Available online at www.sciencedirect.com
ScienceDirect
Computer Vision and Image Understanding 109 (2008) 163–175
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation

Kamal Hammouche [a], Moussa Diaf [a], Patrick Siarry [b,*]

[a] Université Mouloud Mammeri, Département Automatique, Tizi-Ouzou, Algérie
[b] Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, EA 3956), Université Paris XII Val de Marne, 61 avenue du Général de Gaulle, 94010 Créteil, France

Received 15 December 2005; accepted 4 September 2007
Available online 20 September 2007

Abstract

In this paper, a multilevel thresholding method which allows the determination of the appropriate number of thresholds as well as the adequate threshold values is proposed. This method combines a genetic algorithm with a wavelet transform. First, the length of the original histogram is reduced by using the wavelet transform. Based on this lower resolution version of the histogram, the number of thresholds and the threshold values are determined by using a genetic algorithm. The thresholds are then projected onto the original space. In this step, a refinement procedure may be added to detect accurate threshold values. Experiments and comparative results with multilevel thresholding methods over a synthetic histogram and real images show the efficiency of the proposed method.

Keywords: Thresholding; Image segmentation; Genetic algorithm

1. Introduction

Image thresholding is widely used as a popular tool in image segmentation. It is useful to separate objects from background, or discriminate objects from objects that have distinct grey levels. Thresholding involves bi-level thresholding and multilevel thresholding. Bi-level thresholding classifies the pixels into two groups, one including those pixels with grey levels above a certain threshold, the other including the rest. Multilevel thresholding divides the pixels into several classes. The pixels belonging to the same class have grey levels within a specific range defined by several thresholds. Both bi-level and multilevel thresholding methods can be classified into parametric and non-parametric approaches. The non-parametric approach is based on a search of the thresholds optimizing an objective function, such as the between-class variance (Otsu's function) [1], entropy (Kapur's function) [2]. In the parametric approach, the grey level distribution of each class has a probability density function that is assumed to obey a given distribution. An attempt to find an estimate of the parameters of the distribution that best fit the given histogram data is made by using the least-squares method. It typically leads to a nonlinear optimization problem, of which solving is time-consuming. A great number of thresholding methods of parametric or non-parametric type have been proposed in order to perform bi-level thresholding [3–6]. They are extendable to multilevel thresholding as well. However, the amount of thresholding computation significantly increases with this extension. To overcome this problem, several techniques have been proposed. In [7], the Otsu's function is modified in order to be optimized by a fast recursive algorithm along with a look-up-table. In [8], Lin has proposed a fast thresholding computation using the Otsu's function. His approach is based on the search of the thresholds where zero first partial derivatives of the Otsu's function occur by using a successive substitution technique. In [9], the resolution of the histogram is

* Corresponding author.
E-mail addresses: kamal_hammouche@yahoo.fr (K. Hammouche), diaf_moussa@yahoo.com (M. Diaf), siarry@univ-paris12.fr (P. Siarry).

1077-3142/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2007.09.001

reduced using the wavelet transform. Using the reduced histogram, the optimal thresholds are determined by optimizing the Otsu's function based on an exhaustive search. The selected threshold values are then expanded to the original scale.

The pairwise nearest neighbour method previously used in hierarchical clustering has been adapted to multilevel thresholding [10]. Among all possible thresholds, the one that increases the objective function is selected. The selected threshold is then removed and the means of the classes are updated. The selection and removing processes are repeated until the desired number of thresholds is reached. Another hierarchical clustering method used to solve the multilevel thresholding problem was recently proposed [11]. Initially, each non-empty grey level of the histogram is considered as a separate mode representing a cluster. Then the similarities between adjacent clusters are computed and the most similar pairs are merged. The estimated thresholds defined as the highest grey levels of the clusters are obtained by iterating this operation until the desired number of clusters is found.

Another fast multilevel thresholding technique has been proposed by Yin [12]. The thresholds optimizing the Otsu's or the Kapur's functions are searched by using an iterative scheme. This technique starts from random initial thresholds. Then, these thresholds are iteratively adjusted to improve the value of the objective function. This improvement process stops when the value of the objective function does not increase between two consecutive iterations. The implementation of this method is similar to the one presented by Luo and Tian, where the Kapur's function is maximized by using the Iterated Conditional Modes (ICM) algorithm [13].

Several techniques using genetic algorithms (GAs) have also been proposed to solve the multilevel thresholding problem [14–20]. GAs are optimization algorithms based on the mechanics of natural selection and natural genetics. Yin [14] has proposed a fast thresholding method based on a genetic algorithm, where the objective function is similar to Otsu's or Kapur's function. The solution is encoded as a binary string T such that $T = t_1, t_2, \ldots, t_{k-1}$, where t_i, the value of ith threshold, has $\log_2(L)$ bits and a value within $[0, L-1]$. This same technique has been used in [15,16]. In [15], the objective function is similar to Kapur's function and in [16] it is assimilated to the relative entropy [21]. Chang and Yan have proposed a thresholding technique using a Conditional Probability Entropy (CPE) based on Bayesian theory. They have employed a GA to maximize the CPE in order to determine the thresholds [17]. CPE considers that the pixels with the same grey level in an image may belong to different classes with different probabilities. An optimal classification method for these pixels is to classify them in the class with higher probability. The chromosome structure in their GA is based on the conditional probability function employed, with two parameters for each class. The chromosome is then constituted of $2k$ parameters and each parameter has $\log_2(L)$ bits. In [18], a multilevel optimal thresholding technique based on an approach using a GA has been proposed. The intensity distributions of objects and background in an image are assumed to be Gaussian distributions with distinct means and standard deviations. The histogram of a given image is fitted to a mixture Gaussian probability density function. The GA is used to estimate the parameters in the mixture density function so that the square error between the density function and the actual histogram is minimal. Tao et al. use a genetic algorithm in order to find the optimal combination of all the fuzzy parameters by maximizing the fuzzy entropy [19]. The fuzzy parameters describe the membership functions of three parts of the image, namely dark, grey and white parts. The optimal parameters are then used to define two threshold values. More recently, Bazi et al. use a genetic algorithm to provide the initial parameters to the expectation-maximization (EM) algorithm. The parameters are the objects and background classes, which are assumed to follow generalized Gaussian distributions [20]. Each chromosome is viewed as a vector representing statistical parameters defining the mixture of the class distributions.

Beside GAs, particle swarm optimization (PSO) is another latest evolutionary optimization technique which was used for the multilevel thresholding [22–24]. In [22], PSO is used in conjunction with the simplex method for the Gaussian curve fitting and for the Otsu's function optimization. The method presented in [23] uses PSO to optimize the cross entropy [25] and in [24] the histogram is approximated by a mixture Gaussian model. The Gaussian's parameter estimates are iteratively computed by combining PSO with EM algorithm. Like the genetic algorithms and PSO, simulated annealing algorithm has been exploited to optimize the modified Otsu's function [26].

The main problem associated with the aforementioned methods is that the number of thresholds with which the grey level image should be segmented cannot be automatically determined. To overcome this problem, Yen et al. proposed a new criterion for multilevel thresholding, called Automatic Thresholding Criterion (ATC) [27]. This criterion is used with a sequential dichotomization technique [27,28]. In this strategy, the histogram is dichotomized in two distributions by using a bi-level thresholding and the distribution with the largest variance is further dichotomized in two more distributions by applying the same bi-level thresholding. This dichotomization process is repeated until a cost function (ATC) reaches its minimum. In [29], the same dichotomization process is repeated until a cost function derived from Otsu's function becomes higher than a specified value. The dichotomization techniques are faster algorithms; unfortunately, they are sub-optimal techniques, they do not allow providing the optimal threshold values.

In this paper, we propose a fast multilevel thresholding technique based on a GA able to determinate the appropriate number of thresholds, as well as appropriate threshold values, by optimizing ATC proposed by Yen et al. The proposed GA uses a new string representation of the chromo-

some, which is different from those previously mentioned. It is combined with a wavelet transform-based technique in order to reduce the time computation. This proposed method can be considered as similar to the multilevel thresholding method presented by Kim et al. [9], except that the proposed GA is used to search for the optimal thresholds. The using of GAs has many advantages over traditional searching techniques [30]. Particularly, GA-based methods are global searching techniques capable, most often, to prevent from trapping into locally optimal solutions. Another advantage is that the GA-based methods can become faster through parallel implementations.

In the next section, the proposed multilevel thresholding technique using a GA is described. In Section 3, the performance of the proposed method is tested on several examples and is compared with other multilevel thresholding methods. Concluding remarks are given in Section 4.

2. The proposed multilevel thresholding method

Suppose that an image I having N pixels with L grey levels $L = \{0, 1, \ldots, L-1\}$ is to be classified into k classes $(C_1, C_2, \ldots, C_k)$ with the set of thresholds $T = \{t_1, t_2, \ldots, t_{k-1}\}$. For convenience, we assume two other thresholds $t_0 = 0$ and $t_k = L-1$. The histogram of the image is indicated by $h(i)$, $i = 0, \ldots, L-1$, where $h(i)$ represents the number of pixels with the grey level i.

The proposed genetic thresholding technique is based on a standard GA. It allows the determination of the number of thresholds as well as appropriate threshold values. The major steps of this method are summarized in Algorithm 1.

Algorithm 1. *Main steps of the proposed thresholding technique*

1. Compute the histogram of the image.
2. Reduce the length of the histogram.
3. Generate an initial population.
4. Store the best string A^* with the best fitness in a separate location.
5. Apply the learning strategy to improve the fitness value of A^*.
6. Generate the next population by performing selection, crossover and mutation operations.
7. Compare the best string A of the current population with A^*. If A has a better fitness value than A^*, then replace A^* with A.
8. Go to step 3 if the desired number of generations is not reached.
9. Expand the best thresholds.
10. Refine the expanded thresholds.

2.1. Reduction of the histogram length

Before searching for thresholds through the GA, the length of the histogram must be reduced in order to accelerate the convergence of the GA. The histogram reduction is performed by using the wavelet transform technique [9]. The original histogram is decomposed into two signals at the next lower level. One signal is the *trend signal*, or the approximation signal, while the other signal is the *detail signal*. At the lower level, the trend signal gives the reduced dimension version of the original histogram, which still contains the overall characteristics of the original histogram.

The wavelet transform at a level r ($r \in Z$) is performed with decimation operation by 2^r after the convolution of the histogram $h(i)$ with the wavelet function $\Psi(t)$ and scaling function $\Phi(t)$.

$$h^r(j) = WT^r[h(i)], \quad r \in h^r(j) + h^r_W(j),$$

where $h^r(j)$ is the trend of the original histogram and $h^r_W(j)$ is the detail of the original histogram at the rth level. Each trend signal at level r is decomposed into two reduced dimension signals at level $r+1$, i.e.:

$$h^r(j) = h^{r+1}(j) + h^{r+1}_W(j).$$

Hence, for a level r, the length of the reduced histogram $h^r(j)$ is denoted L^r, such that $L^r = L/2^r$.

2.2. String representation

In the proposed thresholding GA, the chromosome is encoded as a binary string A of the same size L^r of the reduced histogram, such that $A = a_0, a_1, \ldots, a^r_{L-1}$, where the character a_i is equal to 0 or 1. a_i indicates the peak or the valley of the histogram. If $a_i = 0$, then $(i, h^r(i))$ is a valley, else it is a peak. The position i for which $a_i = 0$ indicates the value of a threshold. Hence, the number of zero-bits occurred in A indicates the number of thresholds.

Example . Let $L^r = 16$ and consider the following string: $A = 1,1,0,1,1,1,0,1,1,1,0,1,1,1,1,1$.

The number of thresholds is equal to 3, since A contains three zero-bits. The three thresholds are then $t_1 = 2$, $t_2 = 6$ and $t_3 = 10$, respectively. They correspond to the positions of zero-bits in the string.

2.3. Fitness computation

To allow determining the optimal number of thresholds as well as the optimal thresholds values, the fitness of a string is computed using the cost function ACT proposed by Yen et al. [27].

Let P_i, m_i and m be 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:

$$P_i = \sum_{j=t_{i-1}}^{t_i-1} p_j, \quad m_i = \frac{\sigma_i}{P_i}, \quad \sigma_i = \sum_{j=t_{i-1}}^{t_i-1} jp_j, \quad m = \sum_{j=0}^{L^r-1} jp_j,$$

where $p_j = h^r(j)/N$ is the normalized probability at level j.

Let σ^2_{W}, σ^2_{B} and σ^2_{T} be the within-class variance, the between-class variance and the total class variance, respectively. They are given by the following expressions:

$$\sigma_W^2(k) = \sum_{i=0}^{k-1} \sum_{j=t_i}^{t_{i+1}-1} (j - m_{i+1})^2 p_j,$$

$$\sigma_B^2(k) = \sum_{i=1}^{k} P_i (m_i - m)^2, \quad \sigma_T^2 = \sum_{j=0}^{L^r-1} (j-m)^2 p_j.$$

The fitness $F(k)$ of a string is:

$$F(k) = \rho * (\mathrm{Disk}(k))^{1/2} + (\log_2(k))^2.$$

Here $\mathrm{Disk}(k)$ represents the within-class variance $\mathrm{Disk}(k) = \sigma_W^2(k) = \sigma_T^2 - \sigma_B^2(k)$.

The first term of $F(k)$ 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, ρ is a positive weighting constant. The $(k-1)$ number of thresholds is determined by counting the number of zero-bits in the string and the threshold values are determined by the positions occupied by these zero-bits in the string. The function $F(k)$ has a unique minimum, which is an important advantage. The optimum class number k^* and the (k^*-1) best thresholds can be determined by the following equation:

$$F(k^*) = \min\{F(k)\}.$$

2.4. Population initialization

The genetic algorithm starts with a randomly generated population of solutions. The initial population is of fixed size P: A_1, A_2, …, A_P. For each string i in the population ($i = 1, 2, \ldots, P$), L^r bits (0 or 1) are randomly generated.

2.5. Learning strategy

In the proposed GA, the best string with the best fitness value in each generation is copied to an isolated place in order to record the current best solution. Then, before starting the next generation, the best string can improve its fitness value by looking at the neighbouring values. Let the best string of the current population be $T = \{t_1, t_2, \ldots, t_{k-1}\}$, it has a very high probability to improve its fitness value by ranging (t_1 to $t_1 - 1$, $t_1 + 1$), (t_2 to $t_2 - 1$, $t_2 + 1$), …, (t_{k-1} to $t_{k-1} - 1$, $t_{k-1} + 1$). As it was proved in [14], this simple movement may save many generations. This learning strategy does not affect the behaviour of the GA and can accelerate the movement towards the optimal thresholds.

2.6. Genetic operations

The current population evolves to the next population of the same size using three standard genetic operations: selection, crossover and mutation. The evolution process is iterated until a specified number of generations is reached.

2.6.1. Selection

Selection is a process which mimics the natural survival of the fittest creatures. Each string has a fitness value obtained by evaluating the fitness function. The probability of each string to be selected is proportional to its fitness value. In this paper, the tournament selection procedure is performed as follows: two strings A' and A'' of the current population are randomly selected and the string with the best fitness value is chosen to belong to the mating pool. This procedure is repeated, until filling a mating pool of the same size P that the population.

2.6.2. Crossover

The crossover operator chooses two strings A' and A'' of the current population. Single crossover is applied as follows: generate a random integer number q within $[0, L^r - 1]$ and create two offspring by swapping all the characters of A' and A'' after position q. The crossover is performed with the crossover probability P_c. A random number can be generated within $[0, 1]$, associated with each pair of strings selected in the mating pool. If the random value is less than P_c, then the crossover is performed, otherwise no crossover is performed.

2.6.3. Mutation

Mutation is an occasional alteration of a character with a low probability P_m. The proposed mutation is performed in two steps. First, a standard mutation is used in the following way: for each string produced by crossover operation, a random value is generated within $[0, 1]$. If the random number is less than P_m, then a character at a random position is chosen and its value is altered (i.e. one changes 0 to 1, or 1 to 0).

However, the crossover and standard mutation operators can create strings with several successive zero-bits. In this situation, several thresholds with successive values appear. To overcome this undesirable situation, a solution consists in keeping, among successive zero-bits, only the first one, and in mutating the remaining successive zero-bits.

Example . Let $L^r = 16$ and consider the following string: $A = 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1$.

The number of thresholds is equal to 10, since A contains 10 characters with zero value. The values of the 10 thresholds are, respectively, $t_1 = 2$, $t_2 = 3$, $t_3 = 6$, $t_4 = 7$, $t_5 = 8$, $t_6 = 10$, $t_7 = 11$, $t_8 = 12$, $t_9 = 13$ and $t_{10} = 14$. They form three threshold series with successive values: (t_1, t_2), (t_3, t_4, t_5) and $(t_6, t_7, t_8, t_9, t_{10})$. By applying the proposed mutation, A becomes $A = 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1$ and the number of thresholds is reduced to 3. The corresponding threshold values are $t_1 = 2$, $t_2 = 6$ and $t_3 = 10$, respectively.

2.7. Expansion of the best thresholds

Because of the reduced dimension of the histogram, the threshold values t_i determined by the GA are at lower level,

i.e. $t_i \in [0\ L]$. Thus, the thresholds determined by the GA must be expanded in the original space. In this case, each threshold t_i is multiplied by a factor 2^r, as follows [9]:

$$\hat{t}_i = t_i x 2^r, \quad \text{for } i = 1, \ldots, k-1, \text{ such that } \quad \hat{t}_i \in [0\ L].$$

2.8. Refinement of the expanded thresholds

Since a GA is a stochastic technique, the expanded threshold values change at each run of the algorithm and are generally located in a range around the desired optimal threshold values. In these conditions, a refinement procedure can be added to obtain stable and more accurate threshold values. The refinement procedure that we have used is described by Algorithm 2 where only very few iterations are needed before convergence.

Algorithm 2. *Refinement procedure*

1. Compute the mean grey level $m_i(s)$ of the class C_i, $i = 1, 2, \ldots, k-1$ where s denotes the time of iteration.
2. Update the value of $t_i(s)$ according to the following equation:

$$t_i(s+1) = \frac{m_i(s) + m_{i+1}(s)}{2}.$$

3. Repeat steps 1 and 2 until the iteration converges, i.e. $t_i(s+1) = t_i(s)$ $i = 1, 2, \ldots, k-1$.

3. Experimental results and comparison with other methods

In this section, we will evaluate the performance of the proposed multilevel thresholding method using a GA. Some experiments with a synthetic histogram and real images are presented to illustrate the key features of the proposed method in the determination of the number of thresholds and its efficiency for thresholding computation. Comparisons are performed with results provided by other multilevel thresholding methods.

Eight well-known images named *Lena*, *Blood*, *Peppers*, *House*, *Airplane*, *Lac*, *Boats* and *Bridge* with 256 grey levels are used. The seven first images have a size (256×256), while the *Bridge* image is (512×512). These images are gathered in Figs. 2 and 3 shows their respective histograms. We add an artificial histogram in order to objectively show the accuracy of the proposed approach in the determination of the appropriate number of thresholds and the computation of threshold values. The artificial histogram is constructed with $k = 5$ distributions (cf. Fig. 1(a)). Each distribution is assumed to be Gaussian. The probability distribution is described below:

$$h(i) = \sum_{j=1}^{5} \frac{1}{\sqrt{2\pi}} \left[\frac{P_j}{\sigma_j} \exp(-(i - m_j)^2 / 2\sigma_j^2) \right].$$

The probability distribution is plotted in Fig. 2(a), where:

$$P_1 = 12,000, \quad m_1 = 40, \quad \sigma_1 = 10,$$
$$P_2 = 5500, \quad m_2 = 90, \quad \sigma_2 = 10,$$
$$P_3 = 6000, \quad m_3 = 150, \quad \sigma_3 = 20,$$
$$P_4 = 10,000, \quad m_4 = 200, \quad \sigma_4 = 10,$$
$$P_5 = 20,000, \quad m_5 = 230, \quad \sigma_5 = 10.$$

The proposed multilevel thresholding technique using a GA is implemented with the following parameters: $P_c = 0.9$, $P_m = 0.001$. The size P of the population depends on the length of the chromosome and on the resolution level r used in the wavelet transform. In all our experiments, P is expressed in function of the resolution level r as $300/2^r$ and the GA is executed for a maximum of $N_g = 100$ generations. The wavelet transform is performed with 'db1' wavelet. Additional results are presented in order to investigate the influence of the resolution level r. The choice of the constant ρ in the objective function is crucial. After several simulations using a large set of test images, we have found that ρ must depend on the resolution level r. It is fixed to 0.5×2^r for all test images expect for the *Bridge* image where it is fixed to 2^r.

Table 1 shows the number of thresholds, the threshold values and the CPU time achieved by the proposed method, when the level r of the transform wavelet varies from 0 to 3. $r = 0$ corresponds to the original space of the histogram (without reduction). Our experiments are performed on a Pentium IV-2.8 GHz PC with 256 Mb RAM. Without the reduction of the length of the histogram, the length of the chromosome can be large and the

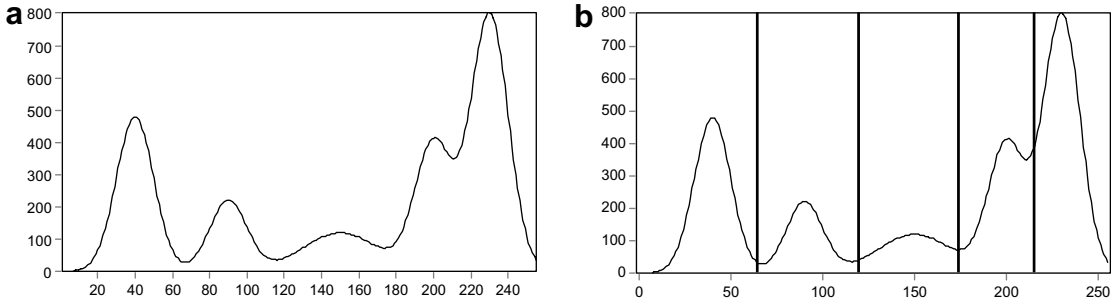

Fig. 1. Artificial histogram.

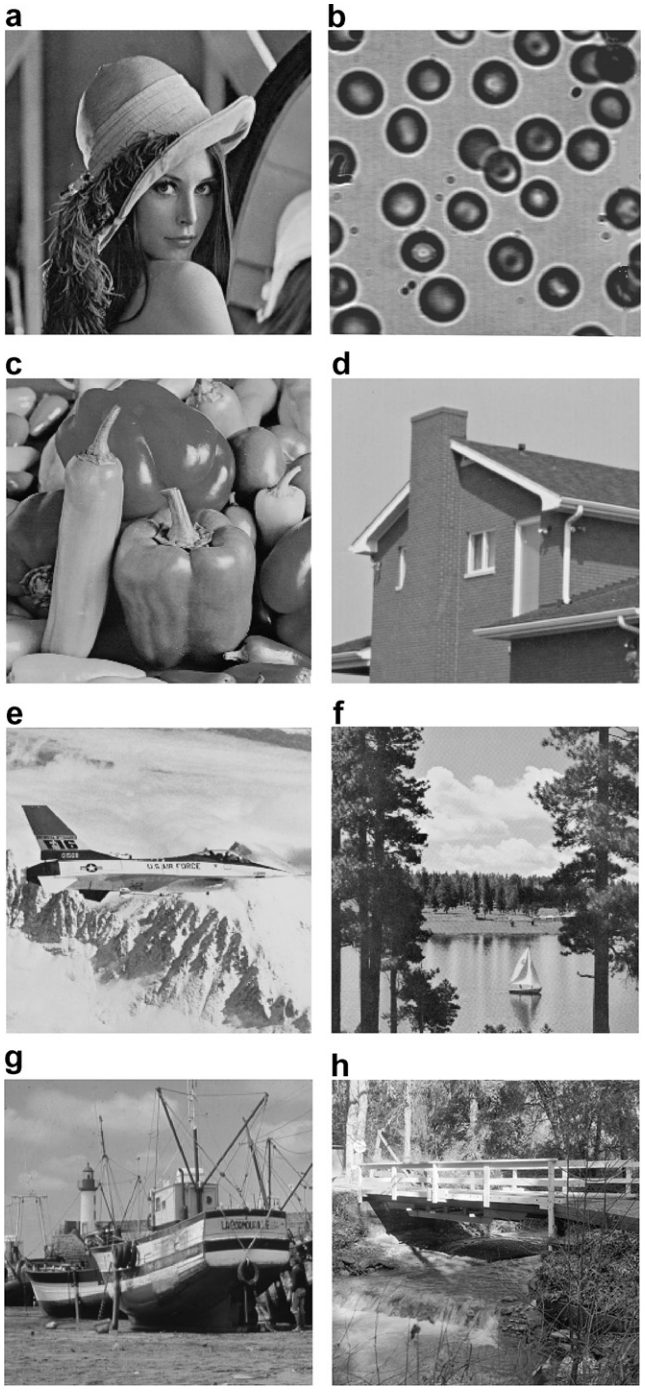

Fig. 2. Test images. (a) Lena. (b) Blood. (c) Peppers. (d) House. (e) Airplane. (f) Lac. (g) Boats. (h) Bridge.

population size P must be large, in order to cover the space of search solutions. Consequently, the computation time of the GA increases. The reduction in the length of the histogram enables reducing the computation time, while the number of thresholds and the threshold values are not really affected by this reduction. It can be pointed out that the computation time is not affected by the number of classes.

For the artificial histogram, the proposed multilevel thresholding technique using a GA with $P = 38$ and $r = 3$ converges to $k^* = 5$, with the four thresholds $T^* = (65–120–174–215)$. The corresponding thresholds are displayed in Fig. 1(b). The results of the proposed method are consistent with the number of classes in the artificial histogram and the corresponding threshold values are located on the valleys of the histogram.

Fig. 4 shows the threshold position on the histograms for the test images of Fig. 2, and the corresponding thresholded images are displayed in Fig. 5. For *Lena*, *Peppers*, *House*, *Airplane*, *Lac*, *Boats* and *Bridge* images, almost all important components are preserved in the thresholded images, since the homogeneous regions are well apparent and their outlines are very clear. While for *Blood* image, the modes of the histogram are not well apparent, however, the visual inspection of the thresholded image shows that the pixel classes are quite well separated, since the major regions (blood cells, blood plasma, cell nuclei and cell membrane) are well delimited (Fig. 5(b)).

3.1. Comparison with other multilevel thresholding methods

In this sub-section, some experimental results from different multilevel thresholding methods will be examined and compared with the proposed method, over the eight real images and the artificial histogram. All methods use the cost function $F(k)$ to decide whether the number of thresholds has reached the optimal value or not. The smaller the value of $F(k)$, better is the segmentation [4,27,28]. The quality of thresholded images can also be evaluated through the uniformity measure, which is widely mentioned in the literature [3,12,14]:

$$U = 1 - 2(k-1)\frac{\sum_{j=0}^{k-1}\sum_{i\in R_{j_i}}(f_i - m_j)^2}{N(f_{\max} - f_{\min})^2},$$

where: $(k-1)$ is the number of thresholds,

R_j is the segmented region j,
f_i is the grey level of pixel I,
m_j is the mean of the grey levels of those pixels in segmented region j,
N is the number of total pixels in the given image,
$f_{\max}$ is the maximal grey level of the pixels in the given image,
$f_{\min}$ is the minimal grey level of the pixels in the given image.

The value of the uniformity measure is between 0 and 1. A higher value of uniformity means that the quality of the thresholded image is better.

3.1.1. Multilevel thresholding methods used for comparison

Two multilevel thresholding techniques based on a GA and an iterative scheme are introduced and compared with the proposed method. In order to make an objective comparison, the results through the exhaustive search method are also presented.

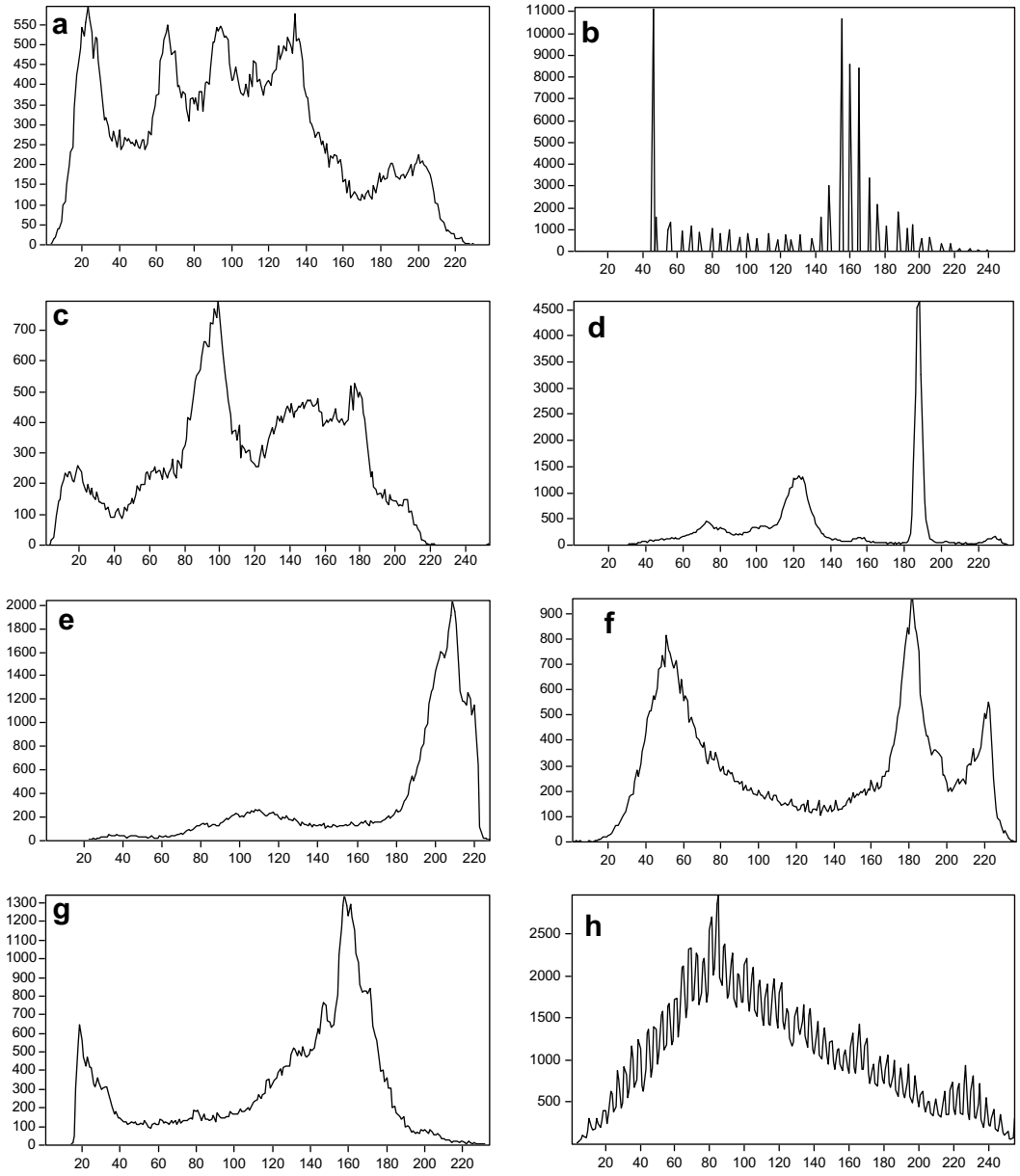

Fig. 3. Grey level histograms of test images. (a) Lena. (b) Blood. (c) Peppers. (d) House. (e) Airplane. (f) Lac. (g) Boats. (h) Bridge.

3.1.1.1. Multilevel thresholding method using a genetic algorithm. The optimal thresholding can be performed through a search of the thresholds optimizing an objective function $G(k)$, using a genetic algorithm [14–16]. This genetic algorithm, which is based on a standard GA, starts with a randomly generated population of solutions. The initial population is of fixed size P: T_1, T_2, …, T_P. Each solution is coded as a binary string T, such that $T = t_1$, t_2, …, t_{k-1}, where $t_i \neq t_j$ for $i \neq j$ and t_i is a $\log_2 L$-bit character, having a value within $[0, L-1]$, i.e. a_i indicates the value of the ith threshold. Then the current population evolves to the next population of the same size, using three genetic operations: selection, crossover and mutation. The evolution process is iterated until a near-optimal solution is obtained or a specified number of generations is reached.

The steps of the method are summarized in Algorithm 3.

Algorithm 3. *Steps of the GA*

1. Generate an initial population.
2. Store the best string T^* with the best fitness in a separate location.
3. Apply a learning strategy to improve the fitness value of T^* (see Section 2.5; process used only in [5]).
4. Generate the next population by performing selection, crossover and mutation operations.
5. Compare the best string T of the current population with T^*. If T has a better fitness value than T^*, then replace T^* with T.
6. Go to step 3 if the desired number of generations is not reached.

Table 1
Number of thresholds, threshold values and time computation achieved by the proposed method

Contents	Original space, $P = 300$			Level 1 space, $P = 150$			Level 2 space, $P = 75$			Level 3 space, $P = 38$		
	k^*	T^*	CPU time (ms)	k^*	T^*	CPU time (ms)	k^*	T^*	CPU time (ms)	k^*	T^*	CPU time (ms)
Artificial histogram	5	66	1100	5	66	300	5	66	80	5	65	32
		120			120			120			120	
		174			174			174			174	
		215			215			215			215	
Lena	5	47	1100	5	47	300	5	47	80	5	47	32
		84			83			83			83	
		119			119			119			119	
		164			164			164			164	
Blood	4	80	1100	4	76	300	4	76	80	4	76	32
		130			130			130			130	
		172			177			177			177	
Peppers	4	62	1100	4	61	300	4	60	80	4	61	32
		116			115			115			115	
		160			159			159			159	
House	3	96	1100	3	96	300	3	96	80	3	96	32
		155			155			155			155	
Airplane	3	117	1100	3	110	300	3	111	80	3	110	32
		175			170			171			170	
Lac	4	80	1100	4	79	300	4	79	80	4	79	32
		140			139			139			138	
		193			192			192			192	
Boats	4	59	1100	4	58	300	4	57	80	4	58	32
		111			109			107			109	
		151			150			149			150	
Bridge	8	49	1100	8	46	300	8	46	80	8	46	32
		75			71			71			71	
		98			94			94			94	
		122			118			118			118	
		148			144			144			144	
		176			173			173			173	
		210			208			208			208	

3.1.1.2. Iterative scheme for multilevel thresholding. Another fast multilevel thresholding method was proposed in [12,13], where the thresholds, optimizing an objective function $G(k)$, are searched by using an iterative scheme.

First, the technique starts with $(k-1)$ initial thresholds. Then, these thresholds are iteratively adjusted to improve the value of the objective function $G(k)$. This improvement process stops when the value of the objective function $G(k)$ is not increased between two consecutive iterations. The details of this iterative scheme are presented in Algorithm 4.

Algorithm 4. *Iterative scheme for multilevel thresholding*

1. Start with a set of $(k-1)$ thresholds $T = \{t_1, t_2, \ldots, t_{k-1}\}$ (for example, $t_i = i^*L/(k-1)$). For convenience, two other thresholds $t_0 = 0$ and $t_k = L-1$ are assumed.
2. Compute the value of the objective function $G(k)$ by using the current $T = \{t_0, t_2, \ldots, t_k\}$.
3. Set $i = 1$.
4. Consider the part of grey level histogram on the subrange $[t_i\ t_{i+1}]$; one can find the optimal threshold t^* between t_i and t_{i+1} by using the objective function $G(2)$ (*i.e. in the case of bi-level thresholding*). Replace t_i with t^*.
5. Set $i = i+1$. Go to step 4 until $i > k-1$.
6. Compute the value of the objective function $G(k)$ by using the current $T = \{t_0, t_2, \ldots, t_k\}$.
7. If $G(k)$ is increased, go to step 3; otherwise, stop.

3.1.1.3. Automatic determination of the threshold number. The above GA technique and iterative scheme still leave a problem, which is the determination of the number of thresholds for a given image. In order to compare the performances of two above techniques with the proposed method, the threshold number must be determined automatically. In practice, the multilevel thresholding methods are generally applied by varying the number of thresholds; then the optimal threshold number, which optimizes a cost function, is determined. Here, the algorithm described in Algorithm 5 is used.

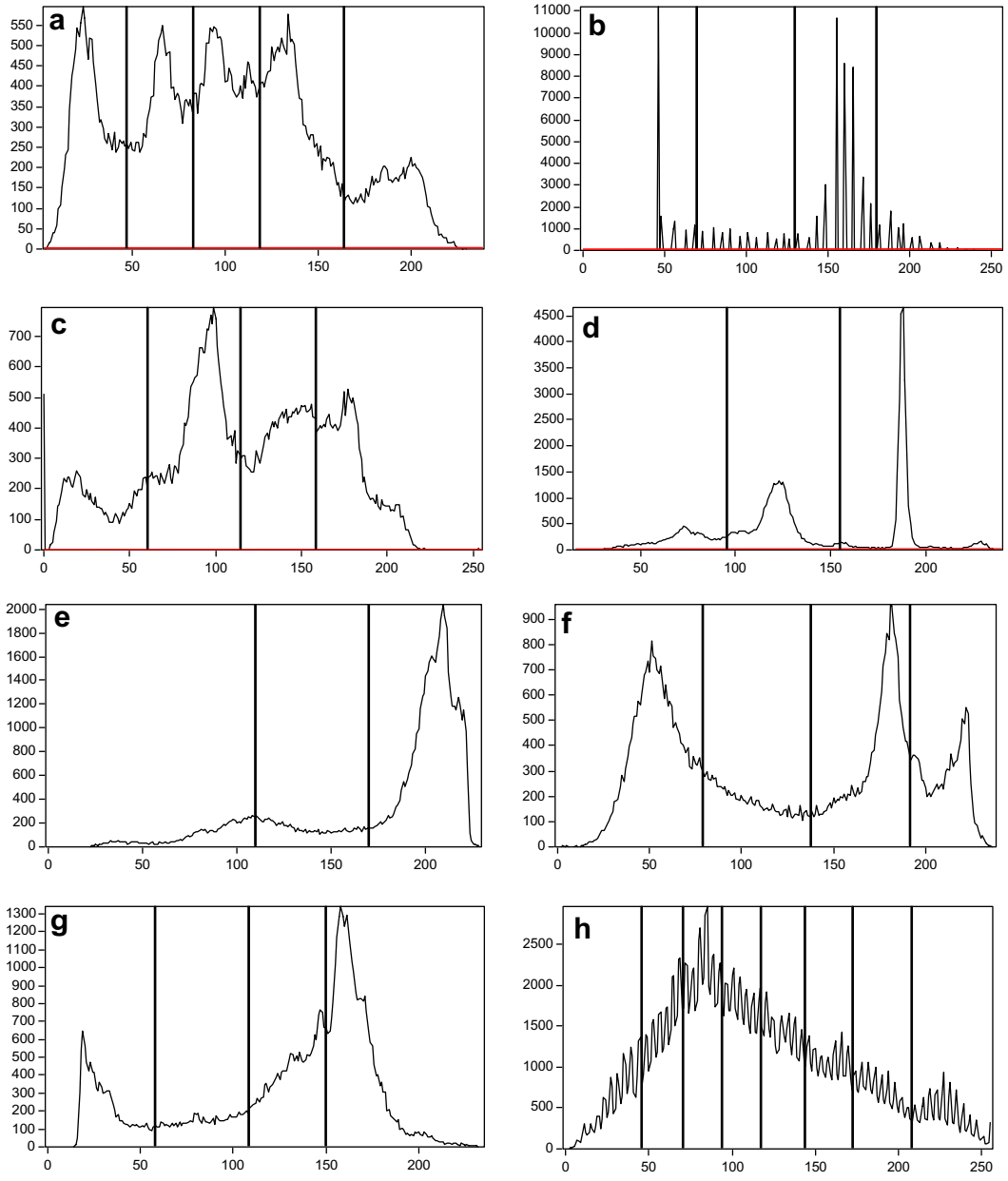

Fig. 4. Grey level histograms with threshold values. (a) Lena. (b) Blood. (c) Peppers. (d) House. (e) Airplane. (f) Lac. (g) Boats. (h) Bridge.

Algorithm 5. *Calculation of the threshold number and the threshold values*

1. $k = 2$.
2. Apply the search method or the GA or the iterative scheme.
3. Compute the value of the cost function $F(k)$ by using the output thresholds $\{t_1, t_2, \ldots, t_{k-1}\}$.
4. If $F(k)$ is decreased, set $k = k + 1$ and go to step 2; otherwise, stop.
5. Output the threshold number $(k^* - 1)$ and the threshold values $\{t^*_1, t^*_2, \ldots, t^*_{k-1}\}$.

The cost function $F(k)$ used in this algorithm is ATC.

3.2. Comparison of performances

We will evaluate the performances of the GA and iterative multilevel thresholding methods by using Otsu's and Kapur's objective functions G, defined below:

$$G_{\text{Otsu}}(k) = \sum_{i=1}^{k} P_i (m_i - m)^2,$$

$$G_{\text{Kapur}}(k) = -\sum_{i=1}^{k} \sum_{j=t_{i-1}}^{t_i - 1} \frac{p_j}{P_i} \operatorname{Log}\left(\frac{p_j}{P_i}\right).$$

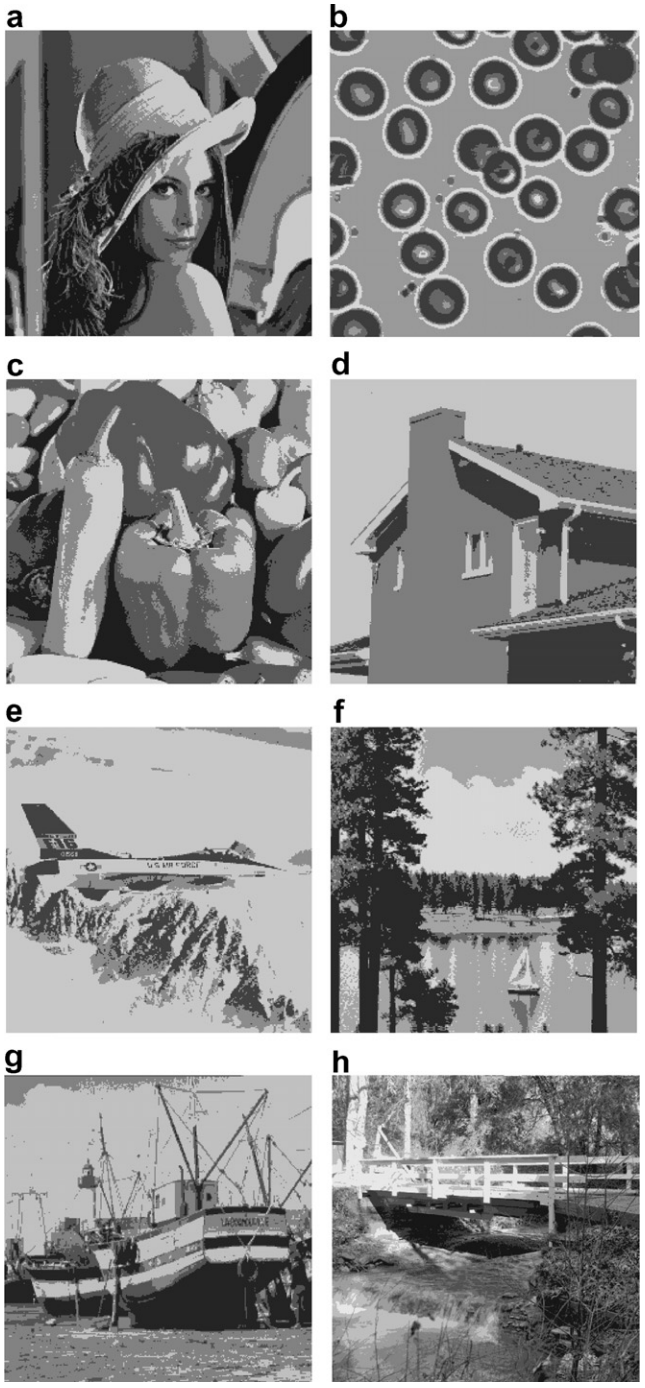

Fig. 5. Thresholded images. (a) Lena. (b) Blood. (c) Peppers. (d) House. (e) Airplane. (f) Lac. (g) Boats. (h) Bridge.

We shall refer to them as GA-Otsu, GA-Kapur, Iterative-Otsu and Iterative-Kapur. The exhaustive search method is also conducted for deriving the optimization solutions for comparison. The corresponding methods are named ES-Otsu and ES-Kapur, respectively. GA-Otsu and GA-Kapur are implemented with the same selection, crossover and mutation operators and with the same parameters of the proposed method: $P_c = 0.9$, $P_m = 0.001$, the size $P = 100$ of the population and a number of generations equal to $N_g = 100$, to ensure the convergence.

For the artificial histogram, Table 2 shows the number of thresholds, the optimal thresholds, the time computation, the value of the cost function F and the corresponding uniformity U value achieved by the exhaustive search methods, GA-Otsu, GA-Kapur, Iterative-Otsu, Iterative-Kapur and the proposed method. The results from the seven mentioned methods obtained over the test images are summarized in Tables 3–10. From these tables, we can make the following observations. The proposed method, GA-Otsu and Iterative-Otsu methods provide the same threshold number that ES-Otsu, for all test images, while GA-Kapur and Iterative-Kapur methods do not give always the same threshold number that ES-Kapur, as it is the case for *Airplane* and *Boats* images. In most cases, the performances of the proposed method, evaluated through the cost function F and the uniformity measure U, are close to those of ES-Otsu, GA-Otsu and Iterative-Otsu methods.

For all images, the performances of the proposed method and the methods based on the Otsu's function are better than those using the Kapur's function, since their cost functions F are smaller and their uniformity measures U are higher. This result occurs because the cost function F and the uniformity measure U are similar to the Otsu's function. It can be also pointed out that the quality function values provided by ES-Kapur can be worse than the values found with GA-Kapur and Iterative-Kapur (see on Tables 3–5, 8), because the maximization of the Kapur's function do not lead to the minimization of the cost function $F(k)$. Therefore, using the entropy (Kapur's function) with the cost function $F(k)$ is not appropriate.

In the view point of the computation time, the proposed method is faster than other methods, even though the wavelet transform and refinement procedures are added. It can be explained through time complexity analysis shown in Table 11. The computational time of the thres-

Table 2
Comparison of performances for the artificial histogram

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	5	66–120–174–215	10.4231	0.98442	900,000
ES-Kapur	4	59–127–187	11.4289	0.97284	70,578
GA-Otsu	5	66–120–174–215	10.4231	0.98442	609
GA-Kapur	4	59–131–187	11.4688	0.97254	1297
Iterative-Otsu	5	65–119–173–214	10.4316	0.98437	453
Iterative-Kapur	4	58–125–186	11.4603	0.97260	1860
Proposed method	5	65–120–174–215	10.4239	0.98442	32

Table 3
Comparison of performances for *Lena* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	5	47–84–119–164	10.8040	0.97877	601,766
ES-Kapur	4	60–109–160	11.0013	0.97159	50,281
GA-Otsu	5	46–80–113–162	10.8952	0.97805	578
GA-Kapur	4	59–107–159	10.9798	0.97177	1118
Iterative-Otsu	5	47–83–118–163	10.8090	0.97874	360
Iterative-Kapur	4	60–108–159	10.9952	0.97160	1516
Proposed method	5	46–83–119–164	10.8040	0.97875	32

Table 4
Comparison of performances for *Blood* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	4	74–132–177	9.0609	0.98123	6468
ES-Kapur	4	91–139–189	9.8446	0.97497	7939
GA-Otsu	4	75–138–181	9.0609	0.98123	406
GA-Kapur	4	94–145–198	10.4550	0.96947	453
Iterative-Otsu	4	73–126–171	9.2997	0.97942	203
Iterative-Kapur	4	90–138–188	9.2963	0.97944	344
Proposed method	4	76–130–175	9.0683	0.98118	32

Table 5
Comparison of performances for *Peppers* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	4	62–116–160	10.9679	0.97590	13,907
ES-Kapur	4	71–130–185	12.0782	0.96763	64,859
GA-Otsu	4	62–116–160	10.9679	0.97590	453
GA-Kapur	5	58–105–145–186	11.7396	0.97501	1938
Iterative-Otsu	4	61–115–159	10.9706	0.97589	343
Iterative-Kapur	5	57–104–144–185	11.7397	0.97501	2000
Proposed method	4	62–116–160	10.9679	0.97590	32

Table 6
Comparison of performances for *House* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	3	96–155	8.6977	0.98152	110
ES-Kapur	2	104	16.9239	0.91841	1
GA-Otsu	3	96–155	8.6977	0.98152	282
GA-Kapur	2	104	16.9239	0.91841	546
Iterative-Otsu	3	97–154	8.7022	0.98150	204
Iterative-Kapur	2	103	17.0098	0.91753	891
Proposed method	3	96–155	8.6977	0.98152	32

Table 7
Comparison of performances for *Airplane* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	3	117–175	9.5471	0.97227	94
ES-Kapur	4	72–128–183	9.8474	0.97440	31,047
GA-Otsu	3	117–175	9.5471	0.97227	281
GA-Kapur	2	162	10.3896	0.96707	516
Iterative-Otsu	3	114–172	9.5541	0.97222	203
Iterative-Kapur	2	161	10.3618	0.96727	812
Proposed method	3	110–170	9.5803	0.97201	32

Table 8
Comparison of performances for *Lac* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	4	80–140–193	10.7422	0.97343	10,031
ES-Kapur	5	72–112–155–194	10.7947	0.97867	2,941,000
GA-Otsu	4	80–140–193	10.7422	0.97343	421
GA-Kapur	5	72–113–158–194	10.7856	0.97874	1594
Iterative-Otsu	4	79–138–191	10.7506	0.97336	312
Iterative-Kapur	5	69–109–152–191	10.8289	0.97840	1813
Proposed method	4	79–138–192	10.7466	0.97339	32

Table 9
Comparison of performances for *Boats* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	4	59–111–151	10.2018	0.9747	8031
ES-Kapur	5	56–103–144–190	10.8262	0.97580	2,214,016
GA-Otsu	4	59–111–151	10.2018	0.97479	406
GA-Kapur	2	116	13.1157	0.95191	563
Iterative-Otsu	4	59–110–150	10.2041	0.97478	265
Iterative-Kapur	2	115	13.0477	0.95245	828
Proposed method	4	58–109–150	10.2054	0.97476	32

Table 10
Comparison of performances for *Bridge* image

Method	Class number K^*	Threshold values T^*	Objective function F	Uniformity U	CPU time (ms)
ES-Otsu	—	—	—	—	—
ES-Kapur	—	—	—	—	—
GA-Otsu	8	47–73–96–119–143–171–207	17.1226	0.98376	1141
GA-Kapur	8	25–60–91–122–150–183–214	17.6771	0.98147	3000
Iterative-Otsu	8	47–73–97–121–147–175–208	17.0829	0.98392	563
Iterative-Kapur	8	34–63–93–123–154–186–219	17.6919	0.98141	3000
Proposed method	8	46–71–94–118–144–173–208	17.1005	0.98385	32

Table 11
Computational complexity

	Complexity	Parameter values
Exhaustive search	$O(L^{(k-1)})$	
GA technique	$O((k-1)*N_g*P*[8*(k-1)+L])$	$N_g = P = 100$
Iterative scheme	$O((k-1)N_i 3L)$	$N_i = 100$
Proposed method	$O\left(N_g * P * \frac{L}{2^{r-1}}\right)$	$N_g = 100, P = \frac{300}{2^r}$

holding methods based on Otsu's and Kapur's functions is very expensive when the exhaustive search is applied. For $(k-1)$ thresholds, the complexity is $O(L^{(k-1)})$, which grows exponentially with the number of thresholds. However, in practice, it is proved that the computation time of Otsu's function is faster than that of Kapur's function.

In each iteration of the iterative scheme, the histogram is looped three times. The computation complexity of the global iterative scheme then becomes $(k-1)N_i 3L$, where N_i is the number of iterations, which is generally less than 100.

We generally agree that the computational time of GAs is reduced only to the computational time of the fitness, because the computation of the selection, crossover and mutation operations is much faster than that of fitness. For the same reason, the computation of the learning strategy can be ignored.

In GA-Otsu and GA-Kapur, each chromosome of length $8(k-1)$ is scanned once to determine the threshold values. The histogram with length L is the looped one in order to compute their fitness. So the complexity of GA-Otsu and GA-Kapur is $O((k-1)*N_g*P\ [8*(k-1)+L])$, where $N_g = P = 100$.

For the proposed method, the wavelet and the refinement procedures are of lower complexity than the GA. Hence only the complexity of the GA is taken into account. As each chromosome of length $\frac{L}{2^r}$ is scanned once to extract the threshold values and as the reduced histogram with length $\frac{L}{2^r}$ is the looped one in order to compute their fitness, the proposed method complexity becomes equal to $O\left(N_g * P * \frac{L}{2^{r-1}}\right)$, where $N_g = 100$ and $P = 300/2^r$.

A simple calculation shows that the proposed method has a complexity lower than those of other methods. Higher speed is due to the reduction of the histogram. On the other hand, the iterative GA-Otsu and GA-Kapur methods can be applied several times to determine the number of thresholds. Furthermore, the iterative methods need some iterations to converge, while for the GA-Otsu

and GA-Kapur methods, the number of generations is higher than that of the proposed method. Let us note also that the computation time of the proposed method is independent from the number of classes, while other methods are linked to the number of classes: indeed, in the GA-Otsu and GA-Kapur methods, the size of the chromosome increases with the threshold number and, in the iterative methods, at each iteration, the number of the updates of the thresholds increases with the number of thresholds. The speed of the proposed method is clearly shown in Table 10, where the *Bridge* image is thresholded by a great number of thresholds. Hence, the proposed method is useful to find multiple thresholds for complex image analysis.

4. Conclusion

In this paper, we have proposed a new multilevel thresholding method based on a genetic algorithm, which enables determining the appropriate number of thresholds, as well as the adequate threshold values. The length of the original histogram is reduced by using a wavelet transform. Based on this lower resolution version of the histogram, the optimal threshold values are determined by using a standard GA. In this GA is used a new string representation of the chromosome, which is different from current representations. A binary encoding is used and the threshold values are directly determined without requiring the use of encoding and decoding operations. In order to improve the performance of the GA, a learning strategy was used and a new mutation operator, that handles problem dependent characteristics, was also proposed. Experiments with a synthetic histogram and real images have proved the robustness of the proposed method, in view of the accuracy of image segmentation, evaluated through the uniformity measure and the cost function. Comparisons with other thresholding methods showed that the proposed method runs faster when the determination of the optimal number of thresholds is required.

References

[1] N. Otsu, A threshold selection method for grey level histograms, IEEE Trans. Syst. Man Cybern. SMC-9 (1) (1979) 62–66.
[2] E.J.N. Kapur, P.K. Sahoo, A.K.C. Wong, A new method for grey-level picture thresholding using the entropy of the histogram, Comput. Vis. Graph. Image Process. 29 (1985) 273–285.
[3] P.K. Sahoo, S. Soltani, A.K.C. Wong, A survey of thresholding techniques, Comput. Vis. Graph. Image Process. 41 (1988) 233–260.
[4] W.B. Lievers, A.K. Pilkey, An evaluation of global thresholding techniques for automatic image segmentation of automotive aluminum sheet alloys, Mater. Sci. Eng. A381 (2004) 134–142.
[5] M. Sezgin, B. Sankur, Survey over image thresholding techniques and quantitative performance evaluation, J. Electron. Imaging 13 (1) (2004) 146–156.
[6] U. Gonzales-Baron, F. Butler, A comparison of seven thresholding techniques with the *k*-means clustering algorithm for measurement of bread-crumb features by digital image analysis, J. Food Eng. 74 (2006) 268–278.
[7] P.-S. Liao, T.-S. Chen, P.-C. Chung, A fast algorithm for multilevel thresholding, J. Inf. Sci. Eng. 17 (2001) 713–727.
[8] K.C. Lin, Fast image thresholding by finding the zero(s) of the first derivative of between-class variance, Mach. Vis. Appl. 13 (2003) 254–262.
[9] B.-G. Kim, J.-I. Shim, D.-J. Park, Fast image segmentation based on multi-resolution analysis and wavelets, Pattern Recognit. Lett. 24 (2003) 2995–3006.
[10] O. Virmajoki, P. Fränti, Fast pairwise nearest neighbour based algorithm for multilevel thresholding, J. Electron. Imaging 12 (4) (2003) 648–659.
[11] A.Z. Arifin, A. Asano, Image segmentation by histogram thresholding using hierarchical cluster analysis, Pattern Recognit. Lett. 27 (2006) 1515–1521.
[12] P.-Y. Yin, L.-H. Chen, A fast iterative scheme for multilevel thresholding methods, Signal Process. 60 (1997) 305–313.
[13] X. Luo, J. Tian, Multi-level thresholding: maximum entropy approach using ICM, in: Proceedings of the 15th International Conference on Pattern Recognition, vol. 3, 2000, pp. 778–781.
[14] P.Y. Yin, A fast scheme for optimal thresholding using genetic algorithms, Signal Process. 72 (1999) 85–95.
[15] C. Jinsong, W. Hongqi, Z. Xiaokuan, Entropic thresholding method using genetic algorithm, in: Proceedings of Geoscience and Remote Sensing Symposium, Hamburg, Germany, vol. 2, 1999, pp. 1247–1249.
[16] Z.-H. Yang, Z.-B. Pu, Z.-Q. Qi, Relative entropy multilevel thresholding method based on genetic algorithm, in: IEEE International Conference on Neural Networks and Signal Processing, Nanjing, China, 2003, pp. 583–586.
[17] Y. Chang, H. Yan, An effective multilevel thresholding approach using conditional probability entropy and genetic algorithm, in: J.S. Jin, P. Eaqdes, D.D. Feng, H. Yan (Eds.), Conferences in Research and Practice in Information Technology, vol. 22, ACS, 2003 (21).
[18] C.-C. Lai, D.C. Tseng, A hybrid approach using Gaussian smoothing and genetic algorithm for multilevel thresholding, Int. J. Hybrid Intell. Syst. 1 (3) (2004) 143–152.
[19] W.-B. Tao, J.-W. Tian, J. Liu, Image segmentation by three-level thresholding based on maximum fuzzy entropy and genetic algorithm, Pattern Recognit. Lett. 24 (2003) 3069–3078.
[20] Y. Bazi, L. Bruzzone, F. Melgani, Image thresholding based on the EM algorithm and the generalized Gaussian distribution, Pattern Recognit. 40 (2007) 619–634.
[21] L.-C. Ramac, P.K. Varsheney, Image thresholding based on Ali-Silvey distance measures, Pattern Recognit. 30 (1997) 1162–1174.
[22] E. Zahara, S.-K.S. Fan, D.M. Tsai, Optimal multi-thresholding using a hybrid optimization approach, Pattern Recognit. Lett. 26 (2005) 1085–1095.
[23] P.-Y. Yin, Multilevel minimum cross entropy threshold selection based on particle swarm optimization, Appl. Math. Comput. (2006), doi:10.1016/j.amc.2006.06.057.
[24] S.-K.S. Fan, Y. Lin, A multi-level thresholding approach using a hybrid optimal estimation algorithm, Pattern Recognit. Lett. 28 (2007) 662–669.
[25] C.H. Li, C.K. Lee, Minimum cross entropy thresholding, Pattern Recognit. 26 (4) (1993) 617–625.
[26] Z. Hou, Q. Hu, W.L. Nowinski, On minimum variance thresholding, Pattern Recognit. Lett. 27 (2006) 1732–1743.
[27] J.C. Yen, F.J. Chang, S. Chang, A new criterion for automatic multilevel thresholding, IEEE Trans. Image Process. IP-4 (1995) 370–378.
[28] M. Sezgin, R. Tasaltin, A new dichotomization technique to multilevel thresholding devoted to inspection applications, Pattern Recognit. Lett. 21 (2000) 151–161.
[29] B.-F. Wu, Y.-L. Chen, C.-C. Chiu, Recursive algorithms for image segmentation based on a discriminant criterion, Int. J. Signal Process. 1 (2004) 55–60.
[30] D.E. Goldberg, Genetic Algorithms: Search, Optimization and Machine Learning, Addison-Wesley, 1989.