A Fully Adaptive and Hybrid Method for Image Segmentation Using Multilevel Thresholding

Salima Ouafel
College of Engineering, MISC laboratory, CICS Group,
Department of Computer Science, University Mentouri – Constantine, Algeria
souadfel@yahoo.fr

Souham Meshoul
College of Engineering, MISC laboratory, CICS Group,
Department of Computer Science, University Mentouri – Constantine, Algeria
smeshoul@umc.edu.dz

Abstract—High level tasks in image analysis and understanding are based on accurate image segmentation which can be accomplished through multilevel thresholding. In this paper, we propose a new method that aims to determine the number of thresholds as well as their values to achieve multilevel thresholding. The method is adaptive as the number of thresholds is not required as a prior knowledge but determined depending on the used image. The main feature of the method is that it combines the fast convergence of Particle Swarm Optimization (PSO) with the jumping property of simulated annealing to escape from local optima to perform a search in a space the dimensions of which represent the number of thresholds and their values. Only the maximum number of thresholds should be provided and the adopted encoding encompasses a continuous part and a discrete part that are updated through continuous and binary PSO equations. Experiments and comparative results with other multilevel thresholding methods using a number of synthetic and real test images show the efficiency of the proposed method.

Index Terms—Image segmentation, Multilevel thresholding, Particle swarm optimization, Simulated annealing

I. INTRODUCTION

Multilevel thresholding is an important technique that has many applications in image processing, including segmentation, clustering and object discrimination. It aims to separate the objects and the background of the image into non overlapping regions. Thresholding may require the use of one or more thresholds depending on the number of classes that should be depicted in an image. Therefore, based on this assumption, thresholding methods can be divided into two main categories namely bi-level thresholding and multi-level thresholding. In bi-level thresholding, pixels within an image are classified into two classes that consist of the object class and the background class depending on whether their values are above or below a given threshold. By extension, multilevel thresholding involves several thresholds to classify pixels into more classes. A good review of bi-level and multilevel thresholding algorithms can be found in [1].

Depending on the way to find the thresholds, both bi-level and multilevel thresholding methods can be divided into parametric and nonparametric approaches. The parametric approach is based on a statistical model of the pixel grey level distribution. Generally, a set of parameters that best fits the model is derived using least square estimation. This typically leads to nonlinear optimization problems that are computationally expensive and time-consuming [2, 3]. In the nonparametric approach, the search of the optimal thresholds is done by optimizing an objective function based on some discriminating criteria such as between class variance [4] and entropy [5].

A large number of thresholding methods have been proposed in the literature in order to perform bi-level thresholding and most of them are easily extendable to multilevel thresholding. Despite the straightforwardness of this extension process, the computational time will increase sharply when the number of thresholds is too high because of the exhaustive searching they perform [6]. This weakness makes the multilevel thresholding methods unsuitable in many applications and explains the need to use faster and robust optimization methods for multilevel thresholding problem [7]. As a consequence, several fast techniques have been proposed in the literature such as recursive algorithms [8]. Huang and Wang [9] propose a two-stage multi-threshold based on Otsu’s method. Wang and Chen [10] applied an improved shuffled frog-leaping algorithm to the three dimensional Otsu thresholding. Li and Tam presented a fast iterative implementation for the minimum cross entropy thresholding method [11]. Chung and Tsai [12] proposed a fast implementation of binarization algorithm using an efficient heap and quantization based data structure. A recursive programming technique using the maximum cross entropy was proposed by Yin [13]. Shelokar et al. [14] selected the optimal threshold by minimizing the sum of
Despite the relative efficiency of most of methods in the literature to find the optimal thresholds, they all require the number of thresholds to be specified in advance. However, in many practical cases, it is difficult, even impossible to determine the exact number of thresholds without a prior knowledge. Few algorithms have been designed to automatically determine the suitable number of thresholds. Fuzzy entropy is used in [35] to select automatically the optimal number of thresholds and their location in the image histogram. 

Yen et al. [36] proposed a new criterion for multilevel thresholding, called Automatic Thresholding Criterion (ATC). This criterion includes a bi-level method that correlates the discrepancy between the original and thresholded image and the number of bits used to represent the segmented image [1, 36]. In this strategy, the histogram image is divided into two more sequentially up to the minimum of the cost function ATC is reached [36]. In each step the distribution with the largest variance is further dichotomized in two more distributions by applying the same bi-level thresholding. A new dichotomization process of the ATC method was proposed in [37].

Although the dichotomization techniques are fast algorithms, they are sub-optimal techniques and so they do not allow providing the optimal threshold values [7]. The GA presented in [7] determines the threshold number as well as the optimal threshold values. In their method, the wavelet transformation is used to reduce the length of the histogram and the GA algorithm optimizes the ATC criterion. Recently, Djerou et al. proposed in [38] an automatic multilevel thresholding approach, based on Binary PSO algorithm, which uses the Otsu’s criterion and the Kapur’s entropy as objective function. In [39] author adapts the intelligent water drops (IWDs) algorithm to optimize a modified Otsu’s criterion for automatic multilevel thresholding.

