PERGAMON

Pattern Recognition 36 (2003) 2793–2804

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Fast adaptive PNN-based thresholding algorithms

Kuo-Liang Chung*,[1], Wan-Yu Chen

Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan ROC

Received 22 October 2002; accepted 2 April 2003

Abstract

Thresholding is a fundamental operation in image processing. Based on the pairwise nearest neighbor technique and the variance criterion, this theme presents two fast adaptive thresholding algorithms. The proposed first algorithm takes $O((m-k)m\tau)$ time where k denotes the number of thresholds specified by the user; m denotes the size of the compact image histogram, and the parameter τ has the constraint $1 \leqslant \tau \leqslant m$. On a set of different real images, experimental results reveal that the proposed first algorithm is faster than the previous three algorithms considerably while having a good feature-preserving capability. The previous three mentioned algorithms need $O(m^k)$ time. Given a specific peak-signal-to-noise ratio (PSNR), we further present the second thresholding algorithm to determine the number of thresholds as few as possible in order to obtain a thresholded image satisfying the given PSNR. The proposed second algorithm takes $O((m-k)m\tau+\gamma N)$ time where N and γ denote the image size and the fewest number of thresholds required, respectively. Some experiments are carried out to demonstrate the thresholded images that are encouraging. Since the time complexities required in our proposed two thresholding algorithms are polynomial, they could meet the real-time demand in image preprocessing.

Keywords: Algorithms; Clustering; Compact Histogram; PNN; PSNR; Thresholding

1. Introduction

Thresholding is an important operation in image segmentation. Previously, Otsu [1] presented a statistical approach to determine the thresholds. Assuming that the gray levels of each object in the input image are normally distributed, Kittler and Illingworth [2] presented an efficient thresholding algorithm. Kapur *et al.* [3] presented an entropy-based thresholding algorithm. All the three developed thresholding algorithms need $O(m^k)$ time where m denotes the size of the compact image histogram and k denotes the number of thresholds to be determined. For a fixed m, the time bound required in the previous three mentioned algorithms are exponentially proportional to k. Following the concepts used in the above three thresholding algorithms, several variants [4–11] were presented. However, they are still time consuming for some large k.

Based on the PNN approach [12,13] and the variance criterion [14], which are originally used in vector quantization, this theme presents two new thresholding algorithms. The proposed first algorithm only takes $O((m-k)m\tau)$ time where the parameter τ has the constraint $1 \leqslant \tau \leqslant m$. On a set of different real images, experimental results reveal that the proposed first algorithm is faster than the previous three algorithms [1–3] considerably while having a good feature-preserving capability. This confirms the theoretical analysis that our proposed algorithm has a polynomial time bound while the time bound required in the previous three algorithms are exponentially proportional to k. Given a specific PSNR, following a modified version of the proposed first algorithm, we present the second thresholding algorithm to determine the number of thresholds as few as possible such that the thresholded image satisfies the given PSNR. The proposed second algorithm takes $O((m-k)m\tau+\gamma N)$ time where N and γ denote the image size and the fewest

* Corresponding author. Tel.: 886-2-27376771; fax: 886-2-27301081.

E-mail address: klchung@cs.ntust.edu.tw (K.-L. Chung).

[1] Supported by the contract NSC89–2218–E011–017.

0031-3203/03/$30.00 © 2003 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/S0031-3203(03)00138-9

number of determined thresholds, respectively. Some experiments are carried out to demonstrate the thresholded images that are encouraging. To the best of our knowledge, this is the first time that such a thresholding algorithm is presented to satisfy the specified image quality requirement.

2. Preliminary

This section gives a brief survey for the previous three algorithms mentioned above. For saving the space of the context, we only consider the case $k = 1$. Let each pixel of the image have gray level in $[0, 1, \ldots, I-1]$, and commonly $I = 256$. The number of pixels with gray level i is denoted by n_i, $0 \leqslant i \leqslant I-1$, and the total number of pixels is denoted $N = n_0 + n_1 + \cdots + n_{I-1}$. Thus, the gray level histogram is defined as a probability distribution: $p(i) = n_i/N$, $p(i) \geqslant 0$, and $\sum_{i=0}^{I-1} p(i) = 1$. Suppose these N pixels is partitioned into two clusters C_1 and C_2 by the threshold T for $0 \leqslant T \leqslant I-1$, where C_1 denotes those pixels with gray levels in $[0, 1, \ldots, T]$ and C_2 denotes those pixels with gray levels in $[T+1, \ldots, I-1]$. In order to unify the parameters used throughout this theme, the probabilities of the cluster occurrence, ω_j, the cluster mean level, μ_j, the cluster variance, σ_j^2, and the mean level of the original image, μ_T, are defined by $\omega_1 = Pr(C_1) = \sum_{i=0}^{T} p(i)$, $\omega_2 = Pr(C_2) = \sum_{i=T+1}^{I-1} p(i)$, $\mu_1 = \sum_{i=0}^{T} ip(i)/\omega_1$, $\mu_2 = \sum_{i=T+1}^{I-1} ip(i)/\omega_2$, $\sigma_1^2 = \sum_{i=0}^{T} (i-\mu_1)^2 p(i)/\omega_1$, $\sigma_2^2 = \sum_{i=T+1}^{I-1} (i-\mu_2)^2 p(i)/\omega_2$, and $\mu_T = \sum_{i=0}^{I-1} ip(i)$, respectively.

2.1. The work by Otsu [1]

In Ref. [1], the threshold $T^\star$ is determined by maximizing the between-cluster variance σ_B^2 or minimizing the within-cluster variance σ_W^2 where $\sigma_B^2(T) = \omega_1(\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2$ and $\sigma_W^2(T) = \omega_1\sigma_1^2 + \omega_2\sigma_2^2$. The found $T^\star$ satisfies $\sigma_B^2(T^\star) = \max \sigma_B^2(T)$ for $0 \leqslant T \leqslant I-1$ and $\sigma_W^2(T^\star) = \min \sigma_W^2(T)$ for $0 \leqslant T \leqslant I-1$ where the total variance of gray levels satisfies $\sigma_T^2 = \sigma_B^2 + \sigma_W^2$.

Suppose we transfer the image histogram into a compact histogram, which will be explained in Section 3. It is not hard to verify that Otsu's thresholding algorithm takes $O(m^k)$ time.

2.2. The work by Kittler and Illingworth [2]

In Ref. [2], Kittler and Illingworth assume that the gray levels of each object (background or foreground) in an image have a Gaussian distribution. The threshold is determined by minimizing the Kullback directed divergence J. The function J is defined in terms of the histogram $p(0), p(1), \ldots, p(I-1)$ and the unknown mixture distribution f: $J(p; f) = \sum_{i=0}^{I-1} p(i)\log(p(i)/f(i))$, where $f(i) = (\omega_1/\sqrt{2\pi}\sigma_1)e^{-1/2((i-\mu_1)/\sigma_1)^2} + (\omega_2/\sqrt{2\pi}\sigma_2)e^{-1/2((i-\mu_2)/\sigma_2)^2}$. J can be rewritten by $J = \sum_{T=0}^{I-1} p(T)\log p(T) - \sum_{T=0}^{I-1} p(T)\log f(T)$. The first summation term at the right-hand side of J is a fixed value, so the minimization of J can be done by minimizing the second summation term. Let $H(T)$ be the second summation term and we have $H(T) = -\sum_{T=0}^{I-1} p(T)\log f(T)$. The value $T^\star$ that minimizes $H(T)$ is the desired threshold and can be obtained by computing $H(T^\star) = \min_{0 \leqslant T \leqslant I-1} H(T)$.

Using Kittler and Illingworth's thresholding algorithm to obtain the threshold, it also takes $O(m^k)$ time when employing the compact image histogram.

2.3. The work by Kapur et al. [3]

In Ref. [3], the threshold is determined based on the entropy concept. Suppose T is selected to partition the histogram into two clusters C_1 and C_2. The entropy of C_1 is given by $H(C_1) = -\sum_{i=0}^{T} p(i)/\omega_1 \log(p(i)/\omega_1) = -1/\omega_1[\sum_{i=0}^{T} p(i)\log p(i) - p(i)\log\omega_1] = \log\omega_1 + H_T/\omega_1$, where $H_T = -\sum_{i=0}^{T} p(i)\log p(i)$. Similarly, the entropy of C_2 is given by $H(C_2) = -\sum_{i=T+1}^{I-1} (p(i)/\omega_2)\log(p(i)/\omega_2) = -1/\omega_2[\sum_{i=T+1}^{I-1} p(i)\log p(i) - p(i)\log\omega_2] = \log\omega_2 + (H_{I-1} - H_T)/\omega_2$. Let $\psi(T)$ denotes the sum of $H(C_1)$ and $H(C_2)$, then it yields to $\psi(T) = \log\omega_1\omega_2 + H_T/\omega_1 + (H_{I-1} - H_T)/\omega_2$. $\psi(T)$ is maximized to obtain the maximum information when summing up $H(C_1)$ and $H(C_2)$. The value $T^\star$ which maximizes $\psi(T)$ is the desired threshold and is obtained by $\psi(T^\star) = \max_{0 \leqslant T < I-1} \psi(T)$.

Using Kapur *et al.*'s thresholding algorithm to obtain the threshold also takes $O(m^k)$ time when giving the compact image histogram.

3. The proposed two algorithms

In this section, based on the PNN concept [12,13] and the variance concept [14] used in vector quantization, we first present a faster thresholding algorithm when compared to the previous three algorithms mentioned above while having a good feature-preserving capability. Next, based on the modified version of our proposed first algorithm, a novel adaptive thresholding algorithm is presented to satisfy the PSNR requirement.

3.1. The first algorithm

It is known that the maximal number of gray levels allowable in the image is I and the probability of gray level i, $0 \leqslant i \leqslant I-1$, is denoted by $p(i)$. Suppose those zero $p(j)$s' are deleted and we pack the remaining nonzero $p(k)$'s, say m nonzero $p(k)$'s, into an array with size m. We thus have a compact image histogram with the probability distribution $\langle p(i_0), p(i_1), \ldots, p(i_{m-1})\rangle$ for $0 \leqslant i_j \leqslant I-1$, $0 \leqslant j \leqslant m-1$, and $p(i_j) \neq 0$. The total number of pixels is denoted by $N = n_{i_0} + n_{i_1} + \cdots + n_{i_{m-1}} = n'_0 + n'_1 + \cdots + n'_{m-1}$ where $n_{i_j} = n'_j = N \times p(i_j)$. We now take a small example to explain how to obtain the compact image histogram from

i	0	...	9	10	11	...	24	25	26	...	29	30	31	...	34	35
n_i	0	...	0	10	0	...	0	20	0	...	0	25	0	...	0	15
$p(i)$	0	...	0	0.1	0	...	0	0.2	0	...	0	0.25	0	...	0	0.15

i	36	...	79	80	81	...	84	85	86	...	89	90	91	...	I-2	I-1
n_i	0		0	5	0		0	15	0		0	10	0		0	0
$p(i)$	0	...	0	0.05	0	...	0	0.15	0	...	0	0.1	0	...	0	0

Fig. 1. An example of image histogram.

j	0	1	2	3	4	5	6
i_j	10	25	30	35	80	85	90
$p(n_{i_j})$	0.1	0.2	0.25	0.15	0.05	0.15	0.1

Fig. 2. The compact image histogram of Fig. 1.

the image histogram. Given an image, suppose there are $N = 100$ nonzero $p(k)$'s and we have the following image histogram as shown in Fig. 1.

According to the above definition, the compact image histogram is shown in Fig. 2. Considering Fig. 2, initially we assign a node with index j to the index pair (i_j, n_{i_j}). For example, we assign node 1 to the index pair $(25, 20)$. For node 1, the grey level $I_j = 25$ is counted as the initial mean of node 1.

Initially, each node is viewed as a cluster. We thus have m clusters, say $C_0, C_1, \ldots$, and C_{m-1}. Let these m clusters be represented by the set S and the size of the cluster C_i be denoted by $|C_i| = n'_i$. For any two clusters, C_j and C_k, for $j \neq k$, $C_j \in S$, and $C_k \in S$, the squared error measure [14] introduced by the cluster C_q after merging C_j and C_k is given by

$$
\begin{aligned}
n'_q\sigma_q^2 &= \sum_{x\in C_q}(x-\bar{X}_q)^2 \\
&= \sum_{x\in C_j}(x^2-2x\bar{X}_q+\bar{X}_q^2)+\sum_{x\in C_k}(x^2-2x\bar{X}_q+\bar{X}_q^2) \\
&= \{[n'_j(\sigma_j^2+\bar{X}_j^2)]-2n'_j\bar{X}_j\bar{X}_q+(n'_j\bar{X}_q^2)\} \\
&\quad +\{[n'_k(\sigma_k^2+\bar{X}_k^2)]-2n'_k\bar{X}_k\bar{X}_q+(n'_k\bar{X}_q^2)\} \\
&= \{n'_j\sigma_j^2+n'_j(\bar{X}_j-\bar{X}_q)^2\}+\{n'_k\sigma_k^2+n'_k(\bar{X}_k-\bar{X}_q)^2\} \\
&= n'_j\sigma_j^2+n'_k\sigma_k^2+\left\{n'_j\left(\frac{n'_k\bar{X}_j-n'_k\bar{X}_k}{n'_j+n'_k}\right)^2\right\} \\
&\quad +\left\{n'_k\left(\frac{n'_j\bar{X}_k-n'_j\bar{X}_j}{n'_k+n'_j}\right)^2\right\} \\
&= n'_j\sigma_j^2+n'_k\sigma_k^2+\frac{n'_jn'_k}{(n'_j+n'_k)}(\bar{X}_j-\bar{X}_k)^2 \\
&= n'_j\sigma_j^2+n'_k\sigma_k^2+\frac{n'_jn'_k}{n'_q}(\bar{X}_j-\bar{X}_k)^2, \qquad (1)
\end{aligned}
$$

where C_q denotes the merged cluster of C_j and C_k; $\bar{X}_q$, $\bar{X}_j$, and $\bar{X}_k$ denote the mean of C_q, C_j, and C_k, respectively; $n'_q = n'_j + n'_k$, n'_j, and n'_k denote the number pixels of C_q, C_j, and C_k, respectively; σ_q^2, σ_j^2, and σ_k^2 denote the variance of C_q C_j, and C_k, respectively. The third term $(n'_jn'_k/n'_q)(\bar{X}_j-\bar{X}_k)^2$ at the right-hand side of Eq. (1) can be viewed as the distance caused by merging clusters C_j and C_k and the distance is denoted by

$$
d(C_j, C_k) = \frac{n'_jn'_k}{n'_q}(\bar{X}_j-\bar{X}_k)^2. \qquad (2)
$$

The distance is symmetric because of $d(C_j, C_k) = d(C_k, C_j)$ and it can be calculated in $O(1)$ time. The smaller the distance (see Eq. (2)) is, the stronger the correlation between C_j and C_k is. For all cluster-pairs, suppose the distance between any two clusters is known. The next merging process is to find the minimal distance among all the distances in all the cluster-pairs.

According to Fig. 2, the simulation of the first merging process is illustrated in Fig. 3. Initially, each nonzero gray level i_j is viewed as a cluster with index j. As shown in Fig. 3(a), the first cluster C_0 with index 0 contains $i_0 = 10$ and $n_{10} = n'_0 = 10$. The second cluster C_1 with index 1 contains $i_1 = 25$ and $n_{25} = n'_1 = 20$. By Eq. (2), for any cluster, we compute the minimum distance between that cluster and the other cluster, then retain the smallest distance. As shown in Fig. 3(b), the minimum distance for each cluster is highlighted by the one-way arrow. For example, the minimum distance for C_0 is 1500 $(=\frac{10\times20}{10+20}(10-25)^2)$ denoting the distance between C_0 and C_1 and the minimum distance for C_1 is 278 $(=\frac{20\times25}{20+25}(25-30)^2)$ denoting the distance between C_1 and C_2. Then, we find the minimal one among these seven minimum distances. In our example, the distance 94 between C_4 and C_5 is the minimal one as shown in Fig. 3(c), where the two gray nodes denote the nearest pair. Then, as shown in Fig. 3(d), we merge C_4 and C_5 into a new cluster with smaller index, i.e. C_4, where the updated mean is 83.5 $(=\frac{5\times80+15\times85}{5+15})$ and the size of the merged cluster becomes 20 $(=5+15)$ which denotes the number of pixels in the merged cluster. We further update the minimum distance of the merged cluster C_4. By Eq. (2), the distance between the new cluster C_4 (merging the old C_4 and C_5) and the cluster C_6 is equal to 260 $(=\frac{2010}{20+10}(83.75-90)^2)$. So, we assign the distance 260 to the arrow line connecting the merged cluster C_4 and the cluster C_6. We repeat the above three manipulations: (1) finding the minimal one among the

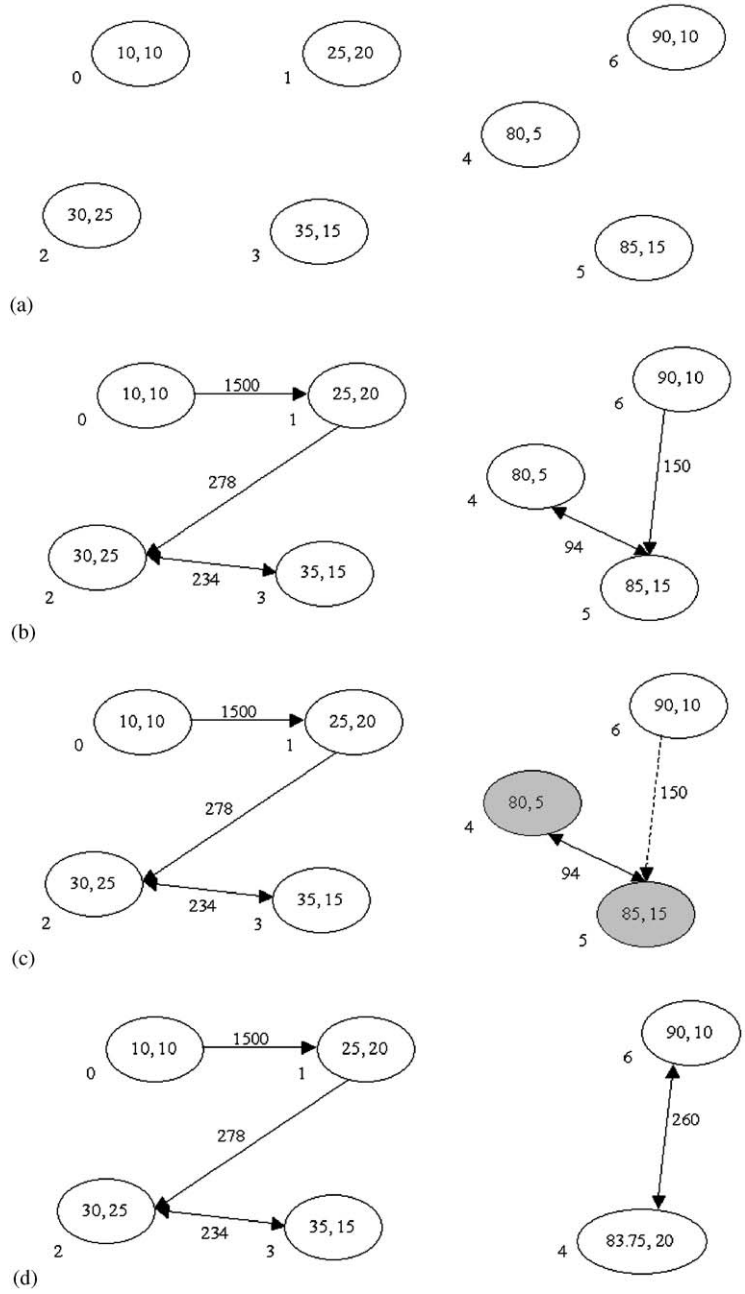


Fig. 3. The simulation of the first merging process for Fig. 2: (a) initial step; (b) calculating the distance; (c) finding the minimal distance; (d) merging and updating.

current different distances, (2) merging the nearest pair, and (3) updating the merged cluster and the related distance, until the desired $k + 1$ merged clusters, i.e. the k thresholds, are obtained.

We now describe how to adopt the table data structure [15] to maintain the above manipulations. Fig. 4(a) depicts the initial table. In this table, the index denotes the cluster's identification. Specifically, in order to determine the best threshold, the maximal gray value in the cluster is still kept in the "*Maximal gray level*" field. From the "*Nearest distance*" field (denoting the minimum distance), we merge the two clusters C_4 and C_5. The updated table is illustrated

Index	Index of the nearest cluster	Mean gray level	Maximal gray level	Nearest distance	The number of pixels
0	1	10	10	1500	10
1	2	25	25	278	20
2	3	30	30	234	25
3	2	35	35	234	15
4	5	80	80	94	5
5	4	85	85	94	15
6	5	90	90	150	10

(a)

Index	Index of the nearest cluster	Mean gray level	Maximal gray level	Nearest distance	The number of pixels
0	1	10	10	1500	10
1	2	25	25	278	20
2	3	30	30	234	25
3	2	35	35	234	15
4	6	83.75	85	260	20
6	4	90	90	260	10

(b)

Fig. 4. The array data structure: (a) initial step; and (b) first merging step.

in Fig. 4(b). In fact, using the linked list data structure [16] is another efficient alternate to implement the above manipulations.

According to the above description, the proposed first algorithm for thresholding is listed below:

Algorithm_1. Find k thresholds.

Input: Compact image histogram and the number of thresholds, k.
Output: The determined k thresholds.
/⋆ Let m be the size of the compact image histogram. ⋆/
Construct the initial table (ref. Fig. 4(a)).
Repeat
  Phase 1: Find two nearest clusters C_j and C_k to be merged.
  Phase 2: Merge the selected two clusters; $m \leftarrow m-1$.
  Phase 3: Update the table (ref. Fig. 2(b)).
Until $m = k+1$.
The k thresholds are determined by selecting the maximal gray level in each cluster.

Following the analysis technique used in Ref. [15], we now analyze the complexity required in the proposed first algorithm. Suppose we have had the compact image histogram. It takes $O(m^2)$ time to construct the initial table since for each cluster, computing its own nearest distance must consider all the remaining m clusters.

In phase 1, we want to find the two nearest clusters. This is equal to a minimum–finding problem and it takes $O(m)$ time.

Table 1
Time complexity comparison

	Time complexity
Otsu's algorithm (OT)	$O(m^k)$
Kittler and Illingworth's algorithm (KI)	$O(m^k)$
Kapur *et al.*'s algorithm (KA)	$O(m^k)$
Proposed Algorithm_1 (OURS1)	$O((m-k)m\tau)$

In phase 2, suppose the merged cluster is denoted by C_q and the corresponding nearest pair are C_j and C_k. The size of the merged cluster is set to be $n'_q = n'_j + n'_k$. The mean of the merged cluster is set to be $\bar{X}_q = (n'_j\bar{X}_j + n'_k\bar{X}_k)/n'_q$. The maximal gray level of the merged cluster C_q is set to be $\max\{MG_j, MG_k\}$, where MG_j (MG_k) denotes the "*Maximal gray level*" in C_j (C_k). The above calculations can be performed in $O(1)$ time.

In phase 3, only the clusters connecting C_j and C_k (before merging) and the merged cluster C_q are needed to update their "*Nearest distance*" and "*Index of the nearest cluster*" fields. Except the merged cluster itself, suppose there are $\tau-1$ ($\tau-1 \leqslant m$) clusters connecting C_j and C_k. Among the merged cluster and these $\tau-1$ clusters, for each cluster, it takes $O(m)$ time to update the nearest distance and the corresponding index of the nearest cluster. So, it takes $O(m\tau)$ time in Phase 3.

Totally, there are $(m-k-1)$ iterations to be performed in the proposed first algorithm, Algorithm_1. For each iteration, there are three phases to be performed and each iteration takes $O(m+1+m\tau) = O(m\tau)$ time. Considering the time to construct the initial table, Algorithm_1 takes $O((m-k)m\tau)$ $(=O((m-k)m\tau + m^2))$ time totally for obtaining the k thresholds.

In summary, the time complexity comparison among the previous three mentioned algorithms (named OT, KI, and KA) and our proposed first Algorithm (named OURS1) is listed in Table 1.

From the above time complexity comparison, it reveals that for $k=1$, i.e. the binary segmentation, all the three previous algorithms are faster to our proposed first algorithm, but for $k \geqslant 3$, our proposed first algorithm is the fastest. In fact, for $k=2$, the proposed first algorithm, OURS1, is still the fastest due to the small leading constant factor in the complexity required in our proposed algorithm. In the next section, some experiments will be carried out to confirm the theoretical analysis. Besides the computational advantage, our proposed algorithm, OURS1, has the similar thresholded images as in the previous three algorithms.

For providing the necessary temporary information, such as all the temporary thresholds and the means of the clusters, to the proposed second algorithm, Algorithms_1, i.e. OURS1, is modified as follows, where all the temporary thresholds and means are kept in the arrays *Thresholds*[] and *Cluster_Means*[], respectively.

Modified Algorithm_1. Find *Thresholds*[] and *Cluster_Means*[].

Input: Compact image histogram and the number of thresholds, k.
Output: The array *Thresholds*$[1..\frac{(k+m-1)(m-k)}{2}]$ and array *Cluster_Means*$[1..\frac{(k+m+1)(m-k)}{2}]$.
/⋆ Let m be the size of the compact image histogram. ⋆/
/⋆ Let the array *Thresholds*[] keep all the temporary thresholds. ⋆/
/⋆ Let the array *Cluster_Means*[] keep all the temporary clusters' means. ⋆/
Construct the initial table (ref. Fig. 4(a)).
$j_1 \leftarrow 0$; $j_2 \leftarrow 0$.
Repeat
  $i_1 = j_1 + 1$; $i_2 = j_2 + 1$.
  $j_1 = j_1 + m - 1$; $j_2 = j_2 + m$.
  Thresholds$[i_1 \dots j_1] \leftarrow (m-1)$ maximal gray levels, i.e. thresholds, in the $(m-1)$ clusters.
  Cluster_Means$[i_2 \dots j_2] \leftarrow m$ means in the m clusters.
  Phase 1: Find two nearest clusters C_j and C_k to be merged.
  Phase 2: Merge the selected two clusters; $m \leftarrow m - 1$.
  Phase 3: Update the table (ref. Fig. 2(b)).
Until $m = k + 1$.

In fact, the time complexity of the modified version of Algorithm_1 is the same as that of Algorithm_1, but it only needs $O(m^2 - k^2)$ extra memory.

3.2. The second algorithm satisfying the given PSNR

Given a specified PSNR, following the proposed Algorithm_1, we now present an adaptive thresholding algorithm such that the resulting thresholded image can satisfy the specified PSNR, but the number of determined thresholds is as few as possible. Before presenting the proposed second algorithm, the PSNR is defined by $PSNR = 10 \times \log_{10}[255^2 N / \sum_{i=0}^{N-1} (I_1(i) - I_2(i))^2]$, where N denotes the total pixels in the image; I_1 denotes the original image and I_2 denotes the current thresholded image.

Following the modified version of Algorithm_1, our proposed second thresholding algorithm is listed below.

Algorithm_2. Find k' thresholds such that the thresholded image satisfies the PSNR requirement.

Input: PNSR_threshold, the input image I_1, and the compact image histogram.
Output: k' thresholds.
$k' \leftarrow 1$.
Find the array *Thresholds*[] and array *Cluster_Means*[] by calling the modified Algorithm_1 with $k = k'$.
/⋆ It takes $O((m-k')m\tau + \gamma N)$ time. ⋆/
$i_1 \leftarrow \frac{(k'+m-1)(m-k')}{2}$; $i_2 \leftarrow \frac{(k'+m+1)(m-k')}{2}$.
While (True)
  Replace I_1 by I_2 using k' thresholds in *Thresholds*$[i_1 \dots (i_1 - k' + 1)]$ and using the $k' + 1$ clusters' means in *Cluster_Means*$[i_2 \dots (i_2 - k')]$.
  /⋆ Obtain the thresholded image I_2 takes $O(N)$ time. ⋆/
  $PSNR \leftarrow$ Compute the PSNR between I_1 and I_2.
  /⋆ It takes $O(N)$ time. ⋆/
  If ($PSNR \geqslant PSNR\_Threshold$)
    Break. /⋆ Stop the algorithm. ⋆/
  EndIf
  $k' \leftarrow k' + 1$.
  $i_1 \leftarrow i_1 - k'$ and $i_2 \leftarrow i_2 - (k' + 1)$.
EndWhile
The k' thresholds are obtained from *Thresholds*$[i_1 \dots (i_1 - k' + 1)]$.

According to the complexity analysis described in Algorithm_1, it is not hard to derive that the time complexity of Algorithm_2 is $O((m-k)m\tau + \gamma N)$, where γ is the number of iterations required in Algorithm_2. To the best of our knowledge, this is the first time that given a specified PSNR, such an adaptive algorithm for thresholding is presented. According to the experiments shown in Section 4, a small γ can achieve a high PSNR.

4. Experimental results

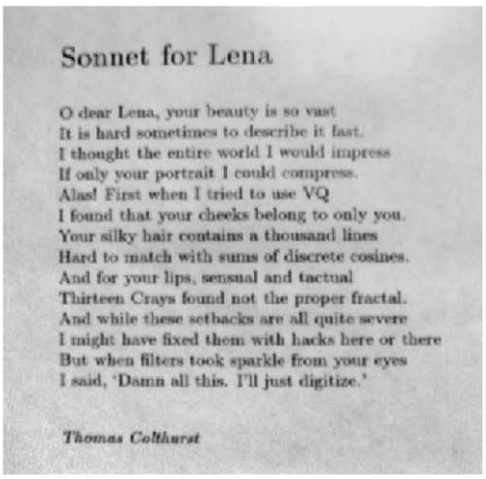
In this section, three types of experiments are carried out to evaluate the performance of the previous three thresholding algorithms and the proposed two algorithms. For convenience, let OURS2 denote our proposed second algorithm. The first experiment is used to evaluate the time performance among OT, KI, KA, and OURS1. The second experiment is used to compare the thresholded images by using OT, KI, KA, and OURS1. The third experiment is used to justify the proposed second algorithm, OURS2. All the related algorithms are implemented using Borland C++ Builder 4.0 language and the Pentium III 600 PC with 128 MB RAM.

4.1. Time comparison among OT, KI, KA, and OURS1

In this experiment, as shown in Figs. 5(a)–(c), three real 512×512 images, namely F16, Lena, and Text, respectively, are used to evaluate the time performance among the mentioned four algorithms. Especially, the constant term in the time complexity for each algorithm will be estimated. That is, the asymptotic time complexity is obtained for each algorithm. In our experiments, the sizes of the compact image histograms are $m = 235$, $m = 215$, and $m = 223$ for F16, Lena, and Text, respectively.

(a)

(b)



(c)

Fig. 5. Original images: (a) F16 and (b) Lena and (c) text.

Table 2
Time comparison for F16 image

	$k=1$	$k=2$	$k=3$	$k=4$
OT	0.0001	0.0216	2.1975	159.2550
KI	0.0003	0.0573	5.8970	425.9250
KA	0.0001	0.0159	1.5310	102.2250
OURS1	0.0015	0.0015	0.0015	0.0015

Table 3
Time comparison for Lena image

	$k=1$	$k=2$	$k=3$	$k=4$
OT	0.0001	0.0180	1.6660	110.3100
KI	0.0003	0.0477	4.4940	296.4250
KA	0.0001	0.0130	1.1140	69.0250
OURS1	0.0010	0.0010	0.0010	0.0010

Table 4
Time comparison for Text image

	$k=1$	$k=2$	$k=3$	$k=4$
OT	0.0001	0.0190	1.8990	127.3477
KI	0.0003	0.0502	5.1050	350.0550
KA	0.0001	0.0137	1.1850	82.1450
OURS1	0.0010	0.0010	0.0010	0.0010

Table 5
Determined thresholds, t_1 and t_2

	t_1, t_2(F16)	t_1, t_2(Lena)	t_1, t_2(Text)
OT	93, 165	92, 150	130, 182
KI	182, 206	72, 119	185, 217
KA	78, 175	96, 163	118, 118
OURS1	118, 177	66, 135	122, 171

Tables 2–4 illustrate the executing time in terms of seconds required in the four algorithms. From the three tables, it is observed that the proposed algorithm OURS1 is faster considerably than OT, KI, and KA when $k \geqslant 2$. For $k \geqslant 2$, the execution time improvement ratio is close to 1 while the thresholded images obtained by using our proposed algorithm OURS1 are encouraging (see Section 4.2) when compared to the previous three algorithms. After calculating the constant term in the time complexity of each algorithm, the detailed average time complexities are $5.2 \times 10^{-8} \times m^k$, $1.38 \times 10^{-7} \times m^k$, $3.2 \times 10^{-8} \times m^k$, and $2.6 \times 10^{-10} \times (m-k)m\tau$, where $\tau = m/2$, for OT, KI, KA, and OURS1, respectively. The experimental results shown in Tables 2–4 also confirm the theoretical analysis described

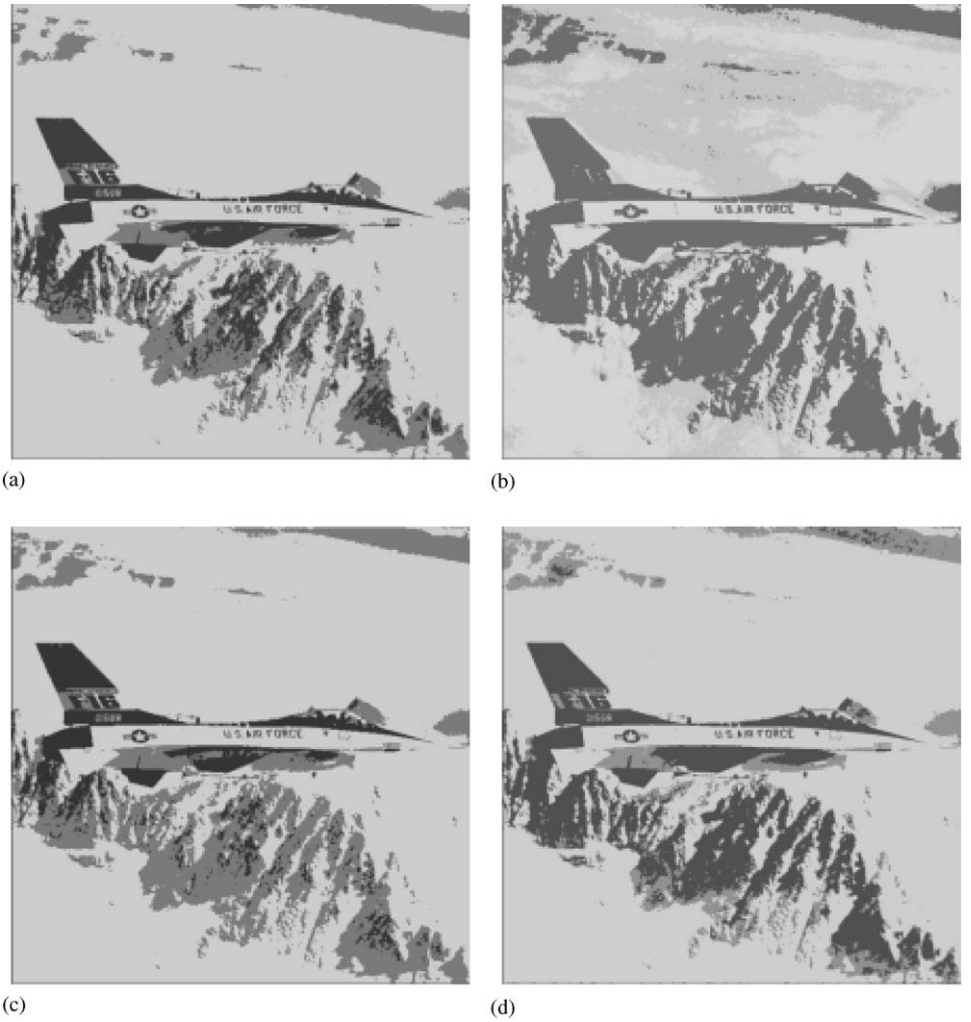


Fig. 6. The thresholded images for F16 with $k = 2$: (a) OT's thresholded image; (b) KI's thresholded image; (c) KA's thresholded image; and (d) OURS1's thresholded image.

Table 6
Feature-preserving comparison of the thresholded image for F16

	OT	KI	KA	OURS1
(1) The entrance with shape □	Fair	Good	Good	Fair
(2) The F-16 mark	Good	Fair	Good	Good
(3) The star signature	Good	Fair	Good	Good
(4) The text "U.S.AIR FORCE"	Good	Good	Good	Good
(5) The belly	Good	Fair	Good	Good
(6) The cloud	Good	Good	Good	Good
(7) The ID number 01568	Fair	Fair	Fair	Good

in Table 1. From Tables 2–4, it is observed that the polynomial time bound required in OURS1 is rather stable for different k while the time bounds required in the previous three algorithms are exponentially proportional to k. However, for $k = 1$, the previous three algorithms are still the fastest.

Fig. 7. The thresholded images for Lena with $k = 2$; (a) OT's thresholded image; (b) KI's thresholded image; (c) KA's thresholded image; and (d) OURS1's thresholded image.

Table 7
Feature-preserving comparison of the thresholded image for Lena

	OT	KI	KA	OURS1
(1) The nose	Good	Fair	Good	Good
(2) The lip	Good	Good	Fair	Good
(3) The cheek	Fair	Good	Fair	Fair
(4) The stumps	Fair	Good	Fair	Good
(5) The shoulder	Good	Fair	Fair	Fair

4.2. Comparison of thresholded images among OT, KI, KA, and OURS1

In the second experiment, we only consider $k = 2$ to compare the thresholded images among the related four algorithms although it is applicable for $k > 2$. Since the criterion used in each algorithm is different, the determined thresholds are somewhat different from each other. Here, the main features of the thresholded images are investigated for the four algorithms.

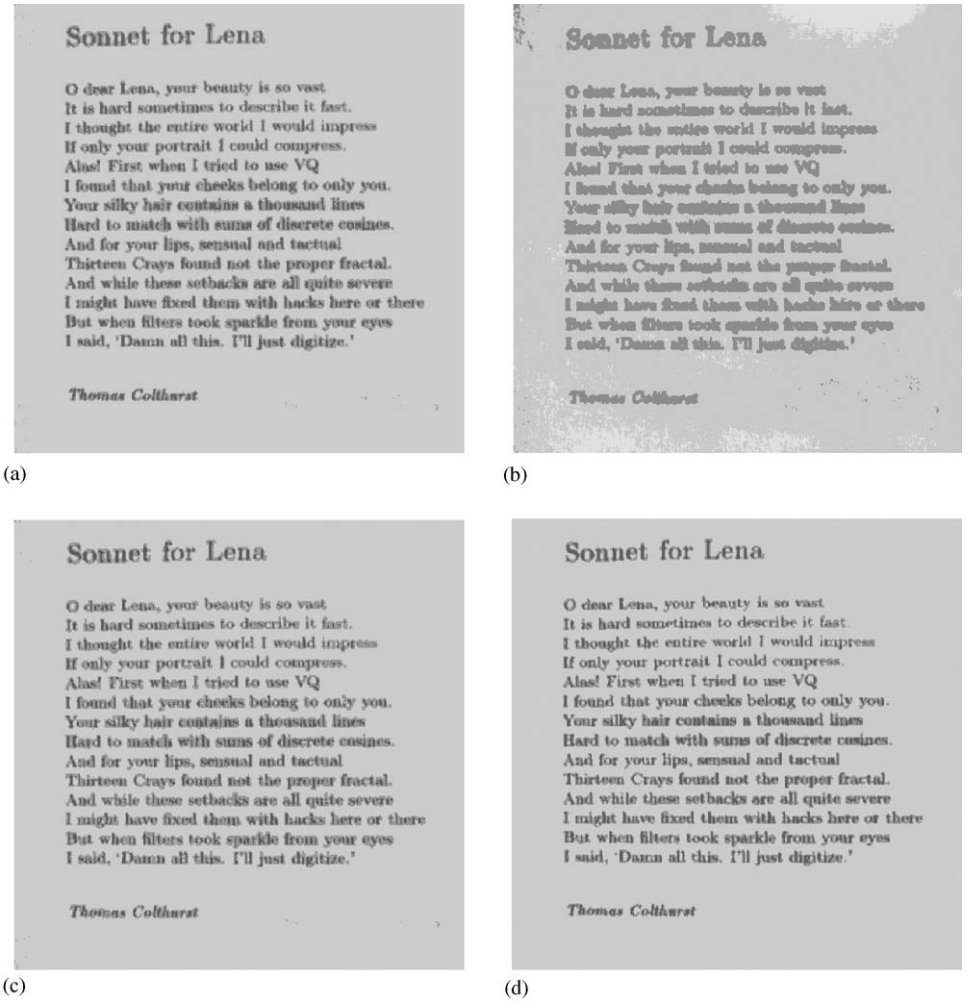


Fig. 8. The thresholded images for Text with $k = 2$; (a) OT's thresholded image; (b) KI's thresholded image; (c) KA's thresholded image; and (d) OURS1's thresholded image.

Table 8
Feature-preserving comparison of the thresholded image for Text

	OT	KI	KA	OURS1
(1) The background	Good	Good	Good	Good
(2) The words' clearness	Good	Fair	Good	Good

Table 5 demonstrates the two determined thresholds, t_1 and t_2, by using the four related algorithms for the same three images as in Section 4.1. From Table 5, it is observed that our proposed algorithm OURS1 has the middle t_1 and t_2 when compared to the previous three algorithms. We now investigate the feature-preserving capability in the resulting thresholded images.

For the thresholded image of F16, we mainly compare seven features: (1) the entrance with shape □, (2) the F–16 mark, (3) the star signature, (4) the text "U.S.AIR FORCE", (5) the belly, (6) the cloud, and (7) the ID number 01568. After comparing the four related thresholded images as shown in Figs. 6(a)–(d) with Fig. 5(a), the above seven features comparison is illustrated in Table 6.

(a) F16. (b) Lena.

Fig. 9. The thresholded image after applying OURS2.

Here, for each feature, we use "fair" or "good" to grade its feature-preserving capability. From Table 6, for F16, it is observed that the thresholded images of OURS1, OT, and KA have the relatively better quality exhibition.

For the thresholded image of Lena, we mainly compare five features: (1) the nose, (2) the lip, (3) the cheek, (4) the stumps, and (5) the shoulder. After comparing the four related thresholded images as shown in Figs. 7(a)–(d) with Fig. 5(b), the above five features comparison is illustrated in Table 7. From Table 7, for Lena, it is observed that the thresholded images of OURS1, OT, and KI have the relatively better quality exhibition.

For the thresholded image of Text, we compare two features: (1) the background and (2) the words' clearness. After comparing the four related thresholded images as shown in Figs. 8(a)–(d), with Fig. 5(c), the above two features comparison is illustrated in Table 8. From Table 8, for Text, it is observed that the thresholded images of OURS1, OT, and KA have the relatively better quality exhibition.

Combining the above feature-preserving comparison, it comes to a conclusion that the thresholded images of our proposed algorithm OURS1 has an encouraging feature-preserving capability.

4.3. Performance of the proposed second algorithm

For human visual system, the thresholded image has PSNR = 30, the image quality is rather satisfactory. Our proposed second algorithm OURS2 can determine the least number of thresholds satisfying the given PSNR quality. For F16, when the PSNR is equal to 30.15, the determined least number of thresholds is $k = 6$ using OURS2. For Lena, when the PSNR is equal to 30.32, the determined least number of thresholds is $k = 6$ too. It implies that OURS2 has the compression effect. From Figs. 9(a) and (b), the thresholded two images are encouraging.

5. Conclusions

Thresholding is a very important operator in image preprocessing. This theme has presented a new PNN-based thresholding algorithm, OURS1, which is quite different from the previous three algorithms [1–3]. For $k \geqslant 2$, i.e. finding at least two thresholds, our proposed OURS1 has a satisfactory feature-preserving capability, but is the fastest when compared to the previous three algorithms. Experimental results confirm the theoretical analysis. According to the modified version of OURS1, we also present an adaptive thresholding algorithm, OURS2, to determine the thresholds as few as possible to satisfy the specified PSNR. Experimental results reveal that the thresholded images are encouraging.

References

[1] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Sys. Man. Cybernet. 9 (1) (1979) 62–66.

[2] J. Kittler, J. Illingworth, Minimum error thresholding, Pattern Recognition 19 (1) (1986) 41–47.

[3] 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 (3) (1985) 273–285.

[4] A.S. Abutaleb, Automatic thresholding of gray-level pictures using two-dimensional entropy, Comput. Vision Graphics Image Process. 47 (1) (1989) 22–32.

[5] A.D. Brink, Thresholding of digital images using two-dimensional entropies, Pattern Recognition 25 (8) (1992) 803–808.
[6] W.T. Chen, C.H. Wen, C.W. Yang, A fast two-dimensional entropic thresholding algorithm, Pattern Recognition 27 (7) (1994) 885–893.
[7] L. Li, J. Gong, W. Chen, Gray-level image thresholding based on Fisher linear projection of two-dimensional histogram, Pattern Recognition 30 (5) (1997) 743–749.
[8] J.Z. Liu, W.Q. Li, Y. Tian, Automatic thresholding of gray-level pictures using two-dimension Otsu method, in: 1991 International Conference on Circuits and Systems, China (Cat. No.91TH0387–1), IEEE Press, New York, Vol. 1, 1991, pp. 325–327.
[9] N.R. Pal, S.K. Pal, Entropic thresholding, Signal Process. 16 (2) (1989) 97–108.
[10] X.J. Wu, Y.J. Zhang, L.Z. Xia, A fast recurring two-dimensional entropic thresholding algorithm, Pattern Recognition 32 (12) (1999) 2055–2061.
[11] C.W. Yang, P.C. Chung, C.I. Chang, Hierarchical fast two-dimensional entropic thresholding algorithm using a histogram pyramid, Opt. Eng. 35 (11) (1996) 3227–3241.
[12] W.H. Equitz, Fast algorithms for vector quantization picture coding, Master's Thesis, Massachusetts Institute of Technology, 1984.
[13] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Preutice-Hall, New York, 1988.
[14] W.H. Equitz, A new vector quantization clustering algorithm, IEEE Trans. Acoust. Speech Signal Process. 37 (10) (1989) 1568–1575.
[15] P. Fränti, T. Kaukoranta, D.F. Shen, K.S. Chang, Fast implementation of the exact PNN algorithm, IEEE Trans. Image Process. 9 (5) (2000) 773–777.
[16] E. Horowitz, S. Sahni, S.A. Freed, Fundamentals of Data Structures in C, Chapter 4: Lists, Computer Science Press, New York, 1993.

About the Authors—KUO-LIANG CHUNG received the B.S., M.S., and Ph.D. degrees in computer science and information engineering from National Taiwan University in 1982, 1984, and 1990, respectively. From 1984 to 1986, he was a soldier. From 1986 to 1987, he was a research assistant in the institute of Information Science, Academic Sinica. He is a professor in the department of Computer science and Information Engineering at National Taiwan University of Science and Technology since 1995. Prof. Chung received the Distinguished Professor Award from the Chinese Institute of Engineers in May 2001. He is also an IEEE senior member. His research interests include image compression, image processing, video compression, coding theory, and algorithms.

About the Authors—WAN-YUE CHEN received the B.S. and M.S. degrees in information management from National Taiwan University of Science and Technology in 1999 and 2001, respectively. Her research interests include image compression, image processing, and algorithms.