PERGAMON

Pattern Recognition 32 (1999) 743–752

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

Binarization of noisy gray-scale character images by thin line modeling

Jeong-Hun Jang*, Ki-Sang Hong

Department of E.E., POSTECH, San 31, Hyoja-dong, Nam-ku, Pohang, Kyungbuk, 790-784, South Korea

Received 13 May 1997; accepted 28 January 1998

Abstract

In this paper, we propose two new methods for the binarization of noisy gray-scale character images obtained in an industrial setting. These methods are different from other conventional binarization methods in that they are specially designed to detect only character-like regions. They exploit the fact that characters are usually composed of thin lines (strokes) of uniform width. We first model the shape of the cross section of a character stroke and discuss how to detect the character stroke. Then, ALGORITHM I, which is a direct realization of our basic idea, is introduced, followed by an advanced algorithm named ALGORITHM II. The key to these algorithms is the local binarization-voting procedure. The performance of our methods is evaluated and compared with that of five other binarization methods using 550 slab ID number images, where a common character segmentation routine is attached to each of the different binarization routines and the segmentation success rate for each method is obtained. Experimental results show that ALGORITHM II results in far better performance than the other methods.

Keywords: Binarization; Noisy gray-scale character image; Thin line modeling; Character segmentation

1. Introduction

Binarization of gray-scale character images is a crucial step in off-line character recognition. Good binarization facilitates segmentation and recognition of characters. Binarization of gray-scale character images is not easy due to noise caused by various noise sources, which can be divided into two main categories. One is electronic devices used for image acquisition (scanners, CCD cameras, frame grabbers and so on) and the other is anything that directly distorts the shape of written or printed characters from which images are obtained. In general, character images obtained by CCD cameras in an industrial setting contain more noise than document images in OCR forms, since characters to be recognized are exposed to a variety of noise sources such as dirty materials, scratches, uneven surfaces, aging, varying outdoor illumination and so on [1]. This is the case we are going to deal with, where more intelligent binarization methods are needed.

More specifically, this paper deals with the binarization of the gray-scale images of ID numbers printed on slabs produced in steel factories. We collected 550 slab images in total, and categorized them into three classes—Class A,B, and C—according to their image qualities. Fig. 1 shows one of slab images we used for our experiments. The quality of this image is better than the average quality of Class B, but worse than that of Class A.

* Corresponding author. E-mail : jeonghun@postech.ac.kr

0031-3203/99/$ – See front matter
PII: S0031-3203(98)00019-3

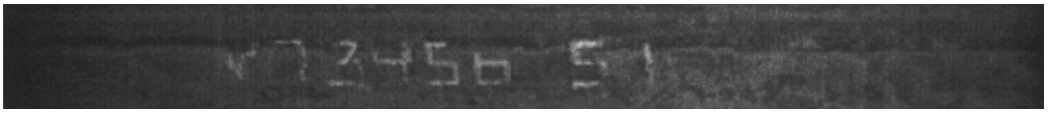


Fig. 1. An example of a slab image.

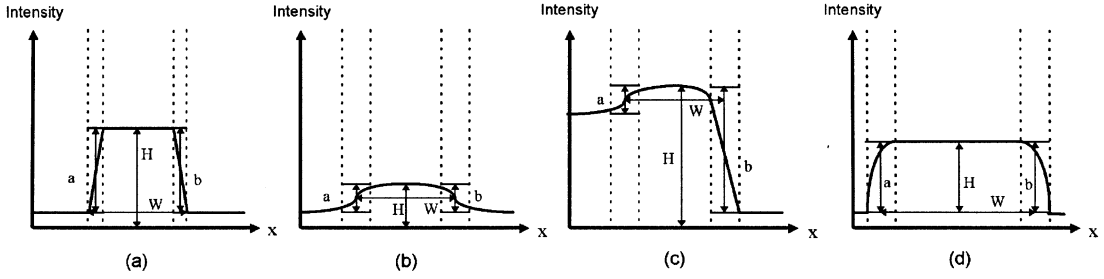


Fig. 2. Shapes of cross sections in several places of a noisy gray-scale character image.

Many binarization algorithms have been suggested in the past. In the early days, global thresholding techniques were developed [2–4], then followed by locally adaptive binarization methods [5–9]. Recently, several researchers proposed new methods in which skeletons of characters are detected directly by extracting topographic features [10–12]. We applied these previous methods to our test images, but failed to get satisfactory results.

It is our opinion that traditional binarization methods are *too general* to binarize *noisy character images* since they are intended to detect relatively bright regions regardless of their shapes, neglecting the fact that characters are usually printed with equal stroke width. As can be seen in Fig. 1, there exist many valleys and holes which are darker than their neighborhoods. It was observed that most conventional binarization methods, when applied to such an image, failed at the boundaries of such regions, which was the expected result.

In this paper, we propose two new algorithms—ALGORITHM I and ALGORITHM II—for the binarization of noisy gray-scale character images. They essentially exploit the fact that, as mentioned above, characters are composed of thin lines (strokes) of uniform width. In this case, we can think of a binarization problem as a kind of thin line detection problem.

In Section 2, we model the shape of the cross section of a gray-scale character image. Our two algorithms, which are motivated by the considerations of Section 2, are described in Section 3. In Section 4, experimental results are shown, followed by conclusions in Section 5.

2. Modeling the shape of the cross section of a gray-scale character image

In Fig. 2, possible shapes of cross sections in several places of a noisy gray-scale character image are illustrated. Fig. 2a represents the ideal shape of the cross section of a character stroke, while Fig. 2b–d represent the cross sections of the background which should be distinguished from a.