PSO is a population-based metaheuristic introduced by Kennedy and Eberhart in 1995 [16] as an alternative to the standard Genetic Algorithm (GA). PSO was inspired by social behavior of bird flocking or fish schooling. Each particle within a swarm searches the solution space for the best solution by changing its position with time according to its own experience, and to the experience of neighboring particles. The main advantages of PSO are its flexibility, its robustness and its inherent parallelism. It has been applied in many kinds of engineering problem widely and has proved its large capability to competing with other classical optimization algorithms [40, 41, 42, 43, 44]. However, observations reveal that the PSO suffers from a premature convergence: it converges rapidly in the early stages of the searching process, but can be trapped in a local optimum in the later stages. To overcome this drawback of diversity loss, different hybrid PSOs with others metaheuristics like SA, GA, DE have been proposed. A good and recent review of hybrid PSO can be found in [45, 46].

In this paper, we propose a new automatic multilevel thresholding technique based on PSO. The new method is able to determine both the appropriate number of thresholds and the appropriate threshold values by optimizing ATC [36]. We adopt AMT-PSOSA (Automatic Multilevel Thresholding with PSO and SA) as an acronym to the algorithm we propose in this paper. AMT-PSOSA uses a new hybrid representation to allow particles to contain different thresholds numbers within a given range defined by minimum and maximum threshold number. Particles are initialized randomly to process different cluster numbers in a specified range and the goal of each particle is to search the optimum number of thresholds and the optimum threshold values. In order to allow PSO jump out of a local optimum, SA is applied to some particles of the swarm if no improvement occurs in their best local fitness during a number of iterations. Thus, at the start of SA most worsening solutions may be accepted, but at the end only improving ones are likely to be accepted. This procedure will help PSO to jump out of a local minimum.

The rest of this paper is organized as follows: Section 2 introduces, briefly, PSO and SA algorithms. Section 3 is devoted to detailed descriptions of AMT-PSOSA. The experimental results are evaluated and discussed in section 4. Finally, conclusions are given in section 5.
II. BACKGROUND

A. Particle swarm optimization

Particle swarm optimization (PSO) is a population-based evolutionary computation method first proposed by Kennedy and Eberhart [16]. It is inspired from the natural behavior of the individuals in a bird flock or fish school, when they search for some target (e.g., food). The PSO algorithm is initialized with a swarm of particles (called potential solutions) randomly distributed over the search area. Particles fly through the problem space by following the current optimum particle and aim to converge to the global optimum of a function attached to the problem.

Each particle $i$ in the swarm is represented by two elements: its current position ($p_i$) and its velocity ($v_i$). Its movement through the search space is influenced by its personal best position $p_{best}$; it has achieved so far and the local best value $lbest$, obtained so far by any particle in the neighbors of the particle. When a particle takes all the population as its topological neighbors, the $lbest$ value is a global best and is called $gbest$. If the problem space is D-dimensional and the swarm size is $N_p$, at each iteration $t$, the particle’s new position and its velocity are updated as follows:

$$
\begin{align*}
    v_i &= wv_i^{t-1} + c_1 \times r_1 \left( p_{best} - p_i^{t-1} \right) + \\
    & \quad + c_2 \times r_2 \left( p_{gbest} - p_i^{t-1} \right) \\
    p_i &= p_i^{t-1} + v_i
\end{align*}
$$

(1)

(2)

where $i = 1, 2, \ldots, N_p, j = 1, 2, \ldots, D$

The parameter $w$ is an inertia weight [47] used in order to control the values of the velocity. The inertia weight $w$ is equivalent to a temperature schedule in the simulated annealing algorithm and controls the influence of the previous velocity: a large value of $w$ favors exploration, while a small value of $w$ favors exploitation. As originally introduced in [47], $w$ decreases linearly during the run from $w_{max}$ to $w_{min}$. $c_1$ and $c_2$ are two constants which control the influence of the social and cognitive components such that $c_1 + c_2 = 4$. $r_1$ and $r_2$ are two random values in the range $[0,1]$. The main steps of PSO algorithm are presented in Algorithm 1.

The particle swarm optimization approach presented above works on continuous space. However, for some optimization problems, a discrete binary representation is better for the particles than a real representation. To deal with this kind of optimization problems, Kennedy and Eberhart proposed in [48] the Binary PSO (BPSO).

Algorithm 1

Initialize all parameters
Initialize particle’s positions and velocities
Set $p_{best}$ for each particle

**Compute $gbest$**

**repeat**

**for each particle do**

Calculate fitness value
Update $p_{best}$ if improvement

**end**

Update $gbest$

**for each particle do**

Calculate particle velocity according to (1)
Update particle position according to (2)

**end**

**until** maximum iterations is not attained

The position update equation is defined by:

$$
\begin{align*}
    p_i^t &= \begin{cases} 
    0 & \text{if } r \geq f(v_i^{t-1}) \\
    1 & \text{if } r < f(v_i^{t-1}) 
    \end{cases} \\
    f(v_i^{t-1}) &= \frac{1}{1 + \exp(-v_i^{t-1})}
\end{align*}
$$

where $r$ is a random value with range $[0, 1]$. To avoid saturation of the sigmoid function in (4), the velocity can be limited in the range $[-V_{max}, V_{max}]$.

Despite the fast convergence of PSO, it has been observed that the PSO algorithm can be trapped in a local minimum. This premature convergence occurs because particles fly to local, or near local, optima, therefore, the balance between the exploration process (searching of the search space) and exploitation process (convergence towards the optimum) is disturbed. In order to overcome this problem, the PSO technique can be combined with some other evolutionary optimization technique to yield an even better performance [43]. In this paper, PSO algorithm is hybridized with the SA algorithm. The hybrid algorithm takes both of fast searching ability in PSO and the advantages of probability jumping property of SA. Other applications of hybrid PSO and SA algorithm can be found [49, 50, 51, 52, 53].

B. Simulated Annealing (SA) algorithm

The original concept of SA was proposed by American physicist Metropolis et al. [54]. It is inspired from the annealing process in metallurgy, which is a method using heat and controlled cooling of a material to increase the size of its crystals. The heat destabilizes the atoms from their first positions and wanders randomly through states of higher energy. It has observed that when the cooling is slow, it gives the
atoms more chances of finding configurations with lower internal energy than the initial one.

Kirkpatrick et al. gives in [18] a mathematical model of the annealing process and uses this model for finding solutions for combinatorial optimized problems, and was the first literature to successfully utilize SA. The principle of SA is to accept solutions of worse quality than the current solution in order to escape from local optima. The probability of accepting such worst solutions is decreased during the search.

A standard SA procedure begins by generating random initial solution called current solution $cs$. At the initial stages, SA attempts to replace the current solution $cs$ by random solution $cs’$ that is generated by making a small random change in the current solution $cs$. The objective function $f(cs’)$ of the new solution is calculated and the metropolis acceptance rule is then used to determine if the generated solution should be accepted or not as the new current solution. A move to the new solution $cs’$ is made if it improves the current solution $cs$ that is $f(cs’)$ is better than $f(cs)$ otherwise it may be accepted with a probability that depends both on the difference between the corresponding function values $f(cs)$ and $f(cs’)$ and also on a global parameter $T$ (called temperature) that is gradually decreased during the cooling process. In rejection case, a new solution is generated and evaluated. Typically this step is repeated until the system reaches a solution that is good enough for the problem, or until a given number of iterations is reached. The probabilistic metropolis acceptance mechanism is defined by the following

$$ P = e^{-\Delta E/T} \tag{5} $$

where

$P$: Probability for acceptance.

$\Delta E$: The fitness difference between both the fitness of the new generated solution $cs’$ and the fitness of the current solution $cs$.

$T$: Temperature value.

The temperature $T$ is decreases during the search process; at the beginning of the search process, the solutions with the worst fitness value are accepted with a high probability and this probability is gradually decreased at the end of the search process.

The pseudo-code of SA algorithm is given in algorithm 2.

Algorithm 2
Let $cs$ the current solution
Evaluate $cs$ and store its fitness value $f(cs)$
Initialize $best_solution$ with the current solution $cs$
Initialize $T_0$ (initial temperature) and $T_f$ (final temperature)
Initialize $T$ to $T_0$
while $T>T_f$ do

for a fixed number of iterations do

Generate new solution $cs’$ in the neighborhood of the current solution $cs$
Evaluate the fitness value of the new solution $cs’$
$\Delta E = f(cs’) - f(cs)$
if $\Delta E < 0$ then $cs = cs’$
else if rand (0,1) $< e^{-\Delta E/T}$ then $cs = cs’$

end
Update $best_solution$ with the new generated solution $cs’$
end

$T = \alpha \times T$
End

The SA algorithm has a strong ability to find the local optimistic result. With its jump probability, it can avoid the problem of local optimum, but its ability of finding the global optimistic result is weak. In order to improve the results of SA algorithm, it can be used with other techniques like PSO to improve the results.

III. AMT-PSOSA: THE PROPOSED AUTOMATIC MULTILEVEL THRESHOLDING METHOD

In this section, we describe a new automatic image segmentation algorithm using multilevel thresholding based on a hybridization of PSO with SA.

Given a gray level image $I$ to be segmented into meaningful regions. Let there be $L$ gray level values lying in the range $\{0, 1, 2, \ldots, (L-1)\}$. For each gray level $i$, we associate $h(i)$ which represents the number of pixels having the $i^{th}$ gray level as a value. Therefore, the probability $p_i$ of the $i^{th}$ gray level is defined as $p_i = h(i)/N$

where $N = \sum_{i=0}^{L-1} h(i)$ denotes total number of pixels in the image $I$.

