A New Two-Scan Algorithm for Labeling Connected Components in Binary Images

Lifeng He, Yuyan Chao, Kenji Suzuki

***Abstract*—This paper proposes a new two-scan algorithm for labeling connected components in binary images. In the first scan of our algorithm, all conventional two-scan labeling algorithms process image lines one by one and process pixels one by one. In comparison, we process image lines two by two and process image pixels two by two. By our algorithm, the average times for checking the neighbor pixels for processing a foreground pixel will decrease, which leads to an efficient labeling processing. Experimental results on various types of images demonstrated that our method is more efficient than conventional label-equivalence-based labeling algorithms.**

***Index Terms*— connected component, labeling, pattern recognition, fast algorithm, computer vision**

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 [1,2]. Especially in real-time applications such as traffic-jam detection, automated surveillance, and target tracking, faster labeling algorithms are always desirable.

Many algorithms have been proposed for addressing this issue, because the improvement of the efficiency of labeling is critical in many applications. For ordinary computer architectures and 2D images, there are mainly two types of labeling algorithms:

(1) Raster-scan algorithms. These algorithms process an image in the raster-scan way. There are multi-scan algorithms [3], [4] the four-scan algorithm [5], two-scan algorithms [6-14], and the one-and-a-half algorithm [15].

(2) Label propagation algorithms. These algorithms access an image in an irregular way. There are run-based algorithms [2], [16] and contour-tracing algorithms [17], [18].

According to experimental results on various types of images, the algorithm proposed in [13], which is an improvement on the two-scan algorithm proposed in [12], is the most efficient one, and has been used for various applications [19]-[21]. For convenience, we denote this algorithm as *HCS* algorithm.

The *HCS* algorithm is a two-scan labeling algorithm. Similar to other two-scan labeling algorithms, it completes labeling in two scans by three processes: (1) provisional label assignment (i.e., assigning a provisional label to each foreground pixel) and equivalent-label finding (i.e., finding the provisional labels assigned to the same connected components); (2) equivalent label record (i.e., using some data structures to record equivalent labels) and label-equivalence resolving (i.e., finding a representative label for all equivalent provisional labels); (3) label replacement (i.e., replacing each provisional label by its representative label).

The *HCS* algorithm uses equivalent label sets and a representative label table to record equivalent labels and resolve the label equivalences. Usually, the smallest label in an equivalent label set is used to the representative label of the set and all labels in the set. For convenience, an equivalent label set with the representative label u is denoted as $S(u)$, and the representative label of a provisional label s is t, denoted as $T[s] = t$.

In the first scan, this algorithm uses the mask shown in Fig. 1 (a), which consists of three scanned neighbor of the current foreground pixels, to assign provisional labels to foreground pixels, and to record and resolve label equivalences. At any moment, all equivalent provisional labels are combined in an equivalent label set with the same representative label.

For the case where the current foreground pixel follows a background pixel (Fig. 1 (b)), if there is no label (foreground pixel) in the mask, it means that the current foreground pixel does not connect with any scanned foreground pixel, and the current foreground pixel belongs to a new connected component in the scanned area. The algorithm assigns a new provisional label m to the current foreground pixel, which is initialized to 1, and establishes the equivalent label set $S(m)=\{m\}$; it sets the representative label table as $T[m] = m$, and $m = m+1$ for later processing. Otherwise, i.e., if there are foreground pixels in the mask, all of such foreground pixels and the current foreground pixel belong to the same connected component. Therefore the current foreground pixel can be assigned any of the labels in the mask. On the other hand, for the case where the current foreground pixel follows another foreground pixel (Fig. 1 (c)), the current foreground pixel can be assigned the same label of that

Manuscript received Feb. 28, 2012. This work was supported in part by the Ministry of Education, Science, Sports and Culture, Japan, Grant-in-Aid for Scientific Research (C), 23500222, 2011.

L. He is with the Shanxi University of Science and Technology, and also with the Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute, Aichi 480-1198, Japan (The corresponding author, Tel: +81-561-1111; Fax: +81-561-1108; e-mail: helifeng@ist.aichi-pu.ac.jp).