Fig. 2b depicts the case in which both a and b are so small that they are below some threshold. In the case of Fig. 2c, a is meaninglessly small whereas b is so large that it is above the given threshold. Fig. 2d shows the case in which both a and b are greater than the threshold value but the width W is relatively large compared to that of an ordinary character stroke. Obviously, conventional binarization methods have the common problem that they detect bright regions in c and d as well as in a. The methods using topographic features also have a problem in that they detect ridges not only in a but also in b and c. We must find the way to exclude the cases of b–d when detecting character regions.

We can infer from Fig. 2, three conditions of a character stroke as follows:

- W, the width of a stroke, is almost the same through all strokes of characters.
- a and b, the intensity differences between the foreground and the background in the boundaries of a stroke, should be greater than some threshold value.
- H, the intensity of a character region, is locally uniform and always larger than that of the background in the neighborhood.

In this paper, two algorithms are proposed which are designed to detect only regions that satisfy the three conditions of a character stroke described above. Section 3.1 introduces ALGORITHM I, which is a direct realization of the underlying idea of our method. ALGORITHM II, which is an advanced version that makes up for weakness in ALGORITHM I, is given in Section 3.2.

3. Proposed binarization algorithms

3.1. Algorithm I

In Fig. 3, the overall flow of ALGORITHM I is given. ALGORITHM I begins by detecting *edges* whose positions are good starting points to find character regions since, as can be seen intuitively in Fig. 2, character strokes correspond to relatively *bright regions* enclosed by *pairs of parallel edge lines*. In Fig. 4, a scene near the character region is illustrated. Here, the center of a 5×5 window W_{out} whose size is about twice the width of a stroke, is placed at an edge pixel position (i, j). By *simple binarization* of the input gray-scale image inside the window W_{out}, we can determine which pixels belong to a bright region in the neighborhood of (i, j). Since it is reasonable to assume that the character region and the background region have locally uniform intensity values, binarization of Fig. 4a inside W_{out} will produce a result like Fig. 4b. We can use as a threshold value an arithmetic mean of the maximum and minimum pixel values within a 3×3 inner window W_{in}. In order to determine whether the above-threshold region is enclosed by a pair of parallel edge lines, the distance between which is approximately equal to the given stroke width, *voting* for that region is performed. In other words, each element of a score array S belonging to the above-threshold region is incremented by 1. After the binarization-voting procedure is carried out at every edge pixel position, the score array, which contains a voting result for each pixel, is binarized globally with a fixed threshold value. Finally, postprocessing is performed.

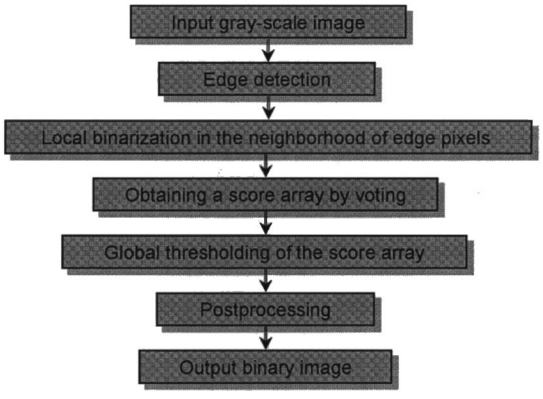


Fig. 3. The overall flow of ALGORITHM I.

Fig. 5 illustrates the shapes of cross sections of several intensity images in the first row and their corresponding score arrays in the second row. In fact, each case of Fig. 5 corresponds to that of Fig. 2. Let N_{out} denote the width of W_{out} and assume $N_{out} = 5$ in these examples. Fig. 5a shows the case where the distance between two parallel edge lines is approximately $N_{out}/2 \approx 2$. In this case, which corresponds to the case for most character regions, score elements between the edge lines will have the value $2N_{out} = 10$, which is evident from Fig. 4a. Fig. 5b shows a trivial case where scores are all 0 since no edges have been detected. In the case of Fig. 5c, where only one edge line has been detected, only score elements adjacent to the edge line will have the value $N_{out} = 5$. Finally, Fig. 5d illustrates the case in which the distance between the two edge lines detected is somewhat greater than $N_{out}/2 \approx 2$. In such a case, only in the middle of the region between the edge lines will scores be $2N_{out} = 10$. If we threshold the score arrays with the value $3N_{out}/2 = 7.5$, only two regions R_1 and R_2 in Fig. 5a and d will remain. R_2 can be effectively removed by using the fact that gradients of gray-scale pixel values at the boundary of R_1 are greater than those of R_2, which is what the postprocessing stage

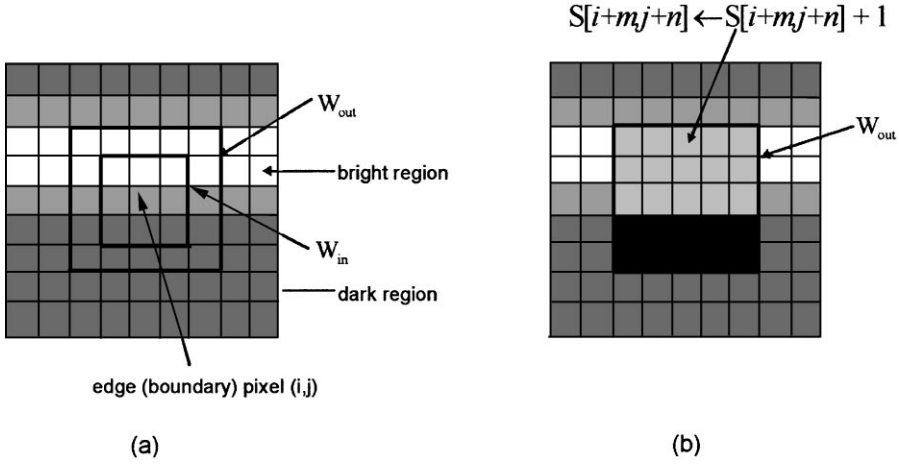


Fig. 4. Local binarization and voting near the character region.

