, cual es la metodología que se debe seguir y que elementos intervienen, para poder así decidir correctamente en que momento implementarlos y cual es la mejor estrategia para ello.

2.1 Registro de Imágenes

Como ya se ha explicado, el Registro de Imágenes es un mapeo tanto espacial como con respecto a intensidades entre dos imágenes. Suponiendo $\boldsymbol{I_1}$ e $\boldsymbol{I_2}$ como Imagen de Referencia e Imagen Objetivo, respectivamente, donde tanto $\mathbf{I_1}$ como $\mathbf{I_2}$ son dos arreglos bidimensionales de

Lo anterior arroja las siguientes preguntas de investigación:

- ¿Cuál de los cuatro algoritmos (Algoritmos Genéticos, Estrategias Evolutivas, Evolución Diferencial o Particle Swarm) puede resolver mejor el problema de registro? ¿Cuál es el menos recomendable?
- ¿Qué ventajas arroja el uso de Algoritmos Evolutivos en el registro de imágenes sobre el uso de otros métodos de optimización basados en otras técnicas?

La implementación de los algoritmos mencionados en ambientes controlados y similares para cada uno, arroja resultados bastante interesantes.

Los algoritmos mas recientes han demostrado ser más novedosos que los anteriores tanto al momento de implementarlos como al momento de utilizarlos, ya que demuestran una gran simplicidad para programarlos y poca complejidad para utilizarlos pero entregan mejores resultados que sus ancestros.

En las siguientes secciones de este articulo se puede encontrar una descripción completa del problema al que nos enfrentamos [Sección 2], una descripción mas formal sobre la implementación de los Algoritmos Evolutivos y como contribuyen al registro de imágenes [Sección 3], un análisis completo de los resultados obtenidos luego de un proceso de prueba de cada uno de estos algoritmos [sección 4] y algunas conclusiones con respecto a la implementación de los Algoritmos Evolutivos en el registro de imágenes, algunas aportaciones y el trabajo a futuro [sección 5].

tamaño determinado ($\mathbf{I_1}$(x,y) e $\mathbf{I_2}$(x,y)), podemos expresar el mapeo entre esas dos imágenes de la siguiente manera:

$$\boldsymbol{I_2 = g\left(I_1(f(x,y))\right)},$$

Donde f es la transformación espacial y g es la transformación de la intensidad

Por lo general, una transformación de intensidad no es necesaria, salvo los casos en los que ocurra un cambio en el sensor con que se obtienen las imágenes. La clave en un problema de registro es **hallar la transformación espacial que pueda alinear las imágenes.**

En general, el Registro de Imágenes se reduce a:

1 Debe seleccionarse la clase de transformación que se aplicara a la (as) imagen (es) de modo que sea capaz de alinear (las) correctamente con la imagen objetivo.
2 Un espacio de rasgos, acompañado de una medida de similitud, deben establecerse, que son menos susceptibles a las distorsiones que puedan estar presentes en la imagen, y más probables a llegar a coincidir.
3 Se seleccionan los métodos de búsqueda que guiaran la búsqueda hacia la mejor coincidencia

2.1.1 Transformaciones

Las transformaciones empleadas para alinear imágenes se clasifican en dos grandes grupos:

- Una transformación **global** está dada por una simple ecuación que se encarga de mapear la imagen completa
- Las transformaciones **locales** mapean de diferente manera a las imágenes, dependiendo de su localización espacial.

Además, una transformación es ***afín*** cuando esta compuesta por las operaciones básicas de escalamiento, rotación y traslación en un plano cartesiano. Este tipo de transformación consta de cuatro parámetros: t_x, t_y, s, θ, que mapean un punto (x_1, y_1) de la imagen de referencia a un punto (x_2, y_2) de la imagen objetivo. Esta dada por la formula

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s * \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

Una transformación ***proyectiva*** es la que mapea una coordenada en el plano (x_p, y_p) a una coordenada en la imagen (x_i, y_i). Este tipo de transformación es necesaria cuando la escena se compone de un plano realzado con respecto al plano de la imagen. Este proceso denominado rectificación esta dado por la formula

$$y_i = \frac{a_{21}x_p + a_{22}y_p + a_{23}}{a_{31}x_p + a_{32}y_p + a_{33}}$$

$$x_i = \frac{a_{11}x_p + a_{12}y_p + a_{13}}{a_{31}x_p + a_{32}y_p + a_{33}}$$

Finalmente, una transformación ***perspectiva*** es la que hace un mapeo de una escena en tres dimensiones (3D) a una imagen en dos dimensiones (2D). Conociendo las coordenadas de los objetos situados en la escena (x_0,y_0,z_0) los puntos correspondientes (x_i, y_i) en la imagen están dados por

$$x_i = \frac{-fx_0}{z_0 - f} \qquad y_i = \frac{-fy_0}{z_0 - f}$$

Donde *f* es la posición del centro del lente de la cámara.

Un nuevo método de transformación que consiste en utilizar ***Splines cúbicos*** para transformar los pixeles de una imagen es modificar la imagen (x, y) mediante un vector de transformación (u, v) que esta determinado por ***B-Splines Cúbicos***, de modo que la imagen transformada I_2 esta dada por $I_2(x, y) = I_1(x, y) + \big(u(x, y), v(x, y)\big)$. El vector de transformación (u, v) esta dado por

$$(u, v) = \sum_{ij} \beta_{ij}^{u} \cdot S(x - x_i, y - y_j)$$

$$(u, v) = \sum_{ij} \beta_{ij}^{v} \cdot S(x - x_i, y - y_j)$$

Los cuales definen al campo de transformación.

2.1.2 Espacio de Rasgos y Medida de Similitud

Un Espacio de Rasgos es la representación de toda la información que será utilizada para hacer el registro. La elección que hagamos en el espacio de rasgos va a determinar que compararemos. Un espacio de rasgos puede estar compuesto por la imagen completa.

El espacio de rasgos es un aspecto fundamental en casi todas las tareas de visión por computadora, y su aplicación afecta a lo siguiente:

- A que propiedades del sensor o de la escena son sensibles los datos
- A que propiedades de la imagen compararemos
- Al costo computacional, ya sea reduciendo el espacio de búsqueda, o aumentando el número de operaciones de computo necesarias

Luego de seleccionar un espacio de rasgos, es necesario establecer una medida de similitud, que se encarga de medir la similitud entre los rasgos de las imágenes obtenidos en el espacio de rasgos.

Entre las medidas de similitud más usadas comúnmente cuando se utilizan rasgos se encuentra la correlación cruzada (cross-correlation), suma de diferencias absolutas y algunas propiedades de invariabilidad de Fourier como la correlación de fase. Cuando se usan curvas o superficies

como espacio de rasgos, se requiere usar medidas como suma de los cuadrados de las diferencias entre puntos cercanos.

Una medida de similitud y un espacio de rasgos bien determinados pueden ayudar a reducir los efectos del ruido en el registro, aunque estos efectos pueden ser eliminados también al momento de establecer el espacio de rasgos en un solo procedimiento previo al proceso de comparación. Establecer un espacio de rasgos correctamente ayuda a disminuir el tamaño del espacio de búsqueda.

2.1.3 Espacio de Búsqueda

El espacio de búsqueda es el espacio de todas las posibles transformaciones. Debido al gran costo computacional que representan las medidas de similitud y los rasgos a comparar, es necesario establecer un espacio de búsqueda y una estrategia de búsqueda. Este paso complementa al método de registro que estemos utilizando y además ayuda a delimitar el problema y encerrar al algoritmo en un espacio de transformaciones definido.

El tamaño y complejidad del espacio de búsqueda están determinados por el hecho de que el método de registro admita o no transformaciones globales o locales.

Los métodos globales consisten, por lo general, en una búsqueda de una transformación que maximice a una medida de similitud, o una búsqueda por los parámetros de dicha transformación. En este caso, el espacio de búsqueda se puede reducir sustancialmente al permitir transformaciones más generales.

En cuanto a los métodos locales, estos se vuelven más complejos, ya que permiten transformaciones muy generales, pero con cualquier cantidad de grados de libertad, haciendo más complejo el espacio de búsqueda.

3 Métodos de Búsqueda: Algoritmos Evolutivos

Un método de búsqueda sirve para explorar el Espacio de Búsqueda definido previamente para encontrar la transformación que mejor pueda alinear las imágenes que estamos registrando. Para esta tarea existen muchas técnicas basadas en gradientes, como ***Descenso de Gradiente, Quasi-Newton, Descenso de Gradiente Estocástico***, etc.

De acuerdo con [2], estas técnicas avanzan paso a paso en la dirección del gradiente inverso de la función objetivo. Estas técnicas hacen búsquedas lineales intentando minimizar la función objetivo, pero su desventaja es que puede ser necesario hacer muchas evaluaciones innecesarias a la función objetivo, o que el algoritmo se estanque en mínimos locales y no se llegue al mínimo global en ningún momento.

Dentro del campo de la Inteligencia Artificial (AI), encontramos un campo que involucra a los problemas de optimización combinatoria. Dentro de los algoritmos que se usan para resolver la optimización combinatoria, encontramos los de Computo Evolutivo, que están basados en progresos evolutivos, tales como crecimiento o desarrollo en una población. Estos algoritmos están inspirados en mecanismos biológicos de evolución [3].

El cómputo evolutivo se basa en los principios Darwinianos para la solución de problemas automatizada y se comenzó a utilizar en los años 50, por Lawrence J. Fogel en Estados Unidos y John Henry Holland desarrollo un método al que llamo Algoritmos Genéticos (GA). En Alemania, Ingo Rechenberg y Hans-Paul Schwefel llamaron a su método, Estrategias Evolutivas (ES). A partir de los años 90, se unifican estas técnicas y se les comienza a llamar Computación Evolutiva, y más adelante, con el surgimiento de nuevas técnicas como Evolución Diferencial (DE) y Particle Swarm Optimization (PSO), se le llamo Programación Genética.

Todos estos algoritmos implementan mecanismos inspirados en evolución biológica, tales como reproducción, mutación, recombinación, selección natural y supervivencia del más apto. Siendo la recombinación (cruza) y la mutación los pilares para el funcionamiento de estos algoritmos, ya que estos son los que brindan la diversidad a los individuos de cada generación.

3.1 Particle-Swarm Optimization - PSO

Matemáticamente, como se menciono antes, un problema de optimización involucra a una función de aptitud que describa al problema, dentro de un conjunto de restricciones que definen el espacio de búsqueda para dicho problema.

Hacia mediados de los años 90, Eberhart y Kennedy enunciaron una alternativa para el problema de optimización no lineal al emular el comportamiento

colectivo de parvadas de pájaros, partículas, el método de Craig Reynolds y la socio cognición, para dar pie a un método llamado Particle Swarm Optimization, en español se puede llamar Optimización por Enjambre de Partículas.

El algoritmo de Optimización por Enjambre de Partículas (PSO) esta inspirado en motivos sociológicos y biológicos, y puede encargarse de la optimización en superficies difíciles, discontinuas y multimodales.

3.1.1 Funcionamiento

En un principio, se crea una población, o enjambre, llamado **S**, compuesto por vectores con posiciones aleatorias $\overrightarrow{x_i}$, cada una con velocidades aleatorias $\overrightarrow{v_i}$. Posteriormente, se establece una relación de vecindad **N** para la población, la cual determina si dos partículas cualesquiera P_i y P_j son vecinos o no. Para cualquier particula **P**, es posible establecer una vecindad con un vector **N(P)** que contenga a todos los vecinos de esa partícula. En el PSO clásico, se estila que **N=S** para cada partícula.

A las variables de posición $\vec{x}(t)$ y velocidad $\vec{v}(t)$ de cada particula se les agrega una pequeña variable $\vec{p}(t)$ que funciona como una memoria en donde se almacena la mejor posición que ha visitado dicha partícula. Cuando se establece una relación de vecindad $N(P) = S$, $\vec{g}(t)$ representara a la partícula globalmente mejor en todo el enjambre.

Los parámetros para el algoritmo son los siguientes:

- Velocidad máxima (V_{max}) que restringe a $\overrightarrow{V_i}(t)$ en un intervalo de [-V_{max} , V_{max})
- Factor de peso ω
- Números aleatorios φ_1 y φ_2 que tienen influencia en las formulas de actualización de la velocidad para $\vec{p}(t)$ y $\vec{g}(t)$ y que oscila entre [0,1]
- Multiplicadores constantes C_1 y C_2 que respectivamente representan valores de confianza para cada partícula y para la población en general

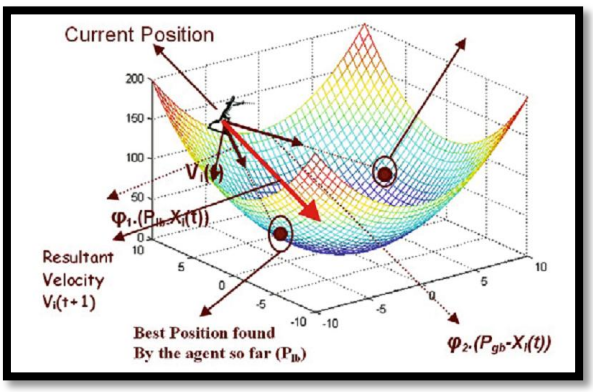

Fig 1. Componentes del Algoritmo

Como se explico antes, los valores iniciales de cada partícula es cero, por lo tanto, $\vec{p}(t)$ y $\vec{g}(t)$ tambien valen cero. Una vez que se inicializan las variables, comienza un proceso iterativo de optimización, en donde las posiciones y las velocidades de todas las partículas se modifican con las siguientes ecuaciones:

- $V_{id}(t+1) = \omega \cdot v_{id}(t) + C_1 \cdot \varphi_1 \cdot \left(P_{id}(t) - x_{id}(t)\right) + C_2 \cdot \varphi_2 \cdot \left(g_{id}(t) - x_{id}(t)\right)$
- $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$

En la ecuación de actualización de la velocidad, $\omega \cdot v_{id}(t)$ representa la velocidad incercial de la particula. Habiendo calculado las velocidades y posiciones para t+1, se completa la primer iteración en el algoritmo y el proceso se repite hasta llegar a uno de los siguientes criterios de paro:

- Se alcanza un cierto numero de iteraciones
- Se encuentra una solución lo suficientemente aceptable

Se alcanza un límite de uso de CPU

3.2 Evolución Diferencial - DE

Por el tiempo en que Eberhart y Kennedy propusieron su PSO, Price y Storn decidieron reemplazar los operadores clásicos de cruza y mutación presentes en Algoritmos Genéticos, por unos operadores diferenciales. Este nuevo tipo de operadores diferenciales seria usado para crear nuevos hijos a partir de los cromosomas de los padres, por lo cual en 1995, se nombro Evolución Diferencial (DE) a este nuevo algoritmo.

3.2.1 Funcionamiento

Como cualquier otro Algoritmo Evolutivo, DE comienza con una población de **N** vectores de búsqueda. Denotados como

$$\overrightarrow{Xi}(t) = [x_{i,1}(t), x_{i,2}(t), \dots, x_{i,D}(t)]$$

Al inicio del problema, los parámetros y variables independientes se inicializan en su correspondiente rango numérico. Una vez inicializados, para cambiar un miembro de la generación $\overrightarrow{Xi}(t)$, es necesario crear un **vector donador** $\overrightarrow{Vi}(t)$.

La estrategia de mutación más común se llama DE/rand/1. Esta estrategia consiste en crear otros tres vectores que servirán como parámetro para determinar el vector donador para cada miembro de la población. Estos vectores se escogen aleatoriamente entre la población. Después, se elige un numero escalar **F** que mide la diferencia escalar entre cualquiera de los dos vectores seleccionados y se le suma al tercer vector. De este modo se obtiene el vector donador para un individuo, mediante esta ecuación

$$v_{i,j}(t+1) = x_{r1,j}(t) + F \cdot (x_{r2,j}(t) - x_{r3,j}(t))$$

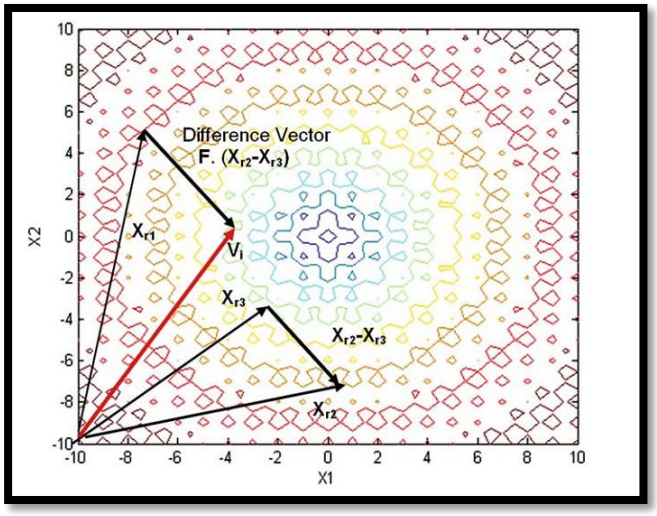

Fig 2. Vector Resultante

Una vez hecha la mutación, para elevar la diversidad de la población, es necesario utilizar métodos de recombinación. En Evolución Diferencial se pueden usar dos tipos de recombinación: **exponencial** y **binomial**. Como en los demás algoritmos evolutivos, se hace un intercambio de genes. En este caso, el vector donador hace un intercambio de sus partes con el individuo $\overrightarrow{Xi}(t)$.

En el caso de la recombinación exponencial, primero se selecciona aleatoriamente un entero ***n*** entre un rango de [0, D-1]. Este entero actúa como punto de partida para el vector objetivo, desde donde comienza el proceso de recombinación con el vector donador. Además, se selecciona otro entero ***L*** entre un rango [1, D], que determina el numero de componentes que el vector donador aporta a la recombinación. El vector recombinado resultante esta dado por

$$\overrightarrow{Ui}(t) = [u_{i,1}(t), u_{i,2}(t), \dots, u_{i,D}(t)]$$

De este modo, por cada individuo, o vector de prueba $\overrightarrow{Xi}(t)$, se crea un hijo $\overrightarrow{Ui}(t)$. Para mantener el tamaño de la población constante, es necesario invocar el proceso de selección natural para determinar que individuos sobreviven para la nueva generación. En Evolucion Diferencial se emplea el principio Darwiniano de la "supervivencia del mas apto" [DAK]. Este proceso esta dado por

$$\overrightarrow{Xi}(t+1) = \overrightarrow{Ui}(t) \quad \text{si} \quad f\left(\overrightarrow{Ui}(t)\right) \leq f\left(\overrightarrow{Xi}(t)\right), \text{ o}$$

$$= \overrightarrow{Xi}(t) \quad \text{si} \quad f\left(\overrightarrow{Xi}(t)\right) < f\left(\overrightarrow{Ui}(t)\right).$$

En este punto, la población puede que mejore o puede que se mantenga igual, pero nunca se deteriora.

3.3 Trabajo Realizado

Habiendo definido el problema que se desea atacar, ahora sigue construir las herramientas con las que guiaremos la búsqueda del mejor alineamiento de las imágenes, es decir, los Algoritmos Evolutivos. El ambiente de programación utilizado para esto es Microsoft Visual Studio 2008, usando el lenguaje de programación C Sharp (C#) por la gran facilidad que brinda cuando queremos usar apuntadores, pero no nos queremos meter directamente con ellos.

Después, hay que construir una herramienta con la que se hagan transformaciones de imágenes, suficientemente poderosa como para hacer transformaciones de gran complejidad y con muchos grados de libertad. En este caso, las transformaciones se hacen mediante B-Splines Cúbicos programados en C++.

Finalmente, hay que poner ambas herramientas a trabajar en conjunto para registrar dos imágenes, la imagen fuente I_1 y la imagen objetivo I_2.

3.3.1 Ambiente de Pruebas

El rendimiento y el buen funcionamiento de los Algoritmos Evolutivos será medido sometiendo a pruebas similares a cada uno. Como se menciono antes, los algoritmos que se utilizaran son: Algoritmos Genéticos, Estrategias Evolutivas y Evolución Diferencial por parte de los Algoritmos Evolutivos, y Particle Swarm Optimization por parte de los Algoritmos Bioinspirados.

Los Algoritmos Evolutivos evalúan una función objetivo y hacen búsquedas sobre un espacio de búsquedas que dicha función pueda generar, dependiendo de las características de la misma.

El ambiente de pruebas esta definido de la siguiente manera:

- Funciones Objetivo
  - Problema de Ackley
  - Función de Maximización $f(x) = \frac{1}{\sum_{i=0}^{n} x_i + 1}$, n = no. dimensiones
  - Función de Minimización $f(x) = \sum_{i=0}^{n} x^2{}_i$, n = no. dimensiones
- Tamaño de Población: 10, 20, 50 y 100 individuos
- Grados de Libertad (Dimensiones) de la función: 2, 4 y 8
- Numero de Generaciones: 100
- Repeticiones: 100

Los puntos a evaluar para cada algoritmo:

- Tiempo de Ejecución
- Valor de aptitud promedio
- Cercanía con el optimo global (cero)

Integrar los Algoritmos Evolutivos con la herramienta de transformación de imágenes sirve para determinar la función objetivo de los algoritmos, ya que el espacio de búsqueda esta determinado por todas las posibles transformaciones que puede sufrir el individuo, que en este caso, cada uno de ellos es una transformación de la imagen fuente (mediante B-Splines Cúbicos). El valor de la aptitud de cada individuo estará determinado mediante la evaluación de la medida de similitud entre ambas imágenes: la imagen transformada I_T y la imagen objetivo I_2.

La medida de similitud es una función que mide el error, o las diferencias en intensidad existentes entre las dos imágenes. Cuanto menor sea el error, mejor es la solución entregada, siendo ***cero*** el valor que se pretende alcanzar, es decir, el optimo global. La medida de similitud esta dada por la función

$$\mathrm{E}_{(x)} = \sum |I(x_T) - I(x_2)|$$

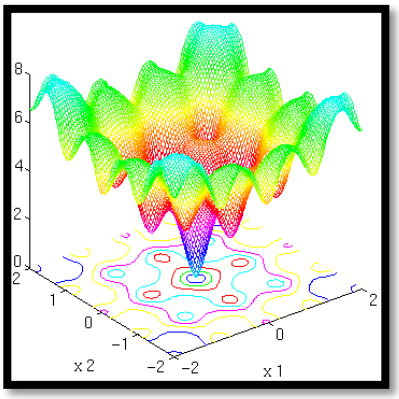

Fig 3. Función de Ackley

4 Análisis de Resultados

4.1 Algoritmos Evolutivos

Las pruebas realizadas a los algoritmos evolutivos son satisfactorias. Los Algoritmos evolutivos muestran un desempeño aceptable al trabajar con las funciones dadas, entregando resultados en tiempos de ejecución no muy grandes y muestran que se acercan mucho al óptimo global antes de completar el número de generaciones establecido.

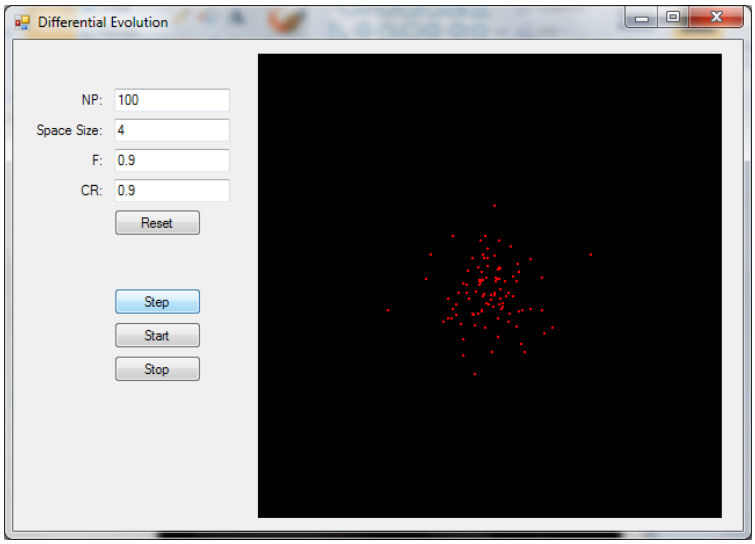

Fig 4. Algoritmo de Evolución Diferencial

Los resultados más sobresalientes los refleja la función de Ackley evaluada mediante los algoritmos PSO y Evolución Diferencial. Ambos muestran que no importa la dimensionalidad de la función, sin duda llegan a acercarse muchísimo al mínimo global. Los valores de aptitud son muy cercanos al cero y la separación que muestran con respecto al mínimo global es casi despreciable, incluso para los casos con mayor dimensionalidad.

En la tabla 1 se muestran los resultados para la evaluación de la función de Ackley con los algoritmos PSO y DE para una población de 100 individuos.

Tamaño de Población	Función Aptitud	Algoritmo	Dimensionalidad	Tiempo de Ejecución Promedio	Valor de Aptitud Promedio	Respecto al Mínimo
100	**Ackley**	Differential Evolution	2	29.03	6.18858E-07	6.18858E-07
			3	30.43	5.45581E-07	5.45581E-07
			4	33.57	5.72903E-07	5.72903E-07
			8	29.14	5.94865E-07	5.94865E-07
100	**Ackley**	Particle Swarm	2	58.8	1.16859E-06	1.16859E-06
			3	73.26	2.92835E-06	2.92835E-06
			4	89	5.12575E-06	5.12575E-06
			8	155.38	0.000457322	0.000457322

Tabla 1. Desempeño de PSO y DE - función de Ackley

Los Algoritmos Evolutivos y las Estrategias Evolutivas también muestran desempeños aceptables para las mismas condiciones, sin embargo son bastante pobres si los comparamos con PSO y DE. Ambos muestran que se acercan al mínimo global, sin embargo se puede apreciar en la Tabla 2 que les toma mas tiempo de ejecución, su valor de aptitud promedio no es tan cercana a cero como PSO y DE, y además quedan muy lejos del mínimo global, en comparación con PSO y DE.

Tamaño de Población	Función Aptitud	Algoritmo	Dimensionalidad	Tiempo de Ejecución Promedio	Valor de Aptitud Promedio	Respecto al Mínimo
100	**Ackley**	Evolution Strategies	2	205.95	0.778751329	0.778751329
			3	211.5	0.913530888	0.913530888
			4	215.19	1.142457382	1.142457382
			8	245.48	1.355920744	1.355920744
100	**Ackley**	Genetic Algorithms	2	292.01	0.635536853	0.635536853
			3	387.58	2.142168369	2.142168369
			4	499.29	3.216676632	3.216676632
			8	872.95	6.38166682	6.38166682

Tabla 2. Desempeño de GA y ES - función de Ackley

4.2 Transformaciones con B-Splines Cúbicos

Hacer transformaciones con B-Splines Cúbicos permite hacer deformaciones más notorias a las imágenes, a diferencia de algoritmos que hacen transformaciones afines en donde lo único que se puede observar son rotaciones, traslaciones o escalamientos. Los B-Splines Cúbicos demostraron ser una herramienta poderosa para hacer transformaciones con muchos grados de libertad y funcionan muy bien hasta para imágenes de gran tamaño, y lo hacen tardando poco tiempo. En la fig. 5 se observa como una imagen es transformada.

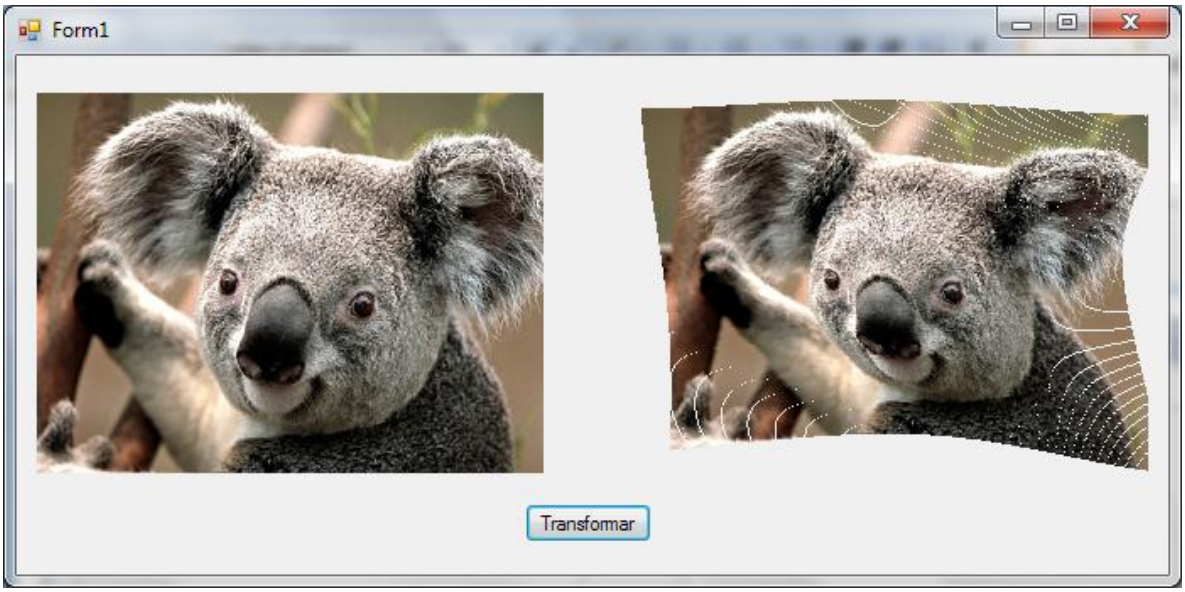

Fig 5. Transformaciones con B-Splines Cúbicos

Los B-Splines Cúbicos demuestran que pueden hacer deformaciones importantes en las imágenes y no presentan problemas si se trata de imágenes en color.

Como se puede observar en la fig. 5, las deformaciones de los Splines se hacen mediante la generalización de curvas de Bezier [4]. Es tarea de algún método de interpolación decidir como se deben dibujar los pixeles de acuerdo con los nuevos valores de transformación. Según [1], "*El problema de interpolar píxeles es bastante sencillo conceptualmente. Una imagen digital es una representación discreta formada por muestras uniformemente distribuidas en una rejilla rectangular. Cada muestra (o conjunto de muestras en imágenes multicanal o en color) es lo que identificamos como un píxel. Una interpolación de píxel es necesaria para obtener nuevos valores de píxel en coordenadas arbitrarias, no necesariamente enteras*".

El algoritmo de interpolación utilizado para las pruebas descritas anteriormente se conoce como el Vecino más cercano. Un método muy básico que lo único que hace es pintar el nuevo pixel transformado tomando información del pixel más cercano en la imagen fuente. En [1] puede encontrarse una descripción mas detallada de los métodos de interpolación de imágenes que existen.

4.3 Integración de Algoritmos Evolutivos y Transformaciones con B-Splines Cúbicos al Registro de Imágenes

Al momento de hacer esta integración, los resultados obtenidos son muy pobres. Las transformaciones de las imágenes se hacen correctamente, sin embargo, en el momento en que los Algoritmos Evolutivos, particularmente Particle Swarm Optimization, deben hacer la búsqueda del individuo (imagen transformada) entregan una transformación que esta muy lejos de parecerse a la imagen objetivo, por lo tanto, al evaluar la función objetivo y calcular el margen de error, el algoritmo obtiene cualquier valor y se queda con el valor mas cercano a cero, devolviendo como mejor solución la que menor valor de aptitud tenga, sin embargo no se acerca en lo mas mínimo al optimo esperado.

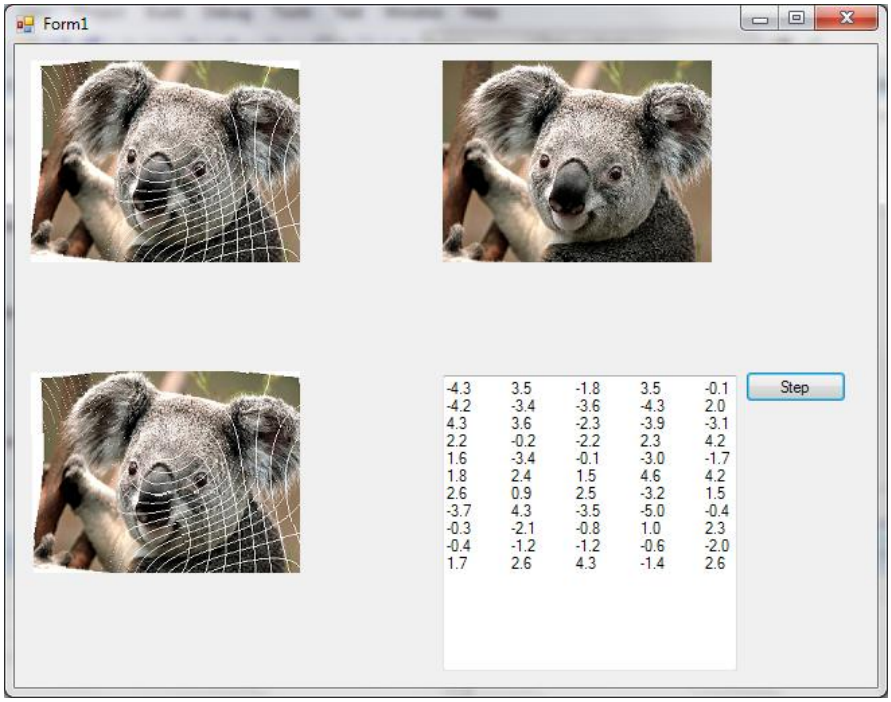

Fig 6. Minimización de Función Objetivo – PSO

Como se aprecia en la fig. 6, el resultado de la minimización de la función objetivo es bastante pobre, la transformación que el algoritmo de PSO reporta como la mas optima esta muy lejos de ser la que buscamos y el tiempo de ejecución de cada prueba oscila entre los 55 y los 70 segundos, dependiendo el espacio de búsqueda que se asigne a la función de aptitud. En la figura, la imagen de la izquierda es la imagen fuente, la de la derecha es la imagen objetivo y la ubicada en la parte inferior es el individuo con mejor aptitud en la prueba.

5 Conclusiones

Los Algoritmos Evolutivos han demostrado trabajar muy bien con funciones de minimización y maximización con muchos grados de libertad, mostrando tiempos de ejecución muy aceptables en los cuatro casos y entregando resultados muy cercanos al mínimo/optimo global, además de que no se atoran en mínimos/máximos locales, como llega a suceder con métodos de búsqueda basados en Métodos de Gradiente.

Hacer transformaciones con B-Splines Cúbicos es una buena estrategia si se busca hacer transformaciones con muchos grados de libertad. Este algoritmo trabaja bastante bien para cualquier tipo de imagen: en color, en escala de grises; imágenes pequeñas o imágenes grandes.

Cuando se intenta integrar estas dos herramientas para hacer Registro de Imágenes, claramente se observan problemas, se puede apreciar también que el problema puede estar localizado al momento de evaluar la función de aptitud o al momento de que el Algoritmo decide en que momento parar y entregar el resultado, ya que no entrega una buena solución.

El valor de este proyecto de investigación es mostrar el buen desempeño de los Algoritmos Evolutivos al enfrentarlos a problemas con muchos grados de libertad, en donde se rescata que a pesar de que los Algoritmos Genéticos y las Estrategias Evolutivas son algoritmos mas antiguos que Evolución Diferencial y Particle Swarm Optimization, funcionan muy bien para este tipo de problemas de optimización. También muestra el desempeño de dos algoritmos recientes y puede brindar un marco de referencia para alguien que este interesado en trabajar con alguno de estos algoritmos.

Otra aportación de esta investigación es mostrar otra forma de transformar imágenes utilizando una herramienta como los B-Splines Cúbicos y la gran ventaja que presentan si se quiere hacer transformaciones complejas.

Si bien la integración de estas dos herramientas no se logro como estaba planeado, queda abierta la posibilidad de retomar el trabajo realizado hasta el momento y lograr una integración adecuada.

6 Referencias Bibliográficas

[1] Nuevos Algoritmos de Interpolación: http://pixinsight.com/forum/index.php?topic=559.0

[2] [KSP], Stefan Klein, Mairus Staring, and Josein P.W. Pluim. "*Evaluation of Optimization Methods for Nonrigid Medical Image Registration Using Mutual Information and B-Splines".* 12 December 2009.

[3] Evolutionary Computation: http://en.wikipedia.org/wiki/Evolutionary_computation

[4]B-Spline Wiki: http://en.wikipedia.org/wiki/B-spline