Y. Chao is with Graduate School of Environment Management, Nagoya Sangyo University, Aichi 488-8711, Japan.

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).

foreground pixel.

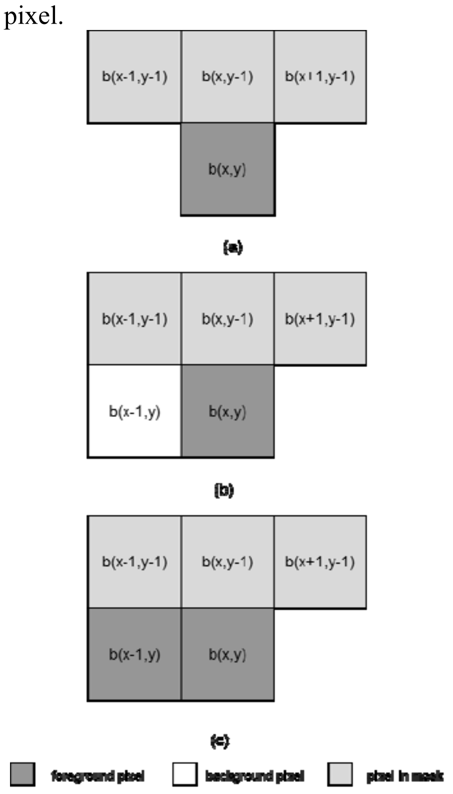

Fig. 1. Mask for the eight-connected connectivity.

In any cases, if there are provisional labels belonging to different equivalent label sets in the mask, all provisional labels in those sets are equivalent labels, and they will be combined together. Suppose that u and v are equivalent labels that belong to $S(T[u])$ and $S(T[v])$, respectively. If $T[u]=T[v]$, the two equivalent label sets are the same, thus, nothing needs to be done. Otherwise, without loss of generality, suppose that $T[u]<T[v]$, i.e., $T[u]$ 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:

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

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.

All conventional two-scan algorithms process image lines one by one. For each foreground pixel, they assign it a provisional label and resolve the connectivity of the pixel and its processed neighbor foreground pixels. To do that, they must check the neighbor pixels of the pixel. It is obvious that the smaller the average number of the checking times for processing a foreground pixel is, the more efficient an algorithm is. However, there are many repeated checking by conventional algorithms. For example, to process the foreground pixel $b(x, y)$ shown in Fig. 2, the *HCS* algorithm will only resolve the connectivity with its processed foreground neighbor pixels $b(x\text{-}1, y\text{-}1)$ and $b(x\text{+}1, y\text{-}1)$ in the lower line of the scan line in the image. The connectivity with its unprocessed foreground neighbor pixels $b(x\text{-}1, y\text{+}1)$ and $b(x, y\text{+}1)$ in the upper line of the scan line will be resolved when processing the next line, where the pixel $b(x, y)$ will be checked again. Such repeatedly checking work can be avoided if, when we process a foreground pixel, we also resolve the connectivity with its unprocessed foreground neighbor pixels.

This paper proposes a new labeling algorithm by processing image lines two by two and processes image pixels tow by two. By our algorithm, the repeatedly checking work mentioned above can be avoided; thus, the average number of checking neighbor pixels for processing a foreground pixel by our algorithm is smaller than that by conventional two-scan labeling algorithms. Experimental results demonstrated that our method is more efficient than conventional label-equivalence-based labeling algorithms.

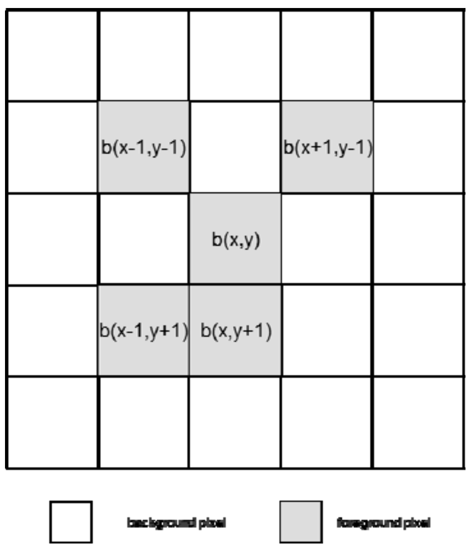

Fig. 2. A simple example for explaining repeatedly checking problem with the *HCS* algorithm.

The rest of this paper is organized as follows: We introduce our algorithm in the next section and show experimental results in section 3. We give our concluding remarks in section 4.

II. The Proposed Algorithm

For an $N\times M$ binary image, we use $b(x, y)$ to denote the pixel at (x, y) in the image, where $1\leq x\leq N$, $1\leq y\leq M$. We also use $b(x, y)$ to denote its value. For convenience, we suppose that the value of foreground pixels is 1 and that of background pixels is 0. All pixels in the edge of an image are considered to be background pixels.

Our proposed algorithm processes image lines two by two and processes pixels two by two, as shown in Fig. 3. For convenience, we call the pixel $b(x, y)$ the *current pixel 1*, and the pixel $b(x, y\text{+}1)$ the *current pixel 2*, respectively.

Similar to the *HCS* algorithm, we process pixels following a foreground pixel and those following a background pixel in different ways.

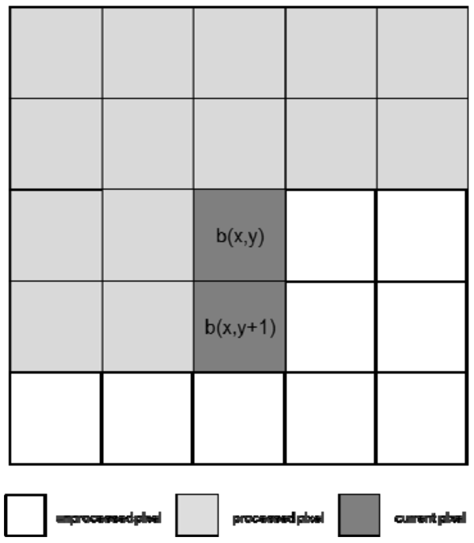

Fig. 3. Processing method in our algorithm.

As shown in Fig. 4, there are four configures for the two current pixels.

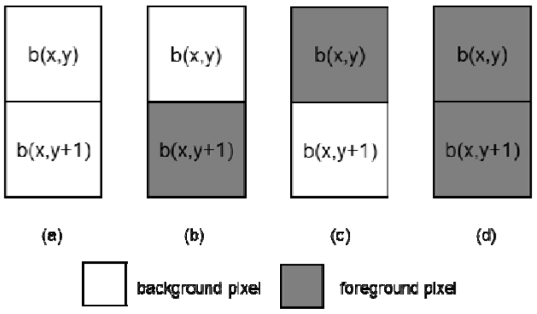

Fig. 4. Four configures of the two current pixels.

If both of the current pixels are background pixel (Fig. 4. (a)), nothing needs to be done.

If the current pixel 1 *b*(*x*, *y*) is foreground pixel (Fig. 4. (c) and (d)), we consider the following two cases:

Case 1. *b*(*x*, *y*) is a foreground pixel following a background pixel. The mask for processing *b*(*x*, *y*) is shown in Fig. 5.

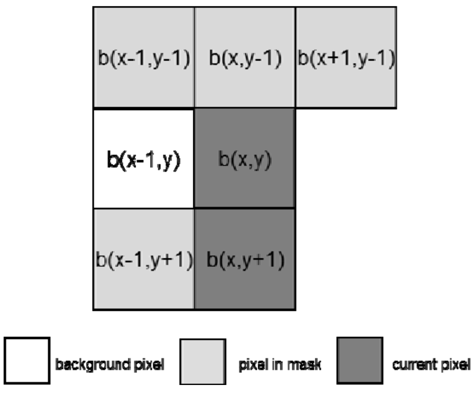

Fig. 5. Mask for processing the current pixel 1 where *b*(*x*-1, *y*) is a background pixel.

In this case, there are 16 configures in the mask, as shown in Fig. 6. Notice that in this case, whether the current pixel 2 *b*(*x*, *y*+1) is a foreground pixel or not does not connect separate block of foreground pixels in the mask, we do not need to consider the connectivity when processing the pixel. Thus, if it is a foreground pixel, we can only assign it the same label of that assigned to the current pixel 1 without checking any other pixels.

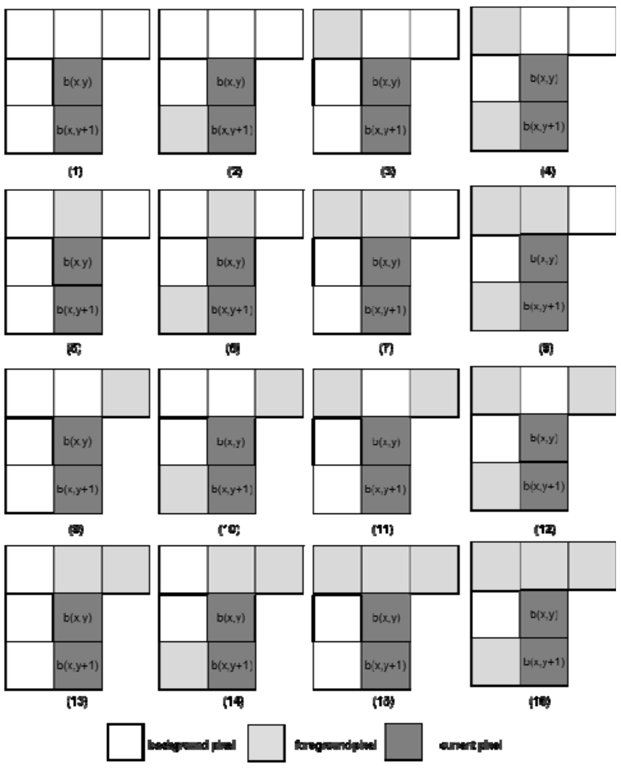

Fig. 6. 16 configures in the mask.

If there is no foreground pixel in the mask (Fig. 6 (1)), we assign a new provisional label to the current pixel 1 *b*(*x*, *y*). Otherwise, i.e., there are foreground pixels in the mask, we can assign the current pixel 1 *b*(*x*, *y*) any label assigned to the foreground pixels[1].

On the other hand, if there is only one block of foregrounds pixel in the mask (Fig. 6 (2), (3), (5), (7), (9), (13) and (15)), no label equivalence needs to be considered. We just need to assign one label among the block to *b*(*x*, *y*). Otherwise, i.e., if there are several separate blocks of foreground pixels in the mask (Fig. 6 (4), (6), (8), (11), (12), (14) and (16)), they are connected by the current pixel 1, and therefore all labels assigned to the pixels of these blocks are equivalent labels. Thus, we need to resolve the label equivalences among the labels if they belong to different equivalent label sets.

Case 2. *b*(*x*, *y*) is a foreground pixel following another foreground pixel, i.e., the pixel *b*(*x*-1, *y*) is a foreground pixel. We assign the *b*(*x*, *y*)'s label to *b*(*x*, *y*). Because all of *b*(*x*, *y*)'s processed neighbor pixels except the pixel *b*(*x*+1, *y*-1) are also *b*(*x*-1, *y*)'s neighbor pixels, the label equivalences among them have been resolved already before processing *b*(*x*, *y*), we do not need to check any of them. The mask for processing the current pixel 1 in this case consists of only one

[1] Because the current pixel 1 *b*(*x*, *y*) is a foreground pixel, all foreground pixels in the mask are connected; thus, all labels assigned to the pixels are equivalent labels. After processing the current pixel 1, all of such labels will be combined in the same equivalent label set.

pixel, as shown in Fig. 7. Only when the pixel $b(x+1, y-1)$ is a foreground pixel, we need consider to resolve the label equivalence of the label assigned to $b(x-1, y)$ and that assigned to $b(x+1, y-1)$.

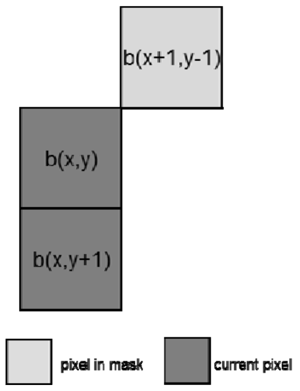

Fig. 7. Mask for processing the current pixel 1 in the case where $b(x-1, y)$ is a foreground pixel.

Moreover, with the same reason described in Case 1, if the current pixel 2 is a foreground pixel, we only need to assign it the same label of the current pixel 1 without checking any of its neighbor pixels.

If the current pixel 1 $b(x, y)$ is background pixel and the current pixel 2 $b(x, y+1)$ is a foreground pixel (Fig. 4. (b)), the mask for processing $b(x, y+1)$ is shown in Fig. 8.

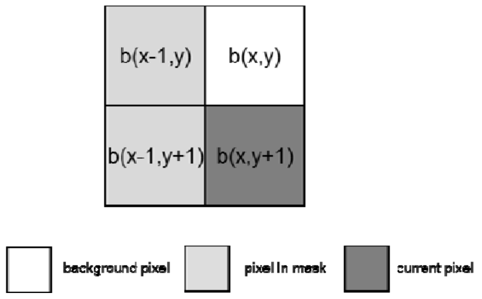

Fig. 8. Mask for processing the current pixel 2 in the case where $b(x, y)$ is a background pixel.

There are four configures of pixels in the mask, shown as in Fig. 9.

In any configure, it is obviously that no separate block of foreground pixel in the mask will be connected by the current pixel 2, therefore, no label equivalence need to be considered.

If there is no foreground pixel in the mask (Fig. 9 (a)), we assign a new provisional label to the current pixel 2 $b(x, y+1)$. Otherwise, if one or both of $b(x-1, y)$ and $b(x-1, y+1)$ are foreground pixels (Fig. 9 (b), (c) and (d)), we assign any label in the mask to the current pixel 2 $b(x, y+1)$.

As soon as the first scan is finished, all provisional labels assigned to each connected component have been combined in an equivalent label set with a unique representative label. During the second scan, similar to all conventional two-scan labeling algorithms, by replacing each provisional label with its representative label, we can complete labeling.

III. Comparative Evaluation

We implemented the *HCS* algorithm and our algorithm with the C language on a PC-based workstation (Intel Pentium D 3.0 GHz + 3.0 GHz CPUs, 2 GB Memory, Mandriva Linux OS). Because our method is an improvement on the first scan (as we described above, the second scan of our method is exactly the same with the *HCS* method), we will only compare the performances of the two methods on the first scan. All data in this section were obtained by averaging of the execution time for 10,000 runs with a single core.

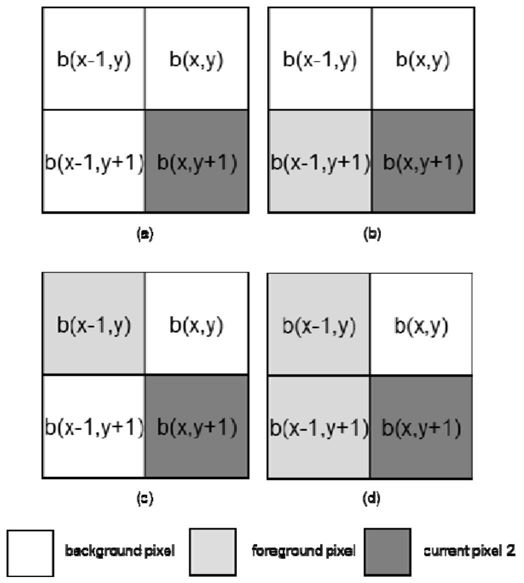

Fig. 9. Four configures of the pixels in the mask for processing the current pixel 2.

Images used for testing included of four types: noise images, natural images, texture images, and medical images.

Noise images consist of forty one 512×512-sized noise images were generated by thresholding of the images containing uniform random noise with 41 different threshold values from 0 to 1000 in steps of 25.

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[2]and the image database of the University of Southern California[3], were used for realistic testing of labeling algorithms. In addition, seven texture images, which were downloaded from the Columbia-Utrecht Reflectance and Texture Database[4], 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 [22].

A. Experimental Results on Noise Images

Because connected components in such noise images have complicated geometric shapes and complex connectivity, as shown in Fig. 10, severe evaluations of labeling algorithms

[2] http://sampl.ece.ohio-state.edu/data/stills/sidba/index.htm
[3] http://sipi.usc.edu/database/
[4] http://www1.cs.columbia.edu/CAVE/software/curet/index.php

can be performed with these images.

The speed-up of our algorithm on the HCS algorithm, which is defined as $(t_1-t_2)/t_1$, where t_1 is the execution time of the HCS algorithm and t_2 is the execution time of our algorithm, versus the density of an image is shown in Fig. 11. We can find that our algorithm is faster than the *HCS* algorithm for all noise images, and the largest speed-up is approximate to 16%.

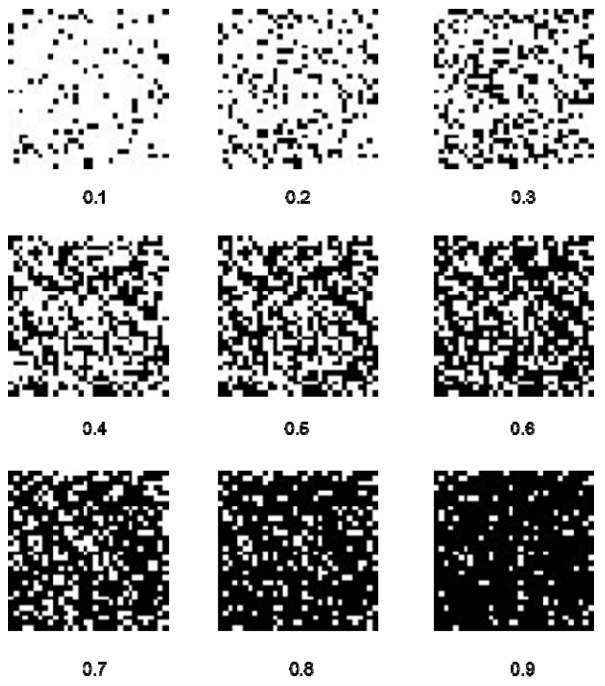

Fig. 10. Samples of noise images with different densities.

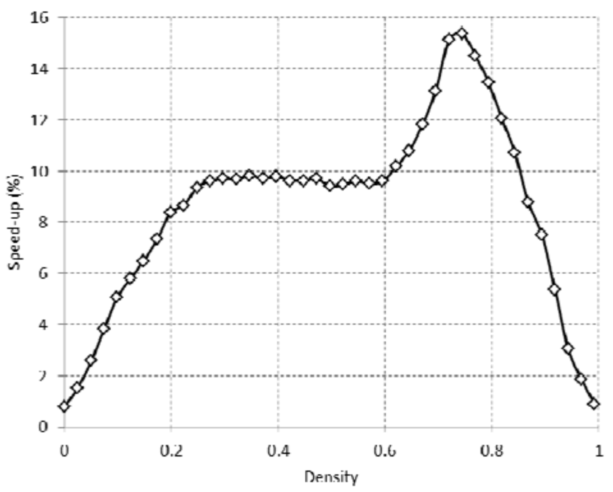

Fig. 11. The speed up of our algorithm on the HCS algorithm versus the density of an image.

The average numbers of times for checking neighbor pixels for processing a foreground pixel in an image of the HCS algorithm and our algorithm are shown in Fig. 12. We can find that for all images, the average numbers of checking times of our algorithm is about 0.5 times smaller than that of the *HCS* algorithm.

B. Experimental Results on Natural Images, Medical Images and Textural Images

The experimental results on the natural images, the medical images, and the textural images are shown in Table I. The execution times of two algorithms on some real images are shown in Fig. 13. We can find that for all items the execution time of our algorithm is smaller than that of the HCS algorithm.

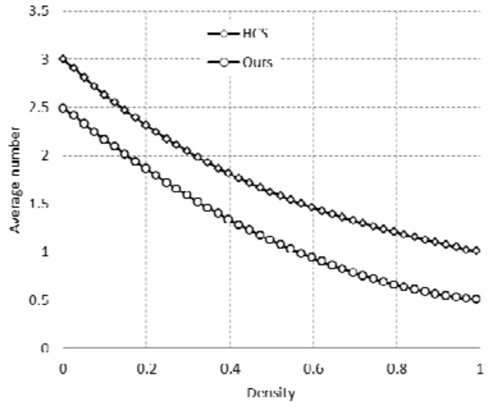

Fig. 12. The average number of times for checking neighbor pixels for processing a foreground pixel in an image.

TABLE I.
COMPARISON OF VARIOUS EXECUTION TIMES [*MSEC*] FOR NATURAL IMAGES, MEDICAL IMAGES, AND TEXTURAL IMAGES.

Image type		The HCS algorithm	Our proposed algorithm
natural	Max.	1.83	1.63
	Mean	0.96	0.87
	Min.	0.44	0.44
medical	Max.	0.98	0.93
	Mean	0.73	0.68
	Min.	0.59	0.54
textural	Max.	1.48	1.41
	Mean	1.09	0.98
	Min.	0.79	0.52

IV. CONCLUSION

In this paper, we presented a new first scan method for two-scan labeling algorithms. Our algorithm scans image lines two by two and processes pixels two by two. By our method, the average number of times for checking neighbor pixels for processing foreground pixels will decrease, which leads efficiently labeling. The experimental results demonstrated that our method was more efficient than conventional scan labeling algorithms' first scan.

There are many works remained for us to do in the future. For example, extending our method for labeling 3D binary images [14], [23], developing, an algorithm for parallel architectures [24], and implementing our algorithm by hardware [25].

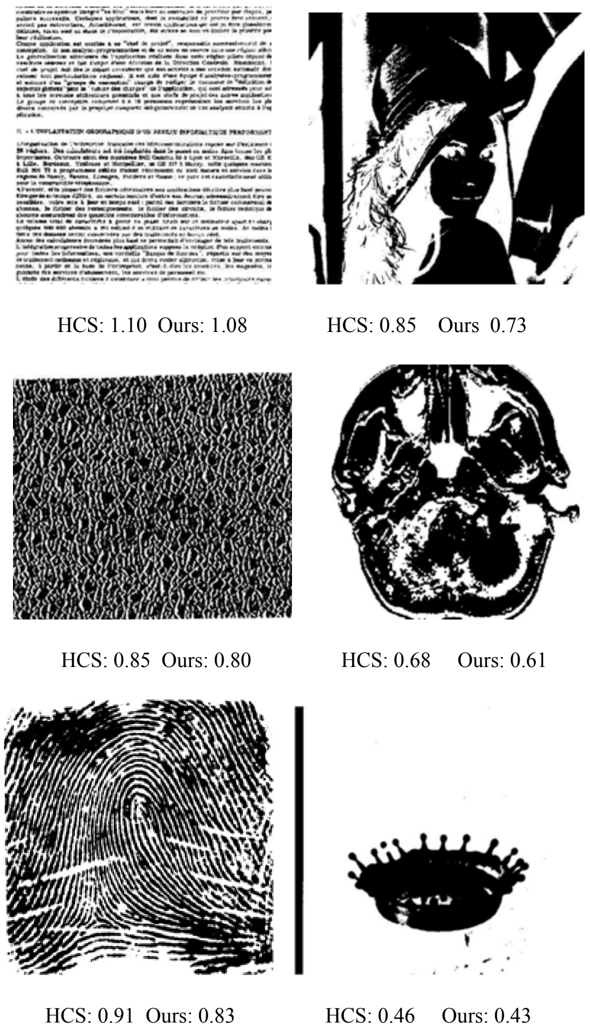

Fig. 13. Execution time (*msec*) on some real images.

REFERENCES

[1] D.H. