ELSEVIER

Signal Processing 60 (1997) 305–313

SIGNAL PROCESSING

A fast iterative scheme for multilevel thresholding methods[1]

Peng-Yeng Yin[a], Ling-Hwei Chen[b,*]

[a] *Department of Information Management, Ming Chuan University, Taipei, Taiwan 11120, ROC*
[b] *Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050, ROC*

Received 5 May 1994; revised 27 May 1995 and 4 April 1997

Abstract

The previously published optimal thresholding techniques based on some objective functions are very efficient in the bi-level thresholding case, but they are impractical when extended to multilevel thresholding. The reason for this is their computational complexity which grows exponentially with the number of thresholds. In this paper, an iterative scheme is proposed to render these optimal thresholding techniques more practical. The proposed algorithm starts with a bi-level thresholding, then uses the initial results to obtain higher-order thresholds. This algorithm is iterative and the convergence is proved. We also introduce some useful programming techniques to make the computation more efficient. The proposed algorithm can therefore determine the number of thresholds automatically as well as save a significant amount of computing time. © 1997 Elsevier Science B.V.

Zusammenfassung

Die bereits veröffentlichten optimalen Schwellwertverfahren, die auf gewissen Zielfunktionen aufbauen, sind im Fall der zweistufigen Schwellwertbildung sehr wirksam, aber sie sind nicht praktikabel, wenn man sie auf mehrstufige Schwellwertbildung erweitert. Der Grund dafür liegt in ihrem Rechenaufwand, der exponentiell mit der Anzahl der Schwellwerte ansteigt. In dieser Arbeit wird ein iteratives Verfahren vorgeschlagen, das diese optimalen Schwellwertverfahren praktisch einsetzbar machen soll. Der vorgeschlagene Algorithmus beginnt mit einer zweistufigen Schwellwertbildung und benutzt darauf diese Anfangsergebnisse, um Schwellwerte höherer Ordnung zu gewinnen. Der Algorithmus arbeitet iterativ und seine Konvergenz wird bewiesen. Wir führen auch einige nützliche Programmiertechniken ein, um die Berechnung noch effizienter zu gestalten. Der vorgeschlagenen Algorithmus kann damit die Anzahl der Schwellen automatisch bestimmen und erlaubt zugleich eine bemerkenswerte Einsparung an Rechenzeit. © 1997 Elsevier Science B.V.

Résumé

Les techniques déjà publiées de seuillage optimal basées sur certaines fonctions objectif sont très efficaces pour le cas du seuillage à deux niveaux, mais elles s'avèrent ne pas être pratiques lorsqu'on les étend au seuillage multi-niveaux. La raison en est leur complexité de calcul qui croît exponentiellement avec le nombre de niveaux. Nous proposons dans cet article une procédure itérative pour rendre ces techniques optimales plus pratiques. L'algorithme proposé part d'un seuillage à deux niveaux, puis utilise les résultats initiaux pour obtenir des seuils d'ordre plus élevé. Cet algorithme est itératif et sa convergence est prouvée. Nous introduisons également certaines techniques de programmation utiles pour rendre le calcul

* Corresponding author.

[1] This research was supported in part by the National Science Council of R.O.C. under contract NSC-82-0408-E-009-425.

PII S0165-1684(97)00080-7

plus efficace. L'algorithme proposé peut de ce fait déterminer le nombre de seuils automatiquement et économiser en même temps un temps de calcul significatif. © 1997 Elsevier Science B.V.

Keywords: Image segmentation; Iterative scheme; Multilevel thresholding

1. Introduction

Image thresholding is widely used for the segmentation of images. If the objects are clearly distinguishable from the background, thresholding is a simple and fast way to segment the objects by detecting the valleys of the gray-level histogram. But thresholding is limited since it ignores the spatial information and it may fail in the case of poor illumination [2, 9]. Thresholding can be classified as bi-level thresholding and multilevel thresholding. Bi-level thresholding classifies the pixels of an image into two groups, one including those pixels with gray levels above a certain threshold, the other including the rest. In general, multilevel thresholding divides the whole range of gray levels into several subranges. Over the years many thresholding techniques have been proposed [12, 7]. They can be classified into two types: optimal thresholding methods [8, 10, 11, 4, 5] and property-based thresholding methods [1, 6, 3, 13]. Optimal thresholding methods select the thresholds by optimizing a certain objective function, while the property-based thresholding methods obtain the thresholds based on some property of the histogram. In this article, we only discuss the issue of optimal thresholding methods and propose a new algorithm to accelerate their speeds.

