Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 874761, 20 pages
doi:10.1155/2012/874761

Research Article

Multithreshold Segmentation Based on Artificial Immune Systems

Erik Cuevas,[1] Valentin Osuna-Enciso,[2] Daniel Zaldivar,[1] Marco Pérez-Cisneros,[1] and Humberto Sossa[2]

[1] *Departamento de Ciencias Computacionales, CUCEI, Universidad de Guadalajara, Avenida Revolución 1500, 44430 Guadalajara, JAL, Mexico*
[2] *Centro de Investigación en Computación-IPN, Avenida Juan de Dios Bátiz s/n, Colonia Nueva Industrial Vallejo, 07738 Mexico, DF, Mexico*

Correspondence should be addressed to Erik Cuevas, erik.cuevas@cucei.udg.mx

Received 19 February 2012; Accepted 18 April 2012

Academic Editor: Yi-Chung Hu

Bio-inspired computing has lately demonstrated its usefulness with remarkable contributions to shape detection, optimization, and classification in pattern recognition. Similarly, multithreshold selection has become a critical step for image analysis and computer vision sparking considerable efforts to design an optimal multi-threshold estimator. This paper presents an algorithm for multi-threshold segmentation which is based on the artificial immune systems(AIS) technique, also known as theclonal selection algorithm (CSA). It follows the clonal selection principle (CSP) from the human immune system which basically generates a response according to the relationship between antigens (Ag), that is, patterns to be recognized and antibodies (Ab), that is, possible solutions. In our approach, the 1D histogram of one image is approximated through a Gaussian mixture model whose parameters are calculated through CSA. Each Gaussian function represents a pixel class and therefore a thresholding point. Unlike the expectation-maximization (EM) algorithm, the CSA-based method shows a fast convergence and a low sensitivity to initial conditions. Remarkably, it also improves complex time-consuming computations commonly required by gradient-based methods. Experimental evidence demonstrates a successful automatic multi-threshold selection based on CSA, comparing its performance to the aforementioned well-known algorithms.

1. Introduction

Several image-processing applications aim to detect and classify relevant features which may be later analyzed to perform several high-level tasks. In particular, image segmentation seeks to group pixels within meaningful regions. Commonly, gray levels belonging to the object are substantially different from those featuring the background. Thresholding is thus a simple but effective tool to isolate objects of interest; its applications include several classics such as document image analysis, whose goal is to extract printed characters [1, 2], logos, graphical content, or musical scores; also it is used for map processing which aims to locate lines, legends, and characters [3]. Moreover, it is employed for scene processing, seeking for object detection, marking [4], and for quality inspection of materials [5, 6].

Thresholding selection techniques can be classified into two categories: bilevel and multilevel. In the former, one limit value is chosen to segment an image into two classes: one representing the object and the other one segmenting the background. When distinct objects are depicted within a given scene, multiple threshold values have to be selected for proper segmentation, which is commonly called multilevel thresholding.

A variety of thresholding approaches have been proposed for image segmentation, including conventional methods [7–10] and intelligent techniques [11, 12]. Extending the segmentation algorithms to a multilevel approach may cause some inconveniences: (i) they may have no systematic or analytic solution when the number of classes to be detected increases, and (ii) they may also show a slow convergence and/or high computational cost [11].

In this work, the segmentation algorithm is based on a parametric model holding a probability density function of gray levels which groups a mixture of several Gaussian density functions (Gaussian mixture). Mixtures represent a flexible method of statistical modelling as they are employed in a wide variety of contexts [13]. Gaussian mixture has received considerable attention in the development of segmentation algorithms despite its performance is influenced by the shape of the image histogram and the accuracy of the estimated model parameters [14]. The associated parameters can be calculated considering the expectation-maximization (EM) algorithm [15, 16] or gradient-based methods such as Levenberg-Marquardt, LM [17]. However, EM algorithms are very sensitive to the choice of the initial values [18]; meanwhile, gradient-based methods are computationally expensive and may easily get stuck within local minima [14]. Therefore, some researchers have attempted to develop methods based on modern global optimization algorithms such as the learning automata (LA) [19] and differential evolution algorithm [20]. In this paper, an alternative approach using a bio-inspired optimization algorithm for determining the parameters of a Gaussian mixture is presented.

On the other hand, biological inspired methods can successfully be transferred into novel computational paradigms as shown by the successful development of artificial neural networks, evolutionary algorithms, swarming algorithms, and so on. The human immune system (HIS) is a highly evolved, parallel, and distributed adaptive system [21] that exhibits remarkable abilities that can be imported into important aspects in the field of computation. This emerging field is known as artificial immune system (AIS) [22] which is a computational system fully inspired by the immunology theory and its functions, including principles and models. AISs have recently reached considerable research interest from different communities [23], focusing on several aspects of optimization, pattern recognition, abnormality detection, data analysis, and machine learning. Artificial immune optimization has been successfully applied to tackle numerous challenging optimization problems with remarkable performance in comparison to other classical techniques [24].

Clonal selection algorithm (CSA) [25] is one of the most widely employed AIS approaches. The CSA is a relatively novel evolutionary optimization algorithm which has been built on the basis of the clonal selection principle (CSP) [26] of HIS. The CSP explains the immune response when an antigenic pattern is recognized by a given antibody.

In the clonal selection algorithm, the antigen (Ag) represents the problem to be optimized and its constraints, while the antibodies (Abs) are the candidate solutions of the problem. The antibody-antigen affinity indicates as well the matching between the solution and the problem. The algorithm performs the selection of antibodies based on affinity either by matching against an antigen pattern or by evaluating the pattern via an objective function. In mathematical grounds, CSA has the ability of getting out of local minima while simultaneously operating over a pool of points within the search space. It does not use the derivatives or any of its related information as it employs probabilistic transition rules instead of deterministic ones. Despite its simple and straightforward implementation, it has been extensively employed in the literature for solving several kinds of challenging engineering problems [27–29].

In this paper, the segmentation process is considered as an optimization problem approximating the 1D histogram of a given image by means of a Gaussian mixture model. The operation parameters are calculated through the CSA. Each Gaussian contained within the histogram represents a pixel class and therefore belongs to the thresholding points. In order to compare the segmentation results with other optimization methods, the number of elements in the Gaussian mixture (classes) is considered already known or given by the user. The experimental results, presented in this work, demonstrate that CSA exhibits fast convergence, relative low computational cost, and no sensitivity to initial conditions by keeping an acceptable segmentation of the image, that is, a better mixture approximation in comparison to the EM- or gradient-based algorithms.

This paper organizes as follows. Section 2 presents the method following the Gaussian approximation of the histogram. Section 3 provides information about the CSA while Section 4 demonstrates the automatic threshold determination. Section 5 discusses some implementation details. Experimental results for the proposed approach are presented in Section 6, followed by the discussion summarized in Section 7.

2. Gaussian Approximation

Let consider an image holding L gray levels $[0,\ldots,L-1]$ whose distribution is displayed within a histogram $h(g)$. In order to simplify the description, the histogram is normalized just as a probability distribution function, yielding

$$\begin{gathered} h(g)=\frac{n_g}{N}, \quad h(g)>0, \\ N=\sum_{g=0}^{L-1} n_g, \qquad \sum_{g=0}^{L-1} h(g)=1, \end{gathered} \tag{2.1}$$

where n_g denotes the number of pixels with gray level g and N being the total number of pixels in the image.

