Technical Correspondence

Configuration-Transition-Based Connected-Component Labeling

Lifeng He, *Senior Member, IEEE*, Xiao Zhao, Yuyan Chao, and Kenji Suzuki, *Senior Member, IEEE*

***Abstract*— This paper proposes a new approach to label-equivalence-based two-scan connected-component labeling. We use two strategies to reduce repeated checking-pixel work for labeling. The first is that instead of scanning image lines one by one and processing pixels one by one as in most conventional two-scan labeling algorithms, we scan image lines alternate lines, and process pixels two by two. The second is that by considering the transition of the configuration of pixels in the mask, we utilize the information detected in processing the last two pixels as much as possible for processing the current two pixels. With our method, any pixel checked in the mask when processing the current two pixels will not be checked again when the next two pixels are processed; thus, the efficiency of labeling can be improved. Experimental results demonstrated that our method was more efficient than all conventional labeling algorithms.**

***Index Terms*— Pattern recognition, image analysis, connected component, labeling.**

I. Introduction

LABELING of connected components in a binary image is one of the most fundamental operations in pattern analysis, pattern recognition, computer (robot) vision, and machine intelligence [1], [3], [6]. By use of the labeling operation, a binary image is transformed into a symbolic image in which all pixels belonging to a connected component are assigned a unique label. Labeling is required whenever a computer or a system needs to recognize independent objects (connected components) in binary images as separate objects.

Many algorithms have been proposed for addressing this issue. For ordinary computer architectures (in contrast to, e.g., parallel architectures) and 2D pixel-based images (in contrast to run-based or hierarchical-tree-based images), there are mainly two classes of labeling algorithms: label-equivalence-based algorithms and label propagation algorithms.

Label-equivalence-based labeling algorithms process an image in the raster-scan way, and complete labeling by the following four basic processing steps:

(1) Provisional label assigning, i.e., assigning to each foreground pixel a provisional label;

(2) Label equivalence recording, i.e., recording all provisional labels that are found to belong to the same connected component as equivalent labels;

(3) Label equivalence resolving, i.e., finding a unique representative label for all equivalent labels;

(4) Label replacing, i.e., replacing each provisional label with its representative label.

According to the number of times of scanning an image for labeling, there are multi-scan algorithms [5], [15] two-scan algorithms [7]–[10], [18], and one-and-a-half-scan algorithms [11].

Recently, some new labeling algorithms were proposed. He *et al.* [9] proposed a new two-scan labeling algorithm, which uses equivalent label sets to record and resolve label equivalences. They also proposed a one-and-a-half scan algorithm [11], which is an improvement on the run-based two-scan algorithm proposed in [8]. On the other hand, Grana *et al.* proposed a block-based two-scan algorithm [4], which resolves connectivity among 2×2 blocks, as shown in Fig. 1. Because all foreground pixels in a 2×2 block are certainly 8-connected, they belong to the same connected component and thus will

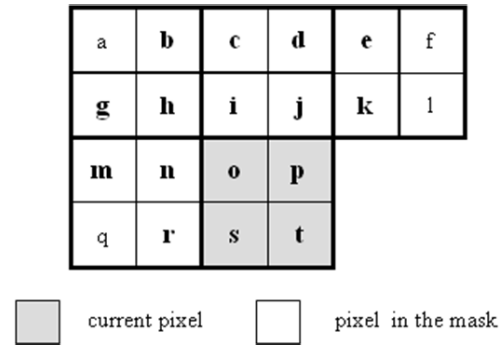


Fig. 1. Mask used in the block-based labeling algorithm, where the pixels in bold face are pixels that should be considered for processing the pixels in the current block.

Manuscript received October 23, 2012; revised April 3, 2013; accepted October 17, 2013. Date of publication November 7, 2013; date of current version January 13, 2014. This work was supported by the Ministry of Education, Science, Sports and Culture, Japan, through Grant-in-Aid for Scientific Research (C) under Grant 23500222. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ton Kalker.

L. He is with the Artificial Intelligence Institute, College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China, and also with the Faculty of Information Science and Technology, Aichi Prefectural University, Aichi 480-1198, Japan (e-mail: helifeng@ist.aichi-pu.ac.jp).

X. Zhao is with the Artificial Intelligence Institute, College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China (e-mail: zhaoxiao@sust.edu.cn).

Y. Chao is with the Graduate School of Environment Management, Nagoya Sangyo University, Aichi 488-8711, Japan, and also with the College of Mechanical and Electrical Engineering, Shaanxi University of Science and Technology, Xi'an 710021, China (e-mail: chao@nagoya-su.ac.jp).

K. Suzuki is with the Department of Radiology, Division of the Biological Sciences, The University of Chicago, Chicago, IL 60637 USA (e-mail: suzuki@uchicago.edu).

Digital Object Identifier 10.1109/TIP.2013.2289968

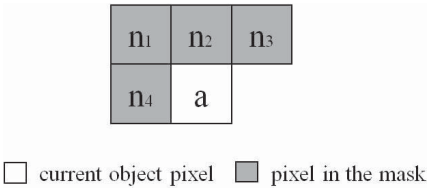


Fig. 2. Mask used in the HCS algorithm.

be assigned the same label. Therefore, instead of assigning to each foreground pixel a provisional label, this algorithm assigns to each foreground-pixel-contained block a provisional label. It uses equivalent-label sets, the same method as proposed in [7], for recording and resolving label equivalences among blocks. In the second scan, it assigns to all foreground pixels in each block the representative label of the block. However, for the current block, the number of neighbors to be considered becomes 16, much larger than pixel-based two-scan algorithms (which is 4), and the decision tree generated by this algorithm contains 210 nodes, with 211 leaves sparse over 14 levels.