Optimal thresholding methods search the thresholds which can make the thresholded classes on the histogram reach the desired characteristics. Usually, it is made by optimizing an objective function. Otsu [8] proposed an optimal thresholding method to maximize the separability measures of the classes in gray levels based on discriminant analysis. Another approach [10, 11, 4] originated by Pun maximizes a posteriori entropy to measure the homogeneity of the thresholded classes. Kittler [5] presented a minimum error method to minimize the pixel classification error rate. All of these methods have one common problem, the computational complexity is exponential when extended to multilevel thresholding due to the exhaustive search.

In this paper, we propose an iterative scheme to make the optimal thresholding techniques practical in the case of multilevel thresholding. The proposed algorithm starts with the bi-level thresholding. Based on the result, the higher level thresholds can be obtained by iteratively adjusting the thresholds to optimize the objective function as much as possible. We also give a formal proof to guarantee the convergence of the proposed algorithm. Furthermore, some programming techniques are introduced to implement the proposed algorithm efficiently. The proposed method can determine the number of thresholds and greatly reduce the computational complexity.

The paper is organized as follows. Section 2 describes the previous optimal thresholding methods. In Section 3, the proposed method is detailed. In Section 4, some experimental results are given to compare the performance of the proposed method with those of the previous optimal thresholding techniques. Finally, in Section 5, some conclusions are made.

2. The optimal thresholding methods

The optimal thresholding methods search the gray-level thresholds such that the desired requirements of the thresholded classes on the histogram are satisfied. This is accomplished by maximizing or minimizing a function which uses the gray-level thresholds as the parameters. This function is often referred to as objective function [12, 7, 8] or criterion measure. In this section, we will present two broadly used methods. One is based on entropy criterion (Kapur method [4]), the other is based on between-class variance (Otsu method [8]).

Let the gray levels of a given image range over $[0, L-1]$ and $h(i)$ denote the occurrence of gray level i. Let $N = \sum_{i=0}^{L-1} h(i)$, $P_i = h(i)/N$, for $0 \leqslant i \leqslant L-1$.

(A) *Bi-level Kapur method*:

$$\text{maximize} \quad f(t) = H_0 + H_1, \tag{1}$$

where

$$\omega_0 = \sum_{i=0}^{t-1} P_i, \qquad H_0 = -\sum_{i=0}^{t-1} \frac{P_i}{\omega_0} \ln \frac{P_i}{\omega_0},$$

$$\omega_1 = \sum_{i=t}^{L-1} P_i, \qquad H_1 = -\sum_{i=t}^{L-1} \frac{P_i}{\omega_1} \ln \frac{P_i}{\omega_1},$$

and the optimal threshold is the gray level that maximizes objective function $f(t)$.

(B) *Bi-level Otsu method*:

$$\text{maximize} \quad f(t) = \omega_0\omega_1(\mu_0 - \mu_1)^2, \tag{2}$$

where

$$\omega_0 = \sum_{i=0}^{t-1} P_i, \qquad \mu_0 = \sum_{i=0}^{t-1} i \frac{P_i}{\omega_0},$$

$$\omega_1 = \sum_{i=t}^{L-1} P_i, \qquad \mu_1 = \sum_{i=t}^{L-1} i \frac{P_i}{\omega_1},$$

and the optimal threshold is the gray level that maximizes objective function $f(t)$.

In bi-level thresholding, both Kapur and Otsu methods can find the optimal threshold efficiently. Kapur and Otsu also extended their methods to multilevel thresholding, however, both of these two methods have the disadvantage that the computational time will be very expensive due to the exhaustive search.

