4

Segmentation with Learning Automata

Erik Cuevas, Daniel Zaldivar and Marco Pérez-Cisneros
Departamento de Electrónica, Universidad de Guadalajara
México

1. Introduction

Several image processing applications aim to detect and mark remarkable features which in turn might be used to perform high-level tasks. In particular, image segmentation seeks to group pixels within meaningful regions. Commonly, gray levels belonging to the object are substantially different from the gray levels featuring the background. Thresholding is thus a simple but effective tool to isolate objects of interest from the background. Its applications include several classics such as document image analysis, whose goal is to extract printed characters (Abak et al., 1997; Kamel & Zhao, 1993) logos, graphical content, or musical scores; also it is used for map processing which aims to locate lines, legends, and characters (Trier & Jain, 1995). It is also used for scene processing, aiming for object detection and marking (Bhanu, 1986); Similarly, it has been employed to quality inspection for materials (Sezgin & Sankur, 2001; Sezgin & Tasaltin, 2000), discarding defective parts.

Thresholding selection techniques can be classified into two categories: bi-level and multi-level. In bi-level thresholding, one limit value is chosen to segment an image into two classes: one represents the object and the other represents the background. When an image is composed of several distinct objects, multiple threshold values have to be selected for proper segmentation. This is called multilevel thresholding.

A variety of thresholding approaches have been proposed for image segmentation, including conventional methods (Guo & Pandit, 1998; Pal & Pal, 1993; Shaoo et al., 1988; Snyder et al., 1990) and intelligent techniques such as in (Chen & Wang, 2005; Chih-Chih, 2006). Extending the algorithm to a multilevel approach may arise some inconveniences: (i) they may have no systematic and analytic solution when the number of classes to be detected increases and (ii) the number of classes is either difficult to be predicted or must be pre-defined. However, this parameter is unknown for many real applications.

In order to solve these problems, an alternative approach using an optimization algorithm based on learning automata for multilevel thresholding is proposed in this paper. In the traditional multilevel optimal thresholding, the intensity distributions belonging to the object or to the background pixels are assumed to follow some Gaussian probability function; therefore a combination of probability density functions is usually adopted to model these functions. The parameters in the combination function are unknown and the parameter estimation is typically assumed to be a nonlinear optimization problem (Gonzalez & Woods, 1990). The unknown parameters that give the best fit to the processed histogram are determined by using a LA algorithm (Thathachar & Sastry, 2002).

The main motivation behind the use of LA as an optimization algorithm for parameter adaptation is to use its capabilities of global optimization when dealing to multimodal

surfaces. Using LA, the search for the optimum is done within a probability space rather than seeking within a parameter space as done by other optimization algorithms (Najim &Poznyak, 1994). Learning automata is referred to as an automaton, acting embedded into an unknown random environment. Such automaton improves its performance to obtain an optimal action. On the other hand, an action is applied to a random environment and gives a fitness value to the selected action of the automata. The response of the environment is used by automata to select its next action. This procedure is continued to reach the optimal action.

LA has been used for solve different sorts of engineering problems. For instance, pattern recognition (Seyed-Hamid, 2008), adaptive control (Zeng et al., 2000) signal processing (Howell & Gordon, 2000) and power systems (Wu, 1995). Recently, some effective algorithms have been proposed for multimodal complex function optimization based on the LA (see (Howell & Gordon, 2000; Thathachar & Sastry, 2002; Zeng & Liu, 2005; Beygi & Meybodi, 2006)). Furthermore, it was shown experimentally that the performance of these optimization algorithms is comparable to or better than the genetic algorithm (GA) in [22]. This work employs the algorithm proposed in (Zeng & Liu, 2005), which is called continuous action reinforcement learning automata (CARLA).

In this chapter, an automatic image multi-threshold approach based on Learning Automata is presented. Hereby the segmentation process is considered to be similar to an optimization problem. First, the algorithm approximates the 1-D histogram of the image using a mix of Gaussian functions whose parameters are calculated using the Learning automata method. Each Gaussian function approximating the histogram represents a pixel class and therefore the threshold points.

This chapter is organized as follows. Section 2 presents the Gaussian approximation to the histogram. Section 3 presents the LA algorithm, while Section 4 shows the determination of the threshold points. In section 5 the implementation details are shown. Experimental results for the proposed approach are presented in Section 6, finally the conclusion are presented in Section 7.normal)

2. Gaussian approximation

Assuming an image has L gray levels $[0,\ldots,L-1]$, following a gray level distribution which can be displayed in the form of the histogram $h(g)$. In order to simplify the description, the histogram is normalized and is considered as a probability distribution function:

$$h(g)=\frac{n_g}{N},\ h(g)\geq 0,$$
$$N=\sum_{g=0}^{L-1} n_g,\ \text{and}\ \sum_{g=0}^{L-1} h(g)=1, \tag{1}$$

Assuming that n_g denotes the number of pixels with gray level g while N is the total number of pixels in the image. The histogram function can be contained into a mix of Gaussian probability functions, yielding:

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

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

The typical mean square error consideration is used to estimate the 3*K* parameters P_i, μ_i and σ_i, *i* = 1, . . ,*K*. For example, the mean square error between the composite Gaussian function $p(x_i)$ and the experimental histogram function $h(x_i)$ is defined as follows:

$$J = \frac{1}{n}\sum_{j=1}^{n}\left[p(x_j) - h(x_j)\right]^2 + \omega \cdot \left|\left(\sum_{i=1}^{K} P_i\right) - 1\right| \tag{3}$$

Assuming an *n*-point histogram as in [13] and ω being the penalty associated with the constrain $\sum_{i=1}^{K} P_i = 1$. In general, the determination of parameters that minimize the square error is not a simple problem. A straightforward method to decrease the partial derivatives of the error function to zero considers a set of simultaneous transcendental equations (Gonzalez & Woods, 1992). An analytical solution is not available due to the non-linear nature of the equations. The algorithm therefore makes use of a numerical procedure following an iterative approach based on the gradient information. However, considering that the gradient descent method may easily get stuck within local minima. In the standard gradient method, the new operation point lies within a neighbourhood distance of the previous point. This is not the case for adaptation algorithm based on stochastic principles such as LA, as the new operating point is determined by probability function and is therefore not considered to be near the previous operating point. This gives the algorithm a higher ability to locate the global minima.