On the other hand, search and label propagation algorithms first search an unlabeled foreground pixel from a given image. If found, they label the foreground pixel with a new label; then, in the later processing, they assign the same label to all foreground pixels connected to the foreground pixel. All such algorithms access an image in an irregular way. Therefore, they are not suitable for pipeline processing, parallel implementation, or systolic-array implementations. Although they are called one-scan algorithms, they usually need to process the background pixels around the foreground pixels twice, and all or some of the foreground pixels at least twice (perhaps more); they are really not a one-scan algorithm. There are also run-based algorithms [1] and contour-tracing algorithms [2]. According to the experimental results reported in [9], these algorithms are inferior to label-equivalence-based algorithms.

This paper presents a new two-scan labeling algorithm. We scan image lines alternate lines, and process pixels two by two. By considering configuration transition in the mask, for processing the current two pixels, we utilize the information detected during processing the last two pixels as much as possible. By our method, we can avoid checking the pixels in the mask repeatedly; thus, the efficiency of labeling can be improved. Experimental results demonstrated that our method was more efficient than all conventional two-scan labeling algorithms for most images.

II. Preliminaries

The two-scan labeling algorithm proposed by He, Chao, and Suzuki in [9] first uses equivalent-label sets to record and resolve label equivalences. For convenience, we call this algorithm the *HCS algorithm*. At any moment in the first scan, all equivalent labels found so far are combined in an equivalent label set, and the minimal label in an equivalent label set is called the *representative label* of the set as well as all the labels in the set. Moreover, an equivalent label set with the representative label t is denoted as $S(t)$, and the representative label of a provisional label b is represented by $r[b]$.

In the first scan, the HCS algorithm uses the mask shown in Fig. 2 for processing the current pixel a. If it is a background pixel, nothing needs to be done. Otherwise, the HCS algorithm processes it as follows: if there is no foreground pixel (label) in the mask, this means that the foreground pixel does not connect to any foreground pixel processed up to now. At this point, in the processed image area, the current foreground pixel belongs to a new connected component consisting of itself only. The HCS algorithm assigns a new provisional label m, which is initialized to 1, to the current pixel a, establishes the equivalent label set $S(m) = \{m\}$ for the new connected component, sets the representative label of m as itself, i.e., $r[m] = m$, and increases m by 1 for later processing.

On the other hand, if there are some foreground pixels (labels) in the mask, then all such foreground pixels and the current foreground pixel are 8-connected, and they belong to the same connected component. In other words, all provisional labels in the mask are equivalent labels, and they should be combined. Suppose that u and v are equivalent labels that belong to $S(r[u])$ and $S(r[v])$, respectively. If $r[u] = r[v]$, the two equivalent label sets are the same, and thus, they belong to the same equivalent label set already; therefore, nothing needs to be done. Otherwise, without loss of generality, suppose that $r[u] < r[v]$, i.e., $r[u]$ is the smallest label in the two equivalent label sets; then the combination of the two equivalent label sets can be completed by the following operations:

$$S(r[u]) = S(r[u]) \cup S(r[v]); \qquad (\forall s \in S(r[v]))(r[s] = r[u]).$$

In this way, at any processing point in the first scan, all equivalent provisional labels found so far are combined in an equivalent label set with the same representative label.

As soon as the first scan is finished, all equivalent labels of each connected component will have been combined in an equivalent label set with a unique representative label. In the second scan, by replacement of each provisional label with its representative label, all foreground pixels of each connected component will be assigned a unique label.

The HCS algorithm was improved in [10] by processing the current foreground pixels following a background pixel and those following a foreground pixel are processed in a different way: for a foreground pixel following a background pixel, i.e., n_4 in the mask shown in Fig. 2 is a background pixel, we do not need to consider pixel n_4 but the other three pixels in the mask; for a foreground pixel following another foreground pixel, i.e., n_4 is a foreground pixel, we can assign pixel n_4's provisional label to it, and then resolve the connectivity with other foreground pixels (if any) in the mask. Because whether the current foreground pixel follows a background pixel or a foreground pixel can be known without any additional computing cost, this algorithm is more efficient than the HCS algorithm. For convenience, we call this algorithm the *HCS_1 algorithm*.

III. Outline of Our Proposed First-Scan Method

In the first scan, unlike most pixel-based two-scan labeling algorithms, which scan image lines one by one and process pixels one by one, our proposed algorithm uses the mask

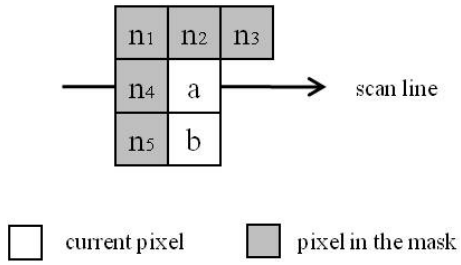


Fig. 3. Mask used in our algorithm.

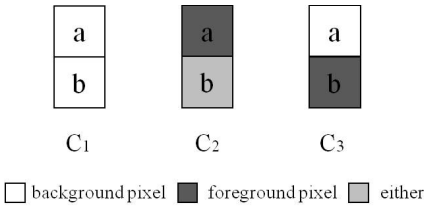


Fig. 4. Three cases need to be considered for the two current pixels a and b.

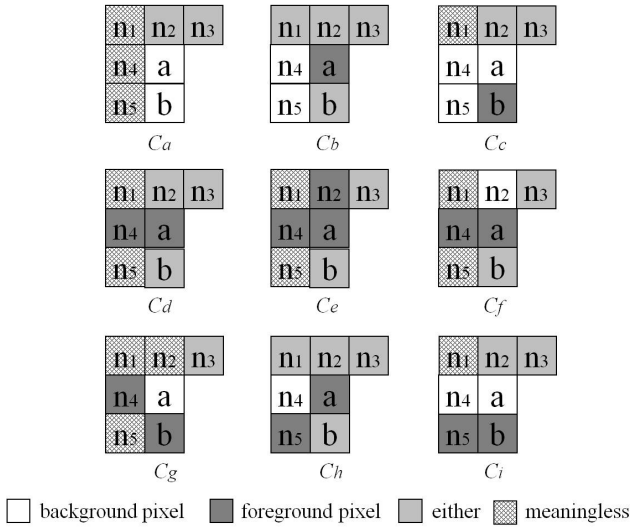


Fig. 5. Configurations of the mask and the two current pixels a and b.

shown in Fig. 3 to scan the given image alternate lines, and to process pixels two by two.

