Applied Mathematics and Computation 212 (2009) 396–408

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Fast incremental algorithm for speeding up the computation of binarization ☆

Kuo-Liang Chung *, Chia-Lun Tsai

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

ARTICLE INFO

Keywords:
Binarization
Heap
Incremental algorithm
Kittler and Illingworth method
Otsu method
Quantization
Within-variance

ABSTRACT

Binarization is an important basic operation in image processing community. Based on the thresholded value, the gray image can be segmented into a binary image, usually consisting of background and foreground. Given the histogram of input gray image, based on minimizing the within-variance (or maximizing the between-variance), the Otsu method can obtain a satisfactory binary image. In this paper, we first transfer the within-variance criterion into a new mathematical formulation, which is very suitable to be implemented in a fast incremental way, and it leads to the same thresholded value. Following our proposed incremental computation scheme, an efficient heap- and quantization-based (HQ-based) data structure is presented to realize its implementation. Under eight real gray images, experimental results show that our proposed HQ-based incremental algorithm for binarization has 36% execution-time improvement ratio in average when compared to the Otsu method. Besides this significant speedup, our proposed HQ-based incremental algorithm can also be applied to speed up the Kittler and Illingworth method for binarization.

1. Introduction

Binarization is a very important preprocessing operation in image processing and computer vision [1–3]. Based on the thresholded value, the gray image can be segmented into a binary image, containing background and foreground. In addition, combining the thresholded value and the Sobel edge detector, the gray image can be transferred to the edge map which has many applications, such as the shape detection [4,5]. How to determine a suitable thresholded value to partition the input gray image into two parts is the key in binarization and has a long history. Usually binarization is also called the threshold selection problem.

In the past years, many efficient threshold selection methods and their applications [6–16] have been developed. Among these developed threshold selection methods, Otsu's 1979 result is a very important pioneering work, which will be surveyed in Section 2, and it is sometimes used as the kernel in the other methods and applications. The motivations of this paper are twofold: (1) present a novel computational platform to speed up the Otsu method significantly and obtain the same thresholded value and (2) apply our proposed computational scheme to speed up the other binarization methods, such as the Kittler and Illingworth method [7] which is based on the mixture of normal distributions.

In this paper, we first transfer the within-variance criterion used in the Otsu method into a new mathematical formulation and this new formula is very suitable to be implemented in a fast incremental way, and it leads to the same thresholded

☆ Supported by National Science Council of ROC under contracts NSC96-2221-E-011-026 and NSC98-2221-E-011-128.
* Corresponding author.
E-mail address: K.L.Chung@mail.ntust.edu.tw (K.-L. Chung).

0096-3003/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2009.02.061

value. Following our proposed novel incremental computation scheme, an efficient heap- and quantization-based (HQ-based) data structure is presented to realize its implementation. Under eight real gray images, experimental results show that our proposed HQ-based incremental algorithm for binarization has 36% execution-time improvement ratio in average when compared to the Otsu method. Besides this significant speedup, our proposed HQ-based incremental algorithm can also be applied to speedup some other binarization methods such as the Kittler and Illingworth method. Further, experimental results also show that our proposed binarization algorithm is much faster than the previous lookup table-based approach by Liao et al. [17] for binarizing gray images.

The rest of this paper is organized as follows: Section 2 first surveys the Otsu method, then presents our proposed HQ-based incremental algorithm for binarization. Section 3 demonstrates the experimental results. Some concluding remarks are addressed in Section 4.

2. Fast binarization of gray images

In this section, there are three subsections. In Section 2.1, the Otsu method is surveyed. In Section 2.2, we transfer the within-variance criterion used in the Otsu method into a new mathematical formulation, which is very suitable to be implemented in a fast incremental way. Further, an efficient HQ-based data structure is presented to realize its implementation. In Section 2.3, our proposed whole HQ-based incremental algorithm for binarization is presented.

2.1. The Otsu method

In this subsection, the Otsu method for binarizing gray images is surveyed. Let N denote the number of pixels in the input gray image and the gray level is ranged from 0 to $L-1$. For the input gray image I, the number of pixels with gray level i is denoted by n_i and the probability of gray level i is defined by $P_i = \frac{n_i}{N}$.

For binarizing the input image I, suppose t is a possible threshold candidate, then the image I can be divided into two classes/parts, namely, $C_1 = \{0, 1, \ldots, t\}$ and $C_2 = \{t+1, t+2, \ldots, L-1\}$. Before introducing the Otsu method, the following useful notations are defined first. The ratios of the first class C_1 and the second class C_2 over the whole image I are given by

$$\omega_1(t) = \sum_{i=0}^{t} P_i \tag{1}$$

and

$$\omega_2(t) = \sum_{i=t+1}^{L-1} P_i, \tag{2}$$

respectively, where

$$\omega_1(t) + \omega_2(t) = 1. \tag{3}$$

By Eqs. (1) and (2), the two means of classes C_1 and C_2 are defined by

$$\mu_1(t) = \sum_{i=0}^{t} iP_i/\omega_1(t) \tag{4}$$

and

$$\mu_2(t) = \sum_{i=t+1}^{L-1} iP_i/\omega_2(t), \tag{5}$$

respectively. By Eqs. (1)–(5), the two variances of classes C_1 and C_2 are defined by

$$\sigma_1^2(t) = \sum_{i=0}^{t} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)} \tag{6}$$

and

$$\sigma_2^2(t) = \sum_{i=t+1}^{L-1} (i-\mu_2(t))^2 \frac{P_i}{\omega_2(t)}, \tag{7}$$

respectively.

By Eqs. (1) and (2) and Eqs. (6) and (7), the within-variance of C_1 and C_2 is defined by

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t). \tag{8}$$

For two classes C_1 and C_2, it is known that $\sigma_I^2 = \sigma_B^2(t) + \sigma_w^2(t)$ where σ_I^2 denotes the variance of the whole gray image I; $\sigma_B^2(t)$ denotes the between-variance of C_1 and C_2. According to the Otsu method, for binarizing the gray image I, it wants to choose a gray level t to minimize the value of $\sigma_w^2(t)$, i.e. maximizing the value of $\sigma_B^2(t)$. We adopt the minimization of $\sigma_w^2(t)$ as the criterion in our proposed algorithm to binarize the gray image I. On the other hand, for binarizing the gray image I, it wants to find the thresholded value t^* satisfying

$$t^* = \arg\min_{0 \leqslant t < L} \{\sigma_w^2(t)\}. \tag{9}$$

According to the determined threshold t^*, the gray image I can be partitioned into two satisfactory classes C_1 and C_2.

2.2. The proposed incremental heap- and quantization(HQ)-based computation scheme

In the Otsu method, for each t, $0 \leqslant t < L$, it must calculate the value of $\sigma_w^2(t)$. After calculating all $\sigma_w^2(t)$'s for $0 \leqslant t < L$, the minimal $\sigma_w^2(t)$, say $\sigma_w^2(t^*)$, is selected from these L values and t^* is the so called determined threshold. By Eq. (8), for each t, the computation effort for calculating $\sigma_w^2(t)$ is bounded by cL where c is constant. According to the time complexity notation [18], let $f(t)$ be the time complexity for t to calculate $\sigma_w^2(t)$, then we have $f(t) \leqslant cL$, i.e. $f(t) = O(L)$. For $0 \leqslant t < L$, it takes $O(L^2)$ time to calculate all $\sigma_w^2(t)$'s.

In our proposed HQ-based computation scheme, initially, it takes $O(L)$ time to perform the prefix computation to obtain $\omega_1(0), \omega_1(1), \ldots$, and $\omega_1(255)$. From Eq. (3), it can also take $O(L)$ time to obtain $\omega_2(t)$ $(= 1 - \omega_1(t))$ for $0 \leqslant t \leqslant 255$. In the initial iteration, it does not need to calculate the final value of $\sigma_w^2(t)$ for $0 \leqslant t \leqslant 255$, but we just calculate the value of $\sigma_w^2(t)$ in an incremental way and the temporary value of $\sigma_w^2(t)$ calculated in the initial iteration is denoted by

$$\sigma_w^2(t,0) = \omega_1(t)\sigma_1^2(t,0) + \omega_2(t)\sigma_2^2(t,0), \tag{10}$$

where $\sigma_1^2(t,0)$ and $\sigma_2^2(t,0)$ are defined by

$$\sigma_1^2(t,0) = \begin{cases} \sum_{i=0}^{v} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)}, & \text{if } t > v, \\ \sum_{i=0}^{t} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)}, & \text{otherwise}, \end{cases} \tag{11}$$

$$\sigma_2^2(t,0) = \begin{cases} 0, & \text{if } t > v, \\ \sum_{i=t+1}^{v} (i-\mu_2(t))^2 \frac{P_i}{\omega_2(t)}, & \text{otherwise}, \end{cases} \tag{12}$$

where $(v+1)$ denotes the number of data that should be included in each incremental computation iteration. Empirically, we set $(v+1) = 32$.

After computing Eq. (10), in the same iteration, we quantize these 256 $\sigma_w^2(t,0)$'s for $0 \leqslant t \leqslant 255$ into 32 groups, namely group_0=$[\sigma_w^2(0,0), \sigma_w^2(1,0), \sigma_w^2(2,0), \ldots, \sigma_w^2(7,0)], \ldots$, and group_31=$[\sigma_w^2(248,0), \sigma_w^2(249,0), \ldots, \sigma_w^2(255,0)]$, where each group contains eight temporary within-variances. For the pth group, $0 \leqslant p \leqslant 31$, we compute its own minimal temporary within-variance which is defined by

$$\sigma_w^{2^*}(t_p,0,p) = \min_{8p \leqslant i < 8(p+1)} \{\sigma_w^2(i,0)\}, \tag{13}$$

where $t_p = \arg\min_{8p \leqslant i < 8(p+1)} \{\sigma_w^2(i,0)\}$.

For $0 \leqslant p \leqslant 31$, after computing Eq. (13), we have 32 minimal temporary within-variances for these 32 groups and they are denoted by the set $\{\sigma_w^{2^*}(t_p,0,p) | 0 \leqslant p \leqslant 31\}$. We now arrange these 32 temporary within-variances to construct a binary tree such that the root contains $\sigma_w^{2^*}(t_0,0,0)$; the root's left (right) son contains $\sigma_w^{2^*}(t_1,0,1)$ $(\sigma_w^{2^*}(t_2,0,2))$, and so on. As shown in Fig. 1, the constructed binary tree is used as the initial heap which will be used in the further iteration. Note that besides containing the minimal temporary within-variance of group_i, $\sigma_w^{2^*}(t_i,0,i)$, each node in Fig. 1 also contains eight temporary within-variances, $\sigma_w^2(8i,0), \sigma_w^2(8i+1,0), \ldots$, and $\sigma_w^2(8(i+1)-1,0)$.

After determining the minimal temporary within-variance of each group, we select the group, say p'th group, with the minimum temporary within-variance from the initial heap by running the minimum-finding operation [18] in the initial iteration on Fig. 1 and we have

$$\sigma_w^{2^*}(t_{p'},0,p') = \min_{0 \leqslant p < 32} \{\sigma_w^{2^*}(t_p,0,p)\}, \tag{14}$$

where $t'_p = \arg\min_{0 \leqslant p < 32} \{\sigma_w^{2^*}(t_p,0,p)\}$. In fact, the minimum-finding operation is the heapifying operation so that the heap condition is satisfied, and thus for every node i other than the root node has the property that the minimal temporary within-variance of the parent node is less than or equal to that of the two sons. Without losing the generality, we suppose $p' = 9$. For group_9, its current own eight temporary within-variances are denoted by $[\sigma_w^2(72,0), \sigma_w^2(73,0), \ldots, \sigma_w^2(79,0)]$. After running the minimum-finding operation on Fig. 1, group_9 is moved to the root of heap and the adjusted heap is shown in Fig. 2.

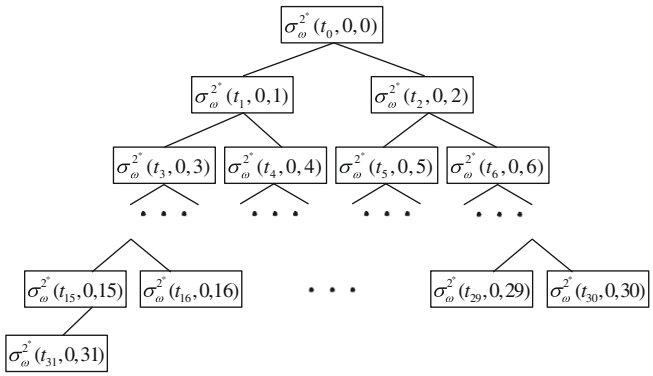

Fig. 1. Initial heap.

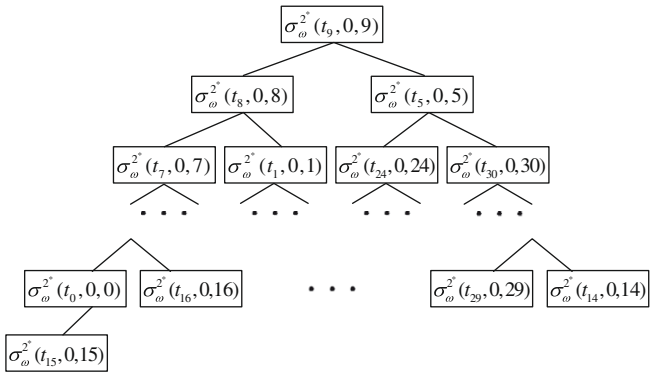

Fig. 2. Adjusted heap after performing minimum-finding operation in the initial iteration.

In the first iteration, group_9 has eight entries for $72 \leqslant t \leqslant 79$ and each entry considers 32 new data and performs

$$\sigma_1^2(t,1) = \sum_{i=32}^{63} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)}, \tag{15}$$

$$\sigma_2^2(t,1) = 0. \tag{16}$$

Since in each iteration, each entry in the minimal group considers 32 new data, there are eight (=256/32) iterations in total. From $\sigma_w^2(t,1) = \omega_1(t)\sigma_1^2(t,1) + \omega_2(t)\sigma_2^2(t,1)$, the set of eight within-variances in group_9 is now changed to $\left[\sigma_w^2(72,1), \sigma_w^2(73,1), \sigma_w^2(74,1), \ldots, \sigma_w^2(79,1)\right]$ while the remaining 31 groups don't update their own temporary within-variances anymore. As shown in Fig. 3, after performing the minimum-finding operation in the first iteration, the root of the adjusted heap contains the minimal group, group_9.

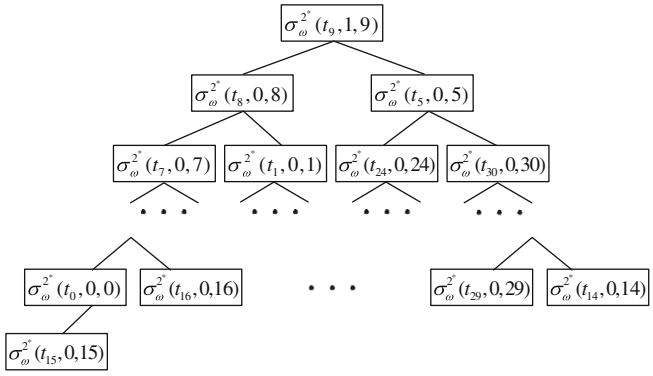

Fig. 3. Root node update of Fig. 2.

In the jth iteration, $2 \leqslant j \leqslant 7$, by the same argument as in the first iteration, suppose the minimal group is group_$\bar{p}$, i.e. group_$\bar{p}$ has minimum within-variance at present, then each entry, $8\bar{p} \leqslant t < 8(\bar{p}+1)$, in group_$\bar{p}$ performs the following operations:

$$\sigma_w^2(t,j) = \sigma_1^2(t,j)\omega_1(t) + \sigma_2^2(t,j)\omega_2(t), \tag{17}$$

where

$$\sigma_1^2(t,j) = \begin{cases} 0, & \text{if } t < 32j, \\ \sum_{i=32j}^{32(j+1)-1} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)}, & \text{if } t \geqslant (32(j+1)-1), \\ \sum_{i=32j}^{t} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)}, & \text{otherwise}, \end{cases}$$

$$\sigma_2^2(t,j) = \begin{cases} 0, & \text{if } t \geqslant (32(j+1)-1), \\ \sum_{i=32k+t+1}^{32(j+1)+t} (i-\mu_2(t))^2 \frac{P_i}{\omega_2(t)}, & \text{if } 32(j+1)+t \leqslant 255, \\ \sum_{i=32k+t+1}^{255} (i-\mu_2(t))^2 \frac{P_i}{\omega_2(t)}, & \text{otherwise}. \end{cases}$$

Before presenting our proposed HQ-based computation scheme to speed up the Otsu method, the following notations are given. Let $P(u)$ denote the accumulated probability whose random variable i is ranged from 0 to u for $0 \leqslant u < L$ and $P(u)$ is defined by

$$P(u) = \sum_{i=0}^{u} P_i.. \tag{18}$$

Let $M(u)$ denote the first-order moment and $M(u)$ is defined by

$$M(u) = \sum_{i=0}^{u} iP_i. \tag{19}$$

Eqs. (18) and (19) can be rewritten as the following recursive forms:

$$P(u+1) = P(u) + P_{u+1}, \tag{20}$$

and

$$M(u+1) = M(u) + (u+1)P_{u+1}, \tag{21}$$

where the two boundary conditions are set to $P(-1) = 0$ and $M(-1) = 0$.

According to Eqs. (1), (3), and (18), the ratio of class C_k is said to $\omega_k(t)$, $1 \leqslant k \leqslant 2$, and can be rewritten as

$$\omega_1(t) = P(t), \tag{22}$$

and

$$\omega_2(t) = 1 - P(t). \tag{23}$$

By Eqs. (4), (5), (19), (21), (22) and (23), $\mu_k(t)$, $1 \leqslant k \leqslant 2$, is rewritten as

$$\mu_1(t) = \frac{M(t)}{P(t)}, \tag{24}$$

and

$$\mu_2(t) = \frac{M(L-1) - M(t)}{1 - P(t)}. \tag{25}$$

It is known $\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$. Let the symbol j denote the order of the iteration in our proposed HQ-based incremental approach for binarization. For example, when $j = 2$, it means that the order of the iteration is the second iteration. Let the symbol k denote the number of new data considered in each entry of the group. For example, in the previous description, k is set to 32 empirically since the quantization level is 32. The number of groups in our approach is set to $\lceil L/g \rceil$ where the group size g denotes the number of entries in the group, thus $\sigma_1^2(t)$ in Eq. (6) is rewritten as

$$\sigma_1^2(t) = \sum_{j=0}^{s_1} \sigma_1^2(t,j), \tag{26}$$

where $s_1 = \lceil (t+1)/k \rceil - 1$, $\sigma_1^2(t,j) = \sum_{i=jk}^{\min(d_1,t)} (i-\mu_1(t))^2 \frac{P_i}{\omega_1(t)}$, and $d_1 = (j+1)k - 1$. Similarly, $\sigma_2^2(t)$ in Eq. (7) is rewritten as

$$\sigma_2^2(t) = \sum_{j=0}^{s_2} \sigma_2^2(t,j), \tag{27}$$

where $s_2 = \lceil (L-t-1)/k \rceil - 1$, $\sigma_2^2(t,j) = \sum_{i=jk+t+1}^{\min(d_2,L-1)} (i-\mu_2(t))^2 \frac{P_i}{\omega_2(t)}$, and $d_2 = (j+1)k + t$. Combining Eqs. (26) and (27), we have

$$\sigma_w^2(t,j) = \sigma_1^2(t,j)\omega_1(t) + \sigma_2^2(t,j)\omega_1(t). \tag{28}$$

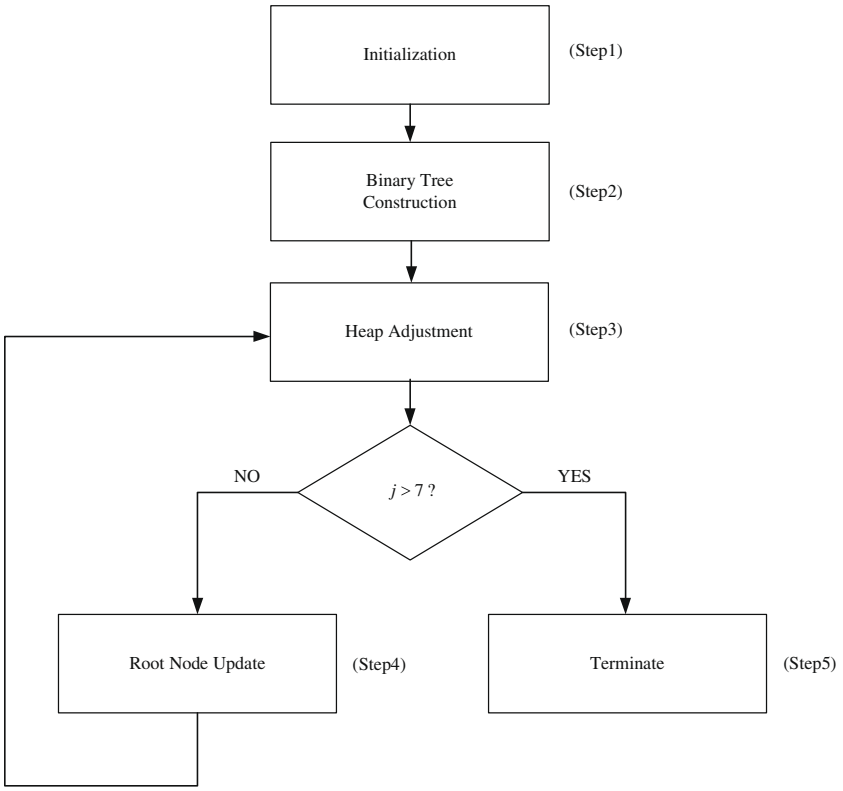

Fig. 4. Flowchart of our proposed HQ-based incremental algorithm.

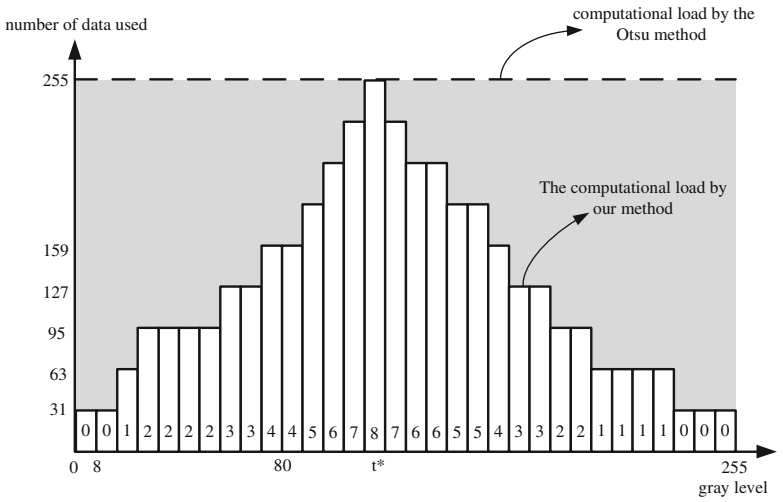

Fig. 5. The depiction of computation-saving advantage of our proposed algorithm.

After computing Eq. (28), we quantize these L $\sigma_w^2(t,j)$'s, $0 \leqslant t < L$, into $\lceil L/g \rceil$ groups, where each group contains g temporary within-variances. For the pth group, $0 \leqslant p < \lceil L/g \rceil$, we compute its own minimum by

$$\sigma_w^{2^*}(t_p,j,p) = \min_{gp \leqslant i < g(p+1)} \{\sigma_w^2(i,j)\}, \tag{29}$$

where $t_p = \arg\min_{gp \leqslant i < g(p+1)} \{\sigma_w^2(i,j)\}$. After determining the minimal group in the jth iteration and performing the minimum-finding operation on the heap, the root node of the heap is changed and denoted by

$$\sigma_w^{2^*}(t_{p'},j,p') = \min_{0 \leqslant p < \lceil L/g \rceil} \{\sigma_w^{2^*}(t_p,j,p)\}, \tag{30}$$

where $t'_p = \arg\min_{0 \leqslant p < \lceil L/g \rceil} \{\sigma_w^{2^*}(t_p,j,p)\}$.

After running the seventh iteration and performing the minimum-finding operation, if the same minimal group is retained in the root node of the heap, we can find this group's minimal temporary within-variance from eight temporary

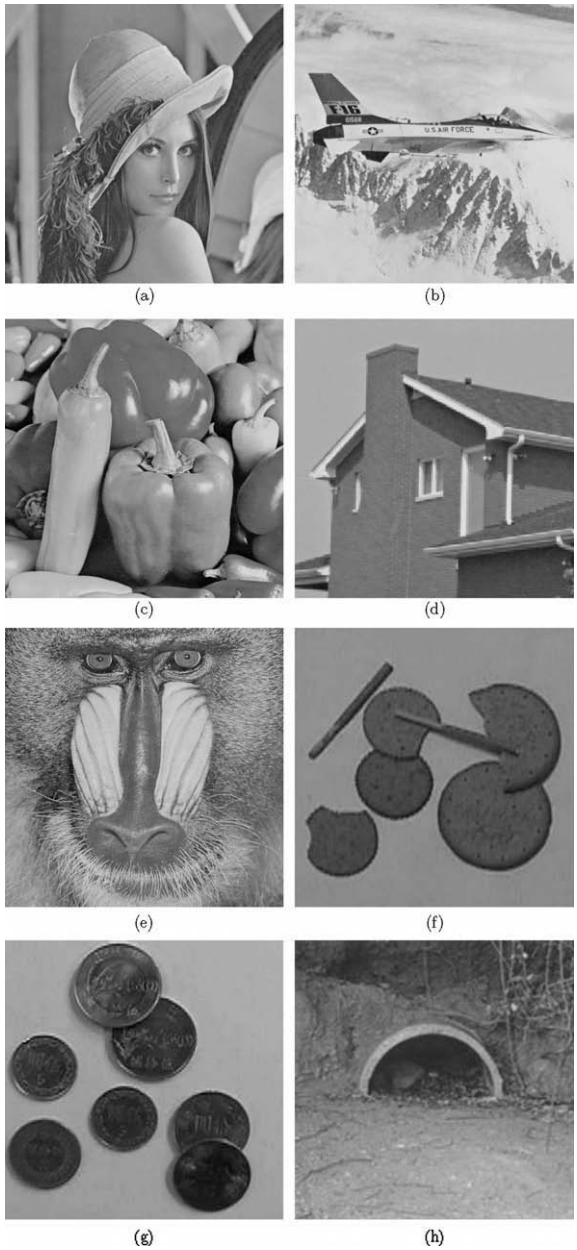

Fig. 6. The eight testing gray images. (a) Lena; (b) F16 Jet; (c) Peppers; (d) House; (e) Baboon; (f) Cakes; (g) Coins and (h) Culvert.

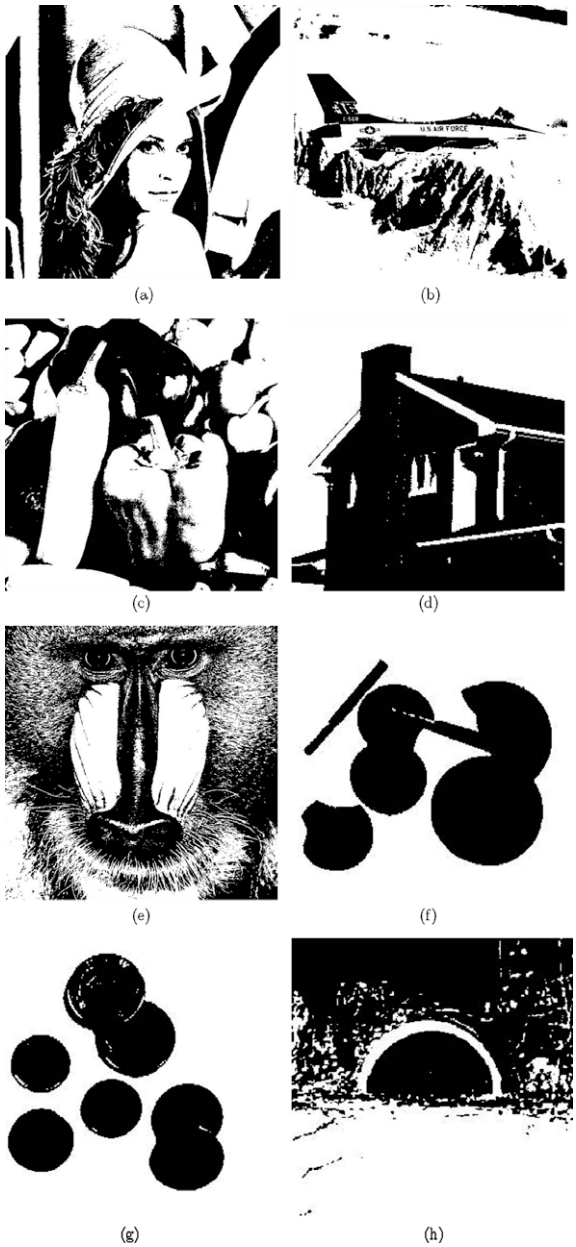

Fig. 7. The eight binarized images of Fig. 6. (a) The binarized Lena; (b) the binarized F16 Jet; (c) the binarized Peppers; (d) the binarized House; (e) the binarized Baboon; (f) the binarized Cakes; (g) the binarized Coins and (h) the binarized Culvert.

Table 1
The execution-time performance comparison between the Otsu method and our proposed method by running each one 1000 times.

	Otsu method (ms)	Proposed method (ms)	Improvement ratio (%)
Lena	2422	1562	35
F16 Jet	2406	1547	33
Peppers	2422	1547	36
House	2437	1593	34
Baboon	2422	1640	32
Cakes	2281	1281	43
Coins	2391	1422	40
Culvert	2328	1531	34
Average	2367	1509	36

within-variances and the found minimal temporary within-variance is the finally determined optimal threshold which is exactly the same as the threshold determined by the Otsu method.

2.3. The proposed HQ-based incremental algorithm

According to the above description, our proposed HQ-based incremental algorithm for binarizing gray images is presented in this subsection. For exposition, for a gray image with 256 levels, assume there are 32 groups and each group has eight entries. Totally there are eight iterations to be performed. Our proposed HQ-based algorithm consists of the following eight steps:

Input: A gray image with 256 levels.
Output: The optimal threshold.
Step 1 (*Initialization*). By Eq. (10), we compute the initial temporary within-variances $\sigma_2^2(t,0)$'s for all $0 \leqslant t \leqslant 255$. Then we quantize these 256 $\sigma_w^2(t,0)$'s for $0 \leqslant t \leqslant 255$ into 32 groups, namely group_0=$[\sigma_w^2(0,0), \sigma_w^2(1,0), \sigma_w^2(2,0), \ldots, \sigma_w^2(7,0)], \ldots$, and group_31=$[\sigma_w^2(248,0), \sigma_w^2(249,0), \ldots, \sigma_w^2(255,0)]$, where each group contains eight temporary within-variances. For the pth group, $0 \leqslant p \leqslant 31$, by Eq. (13), we compute its own minimal temporary within-variance and these 32 minimal temporary within-variances are denoted by $\{\sigma_w^{2^*}(t_p,0,p) | 0 \leqslant p \leqslant 31\}$.
Step 2 (*Binary Tree Construction*). We organize these obtained 32 minimal temporary within-variances obtained in Step 1 as a binary tree such that the root node contains $\sigma_w^{2^*}(t_0,0,0)$; the root's left (right) son contains $\sigma_w^{2^*}(t_1,0,1)$ $(\sigma_w^{2^*}(t_2,0,2))$, and so on.
Step 3 (*Heap Adjustment*). We perform the minimum-finding operation on the tree. After heapifying the tree structure, the heap condition is satisfied and the root node saves the minimal group. Consequently, the smallest group is located at the root node of the heap.
Step 4 (*Root Node Update*). If the iteration number j is not greater than 7, for the current minimal group, say group_$\bar{p}$, its eight temporary within-variances $\sigma_w^2(t,j)$'s, $8\bar{p} \leqslant t < 8(\bar{p}+1)$, will be updated by Eq. (28). Perform the assignment $j = j + 1$ and then go to Step 3; otherwise, go to Step 5.

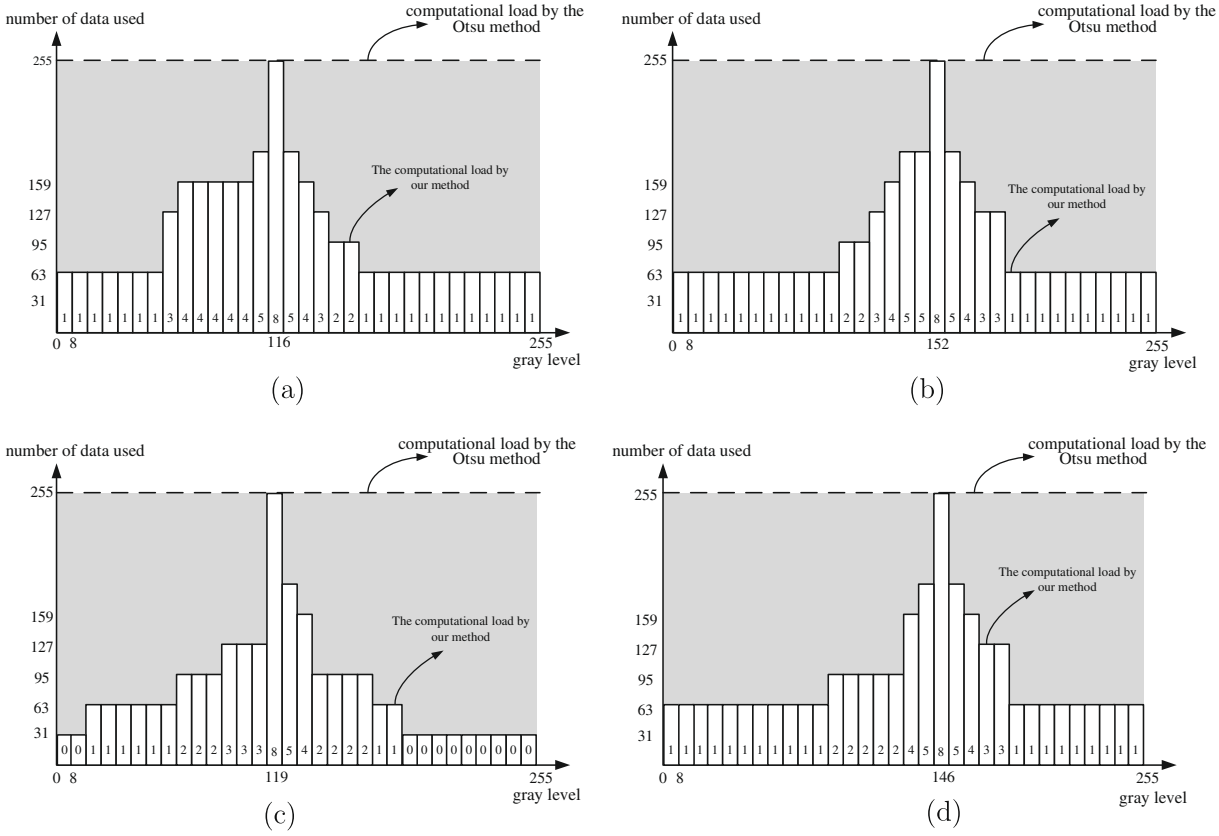

Fig. 8. The illustration of computation-saving advantage of our proposed algorithm. (a) Lena; (b) F16 Jet; (c) Peppers; (d) House; (e) Baboon; (f) Cakes; (g) Coins and (h) Culvert.

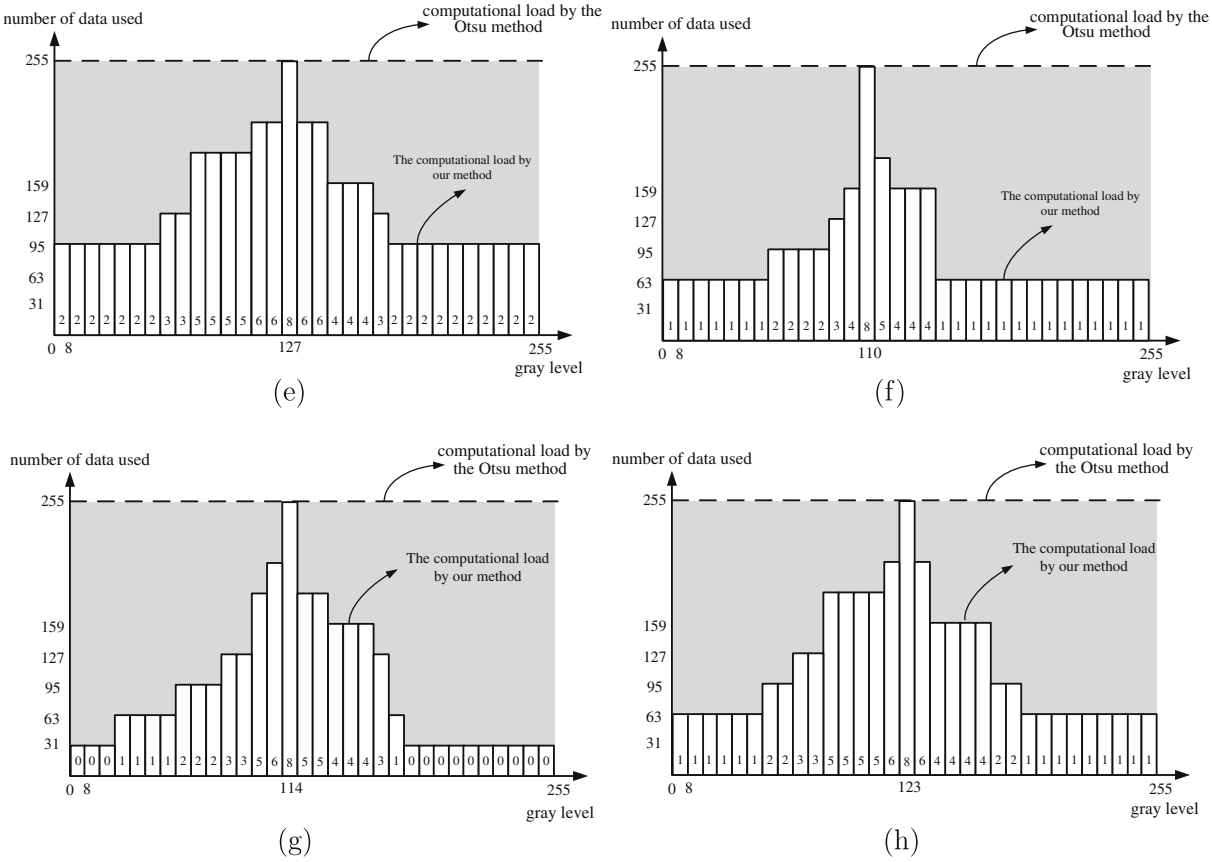

Fig. 8 (*continued*)

Table 2
The execution-time performance comparison between Liao et al.'s method and our proposed method by running each one 1000 times.

	Liao et al.'s method (ms)	Proposed method (ms)	Improvement ratio (%)
Lena	52,188	1562	96
F16 Jet	51,906	1547	96
Peppers	52,078	1547	96
House	52,219	1593	96
Baboon	52,718	1640	96
Cakes	42,281	1297	96
Coins	44,125	1422	96
Culvert	43,563	1500	96
Average	44,297	1531	96

Table 3
The execution-time performance comparison between Kittler and Illingworth's method and our proposed method by running each one 1000 times.

	Kittler et al.'s method (ms)	Proposed method (ms)	Improvement ratio (%)
Lena	14,297	12,828	10
F16 Jet	14,344	12,750	11
Peppers	14,203	12,204	14
House	14,938	11,813	20
Baboon	15,218	12,046	20
Cakes	13,828	11,047	20
Coins	13,453	10,890	19
Culvert	13,531	11,109	17
Average	13,839	11,556	16

Step 5 (*Termination*). Find the smallest within-variance of the eight within-variances in the minimal group and the threshold is thus determined.

Fig. 4 illustrates the flowchart of our proposed HQ-based incremental algorithm. For one gray image with 256 gray levels, assume the symbol t^* is the determined threshold. In the Otsu method, each possible threshold candidate takes the similar computational load for computing Eq. (8). However, all possible threshold candidates considered in our proposed HQ-based incremental algorithm may take different computational loads and for each possible threshold candidate, its own computational load is less than or equal to that of the corresponding possible threshold candidate in the Otsu method. In Fig. 5, the horizontal axis denotes the gray level; the vertical axis denotes the number of data used; the index inside the rectangular bar denotes the total number of iterations needed by the corresponding group. Empirically, in the initial iteration, we compute Eq. (10) and quantize these 256 $\sigma_w^2(t,0)$'s, $0 \leqslant t \leqslant 255$, into 32 groups, namely $\left[\sigma_w^2(0,0), \sigma_w^2(1,0), \sigma_w^2(2,0), \ldots, \sigma_w^2(7,0)\right], \ldots$, and $\left[\sigma_w^2(248,0), \sigma_w^2(249,0), \ldots, \sigma_w^2(255,0)\right]$, where each group contains eight temporary within-variances. In Fig. 5, for example, for the rectangular bar with index 4 and with gray levels 80–87, the corresponding group, group_10, needs only four iterations and its computation load should take 159 data into consideration after we find the final thresholded value t^* in group_14 at the eighth iteration. The white area in Fig. 5 denotes the computational load of our proposed algorithm; the white area and the gray area denote the computational load of the Otsu method. The gray area in Fig. 5 denotes the computation-saving advantage of our proposed algorithm.

3. Experimental results

In this section, some experiments are demonstrated to show that under the same determined threshold, our proposed HQ-based incremental algorithm has better execution-time performance when compared to the Otsu method. The used eight testing gray images, Lena, F16 Jet, Peppers, House, and Baboon are shown in Fig. 6 and they are used to evaluate the performance comparison. All the concerned experiments are performed on the IBM compatible computer with Pentium IV CPU 3.2 GHz and 1 GB RAM. The operating system used is MS-Windows XP and the program developing environment is Borland C++ Builder 6.0.

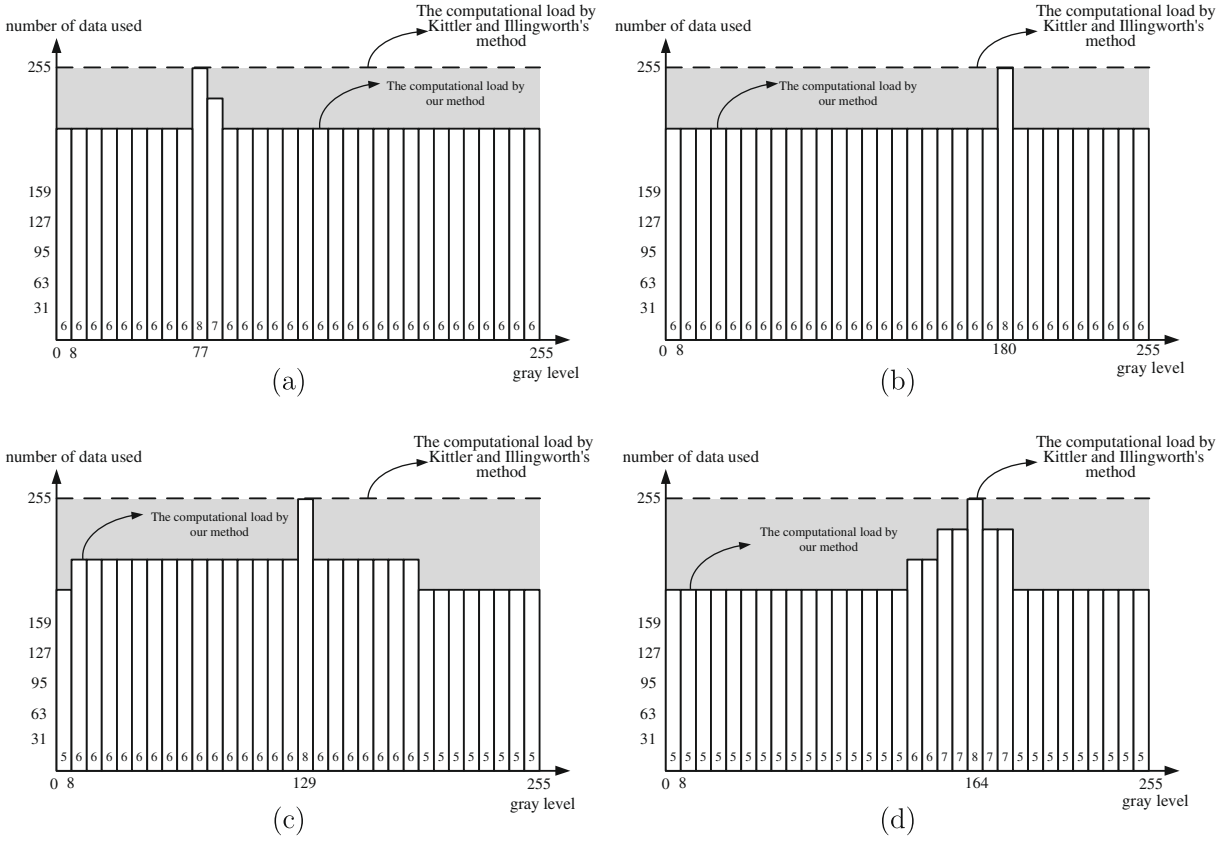

Fig. 9. The illustration of computation-saving advantage of our proposed algorithm. (a) Lena; (b) F16 Jet; (c) Peppers; (d) House; (e) Baboon; (f) Cakes; (g) Coins and (h) Culvert.

After running the Otsu method and our proposed HQ-based incremental algorithm on the eight testing images, both algorithms have the same thresholded values, 116, 152, 119, 146, 127, 110, 114 and 123. The eight binarized images are shown in Fig. 7.

In this paragraph, the execution-time performance comparison between the Otsu algorithm and our proposed algorithm is shown in Table 1. The execution-time requirement is measured by running the relevant algorithm 1000 times and the time unit used is millisecond (ms). From Table 1, for any testing image, the execution-time required in our proposed algorithm is much less than that required in the Otsu algorithm. The average execution-time improvement ratio is 36%. According to the computation-saving depiction in Fig. 5, for any testing image in Fig. 6, the computation-saving advantage of our proposed HQ-based incremental algorithm when compared to the Otsu method is illustrated in the gray area of Fig. 8.

Besides demonstrating the computational advantage of our proposed algorithm when compared to the Otsu method, we now compare the execution-time performance between Liao et al.'s method [17] and our proposed method. For binarizing gray images, Liao et al. build up two lookup tables for saving the $u - v$ interval zeroth-order moment $P(u, v)$ for all intensities u to $v, u \leqslant v$ and the $u - v$ interval first-order moment $S(u, v)$ for all intensities u to $v, u \leqslant v$. Based on the hashing approach, the above two constructed lookup tables are used to speed up the multilevel thresholding. In this research, we only consider the bilevel thresholding of Liao et al.'s method. Table 2 demonstrates that the average execution-time improvement ratio of our proposed method over Liao et al.'s method is 96% because Liao et al.'s method needs to spend considerable time on building up two lookup tables in advance. However, for general k-level thresholding, $k \geqslant 3$, Liao et al.'s method is quite effective and is much faster than the Otsu method. Our proposed HQ-based incremental algorithm for binarization is not easy to be extended for multilevel thresholding and the main contribution of this paper is to present an HQ-based incremental algorithm to speed up the Otsu method for binarizing gray images.

Besides demonstrating the computational advantage of our proposed algorithm when compared to Liao et al.'s method, we finally compare the execution-time performance between Kittler and Illingworth's method [7] and our proposed algorithm for binarization. In this research, we only consider the bilevel thresholding of Kittler and Illingworth's method. For binarizing gray images, Kittler and Illingworth's method used Bayes minimum error criterion to find the optimal threshold. Table 3 demonstrates that the average execution-time improvement ratio of our proposed HQ-based incremental algorithm over Kittler and Illingworth's method is 16%. For any testing image of Fig. 6, the computation-saving advantage of our

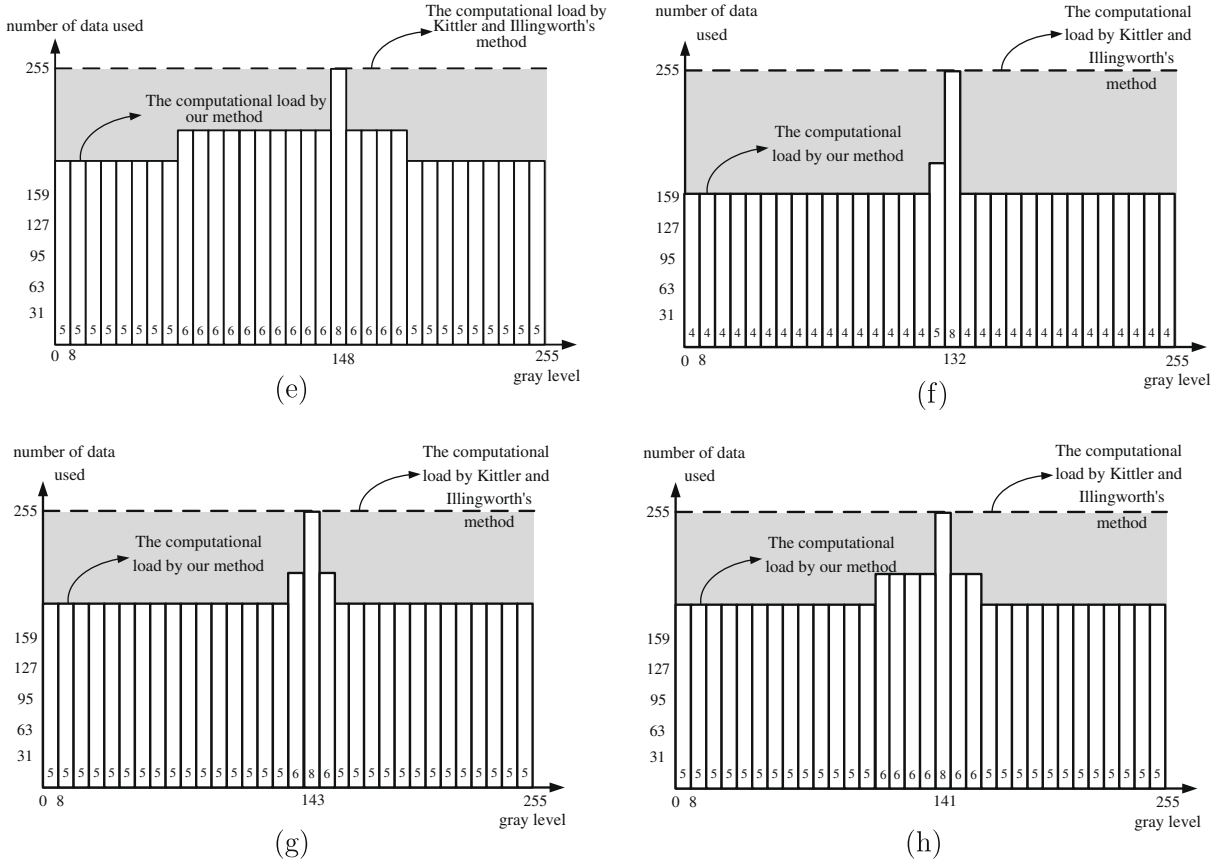

Fig. 9 (*continued*)

Table 4
The execution-time performance comparison between Kittler and Illingworth's method and our proposed method by running each one 1000 times when using the incremental data sequence (128, 64, 32, 16, 8, 4, 2, 2).

	Kittler et al.'s method (ms)	Proposed method (ms)	Improvement ratio (%)
Lena	14,297	12,488	12
F16 Jet	14,344	12,492	12
Peppers	14,203	12,063	15
House	14,938	11,546	22
Baboon	15,218	11,890	21
Cakes	13,828	11,046	20
Coins	13,453	10,875	19
Culvert	13,531	11,093	18
Average	13,839	11,554	17

proposed HQ-based incremental algorithm when compared to Kittler and Illingworth's method is depicted in the gray area of Fig. 9.

If we consider the incremental data sequence (128, 64, 32, 16, 8, 4, 2, 2) where we consider 128 data in the initial iteration; 64 data in the first iteration; ...; 2 data in the sixth iteration; 2 data in the seventh iteration. Table 4 shows that the average execution-time improvement ratio of our proposed HQ-based incremental algorithm over Kittler and Illingworth's method is 17% when using the incremental data sequence (128, 64, 32, 16, 8, 4, 2, 2). How to determine the optimal incremental data sequence is a rather hard problem and we put it as an open problem.

4. Conclusions

Our proposed fast heap- and quantization-based (HQ-based) incremental algorithm for binarizing gray images has been presented. The main two ideas of the proposed HQ-based incremental algorithm are two fold: (1) we first transfer the within-variance criterion used in the Otsu method into a new mathematical formulation, which is very suitable to be implemented in a fast incremental way, and it leads to the same thresholded value and (2) following the proposed incremental computation scheme, an efficient HQ-based data structure is presented to realize its implementation. Under eight real gray images, experimental results show that our proposed HQ-based incremental algorithm for binarization has 36% execution-time improvement ratio in average when compared to the Otsu method. In addition, our proposed HQ-based incremental algorithm can also be applied to speed up the Kittler and Illingworth method for binarization. Maybe the quantization technique used in the proposed HQ-based incremental binarization algorithm can be used in the nearest neighbor and motion estimation applications [19–21]. How to determine the optimal number of data considered in each incremental iteration, i.e. the optimal incremental data sequence, is an interesting open problem. In addition, how to apply the result of this paper to speed up the other computation applications, e.g. global optimization with one parameter, is another interesting research issue.

References

[1] R. Haralick, L. Shapiro, Computer and Robot Vision, vols. I and II, Addison Wesley, New York, 1992.
[2] L. Shapiro, G.C. Stockman, Computer Vision, Prentice-Hall, New Jersey, 2001.
[3] R.C. Gonzalez, R.E. Woods, Digital Image Processing. Chapter 10.3 (2002): Thresholding, second ed., Prentice-Hall, New York, 2002.
[4] K.L. Chung, Y.H. Huang, Speed up the computation of randomized algorithms for detecting lines, circles and ellipses using novel tuning- and LUT-based voting platform, Applied Mathematics and Computation 190 (1) (2007) 132–149.
[5] T.C. Chen, K.L. Chung, An Efficient Randomized Algorithm for Detecting Circles, Computer Vision and Image Understanding, vol. 83, Academic Press, 2001. pp. 172–191.
[6] N. Otsu, A threshold selection method from gray-level histogram, IEEE Transactions on System Man Cybernetics, SMC 9 (1979) 62–66.
[7] J. Kittler, J. Illingworth, Minimum error thresholding, Pattern Recognition 19 (1986) 41–47.
[8] W. Niblack, An Introduction to Digital Image Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1986. pp. 115–116.
[9] P.K. Sahoo, S. Soltani, A.K.C. Wong, Y. Chen, A survey of thresholding techniques, Computer Vision, Graphics, and Image Processing 41 (1988) 233–260.
[10] H. Yan, Unified formulation of a class of image thresholding techniques, Pattern Recognition 29 (1996) 2025–2032.
[11] J. Sauvola, M. Pietikainen, Adaptive document image binarization, Pattern Recognition 33 (2000) 225–236.
[12] I.K. Kim, D.W. Jung, R.H. Park, Document image binarization based on topographic analysis using a water flow model, Pattern Recognition 35 (2002) 265–277.
[13] K.L. Chung, W.Y. Chen, Fast adaptive PNN-based thresholding algorithms, Pattern Recognition 36 (2003) 2793–2804.
[14] B. Wang, X.F. Li, F. Liu, F.Q. Hu, Color text image binarization based on binary texture analysis, Pattern Recognition Letters 26 (2005) 1650–1657.
[15] B. Gatos, I. Pratikakis, S.J. Perantonis, Adaptive degraded document image binarization, Pattern Recognition 39 (2006) 317–327.
[16] Q. Hu, Z. Hou, W.L. Nowinnski, Supervised range-constrained thresholding, IEEE Transactions on Image Processing 15 (1) (2006) 228–240.
[17] P.S. Liao, T.S. Chen, P.C. Chung, A fast algorithm for multilevel thresholding, Journal of Information Science and Engineering 17 (2001) 713–727.
[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second ed., 2001 (Chapter 2, pp. 15–40; Chapter 6, pp. 127–144).
[19] C.H. Lee, L.H. Chen, A fast search algorithm for vector quantization using mean pyramids of codewords, IEEE Transactions on Communications 43 (1995) 1697–1702.
[20] Y.S. Chen, Y.P Hung, C.S. Fuh, Winner-update algorithm for nearest neighbor search, in: Proceedings of the International Conference on Pattern Recognition, Barcelona, Spain, vol. 2, September 2000, pp. 708–711.
[21] K. Lengwehasarit, A. Ortega, Probabilistic partial-distance fast matching algorithms for motion estimation, IEEE Transactions on Circuits and Systems for Video Technology 11 (2) (2001) 139–152.