Some previous experiences have shown that the intelligent approaches actually provide a satisfactory performance in case of image processing problems (Chen & Wang, 2005; Chih-Chih, 2006; Baştürk & Günay, 2009; Lai & Tseng, 2001; Tseng & Lai, 1999). The LA algorithm is therefore adopted in order to find the parameters and their corresponding threshold values.

3. Learning automata

LA operates by selecting actions via a stochastic process. Such actions operate within an environment while being assessed according to a measure of the system performance. Figure 1a shows the typical learning system architecture. The automaton selects an action (**X**) probabilistically. Such actions are applied to the environment, and the performance evaluation function provides a reinforcement signal β. This is used to update the automaton's internal probability distribution whereby actions that achieve desirable performance are reinforced via an increased probability, while those not-performing actions are penalised or left unchanged depending on the particular learning rule which has been employed. Over time, the average performance of the system will improve while a given limit is reached. In terms of optimization problems, the action with the highest probability would correspond to the global minimum as demonstrated by rigorous proofs of convergence available in (Narendra & Thathachar, 1989) and (Najim & Poznyak, 1994).

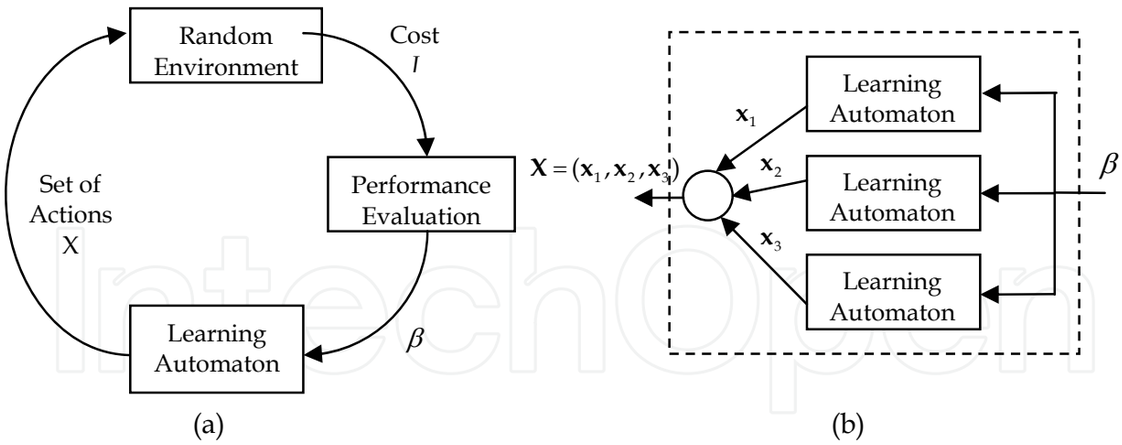


Fig. 1. (a) Reinforcement learning system and (b) Interconnected automata.

A wide variety of learning rules have been reported in the literature. One of the most widely used algorithms is the linear reward/inaction (L_{RI}) scheme, which has been shown to guaranteed convergence properties (see [1008]). In response to action $\mathbf{x}_i$, being selected at time step *k*, the probabilities are updated as follows:

$$\begin{aligned} p_i(n+1) &= p_i(n) + \theta \cdot \beta(n) \cdot (1 - p_i(n)) \\ p_j(n+1) &= p_j(n) - \theta \cdot \beta(n) \cdot p_j(n) \text{ if } i \neq j \end{aligned} \tag{4}$$

being θ a learning rate parameter and $0 < \theta < 1$ and $\beta \in [0,1]$ the reinforcement signal; $\beta = 1$ indicates the maximum reward and $\beta = 0$ is a null reward. Eventually, the probability of successful actions will increase to become close to unity. In case that a single and foremost successful action prevails, the automaton is deemed to have converged.

With a large number of discrete actions, the probability of selecting any particular action becomes low and the convergence time can become excessive. To avoid this, learning automata can be connected in a parallel setup as shown by Figure 1b. Each automaton operates a smaller number of actions and the 'team' works together in a co-operative manner. This scheme can also be used where multiple actions are required.

Discrete stochastic learning automata can be used to determine global optimal states for control applications with multi-modal mean square error surfaces. However, the discrete nature of the automata requires the discretization of a continuous parameter space, and the level of quantization tends to reduce the convergence rate. A sequential approach may be adopted (Howell & Gordon, 2000) to overcome such problem by means of an initial coarse quantization. It may be later refined using a re-quantization around the most successful action. In this paper, an inherently continuous form of the learning automaton is used to speed the learning process and to avoid this additional complexity.