A great merit in doing this is that the current pixel b can be processed efficiently in cases where the current pixel a in the scan line is a foreground pixel. If pixel b is a background pixel, nothing needs to be done; otherwise, because the existence of the pixel could not connect any two independent foreground-pixel parts in the mask,[1] after processing pixel a, we only need to assign to pixel b pixel a's provisional label (without checking any of the other pixels and considering any label equivalence resolving). For simplicity, in the case where the pixel a is a foreground pixel, we will not mention how to process pixel b hereafter.

The combinations that the two current pixels a and b should be considered are the following three cases (see Fig. 4):

C_1 – both pixel a and pixel b are background pixels;
C_2 – pixel a is a foreground pixel and pixel b is either a foreground pixel or a background pixel;
C_3 – pixel a is a background pixel and pixel b is a foreground pixel.

As we will show later, all configurations of the pixels in the mask and the two current pixels that should be considered are the nine configurations shown in Fig. 5, where a meaningless pixel in the mask means whether it is a foreground pixel or not does not affect the processing result, thus, can be removed from consideration. For example, in the configuration C_c shown in Fig. 5, for processing the two current pixels a and b, we do not need to check pixel n_1. When we go to process the following two current pixels, pixel n_1 will be shifted out of the mask. Therefore, in this case, we do not need to consider n_1, it is a meaningless pixel.

For convenience, hereafter we use a shorthand notation, $\{n_1, n_2, n_3, n_4, n_5, a, b\}$, to denote the configuration consisting of the pixels in the mask and the two current pixels.

[1]According to the analysis given in [12], only when two independent foreground-pixel parts in the mask become connected by a foreground pixel being processed, we need to consider resolving the label equivalence of the two parts.

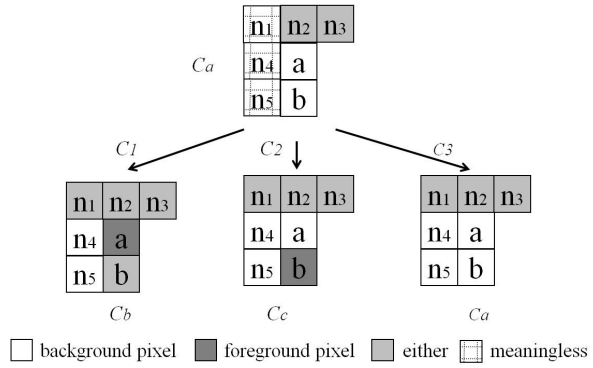


Fig. 6. Transition of the configuration C_a.

Moreover, we use lower-case letters except a and b to denote labels in the mask, # to denote a background pixel, $ to denote a pixel which is either a foreground pixel or a background pixel, and – to denote a meaningless pixel.

For case C_1, we need to do nothing; the configuration is C_a, i.e., {–, $, $, –, –, #, #}, shown in Fig. 5. Because the next two pixels to be processed will be one of the three cases C_1, C_2, and C_3, as shown in Fig. 6, the configuration C_a will transit to one of three configurations {$, $, $, #, #, a, b}, {$, $, $, #, #, #, b}, and {$, $, $, #, #, #, #}, i.e., C_b, C_c, and C_a shown in Fig. 5.

For a case C_2 that follows a case C_1, whose configuration is C_b as shown in Fig. 5, i.e., {$, $, $, #, #, a, b}, we need to consider the pixels n_1, n_2, and n_3. According to the strategy proposed in [9], we will check n_2 first. If n_2 is a foreground pixel, it has been assigned a label u. We assign u to the current pixel a,[2] thus, the configuration becomes

[2]As mentioned above, because all of the foreground pixels in the mask such that they are connected to the current foreground pixel(s) belong to the same connected component, all provisional labels assigned to the foreground pixels will be combined in the same equivalent label set. Therefore, the current foreground pixel(s) can be assigned any provisional label of its foreground-pixel neighbors.

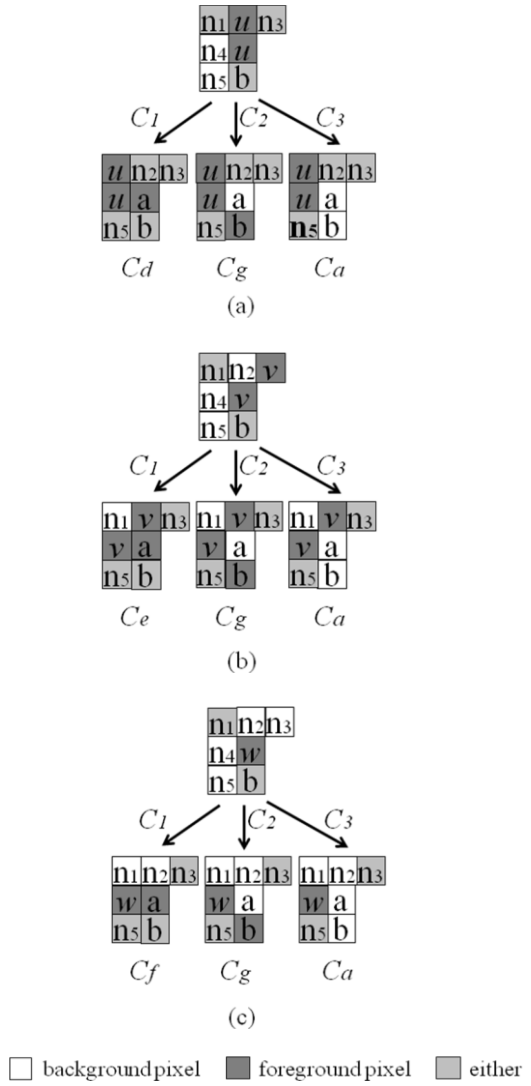


Fig. 7. Transition of the configuration C_b: (a) in the case where n_2 is a foreground pixel u; (b) in the case where n_2 is a background pixel and n_3 is a foreground pixel v; (c) in the case where both n_2 and n_3 are background pixels.

{\$, u, \$, #, #, u, b}. Here, we do not need to check pixel n_1 and pixel n_3, because, if they are foreground pixels, they are known to be eight-connected with pixel n_2 before processing the current two pixels; thus, they should belong to the same equivalent-label set already. Because the next two pixels to be processed will be one of the three cases C_1, C_2, and C_3, the configuration will transit to one of {u, \$, \$, u, \$, a, b}, {u, \$, \$, u, \$, #, b}, and {u, \$, \$, u, \$, #, #} as shown in Fig.7(a), i.e., the configurations C_d, C_g, and C_a shown in Fig. 5, respectively.

Secondly, if pixel n_2 is a background pixel and pixel n_3 is a foreground pixel, then n_3 has been assigned a label v, and we assign the label v to the current pixel a; thus, configuration becomes {\$, #, v, #, #, v, \$}. If pixel n_1 is a foreground pixel, we resolve the label equivalence between the label of pixel n_1 and v. Because the next two pixels to be processed will be one of the three cases C_1, C_2 and C_3, the configuration {\$, #, v, #, #, v, \$} will transit to one of the configurations {#, v, \$, v, \$, a, b}, {#, v, \$, v, \$, #, b} and {#, v, \$, v, \$, #, #} as shown in Fig.7(b); i.e., the configurations C_e, C_g, and C_a shown in Fig. 5, respectively.

Lastly, if both pixel n_2 and pixel n_3 are background pixels, the configuration is {\$, #, #, #, #, a, \$}. We assign to the current pixel a the same provisional label assigned to pixel n_1 if pixel n_1 is a foreground pixel; otherwise, we assign to the pixel a a new provisional label. Let w be the label assigned to pixel a. Because the next two pixels to be processed will also be one of the three cases C_1, C_2, and C_3, this configuration will transit to one of the three configurations {#, #, \$, w, \$, a, b}, {#, #, \$, w, \$, #, b}, and {#, #, \$, w, \$, #, #} as shown in Fig.7(c); i.e., the configurations C_f, C_g, and C_a shown in Fig. 5, respectively.

All other configurations shown in Fig. 5 can be analyzed in a similar way. The configuration transition diagram for our method is shown in Fig. 8, where, for simplicity, all transitions from each configuration to the configuration C_a are omitted.

IV. Comparative Evaluation

We mainly compared our algorithm with the HCS$_I$ algorithm and the block-based algorithm proposed in [4]. For convenience, we denote the block-based algorithm as *BL algorithm*.

An experiment was performed on a Dell Optiplex GX745 Intel(R) Core(TM)2 CPU 6300, 1.86 GHz & 1.86 GHz, 1.99 GB RAM with Microsoft Windows XP Professional Version 2002 Service Pack, using a single core for the processing. All algorithms used for our comparison were implemented in C++ language by use of the OpenCV library and compiled on Windows by use of Visual Studio 2008. The program related to the HCS$_I$ algorithm was provided by its authors, and that of the BL algorithm was downloaded from the authors' providing website [19]. All experimental results presented in this section were obtained by selecting the minimum time over 10 runs in order to remove possible outliers due to other tasks executed by the operating system. Moreover, for any image, the labeling result obtained by any algorithm is exactly the same.

Images used for testing were composed of four types: artificial images, natural images, texture images, and medical images.

Artificial images contain specialized patterns (stair-like, spiral-like, saw-tooth-like, checker-board-like, and honeycomb-like connected components) [9] and noise images. The dataset of noise images was also downloaded from [19]; these consisted of six different image sizes (128×128, 256×256, 512×512, 1024×1024, 2048×2048, and 4096×4096 pixels) with 6 different foreground densities (from 0.1 to 0.9). For every combination of size and density, there are 10 images; thus, there are 540 images in total. Because connected components in such noise images have complicated geometric shapes and complex connectivity, severe evaluations of labeling algorithms can be performed with these images.

On the other hand, 50 natural images, including landscape, aerial, fingerprint, portrait, still-life, snapshot, and text images, obtained from the Standard Image Database (SIDBA) developed by the University of Tokyo [20] and the image database

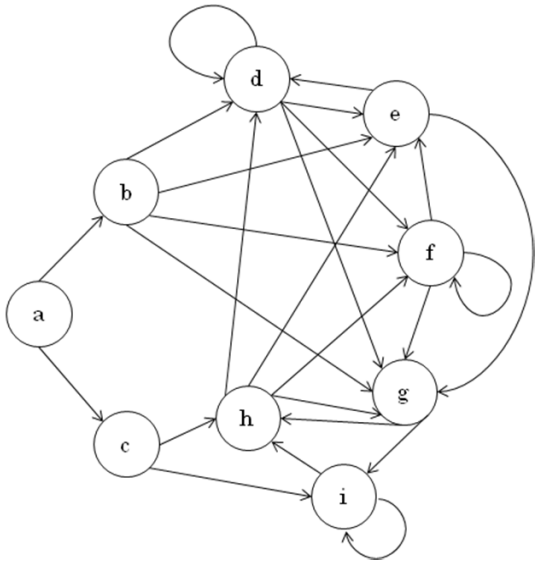


Fig. 8. Configuration transition diagram.

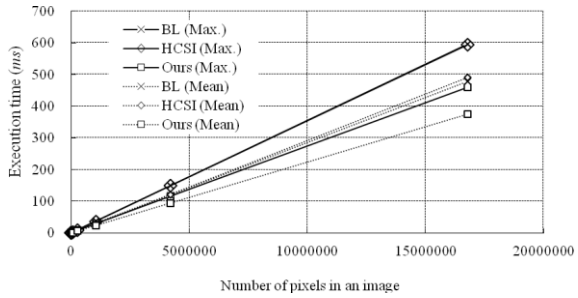


Fig. 9. Execution time (*ms*) versus the number of pixels in an image.

of the University of Southern California [21], were used for realistic testing of labeling algorithms. In addition, seven texture images, which were downloaded from the Columbia-Utrecht Reflectance and Texture Database [22], and 25 medical images obtained from a medical image database of The University of Chicago, were used for testing. All of these images were 512×512 pixels in size, and they were transformed into binary images by means of Otsu's threshold selection method [14].

A. Execution Time Versus the Size of an Image

We used all noise images to test the linearity of the execution time versus image size. The results are shown in Fig. 9. We can see that both the maximum execution time and the average execution time for all of the four algorithms have the ideal linear characteristics versus image size. The maximum execution time of the BL algorithm was almost the same as that of the HCS_1 algorithm, and the execution time of the BL algorithm was a little shorter than that of the HCS_1 algorithm. Moreover, the maximum execution time of our algorithm was even smaller than the minimum execution times of the other two algorithms.

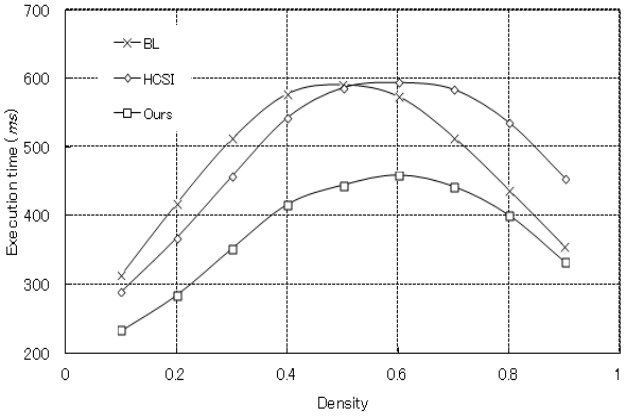


Fig. 10. Execution time (*ms*) versus the density.

B. Execution Time Versus the Density of an Image

Noise images with a size of 4096×4096 pixels were used for testing the execution time versus the density of an image. The results are shown in Fig. 10, where the value for each density is the average execution time on the ten images of that density. The performance of the HCS_1 algorithm was better than that of the BL algorithm for the images with densities lower than 0.5, but worse for the images with densities larger than 0.5. For all density images, our algorithm was faster than the two others.

C. Comparisons in Terms of the Maximum, Average, and Minimum Execution Times

Natural images, medical images, texture images, and artificial images with specialized shape patterns were used for this test.

The maximum, average, and minimum execution times for each kind of image of the algorithms used for comparison are shown in Table 1, where σ indicates the standard deviation. We find that, in most cases, the efficiency of the BL algorithm was better than that of the HCSI algorithm, and that of our algorithm was much better than those of the other two. Moreover, in any case, the standard deviation of our algorithm is the smallest one.

The execution time (in *ms*) for the selected six images is illustrated in Fig. 11, where the foreground pixels are displayed in black, and where the value of D indicates the density of an image.

V. Discussion

A. Comparisons With the HCS_1 Algorithm

For processing a foreground pixel, the HCS algorithm needs to consider four pixels in the mask. Because there are common pixels in the mask of two sequential foreground pixels, a pixel might be checked several times. For example, when processing the pixel f shown in Fig. 10 (thus, pixel f is checked), the HCS algorithm will check the pixels a, b, c, and e. When continuing to process the pixel g, it will check the pixels f, b, c, and d. Thus, the pixels f, b, and c are checked

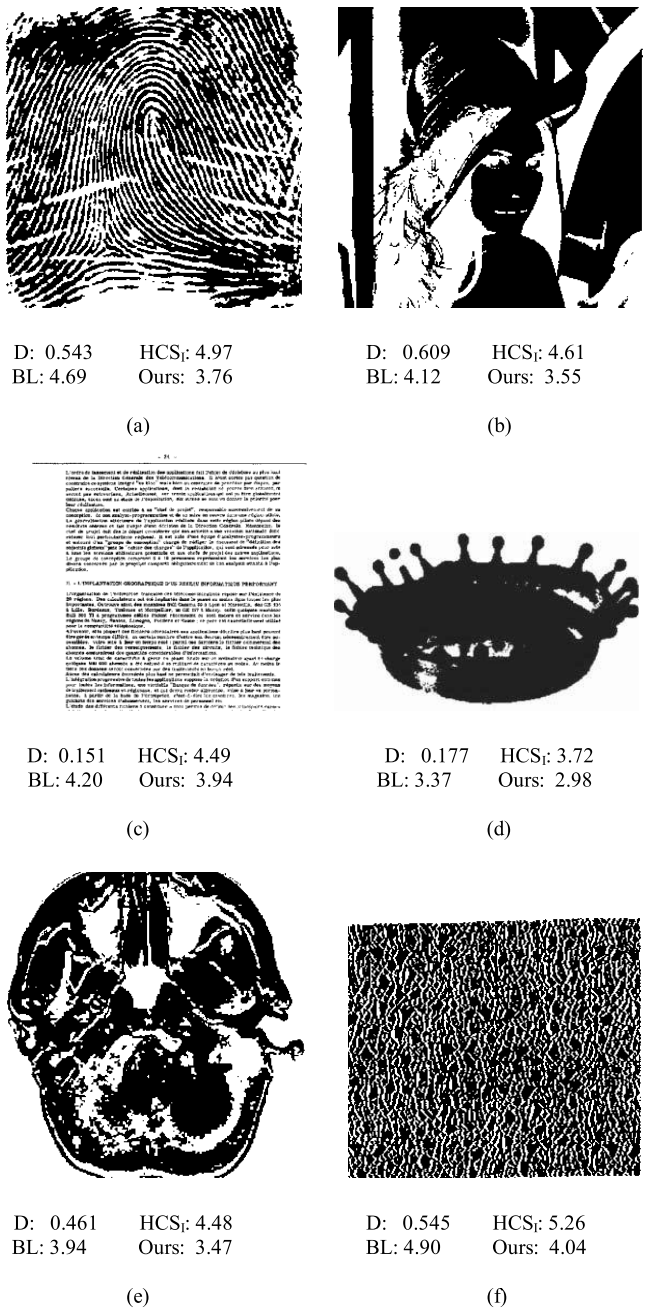


Fig. 11. The density and execution time (*ms*) for the selected six images: (a) a fingerprint image; (b) a portrait image; (c) a text image; (d) a snapshot image; (e) a medical image; (f) a texture image.

repeatedly. Moreover, when processing the pixel j later in the next line, the pixel f will be checked again.

The HCS$_I$ algorithm improved the HCS algorithm by processing foreground pixels following a background pixel, and those foreground pixels following a foreground pixel are processed in a different way. By use of this idea, the pixel followed by the current foreground pixel can be removed from the mask [10]. In other words, for processing the current foreground pixel, the algorithm does not need to check its left neighbor. Thus, for processing a foreground pixel, the average number of times for checking the processed neighbor pixels in the first scan can be reduced. Because whether the current foreground pixel follows a background pixel or a foreground pixel can be known without any additional computational cost, this algorithm is more efficient than the HCS algorithm for all images.

TABLE I
VARIOUS EXECUTION TIMES (IN *ms*) FOR VARIOUS TYPES OF IMAGES

Image Type		HCS$_I$	BL	Ours
Natural	Max.	5.64	5.47	4.30
	Mean	4.48	4.09	3.53
	Min.	3.63	2.98	2.95
	σ	0.79	0.81	0.58
Medical	Max.	5.58	4.41	3.60
	Mean	4.32	3.79	3.36
	Min.	3.89	3.39	3.19
	σ	0.33	0.27	0.11
Textural	Max.	5.41	5.50	4.15
	Mean	5.12	4.61	3.85
	Min.	4.67	4.23	3.54
	σ	0.25	0.31	0.22
Artificial	Max.	6.57	5.72	4.14
	Mean	3.61	3.20	2.39
	Min.	1.20	1.28	0.81
	σ	2.56	2.13	1.63

However, the improvement made by the HCS$_I$ algorithm is not sufficient. For example, when processing the foreground pixel f shown in Fig. 12, it will check the pixels a, b, and c; thus, the pixel c is known as a background pixel. When continuing to process the foreground pixel g, the pixel c will be checked again. In other words, the information that the pixel c is a background pixel obtained during processing the pixel f is not utilized for processing the pixel g, and repeated checking work is done. Moreover, when processing the pixel j later, the same as the HCS algorithm, the HCS$_I$ algorithm will check the pixel f, and the information that pixel f is a foreground pixel is not utilized. This is also a repeated checking.

In comparison, in our algorithm, the information that the pixel c is a foreground pixel obtained during processing the foreground pixel f is used for processing the foreground pixel g, since we just assign the provisional label of the foreground pixel f to the foreground pixel g without checking any pixel. Moreover, in our method, the foreground pixels f and j will be processed simultaneously, where the foreground pixel j is assigned the same provisional label assigned to the pixel f

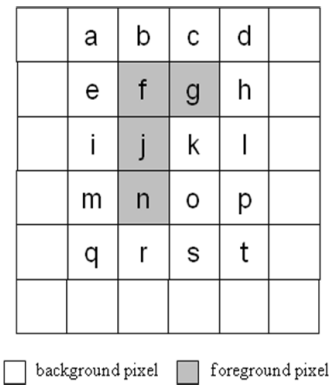
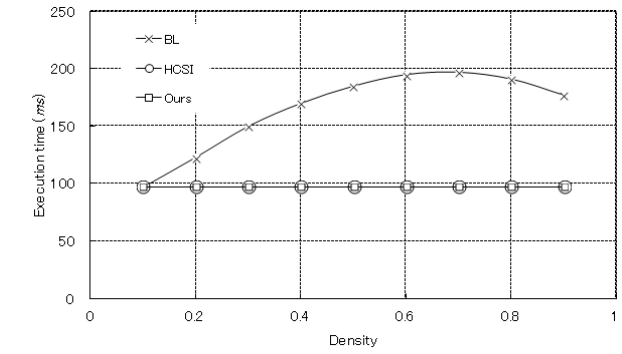
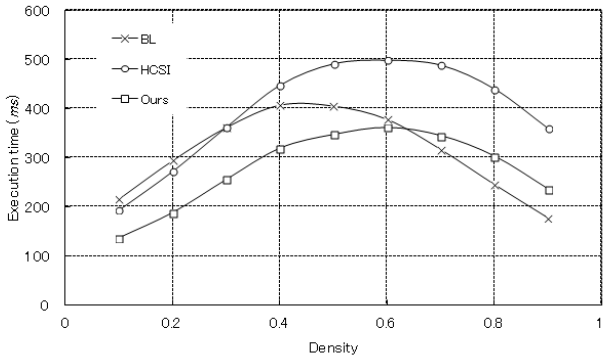


Fig. 12. An example for explaining the problem of the HCS_I algorithm.



(a)



(b)

Fig. 13. Execution time (*ms*) versus the density of foreground pixels in an image: (a) second scan, and (b) first scan.

without checking any pixel. Thus, the repeated checking work made by the HCS_I algorithm is avoided in our algorithm.

Because the second scan made by the HCS_I algorithm and that by our proposed method are exactly the same, the execution time of the second scan of the HCS_I algorithm and that of our proposed algorithm are almost the same for each image, as shown in Fig. 13 (a); thus, the efficiency of our proposed algorithm related to the HCS_I algorithm is achieved in the first scan (see Fig. 13 (b)) by avoiding checking pixels repeatedly.

B. Comparisons With the BL Algorithm

The main idea of the BL algorithm is to consider each 2×2 pixel block as a "super pixel". Because all foreground pixels in a block are certainly 8-connected, and will thus be assigned the same label, the connectivity of the foreground pixels in a block need not be resolved. Instead of assigning to each foreground pixel a provisional label, it assigns to each foreground block[3] a provisional label,[4] and resolves label equivalences among foreground blocks by use of the equivalent-label-set technique proposed in [6], [9]. During the second scan, it assigns to each foreground pixel in a block the representative label of the block. Thus, it can reduce provisional label assignment work in the first scan. Moreover, it can avoid repeatedly checking the pixels in the block.

The main problems with the BL algorithm are: (1) the number of neighbor pixels to be considered for processing a block becomes 12, i.e., the average number of the pixels checked for processing a pixel in the block is $12/4 = 3$, equal to that in the HCS_I algorithm. Moreover, the decision tree generated by this algorithm contains 210 nodes, with 211 leaves sparse over 14 levels. The length of its program source codes provided by the authors is over 1600 lines. Thus, this algorithm has too many rules for a reader to understand and verify. (2) Although this algorithm can avoid repeatedly checking the pixels in the block, similar to the HCS_I algorithm, many pixels will still be checked repeatedly. For example, in the BL algorithm, the pixels d, e, i, j, and k in the mask shown in Fig. 1, which are checked for processing the current block, together with the pixels o, p, and t in the current block might be checked again when the next block is processed. In comparison, our algorithm is much simpler than the BL algorithm, and easy to follow. The length of the codes of our algorithm is less than 600 lines. In our algorithm, when a pixel is checked for processing the current pixels, the information that the pixel is a foreground pixel or a background pixel will be utilized for processing the next pixel. In other words, any pixel in the mask checked in processing the current two pixels will not be checked again when processing the next two pixels. Thus, the repeated work for checking pixels has been reduced as much as possible. For any image used in our experimental test, our algorithm is faster than the BL algorithm.

Moreover, the second scans of both the HCS_I algorithm and our proposed algorithm just replaces the value of each pixel a with its representative value $r[a]$ without checking its value. Therefore, for all images with the same size, the execution times are almost the same (see Fig. 13 (a)). However, when the BL algorithm processes a block in the second scan, it first needs to check the value of the representative label of a block. If the value is 0, it means that there is no foreground pixel in the block, and the BL algorithm assigns 0 to each pixel of the block. Otherwise, it means that there

[3]A *foreground block* is a block such that there is at least one foreground pixel in the block.

[4]In fact, this idea is similar to that proposed in [11], where each run (the maximum set of continuous foreground pixels in a row) is assigned a provisional label.

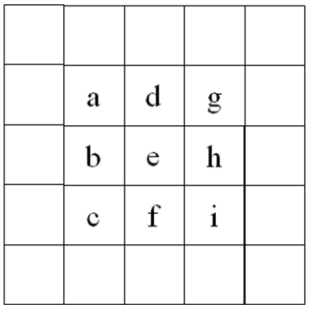


Fig. 14. An example for explaining the application of our strategy.

is at least a foreground pixel in the block. In this case, the BL algorithm checks each pixel in the block. For each foreground pixel in the block, it assigns to the pixel the representative label of the block, and for each background pixel, it assigns 0 to the pixel. Therefore, in the second scan, the BL algorithm usually does more work than the other two algorithms, and the more of foreground block an image is, the more time the BL algorithm will take to process it (see Fig. 13 (a)).

On the other hand, in the first scan, the BL algorithm assigns a provisional label to each foreground block, whereas both the HCS_I algorithm and our proposed algorithm assign a provisional label to each foreground pixel; thus, for high-density images, the BL algorithm will do much less such work than the other two. Therefore, when the density of an image is large than 0.65, the first-scan method of the BL algorithm will be more efficient than that of our proposed algorithm, as shown in Fig. 12 (b), and when the density of an image increases, the efficiency difference between our algorithm and that of the BL algorithm will decrease, as shown in Fig. 10.

In principle, our proposed method can be used for improving the BL algorithm to avoid checking pixels repeatedly; however, it will make the algorithm much more complicated.

C. Application of Our Strategy to Other Moving Window Operations

Our strategy, which reduces computational cost by utilizing information obtained previously for currently processing, can be applied to other moving window operations. For example, by the moving average method for noise reduction, the value of pixel e, say, V_e, in Fig. 14 can be calculated by $(a + b + c + d + e + f + g + h + i)/9$. On the other hand, if we record the information $M_1 = a + b + c$ and $M_2 = d + e + f$ before processing pixel e, when processing pixel e, after calculating $M_3 = g + h + i$, then we can also calculate V_e by $(M_1 + M_2 + M_3)/9$. Obviously, the computational cost of the latter method is smaller than that of the former one.

VI. Conclusion

In this paper, we presented two strategies for improving the first scan of label-equivalence-based two-scan-labeling algorithms. By processing image lines two by two, and considering the transition of configurations of pixels in the mask, we can reduce the repeated work for checking pixels and, therefore, achieve a more efficient processing than conventional labeling algorithms. In future work, we plan to extend our algorithm to include three-dimensional connected component labeling [12], [13], [17].

Acknowledgment

We thank the anonymous referees for their valuable comments that improved this paper greatly. We are grateful to our editor, Prof. Ton Kalker, for his kind cooperation and a lot of valuable advices. We also thank Ms. E. F. Lanzl for proofreading this paper.

References

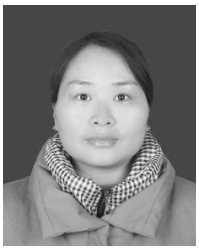
[1] D. H. Ballard, *Computer Vision*. Upper Saddle River, NJ, USA: Prentice-Hall, 1982.
[2] F. Chang, C. J. Chen, and C. J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Comput. Vis. Image Understand.*, vol. 93, no. 2, pp. 206–220, 2004.
[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA, USA: Addison-Wesley, 1992.
[4] C. Grana, D. Borghesani, and R. Cucchiara, "Optimized block-based connected components labeling with decision trees," *IEEE Trans. Image Process.*, vol. 9, no. 6, pp. 1596–1609, Jun. 2010.
[5] R. M. Haralick, "Some neighborhood operations," in *Real Time/Parallel Computing Image Analysis*. New York, NY, USA: Plenum Press, 1981, pp. 11–35.
[6] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. 1. Reading, MA, USA: Addison-Wesley, 1992, pp. 28–48.
[7] L. He, Y. Chao, and K. Suzuki, "A linear-time two-scan labeling algorithm," in *Proc. IEEE ICIP*, Sep. 2007, pp. 241–244.
[8] L. He, Y. Chao, and K. Suzuki, "A run-based two-scan labeling algorithm," *IEEE Trans. Image Process.*, vol. 17, no. 5, pp. 749–756, May 2008.
[9] L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," *Pattern Recognit.*, vol. 42, no. 9, pp. 1977–1987, 2009.
[10] L. He, Y. Chao, and K. Suzuki, "An efficient first-scan method for label-equivalence-based labeling algorithms," *Pattern Recognit. Lett.*, vol. 31, no. 1, pp. 28–35, 2010.
[11] L. He, Y. Chao, and K. Suzuki, "A run-based one-and-a-half-scan connected-component labeling algorithm," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 24, no. 4, pp. 557–579, 2010.
[12] L. He, Y. Chao, and K. Suzuki, "Two efficient label-equivalence-based connected-component labeling algorithms for three-dimensional binary images," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2122–2134, Aug. 2011.
[13] Q. Hu, G. Qian, and W. L. Nowinski, "Fast connected-component labeling in three-dimensional binary images based on iterative recursion," *Comput. Vis. Image Understand.*, vol. 99, no. 3, pp. 414–434, 2005.
[14] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
[15] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, 1970.
[16] Y. Shima, T. Murakami, M. Koga, H. Yashiro, and H. Fujisawa, "A high-speed algorithm for propagation-type labeling based on block sorting of runs in binary images," in *Proc. 10th Int. Conf. Pattern Recognit.*, Jun. 1990, pp. 655–658.
[17] J. K. Udupa and V. G. Ajjanagadde, "Boundary and object labeling in three-dimensional images," *Comput. Vis., Graph., Image Process.*, vol. 51, no. 3, pp. 355–369, 1990.
[18] K. Wu, E. Otoo, and K. Suzuki, "Optimizing two-pass connected-component labeling algorithms," *Pattern Anal. Appl.*, vol. 12, no. 2, pp. 117–135, 2008.
[19] (2012, Aug.). *The BL Algorithm* [Online]. Available: http://imagelab.ing.unimore.it/imagelab/labeling.asp
[20] (2012, Aug.). *The Standard Image Database* [Online]. Available: http://sampl.ece.ohio-state.edu/data/stills/sidba/index.htm

[21] (2012, Aug.). *The Image Database of the University of Southern California* [Online]. Available: http://sipi.usc.edu/database/

[22] (2012, Aug.). *The Columbia-Utrecht Reflectance and Texture Database* [Online]. Available: http://www1.cs.columbia.edu/CAVE/software/curet/index.php

Lifeng He (M'10) received the B.E. degree from the Northwest Institute of Light Industry, China, in 1982, the second B.E. degree from Xian Jiaotong University, China, in 1986, and the M.S. and Ph.D. degrees in AI and computer science from the Nagoya Institute of Technology, Japan, in 1994 and 1997, respectively. He is a Professor with Aichi Prefectural University, Japan, and a Guest Professor with the Shaanxi University of Science and Technology, China. From 2006 to 2007, he was with the University of Chicago, USA, as a Research Associate. His research interests include intelligent image processing, computer vision, automated reasoning, pattern recognition, string searching, and artificial intelligence.

He has been serving as a referee for more than ten journals in computer science fields, including the IEEE TRANSACTIONS ON NEURAL NETWORK, the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON COMPUTERS, *Pattern Recognition, Computer Vision and Image Understanding*, and *Pattern Recognition Letter*. He is a member of IPSJ, IEICE, and AAR.



Xiao Zhao received the B.E. and M.S. degrees from the Shaanxi University of Science and Technology, China, in 2001 and 2006, respectively. From 2001 to 2006, she was an Assistant Professor with the College of Electrical and Information Engineering, Shaanxi University of Science and Technology. Since 2007, she has been a Lecturer. Her research interests include image processing, artificial intelligence, pattern recognition, and string searching.

Yuyan Chao received the B.E. degree from the Northwest Institute of Light Industry, China, in 1984, and the M.S. and Ph.D. degrees from Nagoya University, Japan, in 1997 and 2000, respectively. From 2000 to 2002, she was a Special Foreign Researcher of the Japan Society for the Promotion of Science, Nagoya Institute of Technology. She is a Professor with Nagoya Sangyo University, Japan, and a Guest Professor with the Shaanxi University of Science and Technology, China. Her research interests include image processing, graphic understanding, CAD, pattern recognition, and automated reasoning.

Kenji Suzuki received the Ph.D. degree in information engineering from Nagoya University in 2001. From 1993 to 2001, he was with Hitachi Medical Corporation, and then Aichi Prefectural University as faculty. In 2001, he joined the Department of Radiology, University of Chicago. Since 2006, he has been an Assistant Professor of the Radiology, Medical Physics, and Cancer Research Center. His research interests include computer-aided diagnosis and machine learning in medical imaging. He has published 230 papers (including 95 peer-reviewed journal papers). He has an h-index of 28. He is an inventor on 28 patents that were licensed to several companies and commercialized. He has published nine books and 18 book chapters. He was awarded and co-awarded 44 grants including NIH R01. He served as a referee for 62 international journals, an organizer of 16 international conferences, and a program committee member of 112 international conferences. He has been serving as the Editor-in-Chief and an Associate Editor of 28 leading international journals including *Medical Physics and Academic Radiology*. He received the Paul Hodges Award, the three RSNA Certificate of Merit Awards and Research Trainee Prize, the Cancer Research Foundation Young Investigator Award, the SPIE Honorable Mention Poster Award, the IEEE Outstanding Member Award, and the Kurt Rossmann Excellence in Teaching Award.