Suppose image $I$ is composed of $M+1$ regions. Hence, $M$ thresholds, $\{t_0, t_1, \ldots, t_M\}$ are required to achieve the subdivision of the image into regions: $C_0$ for $[0, \ldots, t_0]$ , $C_i$ for $[t_i, \ldots, t_{i-1}]$ , $\ldots$ , $C_M$ for $[t_M, \ldots, L]$ such that $t_0 \leq t_1 \leq \ldots \leq t_{M-1} \leq t_M$. The thresholding problem consists in choosing the set of optimal thresholds $(t_0^*, t_1^*, \ldots, t_M^*)$ that maximizes an objective function $f$ that is:

$$ (t_0^*, t_1^*, \ldots, t_M^*) = \arg \max (f(t)) \tag{6} $$

where $t$ is a potential solution in the feasible region.

The aim of AMT-PSOSA algorithm is to search both the numbers of thresholds as well as the optimal threshold values using the global search capability of PSO and its fast convergence combined with the efficiency of SA to escape from local minimum.

The framework of the AMT-PSOSA algorithm is given in algorithm 3 below.
Algorithm 3

Initialize all parameters (listed in Table 1)

for each particle $X_i$ do
  Initialize particle’s position
  Initialize particle’s velocities stochastically
  Initialize $pbest$
end

initialize $gbest$ position and $M_{best}$

iter ← 0

while (iter < iter_max) do
  for each particle $X_i$ do
    Update particle velocity
    Enforce velocity bounds
    Update particle’s position
    Enforce position bounds
    Evaluate particle’s fitness
    Record new $pbest$ value of the particle
  end

  Record new $gbest$ and $M_{best}$

  Select particles that are evolved by SA

  for each selected particle $X_i$ do
    $T ← T_0$
    $current\_solution ← pbest_i$
    $current\_fitness ← fitness(current\_solution)$
    while $(T > T_l)$ do
      iter_SA ← 0, // iteration of SA algorithm
      while (iter_SA < iter_SA_max) do
        Generate $neigh\_solution$
        Evaluate $neigh\_solution$‘s fitness
        if accept ($neigh\_solution$) then
          $current\_solution ← neigh\_solution$
        end
        iter_SA ← iter_SA + 1
      end
      $T = \alpha * T$
    end
  end

  Update $gbest$ position and $M_{best}$
  iter ← iter + 1
end

Segment the image using the optimal number of thresholds $M_{best}$ and the optimal thresholds values given by $gbest$.

A. Particle representation

Since in this paper, we have to optimize the threshold values and the number of thresholds, we have chosen to represent the particle with both real values and binary values. The real values are used to select the optimal threshold values which are updated by Real PSO algorithm. Binary values are used to select the optimal number of thresholds using the Binary PSO. The Real PSO and Binary PSO evolve in parallel contribute in the evaluation of the fitness function.

In this representation, a binary mask is associated with each threshold value. The value of this mask determines whether the threshold value is used in the thresholding process or not. The initial population $Pop = \{X_1, X_2, X_3, ..., X_{Np}\}$ is made up of $Np$ possible particles (solutions). For a user-defined maximum thresholds number $T_{max}$, a single particle $X_i$ is a vector of $T_{max}$ binary masks and $T_{max}$ real numbers that represent the $T_{max}$ thresholds values.

For a particle $X_i$, each probably threshold value $t_{ij} (j = 1 ... T_{max})$ is associated with a mask $\gamma_{ij}$. The threshold value $t_{ij}$ is taken into account if its corresponding mask is set to 1 that is: $\gamma_{ij} = 1$ and discarded otherwise that is $\gamma_{ij} = 0$. Therefore, for each particle, the total number of masks set to 1 gives the number of thresholds $M_i$ encoded in it, that is:

$$M_i = \sum_{j=1}^{T_{max}} \gamma_{ij}$$

For a $T_{max}$ thresholds problem, the position of a particle is formulated as:

$$X_i = \{\gamma_{i1}, \gamma_{i2}, ..., \gamma_{imax}, t_{i1}, t_{i2}, ..., t_{ir_{max}}\}$$

$$\gamma_{ij} \in \{0,1\}, \text{ and } 0 \leq t_{i1} \leq t_{i2} \leq ... \leq t_{ir_{max}} \leq L - 1$$

subject to $1 \leq i \leq Np$, $1 \leq j \leq T_{max}$

If after initialziation, it is found that no mask could be set to “1” in a particle (all threshold values are invalid), one random mask is selected and set to 1. Thus a valid particle should include at least one mask set to 1 in its position vector.

Two examples of the particle structure in the proposed approach are shown on Figure 1.

<table>
<thead>
<tr>
<th>masks_field</th>
<th>threshold_values</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1,0,10, 12, 45, 160, 225, 225]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>masks_field</th>
<th>threshold_values</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,0,1,0, 10, 50, 100, 210, 240]</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Two examples of the particle structure in the AMT-PSOSA algorithm.

In the first example, particle $X_i$ represents 3 classes (2 thresholds), and the associated thresholds values are 45 and 225. Thresholds values 12, 160 and 250 are invalid and not used to thresholding the image.

In the second example, particle $X_i$ represents 4 classes (3 thresholds), and the associated thresholds values are 10, 100 and 240. Thresholds values 50 and 210 are invalid and not used to thresholding the image.

The position of each particle is changed using the following rule: the threshold values are update using continuous PSO equations whereas the mask field is changed using equations of Binary PSO. In addition the velocity of each particle must lie within the interval [0,1].
B. Population initialization

To generate the initial population of particles, we use in this paper the random generation strategy until all particles in a population are created. Each individual particle \(X_t\) of the population is initialized by randomly chosen threshold values within the range \([g_{\min}, g_{\max}]\), where \(g_{\min}\) and \(g_{\max}\) are the minimum and the maximum gray levels in the image, respectively.

\[
t_i = g_{\min} + \text{rand()} \times (g_{\max} - g_{\min})
\]

\((i = 1, \ldots, Np, \text{ and } j = 1, \ldots, T_{\max})\)

In addition, \(M_j \left(1 \leq M_j \leq T_{\max} \right)\) masks are randomly chosen and their values are set to “1”.

C. Fitness evaluation

The fitness of a particle indicates the degree of goodness of the solution it represents. In this work the fitness of a particle is computed using the cost function ACT [36].

Let \(M_i, \omega_j, \mu_i, \text{ and } \mu\) be the number of thresholds encoded in a particle, the probability of the class \(C_i\), the mean grey level of the class \(C_i\), and the total mean grey level of the image, respectively:

\[
\omega_i = \sum_{j=t_i}^{L-1} p_j, \quad \mu_i = \sum_{j=t_i}^{L-1} ip_j \omega_i, \quad \mu = \sum_{j=0}^{L-1} ip_j
\]

where \(p_j\) is the normalized probability at level \(j\)

Let \(\sigma^2_w, \sigma^2_h\) and \(\sigma^2_T\) be the within-class variance, the between-class variance and the total class variance, respectively. They are given by the following expressions:

\[
\sigma^2_w(M) = \sum_{j=0}^{L-1} \sum_{i=0}^{L-1} (j - \mu)^2 p_j, \\
\sigma^2_h = \sum_{j=0}^{L-1} \omega \mu - \mu^2, \\
\sigma^2_T = \sum_{j=0}^{L-1} \sum_{i=0}^{L-1} (j - \mu)^2 p_j
\]

The ACT function proposed by Yen et al. is defined as follows:

\[
F(M) = \rho \times (\text{Disk}(M))^{1/2} + (\log_2(M))^2
\]

The \(\text{Disk}(M)\) represents the within-class variance and is defined as:

\[
\text{Disk}(M) = \sigma^2_T - \sigma^2_W(M)
\]

The first term of \(F(M)\) measures the cost incurred by the discrepancy between the thresholded image and the original image. The second term measures the cost resulted from the number of bits used to represent the thresholded image. In this equation, \(\rho\) is a positive weighting constant and it is set to 0.8 in this study as recommended in [36]. The number of thresholds is determined by counting the number of “1” in the mask field of the particle position and the corresponding threshold values are determined accordingly.

The fitness \(\text{fitness}(M)\) of a particle which encoded \(M\) thresholds is defined as:

\[
\text{fitness}(M) = \frac{1}{F(M)}
\]

(10)

When the algorithm converges, the found number of thresholds \(M^*\) and the corresponding found thresholds values are determined by the particle that has the maximum value of fitness function throughout the entire run.

D. SA-based local search

As a metaheuristic based method, PSO has a strong ability in finding the most optimistic result. Meanwhile, it can be trapped in a local optimum. SA has a strong ability in finding a local optimistic result, but its ability in finding the global optimistic result is weak [52]. Combining PSO and SA leads to the combined effect of two algorithms: the fast convergence of the PSO and its efficient global search and the efficient local search of SA and its ability to avoid local minima, which gives better results.

There are many ways to combine PSO with SA, but the best one is which take advantage of probability jumping property of SA without increasing the computational cost or altering the fast convergence ability of PSO. In this paper, we apply the jump property of SA to the process of updating the local best positions of selected particles in the swarm. Indeed, it would be helpful for the best particles to jump out the local minima, and as a result, the whole swarm would move to better positions. SA algorithm is applied for a particle if it is selected with an adaptive probability \(P_e\) as in [22]. Let \(g_{\text{best}}\) the global best fitness of the current iteration; \(p_{\text{best}}\) be the average fitness value of the population and \(p_{\text{best}}\) be the fitness value of the solution (particle) to be evolved. The expression of probability, \(P_e\) is given below:

\[
P_e = \begin{cases} 
\frac{k_1 \times (g_{\text{best}} - p_{\text{best}})}{(g_{\text{best}} - P_{\text{best}})} & \text{if } P_{\text{best}} > P_{\text{best}} \\
k_2 & \text{if } P_{\text{best}} \leq P_{\text{best}}
\end{cases}
\]

(11)

Here, values of \(k_2\) and \(k_4\) are kept equal to 0.5 [22]. This adaptive probability helps PSO to avoid getting stuck at local optimum.

The value of \(P_e\) increases when the fitness of the particle is quite poor. In contrast when the fitness of the particle is a good solution, \(P_e\) will be low.

SA starts with a local best solution \(p_{\text{best}}\) of a selected particle \(X_t\) and generates a random neighbor solution using a heuristic strategy. A valid threshold \(t_i\)
(γij = 1)in X, is chosen to undergo a perturbation operation with probability equal to 0.2. The threshold value is then modified as follows:

\[ t_{ij}^{\text{new}} = t_{ij}^{\text{old}} + \delta \ast (g_{\text{max}} - g_{\text{min}}) \]

where \( t_{ij}^{\text{new}} \in [g_{\text{min}}, g_{\text{max}}] \) (12)

where \( t_{ij}^{\text{new}} \) and \( t_{ij}^{\text{old}} \) represent the new and the old threshold value. \( \delta \) is a random number between [-1, 1]. Thus the solution encoded by the particle is reconfigured, although the number of thresholds belonging to it remains unaltered.

After the SA-based local search is completed, the gbest of the whole swarm should be updated if a solution with better quality is found during the local search.

IV. EXPERIMENTAL RESULTS

In order to assess the ability of the proposed algorithm to determine the number of thresholds and to achieve good quality image segmentation, synthetic and real images (Lena, Pepper, Baboon, Airplane, Hunter and Cameraman) have been used.

Figures 2 and 3 show these test images along with their histograms. All real images are of size (512 x 512) except the Pepper image which is of size (256 x 256).

Experiments have been carried out for several number of thresholds \( M \) ranging from 2 to 5. First experiments have been conducted to set the control parameters used in the AMT-PSOSA algorithm. Their values are gathered in Table 1.

The quality of the thresholded image has been evaluated using the uniformity measure defined by the following equation:

\[ U = 1 - \frac{2}{N \ast (g_{\text{max}} - g_{\text{min}})^2} \sum_{j=0}^{M} \sum_{i \in R_j} (g_i - \mu_j)^2 \]

(13)

Where

- \( M \) is the number of thresholds,
- \( R_j \) is the segmented region \( j \),
- \( g_i \) is the grey level of pixel \( i \),
- \( \mu_j \) is the mean of grey levels of those pixels in segmented region \( j \),
- \( N \) is the total number of pixels in the given image,
- \( g_{\text{max}} \) is the maximal grey level of the pixels in the given image,
- \( g_{\text{min}} \) is the minimum grey level of the pixels in the given image.

The domain of the uniformity measure is the range \([0,1]\). A higher value of uniformity means that the quality of the thresholded image is better.
A Fully Adaptive and Hybrid Method for Image Segmentation Using Multilevel Thresholding

Copyright © 2013 MECS


TABLE 1. PARAMETER SETTINGS FOR AMT-PSOSA ALGORITHM

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (Np)</td>
<td>50</td>
</tr>
<tr>
<td>Maximum Inertia weight (w_max)</td>
<td>0.9</td>
</tr>
<tr>
<td>Minimum Inertia weight (w_min)</td>
<td>0.4</td>
</tr>
<tr>
<td>Maximum velocity (V_max)</td>
<td>+1.0</td>
</tr>
<tr>
<td>Minimum velocity (V_min)</td>
<td>-1.0</td>
</tr>
<tr>
<td>Cognitive coefficient (C1)</td>
<td>1.429</td>
</tr>
<tr>
<td>Cognitive coefficient (C2)</td>
<td>1.429</td>
</tr>
<tr>
<td>Maximum number of iterations (iter_max)</td>
<td>1000</td>
</tr>
<tr>
<td>Initial temperature (T0)</td>
<td>1000</td>
</tr>
<tr>
<td>Final temperature (Tf)</td>
<td>1</td>
</tr>
<tr>
<td>Annealing rate (α)</td>
<td>0.99</td>
</tr>
<tr>
<td>maximum number of iteration (iter_SA_max)</td>
<td>1000</td>
</tr>
</tbody>
</table>

TABLE 2. PARAMETER SETTINGS FOR THE GA AND DMTBPSO ALGORITHMS.

<table>
<thead>
<tr>
<th>Parameter set for the GA algorithm</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (Np)</td>
<td>100</td>
</tr>
<tr>
<td>Crossover probability (P_c)</td>
<td>0.9</td>
</tr>
<tr>
<td>Mutation probability (P_m)</td>
<td>0.1</td>
</tr>
<tr>
<td>Maximum number of iterations</td>
<td>1000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter set for the DMTBPSO algorithm</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size (Np)</td>
<td>100</td>
</tr>
<tr>
<td>P_c</td>
<td>0.5</td>
</tr>
<tr>
<td>Inertia weight (w)</td>
<td>0.72</td>
</tr>
<tr>
<td>Maximum velocity (V_max)</td>
<td>255</td>
</tr>
<tr>
<td>Cognitive coefficient (C1)</td>
<td>2</td>
</tr>
<tr>
<td>Cognitive coefficient (C2)</td>
<td>2</td>
</tr>
<tr>
<td>Maximum number of iterations</td>
<td>1000</td>
</tr>
</tbody>
</table>

To further appreciate the performance of the proposed method, experimental results from the multilevel thresholding technique based on the recently developed automatic multilevel thresholding algorithm DMTBPSO [38] and on a classical PSO and GA algorithms have been examined and compared with the proposed method AMT-PSOSA. In order to make an objective comparison, the results through an exhaustive search method are also presented. GA-based method, PSO-based method and the exhaustive search method use Otsu’s criterion as objective function defined below:

\[ f_{\text{Otsu}}(M) = \sum_{k=0}^{M} \phi_k (\mu_k - \mu)^2 \]

In the following we will refer to them as ES-Otsu, GA-Otsu, PSO-Otsu and DMTBPSO. Parameter values of GA and DMTBPSO algorithms are presented in Table 2. We employ the best possible parameter settings recommended for DMTBPSO algorithm in [38]. PSO-Otsu is implemented using the same parameter values of AMT-PSOSA.

For GA-Otsu, PSO-Otsu and ES-Otsu, the number of thresholds must be given in advance. In order to compare them with the proposed AMT-PSOSA, we must reformulate them in such a way that the number of thresholds is extracted automatically. As in [7], GA-Otsu, PSO-Otsu and ES-Otsu are executed in this paper with varying gradually the number of thresholds. The optimal threshold number, which optimizes the cost function ATC defined, is recorded.

The main steps of this algorithm are given in algorithm 4 below [7].

Algorithm 4

1. \( M = 1 \)
2. Apply the ES-Otsu method or the GA-Otsu method or PSO-Otsu method using the number \( M \) of thresholds
3. Compute the value of the cost function \( F(M) \) by using the output optimal thresholds values
4. If \( F(M) \) is decreased, set \( M = M + 1 \) and go to step 2 else go to step 5
5. Output the threshold number \( M^* \) and the optimal thresholds values obtained from the corresponding multilevel thresholding method.

Moreover, in order to investigate the effects of the hybridization of the PSO with SA, we have compared AMT-PSOSA with a classical PSO based multilevel thresholding method (AMT-PSO) which uses the same particle representation scheme and fitness function as the AMT-PSOSA.

Table 3 summarize the obtained results for the synthetic and real images using the proposed AMT-PSOSA algorithm and the other multilevel thresholding algorithms including the optimal number of thresholds, the corresponding best ATC function value and the uniformity measure related to the synthetic image and the six real test images. Also the computing time of each algorithm is given for each image. From this table, we can draw the following conclusions: The computational time of the exhaustive search method (ES-Otsu) is very expensive and increases sharply when the number of the thresholds increases. AMT-PSO and AMT-PSOSA algorithms provide the same threshold number than GA-Otsu, PSO-Otsu and ES-Otsu algorithms, for all test images. The results of AMT-PSO and AMT-PSOSA algorithms are better than those of DMTBPSO algorithm. AMT-PSOSA is better that AMT-PSO which means that performance is greatly improved by the SA-local search procedure. In most cases, the performance of the proposed method, evaluated through the optimal threshold number \( M^* \), the cost function \( F(M) \) and the uniformity measure \( U \), are close to those of ES-Otsu. From computation time view point, the proposed method gives a reasonable CPU time, even though the SA-local search is added.
As PSO is a stochastic optimization algorithm, the stability of the PSO-based strategies has also been assessed by gathering optimal number of thresholds and the corresponding objective function \( F(M) \) values over several runs. The variation of the results through different runs gives an indication of the stability of the used algorithm which is influenced by its search abilities. The more results are close to each other, the more stable is the algorithm. The mean \( \mu \) and standard deviation \( \sigma \) are defined as:

\[
\mu = \frac{1}{k} \sum_{i=1}^{k} \sigma_i, \quad \sigma = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (\sigma_i - \mu)^2}
\]

where \( k \) is the number of runs for each stochastic algorithm (\( k = 50 \)), \( \sigma_i \) is the best objective value obtained by the \( i^{th} \) run of the algorithm. Higher standard deviation values show that the algorithm is unstable.
A Fully Adaptive and Hybrid Method for Image Segmentation Using Multilevel Thresholding

The mean and the standard deviation values of the number of thresholds, the corresponding objective values and the uniformity measure through 50 runs for DMTBPSO, AMT-PSO and AMT-PSOSA algorithms has been recorded in Table 4 for the six real test images.

As shown on the Table 4, the proposed algorithm achieves mean values that are closest to optimal ones compared to any of the other strategies. Furthermore, the achieved standard deviation values indicate the superiority of AMT-PSOSA over the other algorithms in terms of stability.

V. CONCLUSION

In this paper, we have proposed a new automatic multilevel thresholding algorithm based on the hybridization of the Particle Swarm Optimization with the Simulated Annealing. The new developed hybrid technique makes full use of the exploration ability of PSO and the exploitation ability of SA and offsets the weaknesses of each. AMT-PSOSA uses both real and discrete values for the particle representation in order to determine the appropriate number of thresholds as well as the optimal threshold values using the ATC function. Experimental results over synthetic and real test images show the efficiency of AMT-PSOSA over exhaustive search method, GA based method, PSO based method and the recently proposed DMTBPSO algorithm.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments.


<table>
<thead>
<tr>
<th>Images</th>
<th>Algorithms</th>
<th>M</th>
<th>ATC function</th>
<th>Uniformity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>deviation</td>
<td>mean</td>
</tr>
<tr>
<td>Lena</td>
<td>DMTBPSO</td>
<td>4.15</td>
<td>0.589</td>
<td>11.01742</td>
</tr>
<tr>
<td></td>
<td>AMT-PSO</td>
<td>3.75</td>
<td>0.046</td>
<td>10.13291</td>
</tr>
<tr>
<td></td>
<td>AMT-PSOSA</td>
<td>4</td>
<td>0</td>
<td>9.95813</td>
</tr>
<tr>
<td>Pepper</td>
<td>DMTBPSO</td>
<td>5.25</td>
<td>0.433</td>
<td>12.24612</td>
</tr>
<tr>
<td></td>
<td>AMT-PSO</td>
<td>3.5</td>
<td>0.049</td>
<td>11.01522</td>
</tr>
<tr>
<td></td>
<td>AMT-PSOSA</td>
<td>4</td>
<td>0</td>
<td>10.75325</td>
</tr>
<tr>
<td>Mandril</td>
<td>DMTBPSO</td>
<td>4.06</td>
<td>0.661</td>
<td>10.82737</td>
</tr>
<tr>
<td></td>
<td>AMT-PSO</td>
<td>3.1</td>
<td>0.030</td>
<td>9.63651</td>
</tr>
<tr>
<td></td>
<td>AMT-PSOSA</td>
<td>3</td>
<td>0</td>
<td>9.60531</td>
</tr>
<tr>
<td>Jetplane</td>
<td>DMTBPSO</td>
<td>4.05</td>
<td>0.668</td>
<td>10.27023</td>
</tr>
<tr>
<td></td>
<td>AMT-PSO</td>
<td>2.95</td>
<td>0.022</td>
<td>9.44559</td>
</tr>
<tr>
<td></td>
<td>AMT-PSOSA</td>
<td>3</td>
<td>0</td>
<td>9.37127</td>
</tr>
<tr>
<td>Hunter</td>
<td>DMTBPSO</td>
<td>4.15</td>
<td>0.572</td>
<td>10.88537</td>
</tr>
<tr>
<td></td>
<td>AMT-PSO</td>
<td>3</td>
<td>0</td>
<td>10.06812</td>
</tr>
<tr>
<td></td>
<td>AMT-PSOSA</td>
<td>3</td>
<td>0</td>
<td>9.95374</td>
</tr>
<tr>
<td>Cameraman</td>
<td>DMTBPSO</td>
<td>4.9</td>
<td>0.72</td>
<td>11.44428</td>
</tr>
<tr>
<td></td>
<td>AMT-PSO</td>
<td>2.9</td>
<td>0.061</td>
<td>10.19008</td>
</tr>
<tr>
<td></td>
<td>AMT-PSOSA</td>
<td>3</td>
<td>0</td>
<td>10.10832</td>
</tr>
</tbody>
</table>

REFERENCES


[42] Poli, R, Analysis of the publications on the applications of particle swarm optimisation, Journal of Artificial Evolution


[50] Fang, L., Chen,P., & Liu, S. Particle swarm optimization with simulated annealing for tsp, in Proceedings of the 6th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED '07), (2007) 206–210, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wis, USA.


OUADFEL Salima, received her state engineer degree, master degree in computer science from Mentouri University in Constantine Algeria and she received a PhD in Computer Science from the University of Batna, Algeria, in 2007. She is currently an Associate Professor at the computer science department of Mentouri University and a researcher at MISC laboratory Constantine City. Her current research includes natural inspired metaheuristics and their applications for image processing.

MESHOUF Souham, received her state engineer degree, master degree and State doctorate degree in computer science from Mentouri University in Constantine Algeria. She is currently an associate professor at the computer science department and a researcher at MISC laboratory Constantine City. Her current interests include computational intelligence and its applications, complex systems, bioinformatics, pattern recognition and image analysis and understanding.