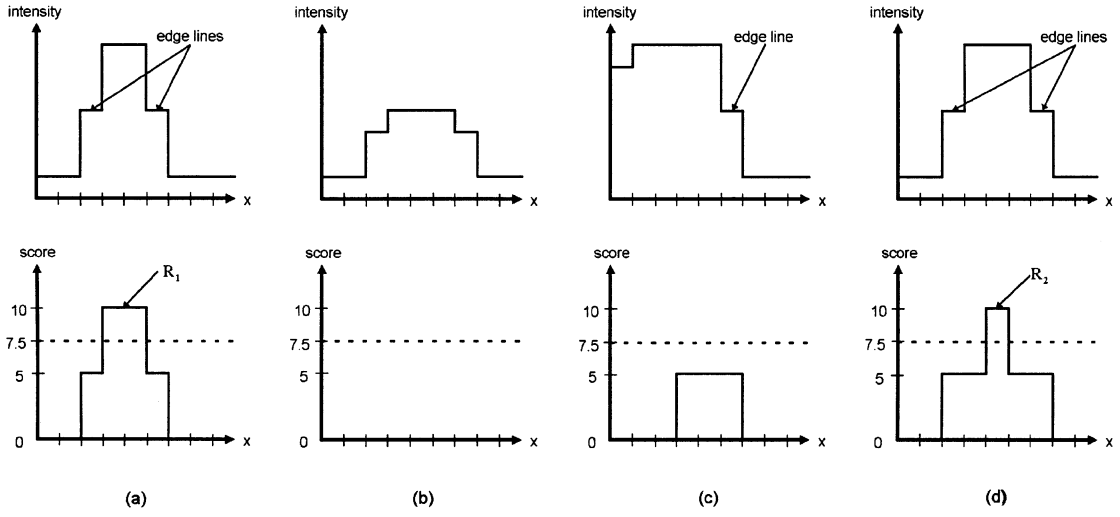


Fig. 5. The shapes of cross sections of several intensity images and their corresponding score arrays.

does. Therefore, we can obtain the desired binarization result. The detailed algorithm is as follows.

Edge detection:

(1) Smooth out an input gray-scale image I with a Gaussian filter. Let I_{gf} denote the resulting image.
(2) Get an edge image I_{edge} from I_{gf}. Edge pixels are assigned the value 1 and the others are assigned the value 0. We used a facet-based zero-crossing edge detector to extract fine edges [13].

Local binarization and voting:

(3) Create a score array S with the same size as I. Initialize each element of S with 0.
(4) Place an $N_{in} \times N_{in}$ window $W_{in}(i,j)$ and an $N_{out} \times N_{out}$ window $W_{out}(i,j)$ on $I_{gf}[i,j]$, where $I_{edge}[i,j] = 1$ (Fig. 4a). The values of N_{in} and N_{out} are dependent on the width of strokes to be detected, and $N_{in} \leqslant N_{out}$.
(5) If we define $W_{in}(i,j)\,[k,l]$ and $W_{out}(i,j)\,[m,n]$ as

$$W_{in}(i,j)\,[k,l] \equiv I_{gf}[i+k, j+l] \quad \text{for } -N_{in}/2 \leqslant k,l \leqslant N_{in}/2, \tag{1}$$

$$W_{out}(i,j)\,[m,n] \equiv I_{gf}[i+m, j+n] \quad \text{for } -N_{out}/2 \leqslant m,n \leqslant N_{out}/2,$$

and let $P_{min}(i,j)$ and $P_{max}(i,j)$ denote the minimum and the maximum values of $W_{in}(i,j)\,[k,l]$ respectively, then, do for $-N_{out}/2 \leqslant m,n \leqslant N_{out}/2$,

$$S[i+m, j+n] \leftarrow S[i+m, j+n] + 1 \tag{2}$$

if $W_{out}(i,j)\,[m,n] \geqslant t(i,j)$, where

$$t(i,j) = \frac{P_{min}(i,j) + P_{max}(i,j)}{2} \tag{3}$$

(Fig. 4b).
(6) Repeat steps (4) and (5) at every pixel position satisfying $I_{edge}[i,j] = 1$.