3.1 CARLA algorithm

The continuous action reinforcement learning automata (CARLA) was developed as an extension of the discrete stochastic learning automata for applications involving searching of continuous action space in a random environment (Howell & Gordon, 2000). Several CARLA can be connected in parallel, in a similar manner to discrete automata (Figure 1b), to search multidimensional action spaces. The interconnection of the automata is through the

environment however, no direct inter-automata communication exist. The automaton's discrete probability distribution is replaced by a continuous probability density function which is used as the basis for action selection. It operates a reward/inaction learning rule similar to the discrete learning automata. Successful actions receive and increase on the probability of future selection via a Gaussian neighborhood function which increases the probability density in the vicinity of such successful action. Table 1 shows the generic pseudo-code for the CARLA algorithm. The initial probability distribution can be chosen as being uniform over a desired range. After a considerable number of iterations, it converges to a probability distribution with a global maximum around the best action value.
If action x is defined over the range $(x_{\min}, x_{\max})$, the probability density function $f(x,n)$ at iteration n is updated according to the following rule:

$$f(x,n+1) = \begin{cases} \alpha \cdot [f(x,n) + \beta(n) \cdot H(x,r)] & \text{if } x \in (x_{\min}, x_{\min}) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

With α being chosen to re-normalize the distribution according to the following condition

$$\int_{x_{\min}}^{x_{\max}} f(x,n+1)dx = 1 \tag{6}$$

with $\beta(n)$ being again the reinforcement signal from the performance evaluation and $H(x,r)$ a symmetric Gaussian neighbourhood function centered on $r = x(n)$. It yields

$$H(x,r) = \lambda \cdot \exp\left(-\frac{(x-r)^2}{2\sigma^2}\right) \tag{7}$$

CARLA Algorithm
Initialize the probability density function to a uniform distribution
Repeat
Select an action using its probability density function
Execute action on the environment
Receive cost/reward for previous action
Update performance evaluation function β
Update probability density function
Until stopping condition

Table 1. Generic pseudo-code for the CARLA algorithm

with λ and σ being parameters that determine the height and width of the neighborhood function. They are defined in terms of the range of actions as follows:

$$\sigma = g_w \cdot (x_{\max} - x_{\min}) \tag{8}$$

$$\lambda = \frac{g_h}{(x_{\max} - x_{\min})} \tag{9}$$

The speed and resolution of learning are thus controlled by free parameters g_w and g_h . Let action $x(n)$ be applied to the environment at iteration n, returning a cost or performance index $J(n)$. Current and previous costs are stored as a reference set $R(n)$. The median and minimum values J_{med} and $J_{\min}$ may thus be calculated, by means of $\beta(n)$ being defined as:

$$\beta(n) = \max\left\{0, \frac{J_{\text{med}} - J(n)}{J_{\text{med}} - J_{\min}}\right\} \tag{10}$$

To avoid problems with infinite storage, and to allow the system to adapt to changing environments, only the last m values of the cost functions are stored in $R(n)$. Equation (10) limits $\beta(n)$ to values between 0 and 1 and only returns nonzero values for costs that are below the median value. It is easy to understand how $\beta(n)$ affects the learning process informally as follows: during the learning, the performance and the number of selecting actions can be wildly variable, generating extremely high computing costs. However, $\beta(n)$ is insensitive to these extremes and to the very high values of $J(n)$ resulting from a poor choice of actions. As learning continues, the automaton converges towards more worthy regions of the parameter space and these actions within such regions are chosen for evaluation increasingly often. While more of such responses are being received, J_{med} gets reduced. Decreasing J_{med} in the performance index effectively enables the automaton to refine its reference around the better responses previously received, and hence resulting in a better discrimination between the competing selected actions.
To define an action value $x(n)$ which has been associated to this probability density function, an uniformly distributed pseudo-random number $z(n)$ is generated within the range of [0,1]. Simple interpolation is then employed to equate this value to the cumulative distribution function:

$$\int_{x_{\min}}^{x(n)} f(x,n)dx = z(n) \tag{11}$$

For implementation purposes, the distribution is stored at discrete points with an equal inter-sample probability. Linear interpolation is used to determine values at intermediate positions (see full details in [19]).

4. Determination of threshold values

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

$$E(T_i) = P_{i+1} \cdot E_1(T_i) + P_i \cdot E_2(T_i), \quad i = 1,2,\ldots,K-1 \tag{12}$$

considering

$$E_1(T_i) = \int_{-\infty}^{T_i} p_{i+1}(x)dx, \tag{13}$$

and

$$E_2(T_i) = \int_{T_i}^{\infty} p_i(x)dx, \tag{14}$$

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

$$AT_i^2 + BT_i + C = 0 \tag{15}$$

considering

$$A = \sigma_i^2 - \sigma_{i+1}^2$$

$$B = 2 \cdot (\mu_i \sigma_{i+1}^2 - \mu_{i+1} \sigma_i^2) \tag{16}$$

$$C = (\sigma_i \mu_{i+1})^2 - (\sigma_{i+1} \mu_i)^2 + 2 \cdot (\sigma_i \sigma_{i+1})^2 \cdot \ln\left(\frac{\sigma_{i+1} P_i}{\sigma_i P_{i+1}}\right)$$

Although the above quadratic equation has two possible solutions, only one of them is feasible (positive and inside the interval). The figure 2 shows the determination process of the threshold points.