The histogram function can thus be contained into a mix of Gaussian probability functions of the form

$$p(x)=\sum_{i=1}^{K} P_i \cdot p_i(x)=\sum_{i=1}^{K} \frac{P_i}{\sqrt{2\pi}\sigma_i} \exp\left[\frac{-(x-\mu_i)^2}{2\sigma_i^2}\right], \tag{2.2}$$

with P_i being the probability of class i, $p_i(x)$ being the probability distribution function of gray-level random variable x in class i, μ_i and σ_i being the mean and standard deviation of the ith probability distribution function, and K being the number of classes within the image. In addition, the constraint $\sum_{i=1}^{K} P_i = 1$ must be satisfied.

The mean square error is used to estimate the $3K$ parameters P_i, μ_i, and σ_i; $i = 1, \ldots, K$. For instance, the mean square error between the Gaussian mixture $p(x_i)$ and the experimental histogram function $h(x_i)$ is now defined as follows:

$$J = \frac{1}{n}\sum_{j=1}^{n}\left[p(x_j) - h(x_j)\right]^2 + \omega \cdot \left|\left(\sum_{i=1}^{K} P_i\right) - 1\right|. \tag{2.3}$$

Assuming an n-point histogram as in [13] and ω being the penalty associated with the constrain $\sum_{i=1}^{K} P_i = 1$. In general, the parameter estimation that minimizes the square error produced by the Gaussian mixture is not a simple problem. A straightforward method is to consider the partial derivatives of the error function to zero by obtaining a set of simultaneous transcendental equations [13]. However, an analytical solution is not always available considering the nonlinear nature of the equation which in turn yields the use of iterative approaches such as gradient-based or maximum likelihood estimation. Unfortunately, such methods may also get easily stuck within local minima.

In the case of other algorithms such as the EM algorithm and the gradient-based methods, the new parameter point lies within a neighbourhood distance of the previous parameter point. However, this is not the case for the CSA which is based on stochastic principles. New operating points are thus determined by a parameter probability function that may yield points lying far away from previous operating points, providing the algorithm with a higher ability to locate and pursue a global minimum.

This paper aims to compare its segmentation results to other optimization methods that have been applied to similar segmentation tasks. Therefore, the number of elements in the Gaussian mixture (classes) is considered as already known or provided by the user. The number of classes, in most cases, represents the number of objects which are contained in the image. However, if the selected number is lower than the object number, it can be assumed that results actually favour the classification of bigger objects yet bearing the expense of ignoring small subjects.

3. Clonal Selection Algorithm

In natural immune systems, only the antibodies (Abs) which are able to recognize the intrusive antigens (nonself cells) are to be selected to proliferate by cloning [21]. Therefore, the fundament of the clonal optimization method is that only capable Abs will proliferate. Particularly, the underlying principles of the CSA are borrowed from the CSP as follows:

(i) maintenance of memory cells which are functionally disconnected from repertoire,

(ii) selection and cloning of most stimulated Abs,

(iii) suppression of nonstimulated cells,

(iv) affinity maturation and reselection of clones showing the highest affinities,

(v) mutation rate proportional to Abs affinities.

From immunology concepts, an antigen is any substance that forces the immune system to produce antibodies against it. Regarding the CSA systems, the antigen concept refers to the pending optimization problem which focuses on circle detection. In CSA, B cells, T cells, and antigen-specific lymphocytes are generally called antibodies. An antibody is a representation of a candidate solution for an antigen, for example, the prototype circle in this work. A selective mechanism guarantees that those antibodies (solutions) that better recognize the antigen and therefore may elicit the response are to be selected holding long life spans. Therefore, such cells are to be named memory cells ($\mathbf{M}$).

3.1. Definitions

In order to describe the CSA, the notation includes boldfaced capital letters indicating matrices and boldfaced small letters indicating vectors. Some relevant concepts are also revisited below:

(i) antigen: the problem to be optimized and its constraints (circle detection),

(ii) antibody: the candidate solutions of the problem (circle candidates),

(iii) affinity: the objective function measurement for an antibody (circle matching),

The limited-length character string $\mathbf{d}$ is the coding of variable vector $\mathbf{x}$ as $\mathbf{d} = \text{encode}(\mathbf{x})$; and $\mathbf{x}$ is called the decoding of antibody $\mathbf{d}$ following $\mathbf{x} = \text{decode}(\mathbf{d})$.

Set $\mathbf{I}$ is called the antibody space; namely, $\mathbf{d} \in \mathbf{I}$. The antibody population space is thus defined as

$$\mathbf{I}^m = \{\mathbf{D} : \mathbf{D} = (\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m), \mathbf{d}_k \in \mathbf{I}, \quad 1 \leq k \leq m\}, \tag{3.1}$$

where the positive integer m is the size of antibody population $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_m\}$which is an m-dimensional group of antibody $\mathbf{d}$, being a spot within the antibody space $\mathbf{I}$.

3.2. CSA Operators

Based on [30], the CSA implements three different operators: the clonal proliferation operator (T_P^C), the affinity maturation operator (T_M^A), and the clonal selection operator (T_S^C). $\mathbf{A}(k)$ is the antibody population at time k that represents the set of antibodies $\mathbf{a}$, such as $\mathbf{A}(k) = \{\mathbf{a}_1(k), \mathbf{a}_2(k), \dots, \mathbf{a}_n(k)\}$. The evolution process of CSA can be described as follows:

$$\mathbf{A}(k) \xrightarrow{T_C^P} \mathbf{Y}(k) \xrightarrow{T_M^A} \mathbf{Z}(k) \cup \mathbf{A}(k) \xrightarrow{T_S^C} \mathbf{A}(k+1). \tag{3.2}$$

3.2.1. Clonal Proliferation Operator (T_P^C)

Define

$$\mathbf{Y}(k) = T_P^C(\mathbf{A}(k)) = \left[T_P^C(\mathbf{a}_1(k)), T_P^C(\mathbf{a}_1(k)), \dots, T_P^C(\mathbf{a}_n(k))\right], \tag{3.3}$$

where $(k) = T_{\mathrm{P}}^C(\mathbf{A}(k)) = \mathbf{e}_i \cdot \mathbf{a}_i(k) i = 1, 2, \dots, n$, and $\mathbf{e}_i$ is a q_i-dimensional identity column vector. Function round(x) gets x to the least integer bigger than x. There are various methods for calculating q_i. In this work, it is calculated as follows:

$$q_i(k) = \text{round}\left[N_c \cdot \frac{F(\mathbf{a}_i(k))}{\sum_{j=1}^{n} F(\mathbf{a}_j(k))}\right] \quad i = 1,2,\ldots,n, \tag{3.4}$$

where N_c is called the clonal size. The value of $q_i(k)$ is proportional to the value of $F(\mathbf{a}_i(k))$. After clonal proliferation, the population becomes

$$\mathbf{Y}(k) = \{\mathbf{Y}_1(k), \mathbf{Y}_2(k), \ldots, \mathbf{Y}_n(k)\}, \tag{3.5}$$

where

$$\begin{aligned} \mathbf{Y}_i(k) &= \{\mathbf{y}_{ij}(k)\} = \{\mathbf{y}_{i1}(k), \mathbf{y}_{i2}(k), \ldots, \mathbf{y}_{iq_i}(k)\}, \\ \mathbf{y}_{ij}(k) &= \mathbf{a}_1(k), \quad j = 1,2,\ldots,q_i,\ i = 1,2,\ldots,n. \end{aligned} \tag{3.6}$$

3.2.2. *Affinity Maturation Operator (T_M^A)*

The affinity maturation operation is performed by hypermutation. Random changes are introduced into the antibodies just like it happens in the immune system. Such changes may lead to increase in the affinity. The hypermutation is performed by the operator T_M^A which is applied to the population $\mathbf{Y}(k)$ as it is obtained by clonal proliferation $\mathbf{Z}(k) = T_M^C(\mathbf{Y}(k))$.

The mutation rate is calculated using the following equation [25]:

$$\alpha = e^{(-\rho \cdot F(ab))}, \tag{3.7}$$

α beingthe mutation rate, F being the objective function value of the antibody (ab) as it is normalized between [0,1], and ρ being a fixed step. In [31], it is demonstrated the importance of including the factor ρ into (3.7) to improve the algorithm performance. The way ρ modifies the shape of the mutation rate is shown by Figure 1.

The number of mutations held by a clone with objective function value F is equal to $L \cdot \alpha$, considering L as the length of the antibody—22 bits are used in this paper. For the binary encoding, mutation operation can be done as follows: each gene within an antibody may be replaced by its opposite number (i.e., 0-1 or 1-0).

Following the affinity maturation operation, the population becomes

$$\begin{aligned} \mathbf{Z}(k) &= \{\mathbf{Z}_1(k), \mathbf{Z}_2(k), \ldots, \mathbf{Z}_n(k)\}, \\ \mathbf{Z}_i(k) &= \{\mathbf{z}_{ij}(k)\} = \{\mathbf{z}_{i1}(k), \mathbf{z}_{i2}(k), \ldots, \mathbf{z}_{iq_1}(k)\}, \\ \mathbf{z}_{ij}(k) &= T_M^A(\mathbf{y}_{ij}(k)), \quad j = 1,2,\ldots,q_1,\ i = 1,2,\ldots,n, \end{aligned} \tag{3.8}$$

where T_M^{A} is the operator as it is defined by (3.7) and applied onto the antibody $\mathbf{y}_{ij}$.

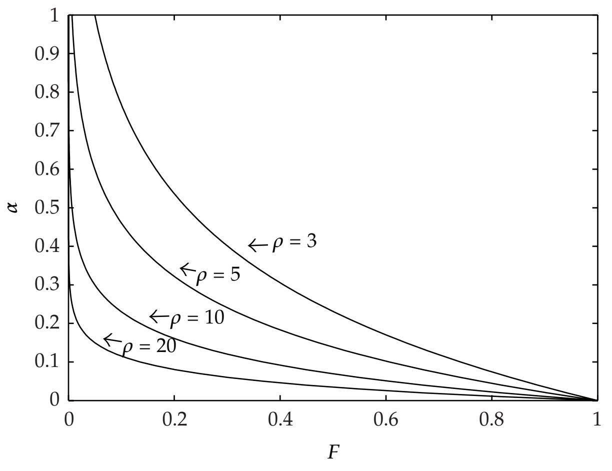


Figure 1: Hypermutation rate versus fitness, considering some size steps.

3.2.3. Clonal Selection Operator (T_S^C)

Define, for all $i = 1, 2, \ldots, n$, $\mathbf{b}_i(k) \in \mathbf{Z}_i(k)$ as the antibody with the highest affinity in $\mathbf{Z}_i(k)$, then $\mathbf{a}_i(k+1) = T_S^C(\mathbf{Z}_i(k) \cup \mathbf{a}_i(k))$, where T_S^C is defined as

$$T_S^C(\mathbf{Z}_i(k) \cup \mathbf{a}_i(k)) = \begin{cases} \mathbf{b}_i(k) & \text{if } F(\mathbf{a}_i(k)) < F(\mathbf{b}_i(k)) \\ \mathbf{a}_i(k) & \text{if } F(\mathbf{a}_i(k)) \geq F(\mathbf{b}_i(k)), \end{cases} \tag{3.9}$$

where $i = 1, 2, \ldots, n$.

Each step of the CSA may be defined as follows.

(1) Initialize randomly a population (Pinit), a set $h = P_r + n$ of candidate solutions of subsets of memory cells ($\mathbf{M}$) which is added to the remaining population (P_r), with the total population being $\mathbf{P}_T = P_r + M$, with $\mathbf{M}$ holding n memory cells.

(2) Select the n best individuals of the population $\mathbf{P}_T$ to build $\mathbf{A}(k)$, according to the affinity measure (objective function).

(3) Reproduce (T_P^C) population $\mathbf{A}(k)$ proportionally to their affinity with the antigen and generate a temporary population of clones $\mathbf{Y}(k)$. The clone number is an increasing function of the affinity with the antigen (3.1).

(4) Mutate (T_M^A) the population $\mathbf{Y}(k)$ of clones according to the affinity of the antibody to the antigen (3.4). A maturated antibody population $\mathbf{Z}(k)$ is thus generated.

(5) Reselect (T_S^C) the best individuals from $\mathbf{Z}(k)$ and $\mathbf{A}(k)$ to compose a new memory set $\mathbf{M} = \mathbf{A}(k+1)$.

(6) Add random P_r novel antibodies (diversity introduction) to the new memory cells $\mathbf{M}$ to build $\mathbf{P}_T$.

(7) Stop if any criteria are reached; otherwise return to (2.2).

Figure 2 shows the full draw of the CSA. The clone number in Step 3 is defined according to (3.1). Although a unique mutation operator is used in Step 5, the mutated values

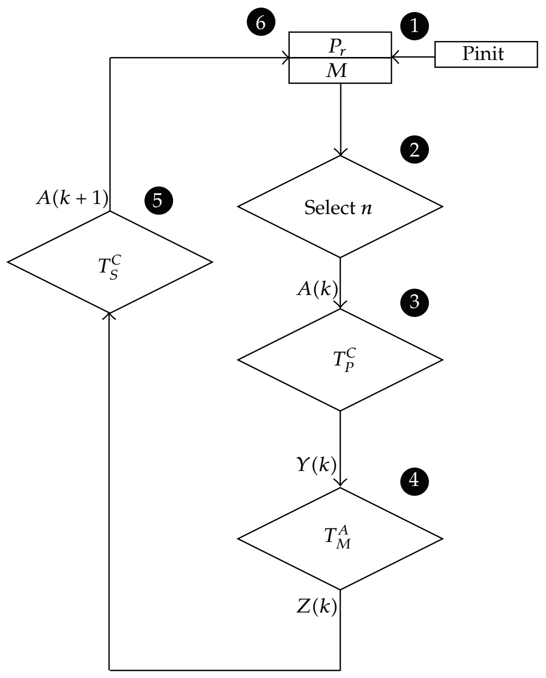


Figure 2: Basic flow diagram of clonal selection algorithm (CSA).

of individuals are inversely proportional to their fitness by means of (3.7); that is, the more Ab shows a better fitness, the less it may change.

The similarity property [32] within the Abs can also affect the convergence speed of the CSA. The idea of the antibody addition based on the immune network theory is introduced for providing diversity to the newly generated Abs in **M**, which may be similar to those already in the old memory **M**. Holding such a diverse Ab pool, the CSA can avoid being trapped into local minima [30], contrasting to well-known genetic algorithms (GAs) which usually tend to bias the whole population of chromosomes towards only the best candidate solution. Therefore, it can effectively handle challenging multimodal optimization tasks [33–36].

The management of population includes a simple and direct searching algorithm for globally optimal multimodal functions. This is also another clear difference in comparison to other evolutionary algorithms, like GA, because it does not require crossover but only cloning and hypermutation of individuals in order to use affinity as selection mechanism. The CSA is adopted in this work to find the parameters P, σ, and μ of Gaussian functions and their corresponding threshold values for the image.

4. Determination of Thresholding Values

The next step is to determine the optimal threshold values. Considering that the data classes are organized such that $\mu_1 < \mu_2 < \cdots < \mu_K$, the threshold values are obtained by computing the overall probability error of two adjacent Gaussian functions, yielding

$$E(T_h) = P_{h+1} \cdot E_1(T_h) + P_i \cdot E_2(T_h), \quad h = 1, 2, \ldots, K-1, \tag{4.1}$$

considering

$$E_1(T_h) = \int_{-\infty}^{T_h} p_{h+1}(x)dx,$$
$$E_2(T_h) = \int_{T_h}^{\infty} p_h(x)dx. \tag{4.2}$$

$E_1(T_h)$ is the probability of mistakenly classifying the pixels in the $(h+1)$th class belonging to the hth class, while $E_2(T_h)$ is the probability of erroneously classifying the pixels in the hth class belonging to the $(h+1)$th class. $P_j{}'$s are the a priori probabilities within the combined probability density function, and $T_h$is the threshold value between the hth and the $(h+1)$th classes. One T_h value is chosen such as the error $E(T_h)$ is minimized. By differentiating $E(T_h)$ with respect to T_h and equating the result to zero, it is possible to use the following equation to define the optimum threshold value T_h:

$$AT_h^2 + BT_h + C = 0, \tag{4.3}$$

considering

$$A = \sigma_h^2 - \sigma_{h+1}^2,$$
$$B = 2 \cdot \left(\mu_h \sigma_{h+1}^2 - \mu_{h+1}\sigma_h^2\right),$$
$$C = \left(\sigma_h \mu_{h+1}\right)^2 - \left(\sigma_{h+1}\mu_h\right)^2 + 2 \cdot \left(\sigma_h \sigma_{h+1}\right)^2 \cdot \ln\left(\frac{\sigma_{h+1} P_h}{\sigma_h P_{h+1}}\right). \tag{4.4}$$

Although the above quadratic equation has two possible solutions, only one of them is feasible; that is, a positive value falling within the interval.

5. Implementation Details

In this work, either an antibody or an antigen will be represented (in binary form) by a bit chain of the form

$$c = \langle c_1, c_2, \ldots, c_L \rangle, \tag{5.1}$$

with c representing a point in an L-dimensional space:

$$c \in S^L. \tag{5.2}$$

The clonal selection algorithm can be stated as follows.

(1) An original population of N individuals (antibodies) is generated, considering the size of 22 bits.

(2) The n best individuals based on the affinity measure are selected. They will represent the memory set.

(3) Such n best individuals are cloned m times.

(4) Performing a hypermutation of the cloned individuals which follows the proportion inside the affinity between antibodies and antigens and generating one improved antibody population.

(5) From the hypermutated population, the individuals with the higher affinity are to be reselected.

(6) As for the original population, the individuals with the highest affinity are replaced, improving the overall cells set.

Once the above steps are completed, the process is started again, until one individual showing the highest affinity is found, that is, finding the minimum stated in (2.3). At this work, the algorithm considers 3 Gaussians to represent the same number of classes. However, it can be easily expanded to more classes. Each single Gaussian has the variables P_i, μ_i, σ_i (with $i = 1, 2, 3$) representing the Hamming shape-space by means of an 22 bits word over the following ranges:

$$\begin{gathered} P_i : [1,\ \max(h)] \\ \mu_i : \left[\min(g),\ \max(g)\right] \\ \sigma_i : \left[1,\ \max\left(g\right)^{*}0.5\right], \end{gathered} \tag{5.3}$$

with g representing the grey level and h is the histogram of the grey level image. Hence, the first step is to generate the initial individual population of the antibody population by means of

$$\mathrm{AB} = 2 \cdot r, \left(N, S_p\right) - 1; \tag{5.4}$$

with S_p representing the bit string assigned to each of the initial individuals N. Later, in order to perform the mapping from binary string to real value, we use

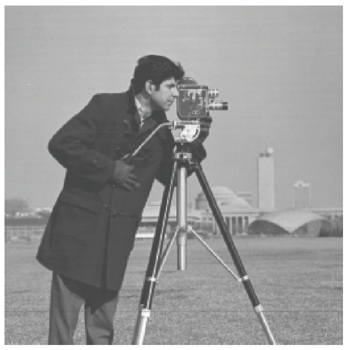
$$(\langle c_L, \dots, c_2, c_1 \rangle)_2 = \left(\sum_{i=0}^{21} c_i \cdot 2^i \right)_{10} = r'. \tag{5.5}$$

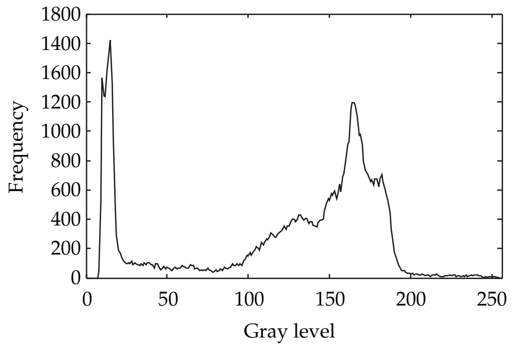
As to find the corresponding real value for r,

$$r = r' \cdot \frac{r_{\max}}{2^{22} - 1}, \tag{5.6}$$

with $r_{\max}$ representing $\max(h), \max(g), \max(g)/2$.

The population is set to 100 individuals (N), including the best 20 individuals (n). The 20 selected individuals are cloned 10 times (m). The corresponding mutation probability is proportional to the resulting error by (2.3). The algorithm is thus executed until the minimum possible value of (2.3) is reached.


(a)



(b)

Figure 3: (a) Original "The Cameraman" image and (b) its correspondent histogram.

6. Experimental Results

6.1. Presentation of Results

In this section, two experiments are tested to evaluate the performance of the proposed algorithm. The first one considers the well-known image of the "The Cameraman" shown in Figure 3(a) as its corresponding histogram is presented by Figure 3(b). The goal is to segment the image with 3 different pixel classes. During learning, the CSA algorithm adjusts 9 parameters in this test. In this experiment, a population of 100 (N) individuals is considered. Each candidate holds 9 dimensions, yielding

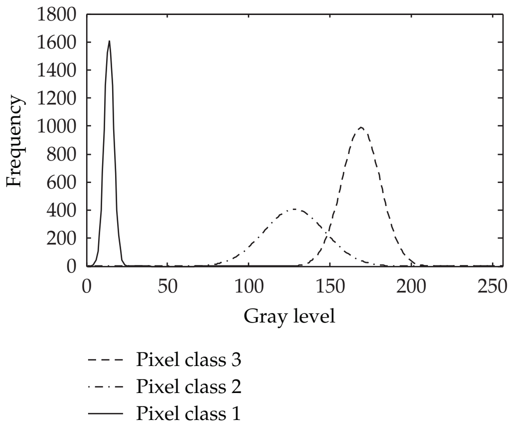
$$I_N = \left\{\sigma_1^N, \sigma_2^N, \sigma_3^N, P_1^N, P_2^N, P_3^N, \mu_1^N, \mu_2^N, \mu_3^N\right\}, \tag{6.1}$$

with N representing the individual's number, in this case, 100. The parameters (P, σ, μ) are randomly initialized.

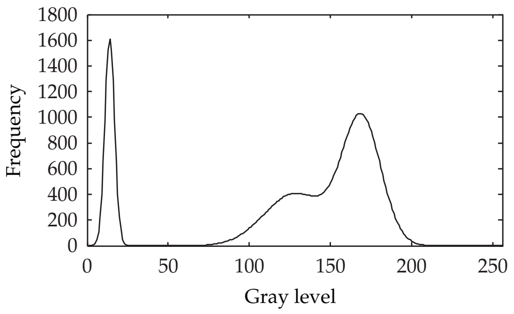
The experiments suggest that, after 130 iterations, the CSA algorithm has converged to the global minimum. Figure 4(a) shows the obtained Gaussian functions (pixel classes), while Figure 4(b) shows the combined graph. The layout in Figure 4(b) suggests an easy combination of the Gaussian functions to approximate the shape of the graph shown in Figure 3(b), representing the histogram of the original image. Figure 5 shows the segmented image whose threshold values are calculated according to (3.4) and (3.5).

In order to test the performance, the algorithm gets to analyze the image shown in Figure 6 whose histogram exhibits a remarkable problem (a set of peaks) regarding class approximation. Such image, due to its complexity, is considered as a benchmark image for other algorithms, including classical approaches as in [7, 10] or intelligent algorithms as in [11, 12]. For this particular image, after 190 generations, the algorithm was capable to achieve the minimum approximation value J (see (2.3)), considering three different classes. Figure 7 shows the approximation quality; meanwhile, Figure 8 presents the obtained segmented image.

The third experiment considers a new image known as "The scene" shown by Figure 9(a). The histogram is presented in Figure 9(b). Now, the algorithm considers 4 pixel classes. The optimization is performed by the CSA algorithm resulting in the Gaussian



(a)



(b)

Figure 4: Applying the CSA algorithm for 3 classes and its results: (a) Gaussian functions for each class, (b) mixed Gaussian functions (approaching the original histogram).

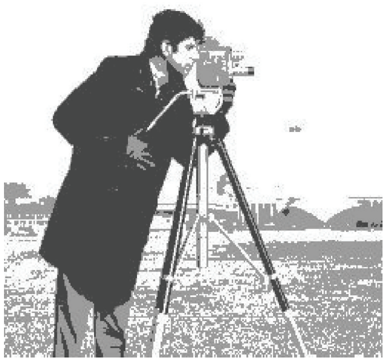
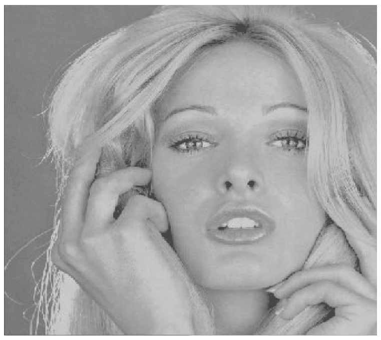
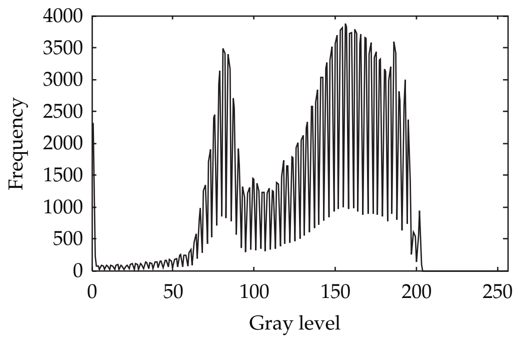


Figure 5: The image after the segmentation is applied, considering three classes.

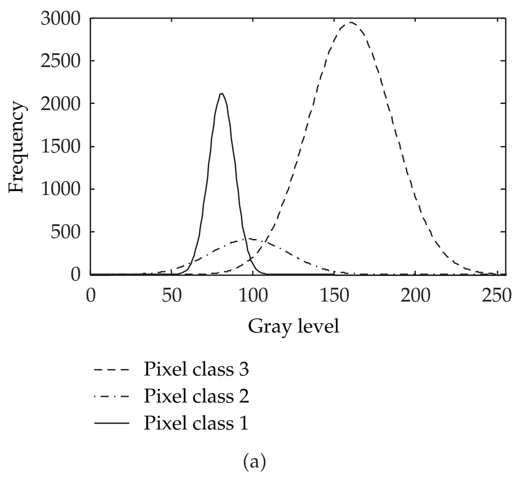


(a)



(b)

Figure 6: Benchmark image and its histogram.



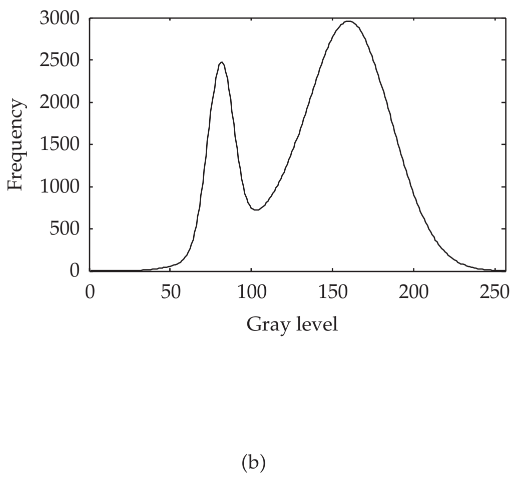


Figure 7: CSA segmentation for 3 classes over the benchmark image: (a) Gaussian functions for each class. (b) Mixed Gaussian functions approaching the look of the original histogram.

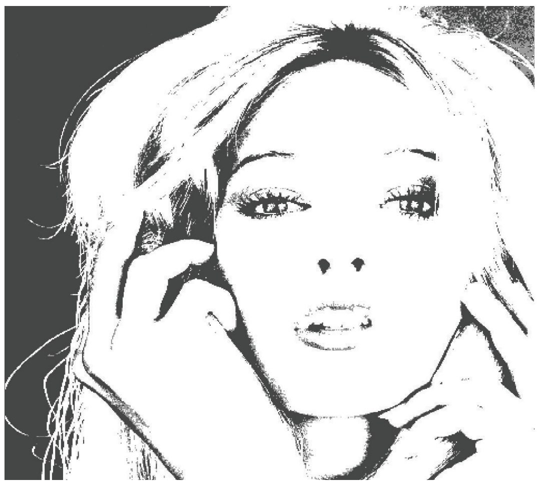


Figure 8: The benchmark image after segmentation considering all three classes.

functions shown by Figure 10(a). Figure 10(b) presents the combined graph from the addition of such Gaussian functions.

After the optimization by the CSA algorithm, the added layout including all 4 Gaussian functions is obtained as shown by Figure 11(a). It is also evident that the resulting function approaches the original histogram as Figure 11(b) shows the resulting image after applying the segmentation algorithm.

6.2. Comparing the CSA versus the EM and LM Methods

In order to enhance the algorithm's analysis, the proposed approach is compared to the EM algorithm and the Levenberg-Marquardt (LM) method which are commonly employed for

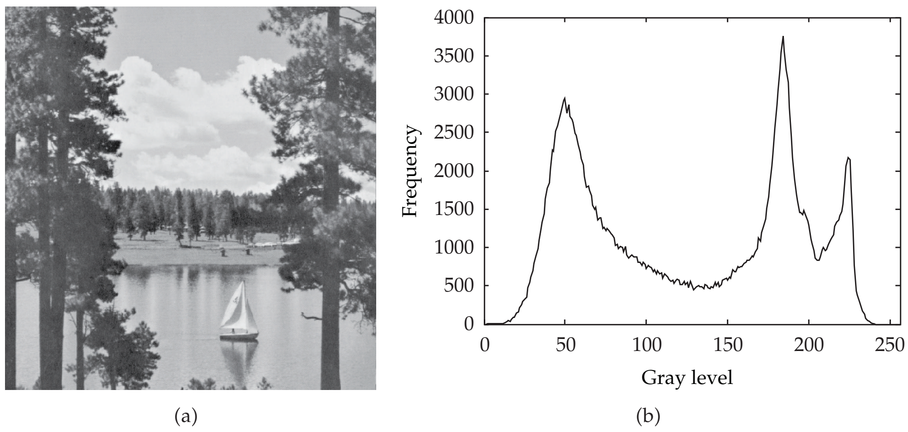


Figure 9: Third experiment data: (a) the original image "The scene" and (b) its corresponding histogram.

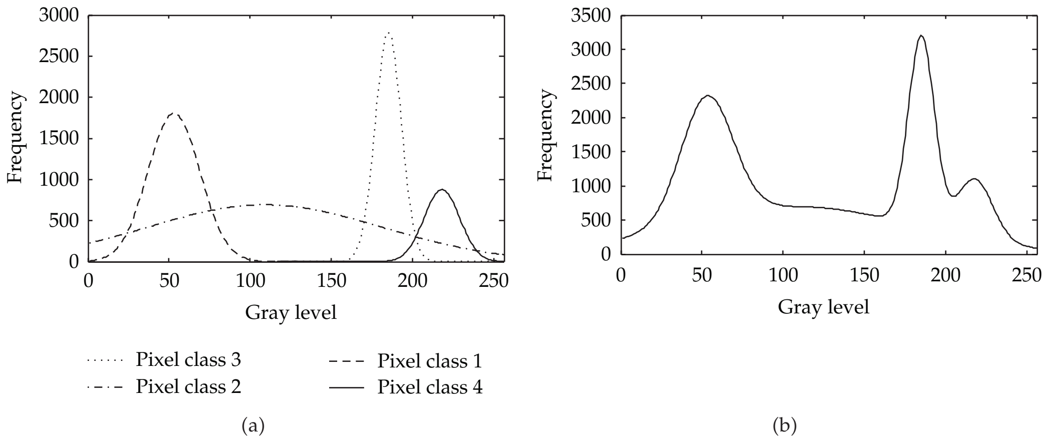


Figure 10: Applying the CSA algorithm for 4 classes: (a) Gaussian functions at each class: (b) adding all four Gaussian functions, it approaches the original histogram.

computing Gaussian mixture parameters. The comparison focuses on the following issues: sensitivity to initial conditions, singularities, convergence, and computational costs.

6.2.1. Sensitivity to the Initial Conditions

In this experiment, initial values of the mixture model for all methods are randomly set while the same histogram is taken in account for the approximation task. Final parameters representing the Gaussian mixture (considering four different classes) after convergence are reported. All algorithms (EM, LM, and CSA) are executed until no change in parameter values is detected. Figure 12(a) shows the image used in this comparison while Figure 12(b) pictures its histogram. All experiments are conducted several times in order to assure consistency. Table 1 exhibits the parameter values (μ_q, σ_q, P_q, $q \in 1, \ldots, 4$) of the obtained Gaussian mixtures, considering the two initial conditions in which the highest contrasts were found. Figure 13 shows the segmented images obtained under such initial conditions. Further

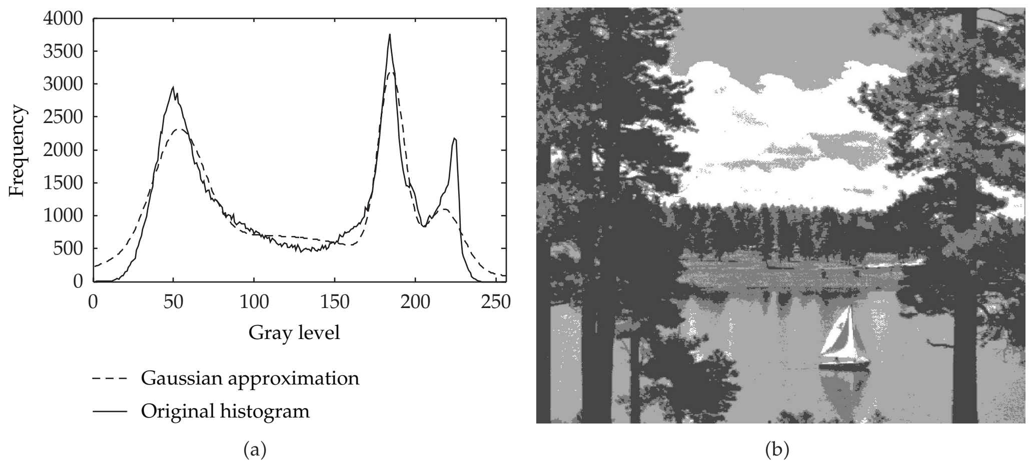


Figure 11: Segmentation of "The Scene" considering four classes for the CSA algorithm. (a) Comparison between the original histogram and the Gaussian approach. (b) The image after the segmentation process.

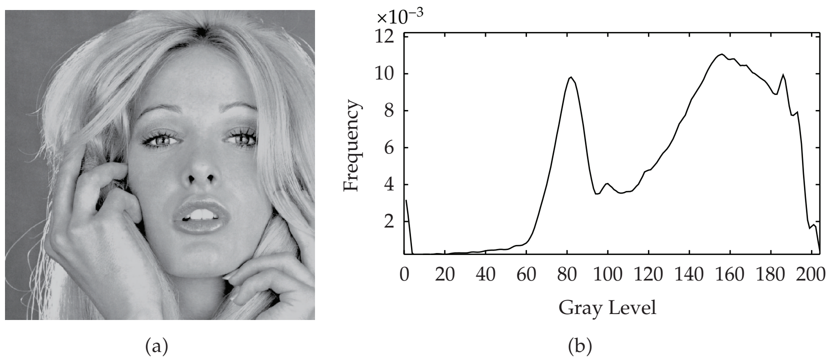


Figure 12: (a) Original image used for the comparison on initial conditions and (b) its corresponding histogram.

analysis on Table 1 shows the acute sensitivity of the EM algorithm to initial conditions. By such sensitivity, it is observed in Figure 13 where a clear pixel misclassification appears in some sections of the image. In case of the LM method, although it does not present a considerable difference in comparison to optimal values, its deviation shows that it is prone to get trapped into a local minimum. On the other hand, the CSA algorithm exhibits the best performance as its parameter values fall the nearest to the optimal approximation performance.

6.2.2. Convergence and Computational Cost

The experiment aims to measure the number of iterations and the computing time spent by the EM, the LM, and the CSA required to calculate the parameters of the Gaussian mixture after considering three different benchmark images. Figure 14 shows the images used in the experiment. Such images are selected, since they are employed in the standard

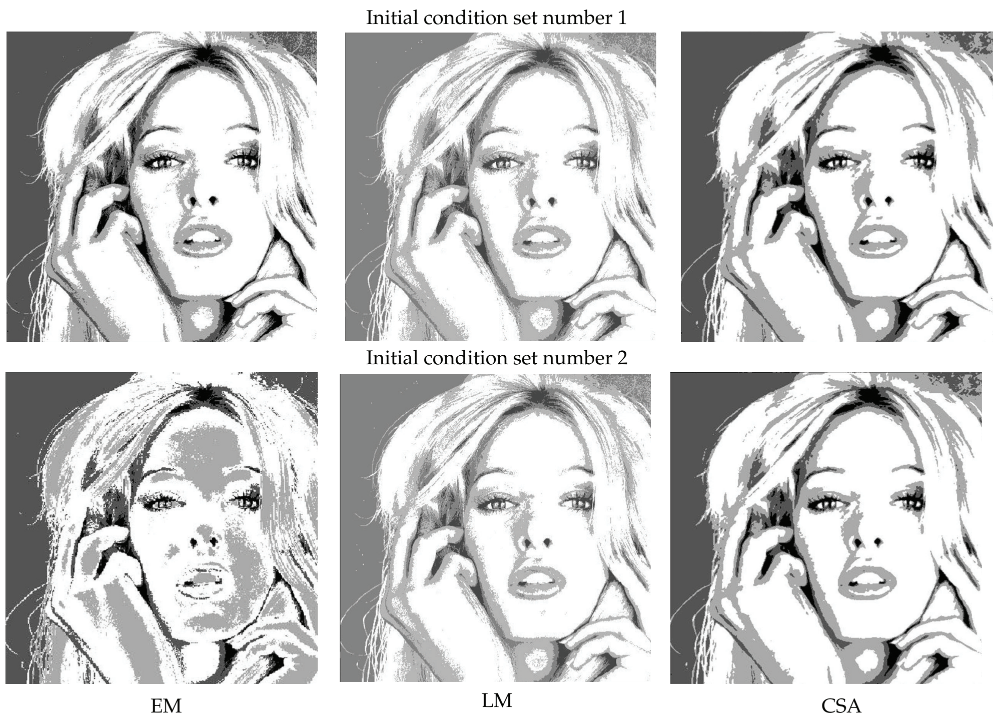

Initial condition set number 1
Initial condition set number 2
EM
LM
CSA

Figure 13: Segmented images after applying the EM, the LM, and the CSA algorithm with different initial conditions.

Table 1: Comparison between the EM, the LM, and the CSA algorithm, considering two different initial conditions.

Parameters	Initial condition 1	EM	LM	CSA	Initial condition 2	EM	LM	CSA
μ_1	40.6	33.13	32.12	32.23	10	20.90	31.80	32.25
μ_2	81.2	81.02	82.05	81.55	100	82.78	80.85	82.00
μ_3	121.8	127.52	127	126.89	138	146.67	128	127.11
μ_4	162.4	167.58	166.80	167.00	200	180.72	165.90	166.50
σ_1	15	25.90	25.50	25.30	10	18.52	20.10	25.01
σ_2	15	9.78	9.70	9.86	5	12.52	9.81	9.45
σ_3	15	17.72	17.05	17.12	8	20.5	15.15	17.23
σ_4	15	17.03	17.52	17.45	22	10.09	18.00	17.22
P_1	0.25	0.0313	0.0310	0.317	0.20	0.0225	0.0312	0.317
P_2	0.25	0.2078	0.2081	0.198	0.30	0.2446	0.2079	0.214
P_3	0.25	0.2508	0.2500	0.249	0.20	0.5232	0.2502	0.245
P_4	0.25	0.5102	0.5110	0.501	0.30	0.2098	0.5108	0.498

segmentation literature. All the experiments consider four classes. Table 2 shows the averaged measurements as they are obtained from 20 experiments. It is evident that the EM is the slowest to converge (iterations), and the LM shows the highest computational cost (time elapsed) because it requires complex Hessian approximations. On the other hand, the CSA shows an acceptable tradeoff between its convergence time and its computational

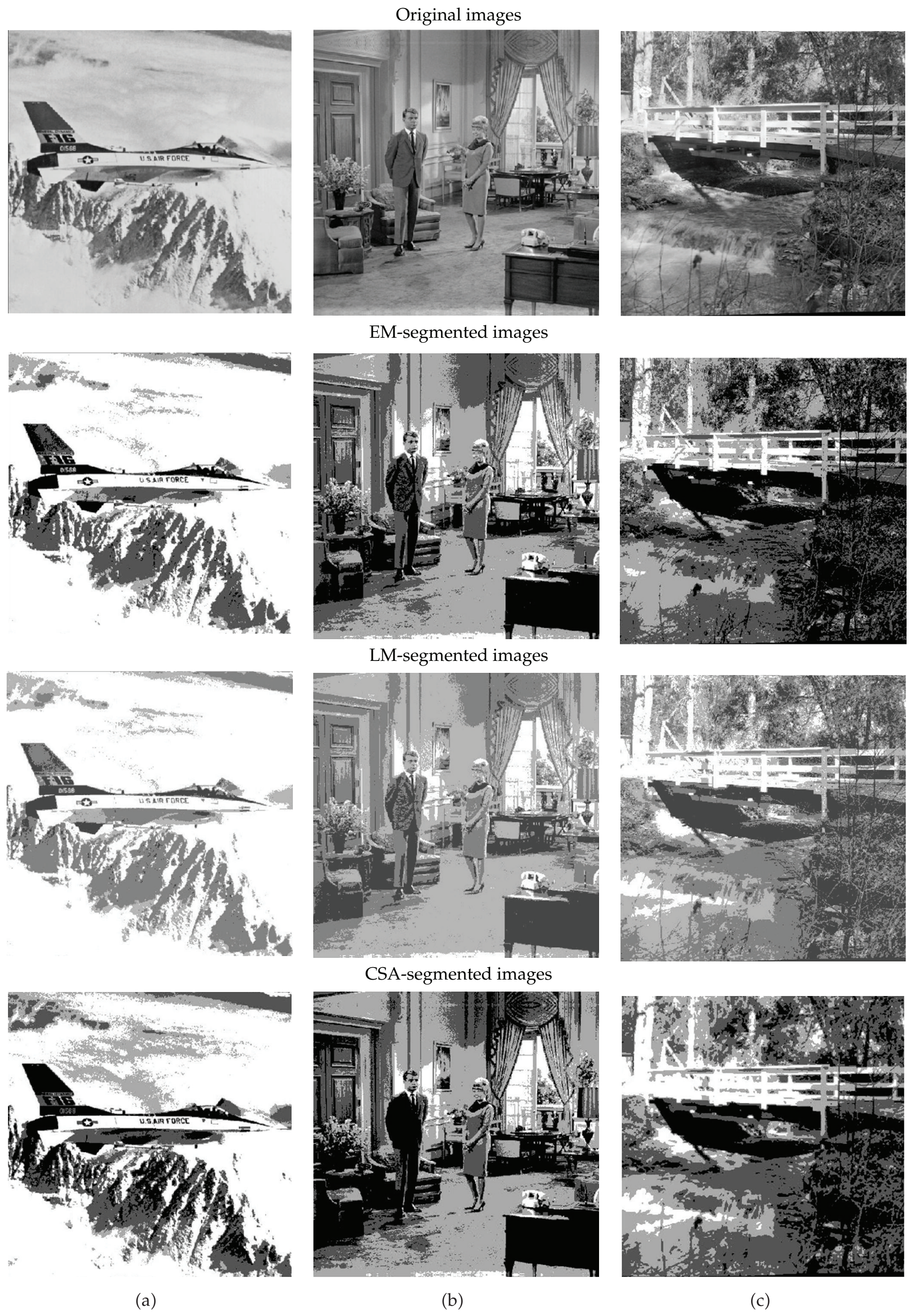


Figure 14: Original benchmark images ((a)–(c)) and segmented images obtained by the EM, the LM, and the CSA algorithms.

Table 2: Iterations and time requirements of the EM, the LM, and the CSA algorithm as they are applied to segment benchmark images (see Figure 14).

Iterations Time elapsed	(a)	(b)	(c)
EM	1855 2.72 s	1833 2.70 s	1870 2.73 s
LM	985 4.03 s	988 4.04 s	958 4.98 s
CSA	201 0.21 s	188 0.18 s	282 0.25 s

cost. Finally, Figure 14 below shows the segmented images as they are generated by each algorithm. By analyzing the images (a)–(c) in Figure 14, it is clear that the CSA approach presents better results when the segmented images are compared with the original ones. In case of the EM and LM algorithms, several stains are identified in regions where the intensity level must be considered homogenous.

7. Conclusions

In this paper, an automatic image multi-threshold approach based on the clonal selection algorithm (CSA) is proposed. The segmentation process is considered to be similar to an optimization problem. The algorithm approximates the 1-D histogram of a given image using a Gaussian mixture model whose parameters are calculated through the CSA. Each Gaussian function approximating the histogram represents a pixel class and therefore one threshold point.

Experimental evidence shows that the CSA has a compromise between its convergence time and its computational cost when it is compared to the expectation-maximization (EM) method and the Levenberg-Marquardt (LM) algorithm. Additionally, the CSA also exhibits a better performance under certain circumstances (initial conditions) on which it is well reported in the literature [14, 18] that the EM has underperformed. Finally, the results have shown that the stochastic search accomplished by the CSA method shows a consistent performance with no regard of the initial value and still showing a greater chance to reach the global minimum.

Although Table 2 indicates that the CSA method can yield better results in comparison to the EM and gradient-based methods, notice that the aim of our paper is not intended to beat all segmentation algorithms which have been proposed earlier but to show that the artificial immune systems can effectively serve as an attractive alternative to evolutionary algorithms which have been employed before to successfully segment images.

References

[1] T. Abak, U. Baris, and B. Sankur, "The performance evaluation of thresholding algorithms for optical character recognition," in *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pp. 697–700, August 1997.

[2] M. Kamel and A. Zhao, "Extraction of binary character/graphics images from grayscale document images," *Graphical Models and Image Processing*, vol. 55, no. 3, pp. 203–217, 1993.

[3] O. D. Trier and A. K. Jain, “Goal-directed evaluation of binarization methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, pp. 1191–1201, 1995.

[4] B. Bhanu, “Automatic target recognition: state of the art survey,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 22, no. 4, pp. 364–379, 1986.

[5] M. Sezgin and B. Sankur, “Selection of thresholding methods for non-destructive testing applications,” in *Proceedings of the IEEE International Conference on Image Processing (ICIP ’01)*, pp. 764–767, October 2001.

[6] M. Sezgin and R. Taşaltín, “A new dichotomization technique to multilevel thresholding devoted to inspection applications,” *Pattern Recognition Letters*, vol. 21, no. 2, pp. 151–161, 2000.

[7] R. Guo and S. M. Pandit, “Automatic threshold selection based on histogram modes and a discriminant criterion,” *Machine Vision and Applications*, vol. 10, no. 5-6, pp. 331–338, 1998.

[8] N. R. Pal and S. K. Pal, “A review on image segmentation techniques,” *Pattern Recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[9] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen, “A survey of thresholding techniques,” *Computer Vision, Graphics and Image Processing*, vol. 41, no. 2, pp. 233–260, 1998.

[10] W. Snyder, G. Bilbro, A. Logenthiran, and S. Rajala, “Optimal thresholding-a new approach,” *Pattern Recognition Letters*, vol. 11, no. 12, pp. 803–810, 1990.

[11] S. Chen and M. Wang, “Seeking multi-thresholds directly from support vectors for image segmentation,” *Neurocomputing*, vol. 67, no. 4, pp. 335–344, 2005.

[12] L. Chih-Chin, “A novel image segmentation approach based on particle swarm optimization,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 89, no. 1, pp. 324–327, 2006.

[13] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, Boston, Mass, USA, 1992.

[14] L. Gupta and T. Sortrakul, “A gaussian-mixture-based image segmentation algorithm,” *Pattern Recognition*, vol. 31, no. 3, pp. 315–325, 1998.

[15] A. P. Dempster, A. P. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1997.

[16] Z. Zhang, C. Chen, J. Sun, and L. Chan, “EM algorithms for gaussian mixtures with split-and-merge operation,” *Pattern Recognition*, vol. 36, no. 9, pp. 1973–1983, 2003.

[17] H. Park, S. Amari, and K. Fukumizu, “Adaptive natural gradient learning algorithms for various stochastic models,” *Neural Networks*, vol. 13, no. 7, pp. 755–764, 2000.

[18] H. Park and T. Ozeki, “Singularity and slow convergence of the em algorithm for gaussian mixtures,” *Neural Processing Letters*, vol. 29, no. 1, pp. 45–59, 2009.

[19] E. Cuevas, D. Zaldivar, and M. Perez-Cisneros, “Seeking multi-thresholds for image segmentation with learning automata,” *Machine Vision and Applications*, vol. 22, no. 5, pp. 805–818, 2011.

[20] E. Cuevas, D. Zaldivar, and M. Pérez-Cisneros, “A novel multi-threshold segmentation approach based on differential evolution optimization,” *Expert Systems with Applications*, vol. 37, no. 7, pp. 5265–5271, 2010.

[21] G. A. Goldsby, T. J. Kindt, J. Kuby, and B. A. Osborne, *Immunology*, W. H. Freeman, New York, NY, USA, 5th edition, 2003.

[22] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, London, UK, 2002.

[23] D. Dasgupta, “Advances in artificial immune systems,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 40–49, 2006.

[24] X. Wang, X. Z. Gao, and S. J. Ovaska, “Artificial immune optimization methods and applications—a survey,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, (SMC ’04)*, pp. 3415–3420, The Hague, The Netherlands, October 2004.

[25] L. N. de Castro and F. J. von Zuben, “Learning and optimization using the clonal selection principle,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.

[26] G. L. Ada and G. Nossal, “The clonal-selection theory,” *Scientific American*, vol. 257, no. 2, pp. 50–57, 1987.

[27] C. A. C. Coello and N. C. Cortés, “Solving multiobjective optimization problems using an artificial immune system,” *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.

[28] F. Campelo, F. G. Guimarães, H. Igarashi, and J. A. Ramírez, “A clonal selection algorithm for optimization in electromagnetics,” *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1736–1739, 2005.

[29] W. Dong, G. Shi, and L. Zhang, “Immune memory clonal selection algorithms for designing stack filters,” *Neurocomputing*, vol. 70, no. 4–6, pp. 777–784, 2007.

[30] M. Gong, L. Jiao, L. Zhang, and H. Du, “Immune secondary response and clonal selection inspired optimizers,” *Progress in Natural Science*, vol. 19, no. 2, pp. 237–253, 2009.

[31] V. Cutello, G. Narzisi, G. Nicosia, and M. Pavone, “Clonal selection algorithms: a comparative case study using effective mutation potentials,” in *Proceedings of the 4th International Conference on Artificial Immune Systems, (ICARIS ’05)*, C. Jacob et al., Ed., pp. 13–28, August 2005.

[32] M. Gong, L. Jiao, and X. Zhang, “A population-based artificial immune system for numerical optimization,” *Neurocomputing*, vol. 72, no. 1–3, pp. 149–161, 2008.

[33] J. Yoo and P. Hajela, “Immune network simulations in multicriterion design,” *Structural and Multidisciplinary Optimization*, vol. 18, no. 2, pp. 85–94, 1999.

[34] X. Wang, X. Z. Gao, and S. J. Ovaska, “A hybrid optimization algorithm in power filter design,” in *Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society (IECON ’05)*, pp. 1335–1340, Raleigh, NC, USA, November 2005.

[35] X. Xu and J. Zhang, “An improved immune evolutionary algorithm for multimodal function optimization,” in *Proceedings of the 3rd International Conference on Natural Computation (ICNC ’07)*, pp. 641–646, Haikou, China, August 2007.

[36] T. Tang and J. Qiu, “An improved multimodal artificial immune algorithm and its convergence analysis,” in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA ’06)*, pp. 3335–3339, Dalian, China, June 2006.