Assume that we want to search c optimal thresholds $[t_1, t_2, \ldots, t_c]$, both of Kapur and Otsu methods must exhaustively search all possible values of $[t_1, t_2, \ldots, t_c]$. This will result in a complexity of $O(L^c)$ which grows exponentially with the number of thresholds. Kapur and Otsu methods suffer another disadvantage that the number of thresholds cannot be determined by the methods themselves. Thus, in practical applications, these methods are unsuitable for multilevel thresholding. In order to treat this problem, in the next section, we will propose an iterative scheme to accelerate Kapur and Otsu methods by obtaining a near-optimal solution. The proposed scheme also can determine the number of thresholds automatically.

3. The proposed algorithm

3.1. Iterative near-optimal thresholding algorithm

In this subsection, we will propose an algorithm which can accelerate the searching speed and determine the number of thresholds. Before this algorithm is detailed, an iterative scheme is presented for the convenience of illustration.

Assume that we want to search c thresholds by optimizing a certain objective function like Kapur and Otsu methods, an iterative scheme can be used to optimize the objective function as much as possible under the sense that the computing speed must be fast (within seconds). First, the proposed scheme starts with c initial thresholds. Then, these thresholds are adjusted iteratively to improve the value of the objective function. The proposed scheme will stop when the value is not increased between two consecutive iterations. The details of the proposed scheme are presented as follows.

Iterative Scheme

Input: A gray-level histogram ranging over $[0, L-1]$, a parameter c as the number of required thresholds, an optimal bi-level thresholding method G (such as Kapur or Otsu methods) and its associated objective function f.

Output: The selected c thresholds.

Step 1: Start with a set of c thresholds $[t_1, t_2, \ldots, t_c]$. For convenience of illustration, we add two additional thresholds: $t_0 = 0$ and $t_{c+1} = L - 1$.

Step 2: Compute the value of the objective function f by using the current $[t_1, t_2, \ldots, t_c]$.

Step 3: Set $i = 1$.

Step 4: Consider the part of gray-level histogram on the subrange $[t_{i-1}, t_{i+1}]$ one can find the optimal threshold t^* between t_{i-1} and t_{i+1} by using G. Replace t_i with t^*.

Step 5: Set $i = i + 1$. Go to Step 4 until $i > c$.

Step 6: Compute the value of the objective function f by using current $[t_1, t_2, \ldots, t_c]$.

Step 7: If f is increased, go to Step 3; otherwise, stop.

The selection of the initial thresholds will be presented in the next algorithm. Step 3 through 7 are iterated until the value of the objective function is not increased between two consecutive iterations. The proposed iterative scheme will stop after a finite number of iterations. The following theorem shows that the convergence is guaranteed.

Theorem. *The proposed iterative scheme will stop after a finite number of iterations.*

Proof. Assume that the objective function to be maximized is $f(T)$, where $T=[t_1,t_2,\ldots,t_c]$. Let the initial configuration of the threshold series be T^0, the configuration relation between two consecutive iterations can be written as

$$T^0 \xrightarrow{*} T^1 \xrightarrow{*} T^2 \xrightarrow{*} \cdots,$$

where $T^j \xrightarrow{*} T^{j+1}$ means that the configuration T^j in the jth iteration is changed to the configuration T^{j+1} after one iteration.

According to the stopping criterion of the proposed iterative scheme, if it does not stop, the objective function will keep monotonically increasing, i.e.,

$$f(T^0)<f(T^1)<f(T^2)<\cdots.$$

Since different values of the objective function must result from different configurations of thresholds, it follows that the configurations will not repeat during these iterations.

Consider the problem of selecting c thresholds from L gray levels. The number of all distinct configuration is $L!/[(L-c)!*c!]$, which is finite. Thus, if the objective function keeps monotonically increasing, the configurations will be exhausted after a finite number of iterations. This means that the proposed iterative scheme will stop after a finite number of iterations.

The above iterative scheme still leaves two problems. One is the selection of initial thresholds, the other is the determination of threshold number for a given image. In the following, we propose a method called iterative near-optimal thresholding method to solve these problems. First, the algorithm employs a hierarchical order of thresholding, i.e., 2-level, 3-level, ..., N-level, and the initialization of k-level thresholding is based on the thresholds of $(k-1)$-level thresholding. For the second problem, the proposed algorithm evaluates the results of each level by the uniformity measure which is broadly used in the literature [12,7], then choose the best one as the final result. The uniformity measure U is given by

$$U=1-2^*N^*\frac{\sum_{j=0}^{N}\sum_{i\in R_j}(f_i-\mu_j)^2}{S^*(f_{\max}-f_{\min})^2},$$

where N is the number of thresholds, R_j the segmented region j, f_i the gray level of pixel i, μ_j the mean of the gray levels of those pixels in segmented region j, S the number of total pixels in the given image, $f_{\max}$ the maximum gray level of pixels in the given image, $f_{\min}$ the minimum gray level of pixels in the given image.

The details of the proposed algorithm are presented as follows.

Iterative Near-optimal Thresholding Algorithm

Input: A gray-level histogram ranging over $[0, L-1]$, a maximum number of thresholds N, an optimal bi-level thresholding method G (such as Kapur or Otsu methods).

Output: A series of appropriate thresholds.

Step 1: Apply G to the whole histogram and get an optimal threshold t_1.

Step 2: Set $n=1$.

Step 3: For each subrange $[t_i,t_{i+1}), i=0,1,\ldots,n$ (for convenience, let $t_0=0$, $t_{n+1}=L-1$), compute local mean μ_i and take gray value m_i with $P_{m_i}=\max_{j=t_i-1}^{t_{i+1}} P_j$. Let $D_i=|P_{\mu_i'}-P_{m_i}|$, where μ_i' is the smallest integer greater than μ_i. Choose k with $D_k=\max_{i=0}^{n} D_i$.

Step 4: Apply G to $[t_k,t_{k+1}]$ to find an optimal threshold t^*. Create an array T with $T[0]=t_0$, $T[1]=t_1,\ldots,T[k]=t_k, T[k+1]=t^*, T[k+2]=t_{k+1},\ldots,T[n+1]=t_n, T[n+2]=t_{n+1}$.

Step 5: Apply the proposed iterative scheme with $T[1], T[2],\ldots,T[n+1]$ as the initial thresholds.

Step 6: Record the final values of $T[1], T[2],\ldots,T[n+1]$ as the thresholds of $(n+1)$-level thresholding.

Step 7: Set $n=n+1$.

Step 8: If $n<N$, goto Step 3.

Step 9: Evaluate the uniformity measure for each level of thresholds and output the thresholds with the highest uniformity measure.

The proposed algorithm starts with a bi-level thresholding (see Step 1), and then constructs a hierarchical order of 1 to N thresholds (see Steps 2–8). The process is to add a new threshold in a candidate subrange which is likely to have multiple peaks, and then apply the iterative scheme to get the thresholds. Finally, the result with the best uniformity measure is output. The proposed algorithm is very fast and provides a way to determine the number of thresholds.

In the next subsection, we describe some programming techniques which can additionally speed up the computations.

3.2. Additional speed-up by some programming techniques

To speed up the computations, some programming techniques are provided to accelerate the evaluation of objective functions.

For Kapur method, consider the objective function:

$$f(t) = -\sum_{i=0}^{t} \frac{P_i}{\omega_0} \ln \frac{P_i}{\omega_0} - \sum_{i=t+1}^{L-1} \frac{P_i}{\omega_1} \ln \frac{P_i}{\omega_1}.$$

Let LLGSUM denote $\sum_{i=0}^{t} P_i \ln P_i$ and RLGSUM denote $\sum_{i=t+1}^{L-1} P_i \ln P_i$. Then the above expression can be rewritten as

$$-\left(\frac{\text{LLGSUM}}{\omega_0} - \ln \omega_0\right) - \left(\frac{\text{RLGSUM}}{\omega_1} - \ln \omega_1\right). \tag{3}$$

If we store P_i and $P_i \ln P_i$ for each gray level i, when the searching point for t is from j to $j+1$, we only need the following five operations to compute Eq. (3):

1. $\omega_0 = \omega_0' + P_{j+1}$,
2. $\omega_1 = \omega_1' - P_{j+1}$,
3. $\text{LLGSUM} = \text{LLGSUM}' + P_{j+1} \ln P_{j+1}$,
4. $\text{RLGSUM} = \text{RLGSUM}' - P_{j+1} \ln P_{j+1}$,
5. Evaluate Eq. (3),

where ω_0', ω_1', LLGSUM′ and RLGSUM′ denote the values of ω_0, ω_1, LLGSUM and RLGSUM at $t = j$, respectively.

The objective function of Otsu method can be implemented similarly. By using these techniques, we can save a lot of computing time.

3.3. Analysis of computational complexity

As discussed in Section 2, the computational complexity of the original Kapur and Otsu method is $O(L^c)$ which grows exponentially with the number of thresholds c. By using the proposed algorithm, in the worst case, the objective function will be computed kN^2L times where k is the number of iterations and N is the maximum number of thresholds. So the complexity of the proposed algorithm in the worst case is $O(kN^2L)$. The complexity of k cannot be derived analytically, many kinds of images (including fruits, characters, commercial advertisements, etc.) have been tested by the proposed algorithm to know the experimental value of k. It is shown that k is less than 10. On the other hand, N is no larger than 5 in the real applications. In the next section, the experimental results show that the proposed algorithm can save the computational time dramatically.

4. Experimental results and comparative performances

In this section, we will evaluate the performances of the following methods: Kapur, Otsu, the proposed algorithm using Kapur's objective function (we shall refer to this as Iterative-Kapur), and the proposed algorithm using Otsu's objective function (we shall refer to this as Iterative-Otsu). Our experiments are performed on Sun Sparc/4 Workstation. Three real images named Japan, Eye and Pepper are used (see Figs. 1(a)–(c)). As shown in the images, Japan has two classes of pixels, Eye has three classes of pixels, and Pepper is more complicated with multiple classes. Their gray-level histograms also show this situation (see Figs. 2(a)–(c)). We additionally add an artificial histogram called Normal constructed with four normal distributions N(30,6), N(100,6), N(130,6) and N(220,6) as illustrated in Fig. 2(d). For Kapur and Otsu methods, we must input the number of

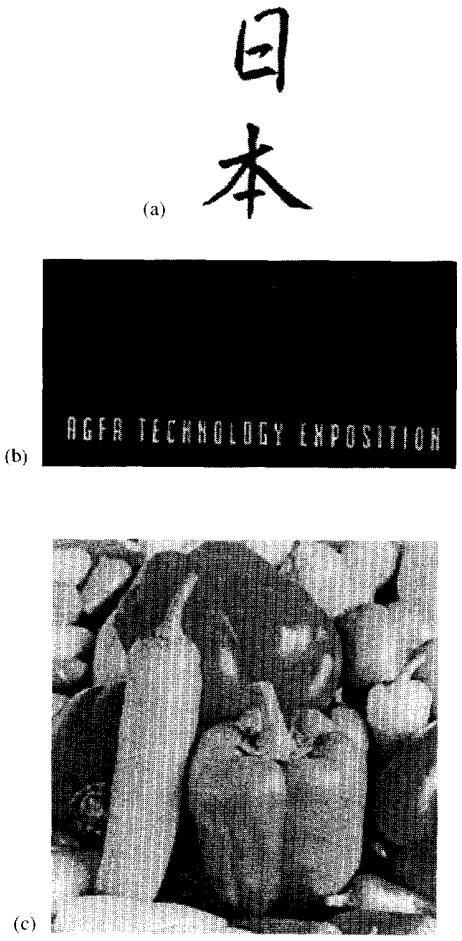

Fig. 1. The tested images. (a) Image Japan; (b) image Eye; (c) image Pepper.

thresholds in order to get the results. The obtained optimal thresholds are illustrated on the gray-level histograms (see Figs. 3(a)–(h)) and they separate the pixel classes quite well. However, the computational time grows exponentially with the number of thresholds (see Table 1). For the proposed methods (Iterative-Kapur and Iterative-Otsu), we construct a 5-level hierarchical order of thresholds for each image, since the number of thresholds is hardly more than five in real applications. The individual hierarchical order and the corresponding uniformity measures are presented in Tables 2 and 3. The output results with the highest uniformity measure are shaded and they are as same as those optimal thresholds (see

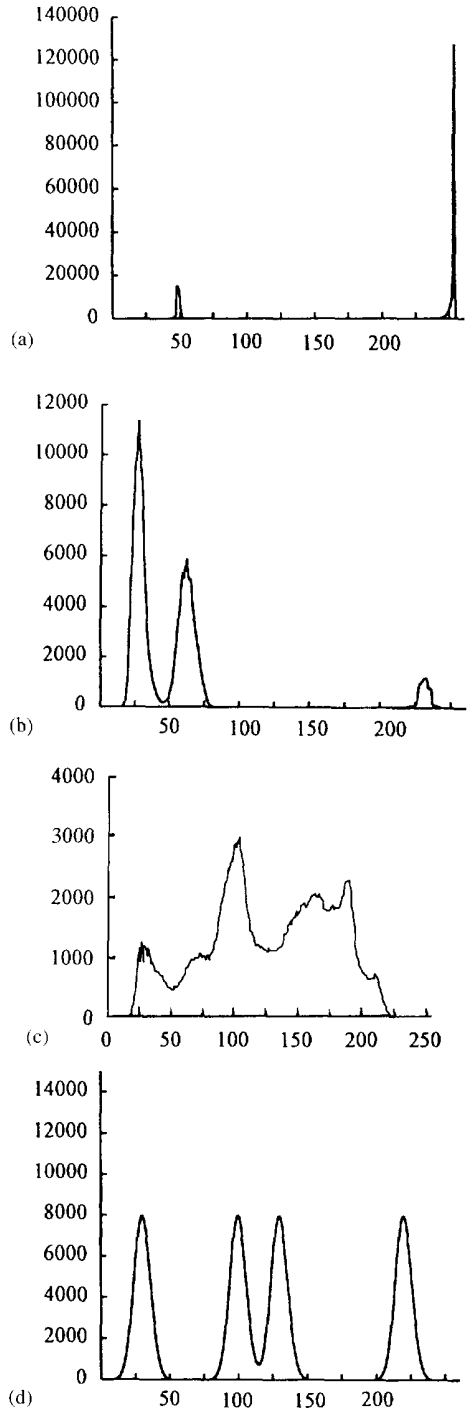

Fig. 2. The gray-level histograms of the tested images. (a) The gray level histogram of Japan; (b) the gray level histogram of Eye; (c) the gray level histogram of Pepper; (d) the gray level histogram of Normal.

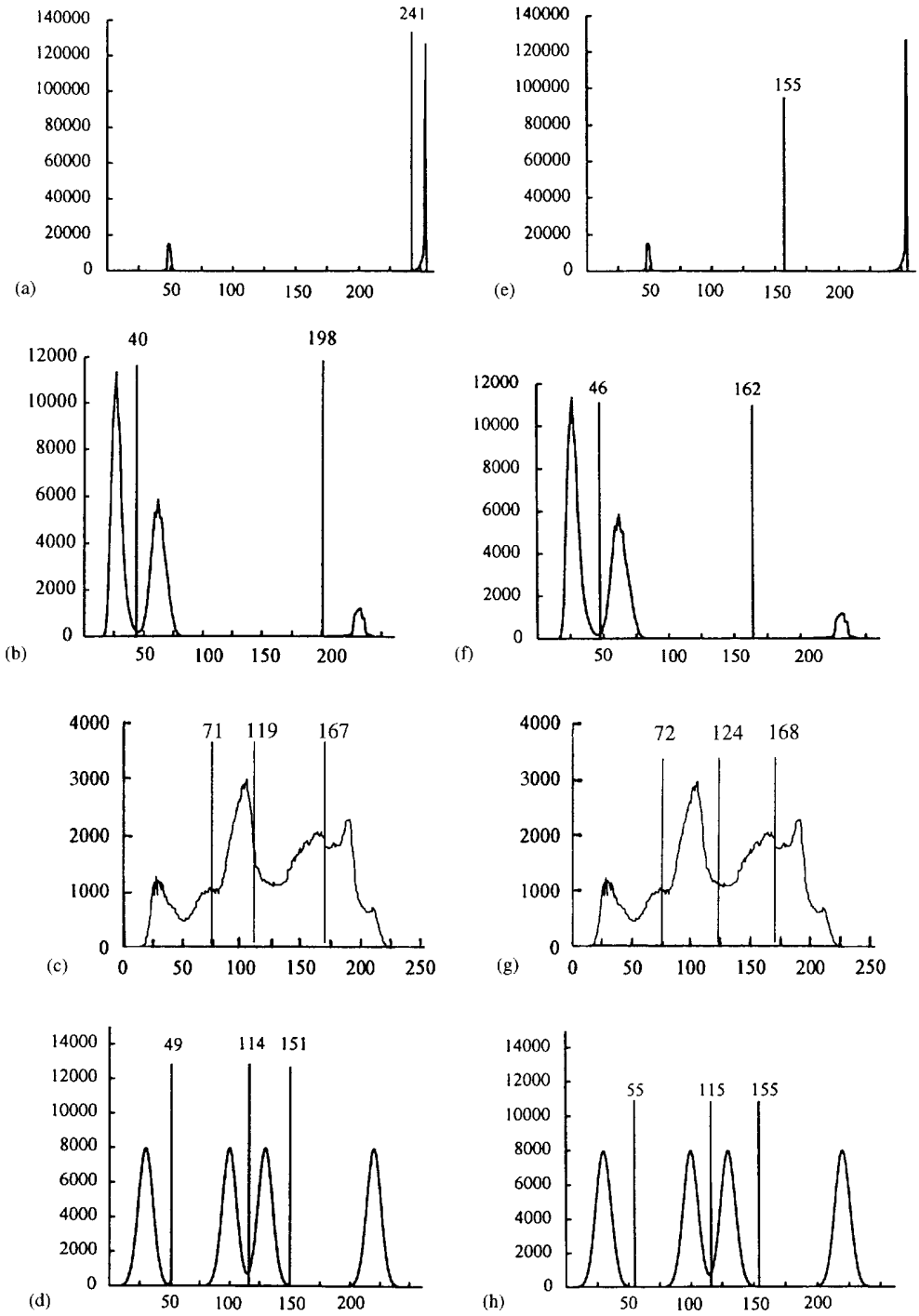

Fig. 3. The obtained optimal thresholds by using Kapur and Otsu methods. (a) The thresholds of Japan by using Kapur; (b) the thresholds of Eye by using Kapur; (c) the thresholds of Pepper by using Kapur; (d) the thresholds of Normal by using Kapur; (e) the thresholds of Japan by using Otsu; (f) the thresholds of Eye by using Otsu; (g) the thresholds of Pepper by using Otsu; (h) the thresholds of Normal by using Otsu.

Table 1
The computing time of thresholding Japan, Eye, Pepper and Normal by using various methods (seconds)

Images	Kapur	Otsu	Iterative-Kapur	Iterative-Otsu
Japan	0.5	0.2	5.8	3.7
Eye	70	26	4.6	2.5
Pepper	9545	2516	5.6	2.4
Normal	5736	2265	6.0	2.1

Table 2
The hierarchical thresholds and uniformity measures of tested images by using Iterative-Kapur. The bold entries indicate the output results

Images	Hierarchical order of thresholds	Uniformity measures
Japan	**241**	**0.969155**
	209,238	0.963848
	209,224,240	0.946042
	203,220,233,244	0.938705
	203,217,228,238,246	0.923737
Eye	198	0.438706
	40,198	**0.782079**
	40,198,224	0.679116
	37,53,198,224	0.630775
	31,40,53,198,224	0.570817
Pepper	92	0.787444
	84,149	0.868958
	71,119,167	**0.896028**
	59,95,145,192	0.868919
	53,90,118,145,193	0.889485
Normal	115	0.148096
	94,134	0.407490
	49,114,151	**0.942589**
	41,110,141,208	0.843569
	41,88,114,141,208	0.876398

Table 3
The hierarchical thresholds and uniformity measures of tested images by using Iterative-Otsu. The bold entries indicate the output results

Images	Hierarchical order of thresholds	Uniformity measures
Japan	**155**	**0.972673**
	100,200	0.811365
	83,154,222	0.877217
	76,132,189,234	0.793070
	73,121,174,222,248	0.679380
Eye	132	0.497231
	46,162	**0.884357**
	46,107,190	0.854454
	43,63,111,191	0.864846
	29,46,64,111,191	0.864282
Pepper	126	0.868559
	77,140	0.873957
	72,124,168	**0.898195**
	58,94,143,171	0.892370
	55,88,119,142,191	0.889065
Normal	153	0.306100
	55,155	0.846718
	55,115,155	**0.943740**
	55,100,117,155	0.937642
	55,95,104,120,155	0.927025

Figs. 3(a)–(h)). The proposed methods can save a lot of computing time (see Table 1) and the number of thresholds is determined automatically.

5. Conclusions

In this paper, we have proposed a fast iterative algorithm to improve the optimal thresholding methods. First, the algorithm starts with a bi-level thresholding and bases on the result to obtain higher level thresholds. The proposed algorithm adjusts the thresholds in an iterative manner to optimize the objective function as much as possible. We have introduced some programming techniques to further improve the computational speed. The experimental results show that the proposed method is computationally fast and determines the number of thresholds correctly.

References

[1] S. Boukharouba, J.M. Rebordao, P.L. Wendel, An amplitude segmentation method based on the distribution function of an image, Comput. Vision Graphics Image Process. 29 (1985) 47–59.

[2] R.M. Haralick, L.G. Shapiro, Image segmentation techniques, Comput. Vision Graphics Image Process. 29 (1985) 100–132.
[3] R.M. Haralick, K. Shanmugam, I. Dinstein, Texture features for image classification, IEEE Trans. Systems Man. Cybernet. 3 (1973) 610–621.
[4] J.N. Kapur, P.K. Sahoo, A.K.C. Wong, A new method for gray-level picture thresholding using the entropy of the histogram, Comput. Vision Graphics Image Process. 29 (1985) 273–285.
[5] J. Kittler, J. Illingworth, Minimum error thresholding, Pattern Recognition 19 (1986) 41–47.
[6] W.L.G. Koontz, P. Narendra, K. Fukunaga, A graph-theoretic approach to nonparametric cluster analysis, IEEE Trans. Comput. 25 (1976) 936–944.
[7] S.U. Lee, S.Y. Chung, R.H. Park, A comparative performance study of several global thresholding techniques for segmentation, Comput. Vision Graphics Image Process. 52 (1990) 171–190.
[8] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Syst. Man. Cybernet. SMC-9 (1979) 62–66.
[9] N.R. Pal and S.K. Pal, A review on image segmentation techniques, Comput. Vision Graphics Image Process. 29 (1993) 1277–1294.
[10] T. Pun, A new method for grey-level picture thresholding using the entropy of the histogram, Signal Processing 2 (1980) 223–237.
[11] T. Pun, Entropy thresholding: A new approach, Comput. Vision Graphics Image Process. 16 (1981) 210–239.
[12] P.K. Sahoo, S. Soltani, A.K.C. Wong, Y.C. Chen, A survey of thresholding techniques, Comput. Vision Graphics Image Process. 41 (1988) 233–260.
[13] W. Tsai, Moment-preserving thresholding: A new approach, Comput. Vision Graphics Image Process. 29 (1985) 377–393.