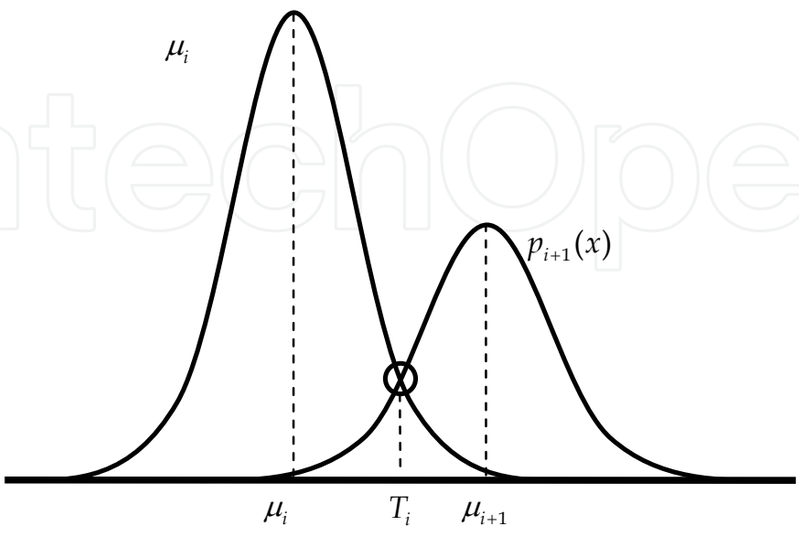


Fig. 2. Determination process of the threshold points

5. Implementation

In the implementation four different pixel classes are used to segment the images. The idea is to show the effectiveness of the algorithm and the little information required for the localization of the threshold points, the implementation could be effortlessly carried out for a bigger number of pixel classes.

To approach the histogram of an image by 4 Gaussian functions (one for each pixel class), it is necessary first, calculate the optimum values of the 3 parameters (P_i, μ_i and σ_i) for each Gaussian function (in this case, 12 values according to equation 2). This problem can be resolved optimizing equation 3, considering that function $p(x)$ is formed by 4 Gaussian functions.

The parameters to be optimized are summarized in table 2. Where k_p^i is the parameter that represents the a priori probability (P), k_σ^i represents the variance (σ) and k_μ^i represents the expected value (μ) of the Gaussian function i.

Parameters			Gaussian
k_P^1	k_σ^1	k_μ^1	1
k_P^2	k_σ^2	k_μ^2	2
k_P^3	k_σ^3	k_μ^3	3
k_P^4	k_σ^4	k_μ^4	4

Table 2. Parameters to be optimized

In LA optimization, each parameter is considered like an Automaton which chooses actions. The actions correspond to values assigned to the parameter, by a probability distribution inside of an interval. The intervals used in this work for the parameters are defined as $k_P^i \in [0,0.5]$, $k_\sigma^i \in [0,60]$, and $k_\mu^i \in [0,255]$.

For this 12-dimensional problem will be 12 different Automatons which represent the parameters to approach the corresponding histogram. One of the main advantages of the LA is that in a multi-dimensional problem the Automatons are coupled only through the enviorement, thus each Automaton operated independently, during the optimization.

Thus, in each instant n each Automaton chooses an action according to their probability distribution, which can be represented in a vector $A(n)$={ k_P^1, k_σ^1, k_μ^1 …, k_P^4, k_σ^4, k_μ^4 }. This vector represents a certain approach of the histogram. Then, the quality of the approach is evaluated (according to equation 3) and converted to a reinforcement signal $\beta(n)$ (through equation 10). Having obtained the reinforcement value $\beta(n)$ as product of the elected approach $A(n)$, the distribution of probability is update for n+1 of each Automaton (according to the equation 5). To simplify parameters of equation 8 and 9 are the same for the 12 Automatons, such that $g_w = 0.02$ and g_h =0.3. In this work is considers to limit to 2000 the iterations on the optimization process.

Next, the optimization algorithm is described:

i	Set iteration n=0.	
ii	Define the action set $A(n)$={ k_P^1, k_σ^1, k_μ^1 …, k_P^4, k_σ^4, k_μ^4 } such that $k_P^i \in$ [0,0.5], $k_\sigma^i \in$ [0,60] and $k_\mu^i \in$ [0,255].	
iii	Define probability density functions at iteration n: $f(k_P^i,n)$, $f(k_\sigma^i,n)$ and $f(k_\mu^i,n)$	
iv	Initialize $f(k_P^i,n)$, $f(k_\sigma^i,n)$ and $f(k_\mu^i,n)$ as an uniform distribution between the defined limits.	
v	Repeat while $n \leq 2000$	
	(a)	Using a pseudo-random number generator for each Automaton, select $z_P^i(n)$, $z_\sigma^i(n)$ and $z_\mu^i(n)$ uniformly between 0 and 1.
	(b)	Select $k_P^i \in$ [0,0.5], $k_\sigma^i \in$ [0,60] and $k_\mu^i \in$ [0,255] where the area under the probability density function is $\int_0^{k_P^i(n)} f(k_P^i,n) = z_P^i(n)$, $\int_0^{k_\sigma^i(n)} f(k_\sigma^i,n) = z_\sigma^i(n)$ and $\int_0^{k_\mu^i(n)} f(k_\mu^i,n) = z_\mu^i(n)$.
	(c)	Evaluate the performance using Eq. (3).
	(d)	Obtain the minimum, $J_{\min}$, and median, J_{med} of $J(n)$.
	(e)	Evaluate $\beta(n)$ via Eq. (10).
	(f)	Update the probability density functions $f(k_P^i,n)$, $f(k_\sigma^i,n)$ and $f(k_\mu^i,n)$ using Eq. (5).
	(g)	Increment iteration number n.

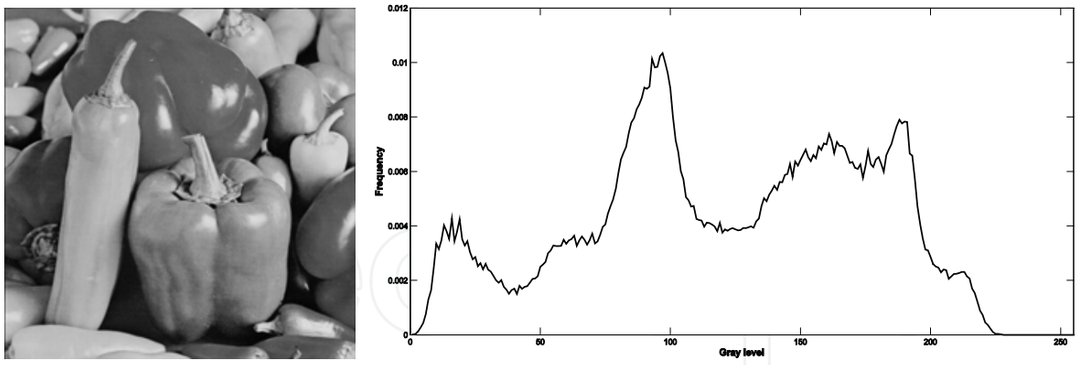
The learning system search in the 12-dimensional parameter space with the aim of reducing the values for J in Eq. (3).

6. Experimental results

In this section the performance of the algorithm is tested by two experiments. In both experiments a 4 pixel class segmentation is consider and an approaching of the original histogram of the image by LA. To test the consistency of the algorithm, 10 independent repetitions were made for each experiment.

In the first experiment the image represented in figure 3a was used, whose original histogram is shown in figure 3b. Considering the proposed LA algorithm (detailed in the previous section) a global minimum was obtained (equation 3), the point defined as k_P^1 =0.0210, k_σ^1 =6, k_μ^1 =15, k_P^2 =0.0404, k_σ^2 =29, k_μ^2 =63, k_P^3 =0.0608, k_σ^3 =10, k_μ^3 =93, k_P^4 =0.1002, k_σ^4 =30, and k_μ^4 =163. The values of these parameters define 4 different Gaussian functions, which are represented in figure 4. From the mix of these 4 Gaussian functions, an approach to the original histogram is obtained as shown in figure 5.

The evolution of the probability densities parameters, whose represent the expected values $f(k_\mu^1,n)$, $f(k_\mu^2,n)$, $f(k_\mu^3,n)$ and $f(k_\mu^4,n)$ of the Gaussian functions are shown in figure 6. It can be seen that most of the convergence is achieved at the first 1500 iterations, after that a gradual sharpening of the distribution occurs. The final probability densities (n=2000) can be taken as the final parameter value.



(a) (b)

Fig. 3. (a) Original image used on the first experiment, (b) and its histogram

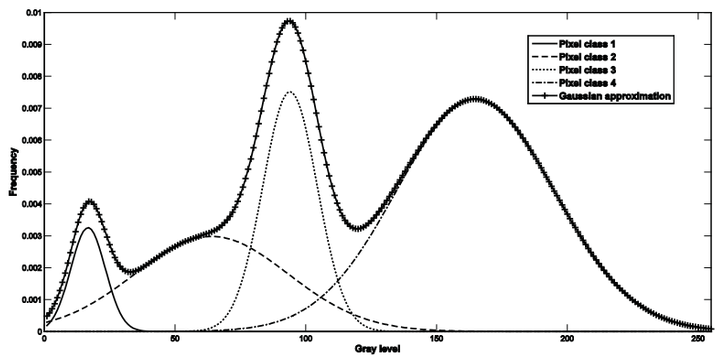


Fig. 4. Gaussian functions obtained by LA

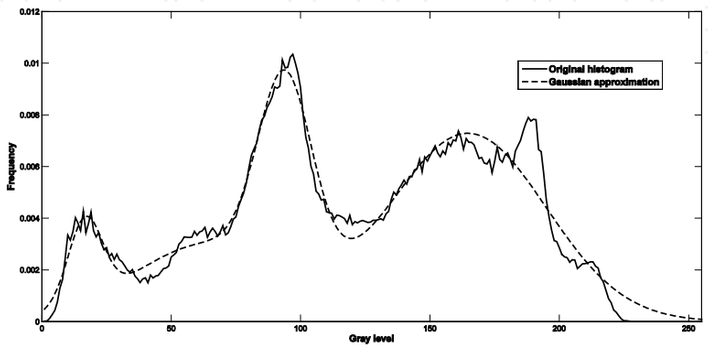
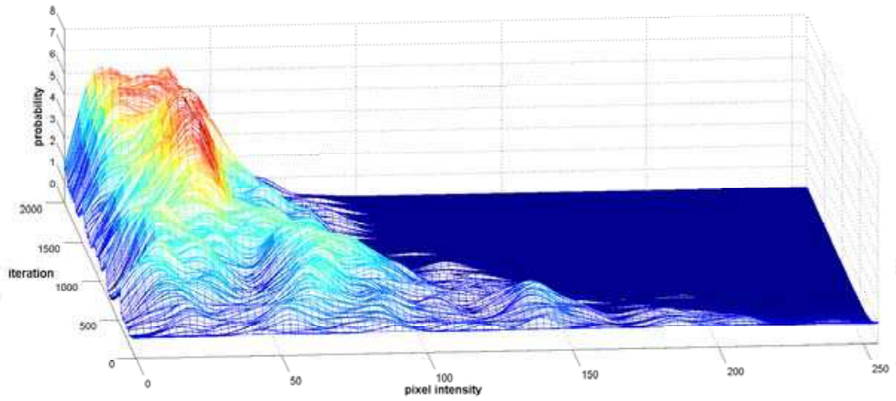
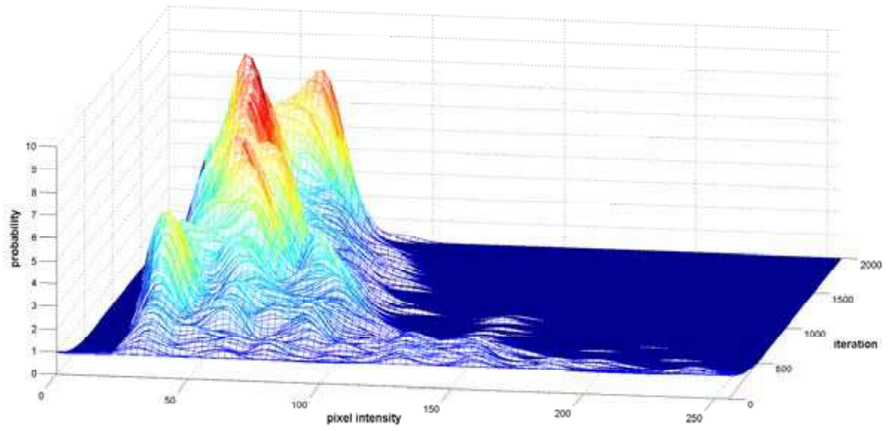


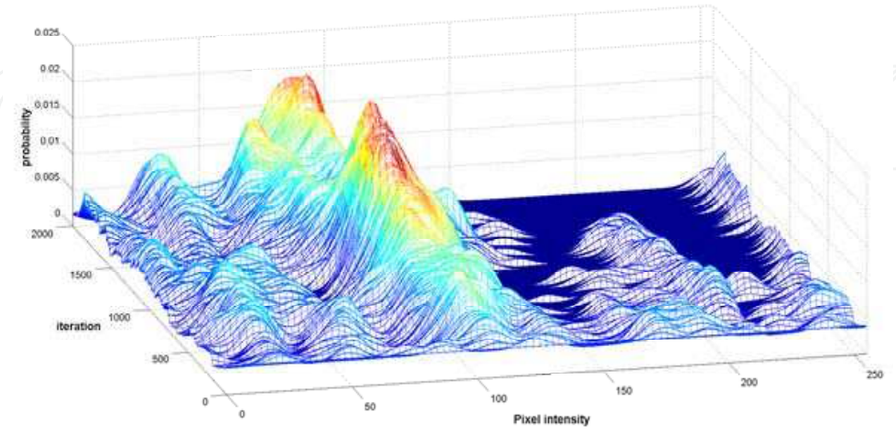
Fig. 5. Comparison between the original histogram and its approach


probability
iteration
pixel intensity

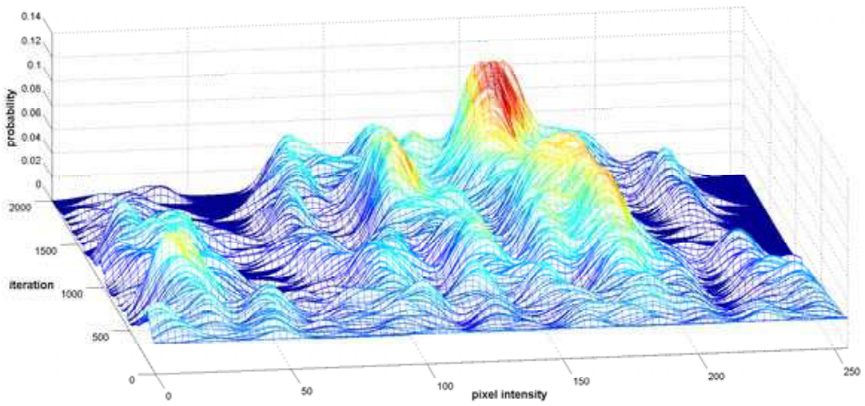
(a)


probability
pixel intensity
iteration

(b)


probability
iteration
Pixel intensity

(c)



(d)

Fig. 6. Evolution of the probability densities parameters, whose represent the expected values (a) $f(k_{\mu}^{1},n)$, (b) $f(k_{\mu}^{2},n)$, (c) $f(k_{\mu}^{3},n)$ and (d) $f(k_{\mu}^{4},n)$, of the Gaussian functions

From the Gaussian functions obtained by LA (figure 4), the threshold values T_i are calculated considering equations 15-16. From these values the image segmented in 4 classes shown in figure 7 is obtained.

Fig. 7. Image segmented in 4 classes by LA

In the second experiment, the image shown in figure 8 was used. The idea is again, to segment it in 4 different pixel classes using the LA approach proposed in this work. After execute the algorithm with the parameters detailed in the previous sections the Gaussian functions obtained are shown in figure 9a.

The mix of Gaussian functions obtained by the LA algorithm approach to the original histogram, as can be seen in figure 9b. From figure 9b is clear that the algorithm approaches each one of the pixel concentrations, distributed in the histogram, except to the first one (presented approximately around the intensity value 7). This effect shows that the algorithm discards the smallest accumulation of pixels and prefer to cover those classes that contribute to generate a smaller error during optimization of the equation 3. The results can be improved if 5 pixel classes were used (instead of segmenting the image by 4 pixel classes).

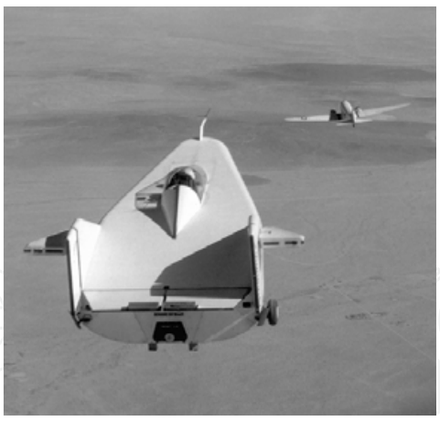
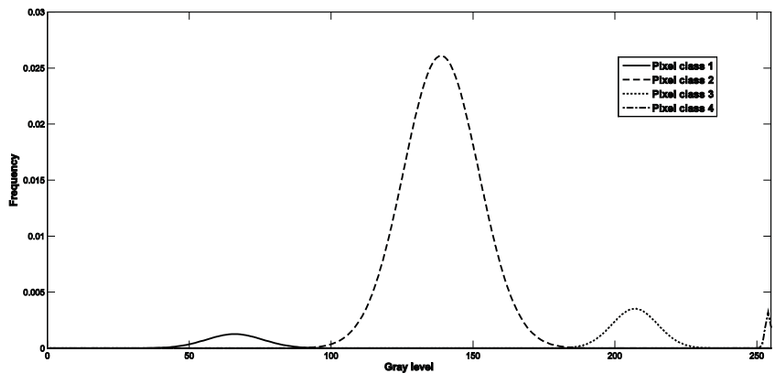
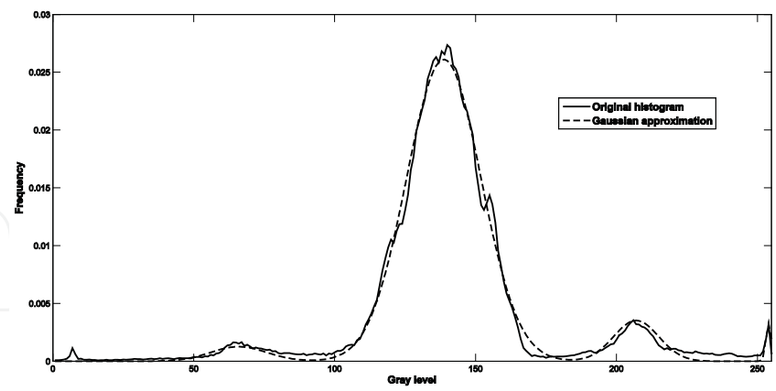


Fig. 8. Image used on the second experiment



(a)

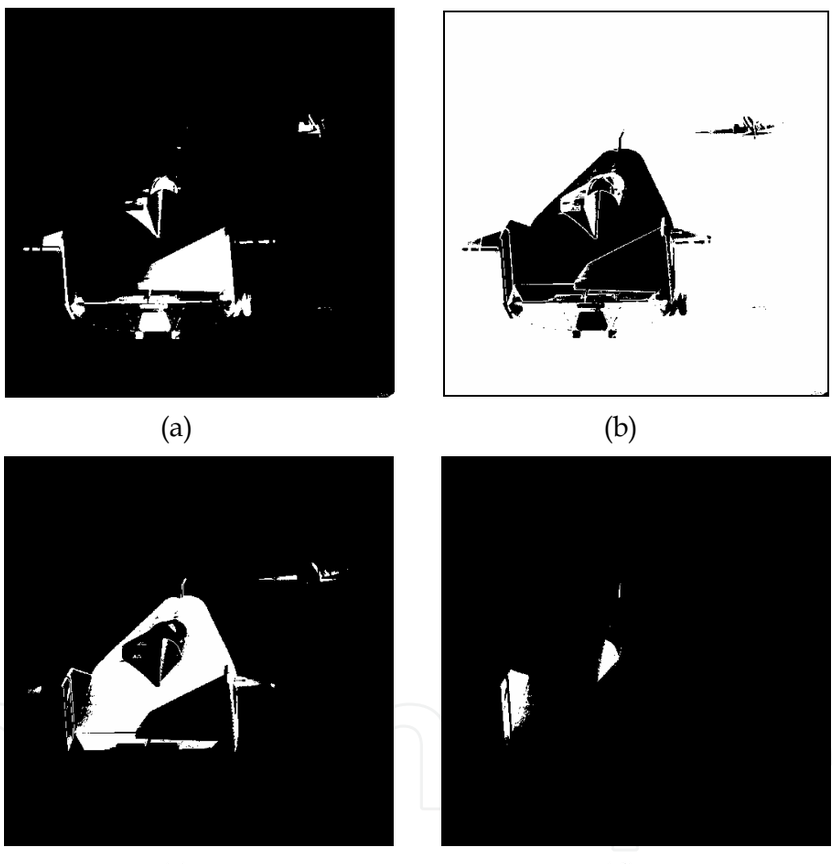


(b)

Fig. 9. (a) Gaussian functions obtained by LA, and (b) its comparison to the original histogram

From the Gaussian functions obtained by LA (figure 9a), the threshold values T_i are calculated considering equations 15-16. From these values the image segmented in 4 classes shown in figure 10 is obtained. Figure 11 shows the separation of each class obtained by the algorithm.

Fig. 10. Segmentation obtained by LA



(a) (b)

(c) (d)

Fig. 11. Separation of each class obtained by the LA algorithm. (a) Pixel class 1, (b) Pixel class 2, (c) Pixel class 3, and (d) Pixel class 3.

6. Conclusions

This works presents a novel segmentation algorithm which includes an automatic threshold determination approach. The overall method can be considered as a Learning automata optimization algorithm. Following the intensity distributions for each object. The intensity

distributions of objects and background in an image are assumed to obey Gaussian distributions with distinct variances and means. The histogram of a given image is approach by a mix of Gaussian probability functions. The LA algorithm is used to estimate the parameters in the mix density function obtaining a minimum square error between the density function and the original histogram. The experimental results reveal that the proposed approach can produce satisfactory results. Further work of extending the proposed approach with other techniques and comparing the results with state of the art image segmentation techniques are in progress.

7. Acknowledgments

The first author would like to thank SEP (Mexico) and DAAD (Germany) for its economical support.

8. References

Abak, T.; Baris, U. & Sankur, B. (1997). The performance of thresholding algorithms for optical character recognition, *Proceedings of International Conference on Document Analytical Recognition*, 697–700.

Baştürk, A. & Günay, E. (2009). Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm, *Expert System with Applications*, 36(8), 2645-2650.

Beygi, H. & Meybodi, M. (2006). A new action-set learning automaton for function optimization. *Int. J. Franklin Inst.* 343, 27–47.

Bhanu, B. (1986). Automatic target recognition: state of the art survey, *IEEE Trans. Aerosp. Electron. Syst.* 22, 364–379.

Chen, S. & Wang, M. (2005). Seeking multi-thresholds directly from support vectors for image segmentation, *Neurocomputing*, 67(4), 335-344.

Chih-Chih, L. (2006). A Novel Image Segmentation Approach Based on Particle Swarm Optimization, *IEICE Trans. Fundamentals*, 89(1), 324-327.

Gonzalez, R. & Woods, R. (1992). Digital Image Processing, Addison Wesley, Reading, MA.

Guo, R. & Pandit, S. (1998). Automatic threshold selection based on histogram modes and discriminant criterion, *Mach. Vis. Appl.*, vol.10, pp.331–338.

Howell, M. & Gordon T. (2000). Continuous action reinforcement learning automata and their application to adaptive digital filter design. *Engineering Applications of Artificial Intelligence.* 14, 549–561.

Kamel, M. & Zhao, A. (1993). Extraction of binary character/graphics images from grayscale document images, *Graph. Models Image Process*. 55 (3) 203–217.

Lai, C. & Tseng, D. (2001). An optimal L-filter for reducing blocking artifacts using genetic algorithms, *Signal Process.*, vol.81, no.7, pp.1525–1535.

Najim, K. & Poznyak, A. (1994). Learning Automata - Theory and Applications. Pergamon Press, Oxford.

Narendra, K. & Thathachar, M. (1989). Learning Automata: an Introduction. Prentice-Hall, London.

Pal, N. & Pal, S. (1993). A review on image segmentation techniques, *Pattern Recognit.*, vol.26, pp.1277–1294.

Seyed-Hamid, Z. (2008). Learning automata based classifier, *Pattern Recognition Letters*. 29, 40-48.

Sezgin, M. & Sankur, B. (2001). Comparison of thresholding methods for non-destructive testing applications, *IEEE International Conference on Image Processing*, 764–767.

Sezgin, M. & Tasaltin, R. (2000). A new dichotomization technique to multilevel thresholding devoted to inspection applications, *Pattern Recognition Lett.* 21 (2) 151–161.

Shaoo, P. ; Soltani, S. ; Wong, A. & Chen, Y. (1988). Survey: A survey of thresholding techniques, *Comput. Vis. Graph. Image Process.*, vol.41, pp.233–260.

Snyder, W. ; Bilbro, G. ; Logenthiran, A. & Rajala, S. (1990). Optimal thresholding: A new approach, *Pattern Recognit. Lett.*, vol.11, pp.803–810.

Thathachar, M. & Sastry, P. (2002). Varieties of learning automata: An overview. *IEEE Trans. Systems. Man Cybernet. Part B: Cybernet.* 32, 711–722.

Trier, O. & Jain, A. (1995). Goal-directed evaluation of binarization methods, IEEE Trans. *Pattern Anal. Mach. Intel.* 17 (12) 1191–1201.

Tseng, D. & Lai, C. (1999). A genetic algorithm for MRF-based segmentation of multispectral textured images, *Pattern Recognit. Lett.*, vol.20, no.14, pp.1499–1510.

Wu, Q. (1995). Learning coordinated control of power systems using inter-connected learning automata. *Int. J. Electr. Power Energy Syst.* 17, 91–99.

Zeng, X. & Liu, Z. (2005). A learning automaton based algorithm for optimization of continuous complex function. *Information Sciences.* 174, 165–175.

Zeng, X. ; Zhou, J. & Vasseur, C. (2000). A strategy for controlling non-linear systems using a learning automaton. *Automatica*. 36, 1517–1524.

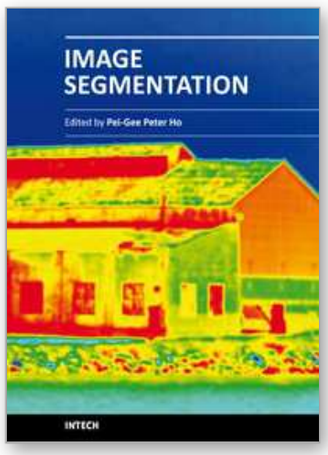


Image Segmentation
Edited by Dr. Pei-Gee Ho

ISBN 978-953-307-228-9
Hard cover, 538 pages
Publisher InTech
Published online 19, April, 2011
Published in print edition April, 2011

It was estimated that 80% of the information received by human is visual. Image processing is evolving fast and continually. During the past 10 years, there has been a significant research increase in image segmentation. To study a specific object in an image, its boundary can be highlighted by an image segmentation procedure. The objective of the image segmentation is to simplify the representation of pictures into meaningful information by partitioning into image regions. Image segmentation is a technique to locate certain objects or boundaries within an image. There are many algorithms and techniques have been developed to solve image segmentation problems, the research topics in this book such as level set, active contour, AR time series image modeling, Support Vector Machines, Pixon based image segmentations, region similarity metric based technique, statistical ANN and JSEG algorithm were written in details. This book brings together many different aspects of the current research on several fields associated to digital image segmentation. Four parts allowed gathering the 27 chapters around the following topics: Survey of Image Segmentation Algorithms, Image Segmentation methods, Image Segmentation Applications and Hardware Implementation. The readers will find the contents in this book enjoyable and get many helpful ideas and overviews on their own study.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen