Solution to the Registration Problem using Differential Evolution and SSD-ARC Function

Felix Calderon  
*Universidad Michoacana de San Nicolás de Hidalgo*  
*División de Estudios de Posgrado.*  
*Facultad de Ingeniería Eléctrica*  
*Santiago Tapia 403 Centro.*  
*Morelia, Michoacán, México. CP 58000*  
calderon@umich.mx

Juan J. Flores  
*Universidad Michoacana de San Nicolás de Hidalgo*  
*División de Estudios de Posgrado.*  
*Facultad de Ingeniería Eléctrica*  
*Santiago Tapia 403 Centro.*  
*Morelia, Michoacán, México. CP 58000*  
juanf@umich.mx

**Abstract**—The problem of image registration is to find the best set of parameters of an affine transformation, which applied to a given image yields the closest match to a target image (possibly with noise). We present a method to perform parametric image registration based on Differential Evolution. Besides using Differential Evolution, we propose to use an error parametric image registration function robust enough to discard misleading information contained in outliers. The results are compared to those obtained using Genetic Algorithms. It is clear that Differential Evolution outperforms Genetic Algorithms in terms of speed (number of evaluations), and quality of the solutions (accuracy). The quality of the solutions provided by Differential Evolution is so good that they do not need to be refined by gradient methods. At the end we present a general analysis and discussion about why DE converges in a better way than GA.

**Keywords**—Differential Evolution; Image Registration;

I. INTRODUCTION

The parametric image registration problem [1] consists of finding a parameter set for an Affine Transformation, which allows us to match an origin image with a target image as closely as possible. Occluded areas and noise (outliers) increase the complexity of the registration task, and some parametric registration techniques exhibit poor results under these circumstances.

Differential Evolution (DE) was developed by Storn and Price [2], [3] around 1995 as an efficient and robust meta-heuristic to optimize functions of arbitrary complexity. Like most algorithms in Evolutionary Computation, DE is a population based optimizer. Most of these methods produce new individuals, by different heuristic techniques to perturb old individuals (e.g. crossover, mutation, etc.). DE produces new individuals adding the scaled difference of two randomly selected individuals to a third one.

In this paper we propose to use DE to solve the image registration problem. Nevertheless, by using DE with the usual error function (i.e. sum of square errors), the solutions are considerably affected by noise (outliers). By using SSD-ARC function [4], the resulting transformation identification system is robust with respect to noise.

We compare our results with those produced by GA and RANSAC [5]. The results were impressive; DE produced better solutions in less time, still retaining the advantages presented by GA.

The rest of the paper is organized as follows: Section II presents the image registration problem using an affine transformation and also describes the problem of estimating a good transformation in the presence of outliers (noisy data); Section III presents the basics of Differential Evolution; Section IV presents the results, and a comparison of our proposal against GA and RANSAC; Section V presents an analysis and discussion of differences in the convergence profiles of DE and GA; Finally, Section VI presents the conclusions.

II. REGISTRATION USING AN AFFINE TRANSFORMATION

The Affine Transformation (AT) [6], [7] allows us to compute rotation, scaling, and shearing of images. An AT uses a six-parameter vector $\Theta$, and maps a pixel at position $p_i$ (with integer coordinates $[x_i, y_i]$) to a new position $\hat{p}_i$ (with real coordinates $[\hat{x}_i, \hat{y}_i]$) given by

$$
\hat{p}_i(\Theta) = \left[ \begin{array}{c} \hat{x}_i \\ \hat{y}_i \end{array} \right] = \left[ \begin{array}{cccccc} x_i & y_i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_i & y_i & 1 \end{array} \right] \Theta
$$

(1)

where $M(p_i)$ is the matrix of coordinates and $\Theta = [\theta_1...\theta_5]$ is the parameter vector. A couple of points $p_i$ and $\hat{p}_i$ are named corresponding points and using only three of them, the affine transformation $\Theta$, can be computed by means of (1).

The image registration problem, tries to find the parameter vector $\Theta$ that matches the origin image $I_1$ with a target image $I_2$. In some cases the image registration is performed using control points, which are selected as interesting points in both images. For instance, for every interesting point $p_i$ in the origin image there exists a corresponding point $\hat{p}_i$ in the target image; correspondences are computed using
proximity and similarity measures. Harley and Zisserman in [5] propose an algorithm based on RANSAC for computing the Homography which uses a set of corresponding points. Thomas and Vernon in [11] also use these control points to compute a polynomial function. Although both algorithms reach good results in the outlier cases, they present weak performance, when the control points are not enough or the correspondences in between are bad computing in case of noisy images.

In general, using more control points and color information results in a more robust Image Registration. However, in practical cases, both images are corrupted by noise and the problem is to find the best $A$ to match a transformation of $I_1$ into $I_2$. A very well known method to evaluate the match quality is to compute the Sum of Squared Differences (SSD) between the source and target images, pixel by pixel, as in Equation (2)

$$E(\Theta) = \sum_{i=1}^{N} [I_1(\hat{p}_i(\Theta)) - I_2(p_i)]^2 = \sum_{i=1}^{N} e_i(\Theta)^2 \quad (2)$$

with a difference vector image $e_i$ given by

$$e_i(\Theta) = I_1(\hat{p}_i(\Theta)) - I_2(p_i) \quad (3)$$

where $I(p_i)$ is the gray level value at pixel $p_i$. Using this error measurement, SSD, the registration image task consists of finding $\Theta^*$ that minimizes $E$ so $\Theta^*$ is the minimizer of Equation (2). This method is named Least Squared (LS); some strategies to minimize Equation (2) are presented in [8].

Using the $\rho$-function, defined by Hampel in [9], the SSD can be defined by Equation (4) with $\rho_{LS}(e_i) = e_i^2$.

$$E_{LS}(\Theta) = \sum_{i=1}^{N} \rho_{LS}(e_i(\Theta)) \quad (4)$$

The influence function is defined by Hampel, as the derivative of the $\rho$-function and it helps to see the contribution of the errors to the solution (see [9]). In the LS case, the influence function is given by (5)

$$\psi_{LS}(e_i) = 2e_i \quad (5)$$

A robust function presented in [10], is the Lorentzian Estimator (LE), which has a $\rho$-function and influence function given by Equation (6) and (7)

$$\rho_{LE}(e_i) = \log \left(1 + \frac{e_i^2}{2\sigma^2}\right) \quad (6)$$
$$\psi_{LE}(e_i) = \frac{2e_i}{2\sigma^2 + e_i^2} \quad (7)$$

note, the term $\frac{1}{2\sigma^2 + e_i^2}$ in (7), reduces the error contribution on the gradient vector and it is not present in Equation (5). This fact explains why LS is notoriously sensitive to outliers.

Salomon et all in [12] used the gray color information in whole images for medical image registration, nevertheless they use a Least Square approach, which is not robust when the image is corrupted by outliers. Falco and Cioppa in [13] propose to use the maximization of mutual information which is a robust similarity measure for satellite image registration. Our approach tries to resolve the affine image registration problem in the presence of outliers; our similarity measure is computed solving an energy equation and it is equivalent to one M-estimator [9]. This approach is quite different to mutual information, which in our opinion is one of the most robust approaches to image registration but we focus on the solution of our energy function, named SSD–ARC which is described in the following sub–section.

A. Parametric SSD–ARC

The Sum of Squared Differences with Adaptive Rest Condition for non–parametric Image Registration was presented, by Calderon et al. in [14], as the minimization of a quadratic energy function $E_{SSD–ARC}$ with a term $(1-l_i)^2$ to reduce huge error contributions. The energy function is given by Equation (8)

$$E_{SSD–ARC}(\Theta, l) = \sum_{i=1}^{N} e_i^2(\Theta)(1-l_i)^2 + \mu \sum_{i=1}^{N} l_i \quad (8)$$

where $l_i \in [0,1]$ is an outlier indicator function under the control of parameter $\mu$.

For this parametric SSD–ARC function, the term $(1-l_i)^2$ allows us to discard outliers. The second term in Equation (8) restricts the number of outliers by means of $\mu$. The minimizer $l_i^*$ for Equation (8) can be computed by solving

$$\frac{\partial E_{SSD–ARC}(\Theta, l)}{\partial l_i} = 0,$$

so the solution for $l_i^*$ is given by Equation (9). We refer to $l_i^*$ as the outlier field.

$$l_i^* = \frac{e_i^2(\Theta)}{\mu + e_i^2(\Theta)} \quad (9)$$

Replacing the value of $l_i^*$ in Equation (8), we have a new parametric SSD–ARC function $E_{SSD–ARC}(\Theta)$ given by Equation (10), which has a unimodal $\rho$–function and an influence function given by Equations (11) and (12), respectively.

$$E_{SSD–ARC}(\Theta) = \sum_{i=1}^{N} \frac{\mu e_i^2(\Theta)}{\mu + e_i^2(\Theta)} \quad (10)$$
$$\rho_{SSD–ARC}(e_i) = \frac{\mu e_i^2}{\mu + e_i^2} \quad (11)$$
$$\psi_{SSD–ARC}(e_i) = \frac{2\mu^2 e_i}{(\mu + e_i^2)^2} \quad (12)$$

Note, the parametric SSD–ARC influence function exhibits a behavior similar to the Lorentzian Estimator influence function.
function. In both functions, large differences give derivatives values close to zero, as you can see in Equations (12) and (7), respectively. The parametric SSD–ARC influence function has a maximum value located at \( e = \sqrt{\mu / 3} \) and values greater than \( 2e \) will have a derivative value close to zero. Nevertheless, there is not a gradient-based algorithm capable to reach the minimum if the initial value gives an error greater than \( 2e \). For this reason, a minimization method in two steps for the parametric SSD–ARC error function was proposed by Calderon et al. in [4]. In the first step a minimization using GA was proposed and in the second step a minimization based on Gradient algorithm was used. Nevertheless in the following section we will describe our experiments in which DE present more accuracy than GA for the SSD-ARC function making the refinement by gradient algorithms unnecessary.

III. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) was developed by Storn and Price [2], [3] around 1995 as an efficient and robust meta-heuristic to optimize functions of arbitrary complexity. Like most algorithms in Evolutionary Computation, DE is a population–based optimizer. Most of these methods produce new individuals, by different heuristic techniques, as perturbations of old ones (e.g. crossover, mutation, etc.). DE produces new individuals adding the scaled difference of two randomly-selected individuals to a third one.

DE maintains two vector populations, containing \( N_{\text{pop}} \times N_{\text{par}} \) real-valued parameters. Population \( \Theta^{(g)} \) contains \( N_{\text{pop}} \) vectors for each generation \( g \), where the \( k \)th individual in \( \Theta^{(g,k)} \) has \( N_{\text{par}} \) parameters. Population \( V^{(g)} \) are mutant vectors produced from \( \Theta^{(g)} \). Each vector in the current population is recombined with a mutant to produce a trial population, \( U^{(g)} \). The trial population is stored in the same array as the mutant population, so only two arrays are needed.

The initial population is generated on a searching space \( \Omega = \{[\omega_0^{\min}, \omega_0^{\max}], \ldots, [\omega_5^{\min}, \omega_5^{\max}]\} \) by random numbers following a uniform probability distribution, with a (possibly) different range for each dimension. Discrete and integral variables are represented by reals and then interpreted in the right way. The Initial population is computed by Equation (13)

\[
\theta_j^{(g,k)} = (\omega_j^{\max} - \omega_j^{\min}) \alpha + \omega_j^{\min} \\
\forall j \in [1, N_{\text{par}}]
\]

where \( \alpha \) is a real random number in \([0, 1]\) and \( [\omega_j^{\min}, \omega_j^{\max}] \) represents the \( \Omega \) space boundaries at the \( j \)th dimension.

Differential mutation adds a scaled randomly chosen, vector difference to a third as you can see in Equation (14)

\[
V^{(g,k)} = \Theta^{(g,r_0)} + F(\Theta^{(g,r_1)} - \Theta^{(g,r_2)}) \\
\forall k \in [1, N_{\text{pop}}]
\]

where \( F \in [0, 1] \). The difference vector indices, \( r_1 \) and \( r_2 \), and the index \( r_0 \) are chosen randomly in \([1, N_{\text{pop}}]\). In our experiments the index \( r_0 \) was chosen equal to the \( k \)th population member, this condition allow us to move a member of the population in some random direction; a desirable condition.

To complement the differential mutation strategy, DE uses uniform crossover, also known as discrete recombination. Crossover takes place according to Equation (15)

\[
v_j^{(g,k)} = \begin{cases} 
\nu_j^{(g,k)} & \text{if } \text{rand}(0,1) \leq C_r \text{ or } j = j_{\text{rand}} \\
\theta_j^{(g,k)} & \text{otherwise}
\end{cases} \\
\forall < j, k > \in [1, N_{\text{par}}] \times [1, N_{\text{pop}}]
\]

where \( C_r \in [0, 1] \) is a user defined parameter that controls the proportion of components copied from the mutant onto the trial vector. \( v_j^{(g,k)} \) is the \( j \)th component of the \( i \)th individual of the \( g \)th generation of population \( V^{(g)} \).

If the trial vector \( U_j^{(g,k)} \) has an equal or lower fitness value than its target vector \( \theta_j^{(g,k)} \), it replaces the target vector in the next generation (16).

\[
\Theta^{(g+1,k)} = \begin{cases} 
U_j^{(g,k)} & \text{if } E(U^{(g,k)}) \leq E(\Theta^{(g,k)}) \\
\Theta^{(g,k)} & \text{otherwise}
\end{cases} \\
k \in [1, N_{\text{pop}}]
\]

The complete algorithm for Image registration using Equation (10) using DE is shown in Algorithm 1.

IV. IMAGE REGISTRATION RESULTS

In order to compare DE and GA we develop two experiments using the Lena and Cameraman Images. For Lena’s image an affine transformation given by \( \Theta = [0.9396, -0.3420, 3, 0.3420, 0.9396, 3] \) was applied and 15% of the image was erased (at the image right) in order to introduce outliers. The origin and target images are shown in figures 1(a) and 1(b). In the cameraman case an affine transformation given by \( \Theta = [1.3, 0, 0, 1.3, 0] \) was applied to the original image and the information outside a circle with radius 98 and center at the image coordinate [98, 98] was erased. The origin and target images for cameraman are shown in Figures 1(c) and 1(d). The image couples 1(a) and 1(b) and 1(c) and 1(d) were used for testing the behavior of the DE and GA algorithms with a population size of 100 and 400 respectively and a search space given by \( \omega_0 \in [0.5, 1.5], \omega_1 \in [-0.5, 0.5], \omega_2 \in [-10, 10], \omega_3 \in [-0.5, 0.5], \omega_4 \in [0.5, 1.5], \omega_5 \in [-10, 10] \). For the GA the Initial population was fixed in 400 and it is reduced to 200 at first generation. Only 100 of the best members are allowed to generate offspring which replaces the 100 of the worst elements with 4% of mutation, see [15] for details, so the evaluation number of the error function is at the begin 400 and 108 at each generation. These parameters for DE
Algorithm 1 DE-Registration

Input: $I_1, I_2, \Omega$
Output: $\Theta^{(\ast, \ast)}, l$

Set $g = 0$, $\mu = 1000$, $N_{pop} = 100$, $F = 0.1$, $C_r = 0.5$
1. Compute a initial population $\Theta^{(g)}$ by Equation (13), compute the Fitness
   for each population member $\Theta^{(g,k)}$ by Equation (10)
2. Mutation. Compute $V^{(g)}$ by Equation (14) doing $r_0$ equal to $k$th population member
3. Crossover. Compute the Population $U^{(g)}$ by Equation (15)
4. Select the best Population member applying Equation (16)
5. Set $g = g + 1$ and repeat steps 2, 3 and 4 until the population reaches convergence.
6. For the best population member $\Theta^{(\ast, \ast)}$, compute $e_i(\Theta^{(\ast, \ast)})$ using Equation (3), and then, the outliers field $l_i$
   by Equation (9)

and GA were selected in order to have the same evaluation numbers per generation of the error function. In all cases
DE and GA algorithms were stopped when the generation number was greater than 10,000 or when the Square-Norm
of the difference between the known transformation vector and the solution was smaller than 0.1.

![Origin](image1.png) ![Target](image2.png)

Figure 1. Origin and Target images for Lena and Cameraman

Table I shows the average of 30 executions for DE and GA
using the origin and target images of Lena and Cameraman.
That table includes, besides the image identification and the
algorithm used in the solution, the number of generations
required to reach the solution, the number of evaluations of
the fitness function, the evaluation fitness per generation,
execution time in seconds and the Square–norm of the
difference between the parameter vector computed and the
reference. GA for this experiment did not reach the solution
in 10,000 generations using more time than DE, while DE
converged in an average of 923 generation for the Lena
image and 1,332 generation for the Cameraman image. In
general, GA needs more time than DE to yield similar
accuracy for these experiments.

Table II shows the parameters computed by DE and GA.
That table includes, for each image, the algorithm used in
the solution, the square-norm of the difference, the vector
parameter, and a reference to the corresponding difference
image. The best and worst executions are shown for DE,
GA, and also one o the best solution reach for RANSAC
which is available in [4]. In all cases the vector parameter
closest to the vector reference was the solution computed
by DE. The images in Figure 2 show the square differences
(at the pixel level) between the target image and the images
computed by the final parameter vector in Table II. In those
figures, white means an error close to zero and black means
a high error. Notice that the worst results for DE are better
than the best results for GA.

V. CONVERGENCE ANALYSIS

To illustrate the concept, we have run a minimization
experiment in two dimensions, so that the populations and
their perturbation distributions could be plotted into intelligi-
ble figures. The objective function used for the experiment
was similar to the error function used for the registration
process, with the difference that we composed it to provide
two unequal minima, and make the convergence conditions
more obvious in the experiments. The objective function is
shown in Equation (17), and its plot in Figure 3. Function
$h$ produces a single well on a quasi-flat surface; function
$f$ composes two wells, displaced at $(-11, -9)$, and$(11,3)$.
Note that the $f$ axis has been elongated to show where
the wells lie. Seen at the same scale as the $x$ and $y$, the
objective function is pretty flat. We chose this function to
make the optimization process harder. In fact, any gradient-
based optimization method is in trouble if it starts in the flat
part of the function.
Table I

| Image    | Algorithm | Number of Generations | Number of Evaluations | Eval. per Gen. | time (sec.) | $|\Theta^* - \hat{\Theta}|$ |
|----------|-----------|-----------------------|-----------------------|----------------|-------------|----------------|
| Lena     | DE        | 923.80                | 92,380.00             | 100            | 394.91      | 0.0449        |
|          | GA        | 10,000.00             | 108,400.00            | 108            | 4,024.64    | 67.2149       |
| Cameraman| DE        | 1,332.38              | 133,237.93            | 100            | 565.20      | 0.0504        |
|          | GA        | 10,000.00             | 108,400.00            | 108            | 3,720.03    | 41.7701       |

(a) best DE (b) worst DE (c) best GA (d) worst GA (e) RANSAC

Premature convergence takes place due to the selection mechanism used by GA. Two parents mate and produce two offsprings. Those two offsprings will be inside the neighborhood defined by the parents\(^1\). If an offspring is better than another individual in the population, the other individual will be replaced by the offspring, reducing the population’s coverage, thus reducing the population’s genetic diversity. This effect takes place over and over again, until the whole population consists of \(n\) clones of the best individual, which generally does not lie at the global optimum.

On the other hand, DE presents a different selection mechanism. An individual moves to a new prospect location only if the new location is better than the old one. This strategy guides the search while preserving the genetic richness of the population.

Most evolutionary computation metaheuristics share two features, they are population-based, and their populations evolve due to the effect of perturbations imposed to their individuals. For instance, GA evolve due to perturbations to the individuals in the population produced by crossover and mutation; in PSO, particles evolve following social and

\[^1\]this situation occurs in both mating schemes: multipoint and real crossover.

\(h(x, y) = \frac{x^2 + y^2}{0.001 + x^2 + y^2} \)  
\(f(x, y) = h(x + 11, y + 9) + 1.1h(x - 11, y - 3) \)

In the search process, GA converges faster than DE, but less accurately. DE, on the other hand, proceeds more slowly towards the global optimum. DE converges to a better result faster, while GA already got stuck at a local optimum or “near” the global optimum. This behavior is known as premature convergence.
cognitive knowledge; in DE, vectors evolve according to perturbations resulting of vector differences.

DE’s perturbation distribution is based on differences; in early generations, those differences range from large to small. In later generations, when the population is converging the range of those differences is smaller. This fact makes the perturbations become smaller and smaller, as the population converges, relieving DE from the need of adjusting or evolving parameters, as in other metaheuristics (e.g. Simulated Annealing or Evolutionary Strategies, etc). On the other hand, GA’s perturbations are distributed not as closely to the origin, yielding coarser grained movements to the individuals. This fact decreases the probability of an offspring or mutant to be located at or near the global optimum.

Figure 4 shows the perturbation distribution differences for generation 0, 100 and 2000. In this figure one can see the convergence process for DE in case of minimizing the Equation (17). Note how the difference will form clustering and near to generation 2000 the population will have differences distribution close to zero; it means a reduction in the population size, which increase the accuracy for a reduced searching space, an automatic condition for DE, which does not have GA.

Consider an individual on any given contour level near an optimum. Mutation in GA selects some of the components of an individual to perform mutation on those particular genes; for the two-dimensional case of our examples, it can either move only one of the coordinates of the individual undergoing mutation. The effect is that all possible results lie in a cross centered at the mutated individual. If the optimum lies outside the mutation cross, and the population reached a point of premature convergence, where the population does not enclose the optimum, the optimum is very hard to reach. Figure 5 shows several contour lines for the objective function (Eq. 17); the optimum is at the center of the contour lines. The GA population has already converged to a single point; the DE population still preserves some genetic diversity (both populations were withdrawn from generation 2000). Vertical and horizontal lines show all possible results for mutations for the GA population. If the optimum is outside their area of influence, the exploratory power of GA is no useful at all. For mutation to reach the goal, or lie within a neighborhood close enough to the optimum, it would have to displace it vertically, and then horizontally; possibly a repeated number of times.

Given that the ranges of x and y are both 40, assuming a uniform probability distribution, the probability to lie within a tolerance of $10^{-5}$ for one variable is $25 \times 10^{-8}$. Thus, the probability of both variables to lie within that tolerance is $6.25 \times 10^{-18}$. GA would take approximately $0.16 \times 10^{18}$ mutations to hit the optimum within that accuracy range. Considering a point mutation rate of 5%, GA would need approximately $6.4 \times 10^{13}$ calls to the mutation operator to hit the optimum within that accuracy range. A luxury we cannot afford.

GA converges to local optima under the same conditions that DE finds global optima most of the time. As a result, the

| Algorithm | $|\Theta^{(*)} - \Theta|_2^2$ | Parameters computed $\Theta^{(*)}$ | Figure |
|-----------|----------------|----------------|-------|
| DE        | 0.0006         | 0.9368 -0.3407 3.0164 0.3470 0.9359 2.9822 | 2(a) |
| GA        | 0.2145         | 0.9366 -0.3419 3.3515 0.3433 0.9405 2.6985 | 2(b) |
| RANSAC    | 0.2890         | 0.9425 -0.3440 2.9706 0.3457 0.9411 2.4632 | 2(c) |

Table II

Comparison between DE’s and GA’s Final Parameters for best and worst computing parameters

For Lena Image

| Algorithm | $|\Theta^{(*)} - \Theta|_2^2$ | Parameters computed $\Theta^{(*)}$ | Figure |
|-----------|----------------|----------------|-------|
| DE        | 0.0088         | 1.3023 -0.0016 -0.0196 0.0022 1.2977 -0.0204 | 2(d) |
| GA        | 0.6703         | 1.3116 -0.0040 -0.6844 0.0029 1.3020 -0.4491 | 2(e) |
| RANSAC    | 0.1667         | 1.2985 -0.0028 0.2242 -0.0004 1.2948 0.3411 | 2(f) |

For Cameraman Image

| Algorithm | $|\Theta^{(*)} - \Theta|_2^2$ | Parameters computed $\Theta^{(*)}$ | Figure |
|-----------|----------------|----------------|-------|
| DE        | 0.0984         | 1.3029 0.0006 -0.3136 -0.0004 1.3004 0.0054 | 2(g) |
| GA        | 89.2889        | 1.4108 -0.0197 -9.3312 -0.1324 1.4390 -1.4712 | 2(h) |
| RANSAC    | 0.1967         | 1.2985 -0.0028 0.2242 -0.0004 1.2948 0.3411 | 2(i) |
Figure 5. Premature convergence of GA. Crossover will not yield any new information, and mutation cannot reach the optimum.

The quality of DE’s solutions is better than those of GA’s. GA always needs a final push using a gradient-based method.

Figure 6(a) shows the behavior of the error function in the generation course until the algorithm convergences. As you can note GA convergences more quickly than DE, nevertheless Figure 6(b) shows, a detail of Figure 6(a), in which DE convergences more slowly and finally gets closer to the global minimum than GA. Accuracy is very desirable, in general, for optimization problems; the final error and norm of the difference are shown in Table III. In this table one can see that DE’s norm of the difference is smaller than GA’s and the final error for DE and GA are comparable.

![Error function](image1)

![Detail last generation](image2)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Gen</th>
<th>Pop</th>
<th>Error function</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED</td>
<td>2000</td>
<td>50</td>
<td>0.999992497964</td>
<td>1.81973842851E-10</td>
</tr>
<tr>
<td>GA</td>
<td>2000</td>
<td>50</td>
<td>1.000389959971</td>
<td>5.9672719421E-4</td>
</tr>
</tbody>
</table>

VI. CONCLUSIONS

A solution for the Image Registration problem using Differential Evolution was presented. The error function SSD–ARC was selected because it is robust in the presence of noise and outliers. Another reason to use an evolutionary optimizer is because this error function cannot be minimized using a gradient-based method if the initial value is placed in the flat zone of the error function. The experiments performances show that stochastic strategies as GA and RANSAC do not have the accuracy presented by DE. DE’s accuracy is enough for a registration image and we consider that other additional strategies are unnecessary. Additionally, in all experiments DE converge in a tenth of the time required by GA, exhibiting an accuracy five or more orders of magnitude than GA’s solution. This increase in the solution’s quality, provided by DE is due to the premature convergence present in GA, whose solution is very often maintained at a local minimum.

ACKNOWLEDGMENT

The authors would like to thank Universidad Michoacana for sponsor this research.

REFERENCES