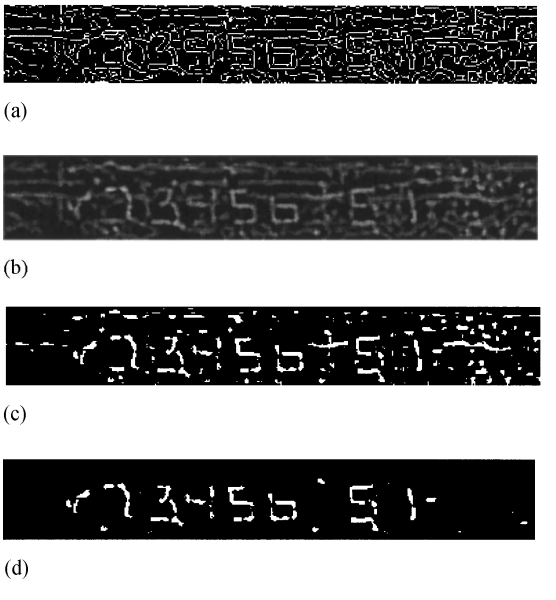
Global thresholding:

(7) Binarize the resulting score array S with a threshold t_s to obtain an image I_{bin}. If we let the size of W_{out} be twice the width of a stroke, t_s is given by $3N_{out}/2$.

Postprocessing:

(8) Apply Yanowitz and Bruckstein's postprocessing algorithm to I_{bin} to remove unnecessary connected components [9]. Yanowitz and Bruckstein's method can be stated simply as follows. After an input image is smoothed by a low-pass filter, the average of gradient magnitudes of gray-scale pixel values at the boundary of each connected component is calculated. Then, connected components having an average gradient magnitude below a given threshold are removed.

The intermediate and final results of ALGORITHM I when applied to the image in Fig. 1 are shown in Fig. 6. Note that other postprocessing algorithms specific to the application can be used before or after step (8) in the above procedure. We removed horizontally long connected components before step (8).



(a) (b) (c) (d)

Fig. 6. ALGORITHM I: (a) Edge image I_{edge} (b) Score array S (c) Binarized score array I_{bin} (d) Final result.

3.2. ALGORITHM II

The overall flow of ALGORITHM II is shown in Fig. 7. It is evident that the performance of ALGORITHM I depends heavily on the performance of the edge detector used. It was observed in the experiment that thin lines (strokes) were detected broken when some edge pixels were missing. In ALGORITHM II, no explicit edge detector is used, thereby avoiding the dependence on an edge detector. A binarization-voting procedure which is slightly different from that of ALGORITHM I is performed at every pixel position. The difference between the maximum and the minimum pixel values inside W_{in} is used as a local contrast measure which has a direct influence on score evaluation.

Another difference of ALGORITHM II from ALGORITHM I is that a process called *background equalization* is added. The uneven surface of the background in an input gray-scale image is iteratively smoothed out through background equalization. During background equalization, m, the normalized sum of gray-scale pixel values inversely weighted by scores computed at the previous stage, is calculated in the neighborhood of each pixel p. If the value of p is less than m, it is replaced with m. The resulting image serves as an input to the next iteration. Then a new score is evaluated for each pixel. As iteration goes on, valleys or holes in the input gray-scale image are gradually filled up while character regions are preserved. Instead of the input gray-scale image, the background-equalized image is used during Yanowitz and Bruckstein's postprocessing stage. More formally, ALGORITHM II consists of the following steps.

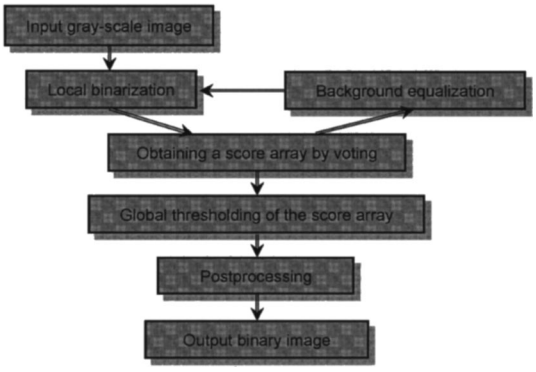


Fig. 7. The overall flow of ALGORITHM II.

Local binarization and voting:

(1) Create a score array S with size equal to that of an input gray-scale image I. Initialize each element of S with 0.
(2) Place an $N_{in} \times N_{in}$ window $W_{in}(i,j)$ and an $N_{out} \times N_{out}$ window $W_{out}(i,j)$ on $I[i,j]$. The values of N_{in} and N_{out} are dependent on the width of strokes to be detected, and $N_{in} \leqslant N_{out}$.
(3) If we define $W_{in}(i,j)\ [k,l]$ and $W_{out}(i,j)\ [m,n]$ as

$$\begin{aligned} &W_{in}(i,j)\ [k,l] \equiv I[i+k, j+l] \\ &\quad \text{for } -N_{in}/2 \leqslant k,l \leqslant N_{in}/2, \\ &W_{out}(i,j)\ [m,n] \equiv I[i+m, j+n] \\ &\quad \text{for } -N_{out}/2 \leqslant m,n \leqslant N_{out}/2, \end{aligned} \tag{4}$$

and let $P_{min}(i,j)$ and $P_{max}(i,j)$ denote the minimum and the maximum values of $W_{in}(i,j)\ [k,l]$ respectively, then, do for $-N_{out}/2 \leqslant m,n \leqslant N_{out}/2$

$$\begin{aligned} &S[i+m, j+n] \leftarrow S[i+m, j+n] + s^{+}(i,j) \\ &\quad \text{if } I[i+m, j+n] \geqslant t(i,j), \\ &S[i+m, j+n] \leftarrow S[i+m, j+n] - s^{-}(i,j) \\ &\quad \text{otherwise,} \end{aligned} \tag{5}$$

where

$$t(i,j) = \frac{P_{min}(i,j) + P_{max}(i,j)}{2}, \tag{6}$$

$$s^{+}(i,j) = \frac{1}{1 + \exp(-\lambda(x - x_0))}, \tag{7}$$

$$x = P_{max}(i,j) - P_{min}(i,j), \tag{8}$$

$$s^{-}(i,j) = 1 - s^{+}(i,j). \tag{9}$$

It is worth noting that $s^{+}(i,j)$ in Eq. (7) represents a sigmoid function monotonically varying from 0 to 1. As λ gets larger, $s^{+}(i,j)$ becomes closer to a thresholding function with the threshold x_0. Therefore,

score increment is proportional to the local contrast $P_{\max}(i,j) - P_{\min}(i,j)$, but is confined to be between 0 and 1.

(4) Repeat steps (2) and (3) at every pixel position. Then, do $S[i,j] \leftarrow 0$ for all i,j if $S[i,j] \leqslant 0$.

Background equalization:

(5) Create an array W with size equal to that of S. Each element of W is given by

$$W[i,j] = \exp\left(-\frac{S^2[i,j]}{2\kappa^2}\right). \tag{10}$$

Note that Eq. (10) represents a function which monotonically decreases from 1 to 0 when $S[i,j] \geqslant 0$.

(6) Create an array $\bar{I}$ with size equal to that of I. The background-equalized image $\bar{I}$ is given by

$$\bar{I}[\mathrm{i,j}] = \begin{cases} b(i,j) & \text{if } I[i,j] < b(i,j), \\ I[i,j] & \text{otherwise,} \end{cases} \tag{11}$$

where

$$b(i,j) = m(i,j) + \mu\sigma(i,j), \tag{12}$$

$$m(i,j) = \frac{1}{P}\sum_{k,l=-M/2}^{M/2} W[i+k,j+l] \times I[i+k,j+l], \tag{13}$$

$$\sigma^2(i,j) = \left(\frac{1}{P}\sum_{k,l=-M/2}^{M/2} W[i+k,j+l] \times \mathrm{I}^2[i+k,j+l]\right) - m^2(i,j), \tag{14}$$

$$P = \sum_{k,l=-M/2}^{M/2} W[i+k,j+l]. \tag{15}$$

$m(i,j)$ and $\sigma(i,j)$ in Eq. (12) represent the mean and the standard deviation of pixel values weighted by W inside an $M \times M$ window placed on $I[i,j]$. $\sigma(i,j)$ plays the role of boosting up the speed of background equalization.

Iteration:

(7) Do $I \leftarrow \bar{I}$.

(8) Repeat step (1)–step (7) K times.

Global thresholding:

(9) Binarize the resulting array S with a threshold t_s to obtain an image I_{bin}.

Postprocessing:

(10) Perform the morphological closing operation with a 3×3 squared SE (structuring element).

(11) After CRE (connected region extraction), connect neighboring CR's having small dissimilarity with appropriate SE's. Let $m(i)$ and $\sigma(i)$ denote the mean and the standard deviation of gray-scale pixel values belonging to the ith CR, C(i). The dissimilarity d between two CR's C(i) and C(j) is given by

$$d = |m(i) - m(j)|, \tag{16}$$

and C(i) and C(j) are connected if $d < k\sigma(i)$ and $d < k\sigma(j)$.

(12) Perform Yanowitz and Bruckstein's postprocessing step. It should be noted that in ALGORITHM II, gradient magnitudes are computed from the background equalized image $\bar{I}$ instead of the input image I.

The intermediate and final results of ALGORITHM II when applied to the image in Fig. 1 are shown in Fig. 8. Note that the background equalized image in Fig. 8c has an almost uniform background while its character regions are preserved, which makes it easier to remove unnecessary connected components in the postprocessing stage.

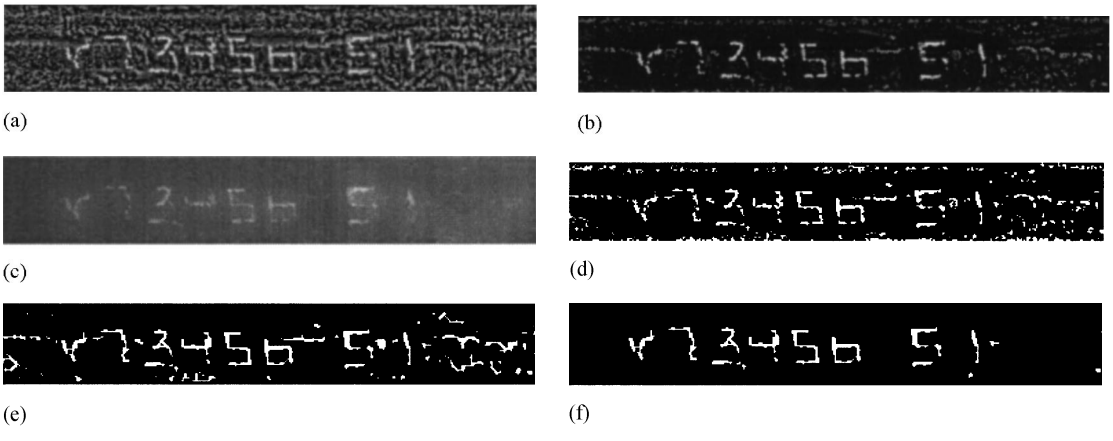


Fig. 8. ALGORITHM II : (a) Score array S after one iteration (b) Score array S after ten iterations (c) Background equalized image after ten iterations (d) Binarized score array I_{bin} (e) Result after postprocessing step (11) (f) Final result.

4. Experimental results

As mentioned in Section 1, we collected 550 slab images in total, and categorized them into three classes—Class A, B, and C—according to their image qualities. Our two algorithms were tested with these images. Five other binarization methods were also tested for the purpose of comparison. They were chosen based on Trier et al.'s experimental results [14,15], and are listed below.

- Bernsen's method [5].
- Eikvil et al.'s method [6].
- Niblack's method [7].
- Trier and Taxt's method [8].
- Yanowitz and Bruckstein's method [9].

We first selected 10 images which have different characteristics and set parameters for each method so that the best binarization results were obtained. In ALGORITHM I, we set $N_{in} = 3$, $N_{out} = 7$, $t_s = 10$, and in ALGORITHM II, we set $N_{in} = 3$, $N_{out} = 5$, $\lambda = 0.1$, $x_0 = 7.0$, $\kappa = 5.0$, $M = 11$, $\mu = 0.4$, $t_s = 11.0$, $K = 10$.

In Fig. 9, binarization results for two sample images which were not used for parameter tuning, are shown.

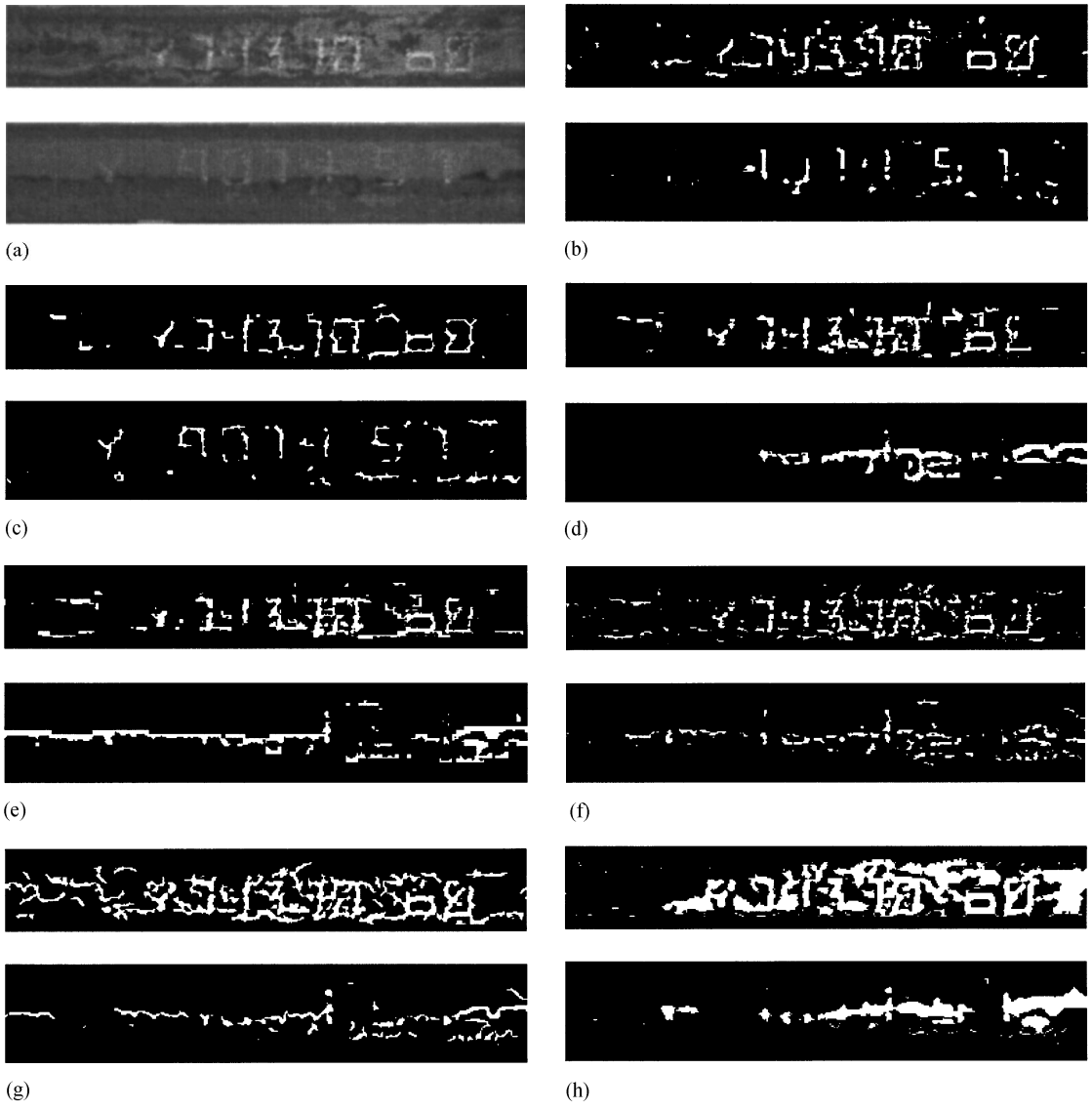


Fig. 9. Binarization results : (a) Input gray-scale images (b) ALGORITHM I (c) ALGORITHM II (d) Bernsen's method (e) Eikvil et al.'s method (f) Niblack's method (g) Trier and Taxt's method (h) Yanowitz and Bruckstein's method.

The sample images given have very different characteristics. The left image in Fig. 9a has a noisier background than the one on the right. On the other hand, the contrast between a foreground and a background is much lower in the right image than the left one. It is easily observed from the figure that ALGORITHM II gives better results than the others for both sample images.

To evaluate the performance of each binarization method quantitatively, we attached the same *character segmentation* routine to each of the different binarization routines. Then, we obtained a segmentation success rate for each method. Fig. 10 shows an example of a segmentation result which is obtained by applying the segmentation algorithm to the binarization result shown in Fig. 8f. For this evaluation, since each character is printed on a slab with approximately equal width, we employed a classical *dissection* method based on projection analysis [16]. Therefore, no character classifier is involved in the

Fig. 10. An example of a character segmentation result.

Table 1
Segmentation success rate and execution time for each binarization method

Binarization method	Class A (133 images)	Class B (246 images)	Class C (171 images)	Total (550 images)	Execution time
ALGORITHM I	122 91.7%	188 76.4%	93 54.4%	403 73.3%	1.7 s
ALGORITHM II	121 91.0%	219 89.0%	118 69.0%	458 83.3%	5.0 s
Bernsen	100 75.2%	121 49.2%	35 20.5%	256 46.5%	0.65 s
Eikvil *et al.*	113 85.0%	148 60.1%	39 22.8%	300 54.5%	1.1 s
Niblack	116 87.2%	135 54.9%	44 25.7%	295 53.6%	0.4 s
Trier / Taxt	107 80.0%	144 58.5%	60 35.1%	311 56.5%	0.5 s
Yanowitz / Bruckstein	92 69.2%	124 50.4%	41 24.0%	257 46.7%	39.2 s

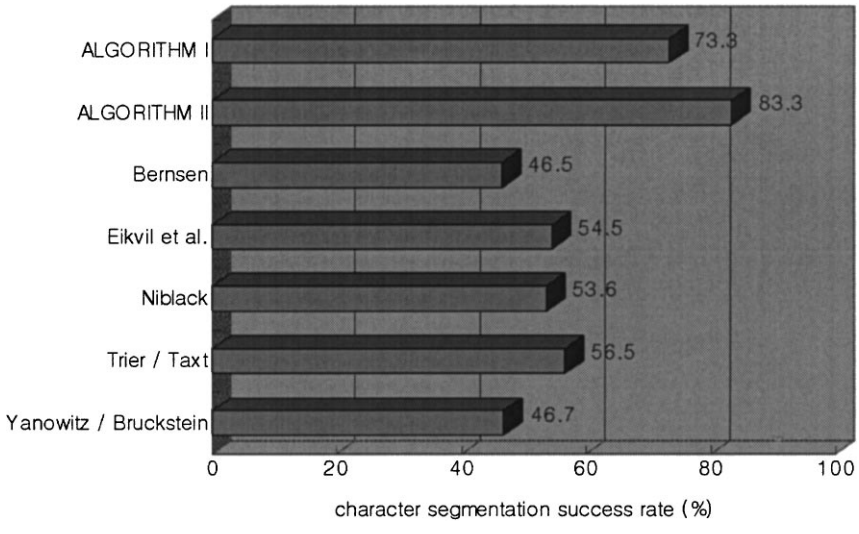


Fig. 11. Segmentation success rates for the tested method (550 slab images).

Table 2
Environment for the experiments

Computer	HP Vectra XU 6/200 (CPU : Pentium Pro 200Mhz, RAM : 64MB)
OS	Windows 95
Compiler	Microsoft Visual C++ 4.0 (no optimization)
Image size	624×71

segmentation stage. The segmentation results are given in Table 1 and Fig. 11. The execution time for each method is also given in the table. These measurements were obtained under the environment described in Table 2. According to Table 1, the success rate of ALGORITHM II is highest among all the methods tested, and ALGORITHM I gives the next highest success rate. The success rates of ALGORITHM I and ALGORITHM II are significantly higher than those of the other methods.

5. Conclusions

This paper has described new methods for the binarization of noisy gray-scale character images obtained in an industrial setting. Our methods are specially designed to binarize gray-scale character images more effectively by using the fact that characters are usually composed of thin lines of uniform width. We applied our methods and other conventional methods to the binarization of steel slab ID images, and experimental results show that our methods give the best binarization results. These experimental results demonstrate that our methods are more adequate than previous methods for the binarization of noisy gray-scale character images.

6. Summary

Binarizaion of gray-scale character images is a crucial step in off-line character recognition. Good binarization facilitates segmentation and recognition of characters. But, this is not an easy task as input gray-scale images tend to be noisy. In general, character images obtained by CCD cameras in an industrial setting contain more noise than document images in OCR forms, since the character to be recognized are exposed to a variety of noise sources such as dirty materials, scratches, uneven surfaces, aging, varying outdoor illumination and so on. In particular, this paper deals with the binarization of the gray-scale images of ID numbers printed on slabs produced in steel factories.

Numerous binarization algorithms have been suggested. In the early days, global thresholding techniques were developed, subsequently followed by locally adaptive binarization methods. Recently, several researchers proposed new methods in which skeletons of characters are detected directly by extracting topographic features. We applied these previous methods to our test images, but failed to get satisfactory results.

It is our opinion that traditional binarization methods are *too general* to binarize *noisy character images* since they are intended to detect relatively bright regions regardless of their shapes, neglecting the fact that characters are usually printed with equal stroke width.

In this paper, we propose two new algorithms—ALGORITHM I and ALGIRITHM II—for the binarization of noisy gray-scale character images. They essentially exploit the fact that, as mentioned above, characters are composed of thin lines (strokes) of uniform width. In this case, we can think of a binarization problem as a kind of thin line detection problem.

The basic idea involved in ALGORITHM I is that strokes of characters correspond to relatively bright regions enclosed by pairs of parallel edge lines. In order to find them, the center of a square window, whose size is about twice the width of a stroke, is placed on each edge pixel position, and simple binarization of the given gray-scale image is performed inside that window. Next, voting for above-threshold pixels is carried out. After the binarization-voting procedure is performed on every edge pixel position, the score array, which contains a voting result for each pixel, is binarized globally with a fixed threshold value. Finally, Yanowitz and Bruckstein's postprocessing algorithm is applied to the binarized score array to remove unnecessary connected components.

The performance of ALGORITHM I depends heavily on the edge detector used. It was observed in our experiments that thin lines (strokes) were detected broken when some edge pixels were missing. In ALGORITHM II, no explicit edge detector is used, thereby avoiding the dependence on an edge detector. The score is evaluated at every pixel position with an algorithm conceptually similar to that of ALGORITHM I.

Another difference of ALGORITHM II from ALGORITHM I is that a process called *background equalization* is added. The uneven surface of the background of an input gray-scale image is iteratively smoothed out through background equalization. The background-equalized image is used instead of the input gray-scale image in Yanowitz and Bruckstein's postprocessing stage.

We tested our two algorithms and five other binarization methods with 550 slab images. To measure the performance of each method, we attached the same *character segmentation* routine to each of the different binarization routines. Then, we obtained a segmentation success rate for each mehtod. According to our experiments, the success rate of ALGORITHM II was highest among all of the methods tested, and ALGORITHM I gave the next highest success rate. This experimental result proves that our methods are more adequate than previous

methods for the binarization of noisy gray-scale character images.

References

[1] J.H. Jang, J.H. Ku, K.S. Hong, J.H. Kim, Two-stage recognition of freight train ID number under outdoor environment, Proc. Int. Conf. Document Analysis and Recognition, Montreal, Canada, 1995, pp. 986–990.
[2] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Systems, Man Cybernet. 9 (1) (1979) 62–66.
[3] J. Kittler, J. Illingworth, On threshold selection using clustering criteria, IEEE Trans. Systems, Man Cybernet. 15 (1985) 652–655.
[4] J. Kittler, J. Illingworth, Minimum error thresholding, Pattern Recognition 19 (1) (1986) 41–47.
[5] J. Bernsen, Dynamic thresholding of grey-level images, Proc. Int. Conf. Pattern Recognition, Paris, France, 1986, pp. 1251–1255.
[6] L. Eikvil, T. Taxt, K. Moen, A fast adaptive method for binarization of document images, Proc. Int. Conf. Document Analysis and Recognition, Saint-Malo, France, 1991, pp. 435–443.
[7] W. Niblack, An Introduction to Digital Image Processing, pp. 115–116. Prentice-Hall, Englewood Cliffs, New Jersey, 1986, 115–116.
[8] Ø.D. Trier, T. Taxt, Improvement of "integrated function algorithm" for binarization of document images, Pattern Recognition Lett. 16 (3) (1995) 277–283.
[9] S.D. Yanowitz, A.M. Bruckstein, A new method for image segmentation, Comput. Vision, Graphics, and Image Process 46 (1) (1989) 82–95.
[10] L. Wang, T. Pavlidis, Direct gray-scale extraction of features for character recognition, IEEE Trans. Pattern Anal. Mach. Intell. 15 (10) (1993) 1053–1067.
[11] B. Sakoda, J. Zhou, T. Pavlidis, Refinement and testing of a character recognition system based on feature extraction in grayscale space, Proc. Int. Conf. Document Analysis and Recognition, Tsukuba, Japan, 1993, pp. 464–469.
[12] S.W. Lee, Y.J. Kim, Direct extraction of topographic features for gray scale character recognition, IEEE Trans. Pattern Anal. Mach. Intell. 17 (7) (1995) 724–729.
[13] R.M. Haralick, L.G. Shapiro, Computer and Robot Vision vol. 1, Addison-Wesley, Reading, Massachusetts, 1992, pp. 392–403.
[14] Ø.D. Trier, T. Taxt, Evaluation of binarization methods for document images, IEEE Trans. Pattern Anal. Mach. Intell. 17 (3) (1995) 312–315.
[15] Ø.D. Trier, A.K. Jain, Goal-directed evaluation of binarization methods, IEEE Trans. Pattern Anal. Mach. Intell. 17 (12) (1995) 1191–1201.
[16] R.G. Casey, E. Lecolinet, A survey of methods and strategies in character segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 18 (7) (1986) 690–706.

About the Author—JEONG-HUN JANG was born in Seoul, Korea, in 1970. He received the B.S. degree in Electrical Engineering from Hanyang University, Korea, in 1994 and the M.S. degree in Electrical & Electronic Engineering from POSTECH, Korea, in 1996. He is now in the Ph.D. program at POSTECH. His research interests include computer vision and pattern recognition.

About the Author—KI-SANG HONG received the B.S. degree in Electronic Engineering from Seoul National University, Korea, in 1977, and the M.S. degree in Electrical & Electronic Engineering from KAIST, Korea, in 1979. He also received the Ph.D. degree from KAIST in 1984. During 1984–1986, he was a researcher in the Korea Atomic Energy Research Institute and in 1986, he joined POSTECH, Korea, where he is currently an associate professor of Electrical & Electronic Engineering. During 1988–1989, he worked in the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, as a visiting professor. His current research interests include computer vision, augmented reality and pattern recognition.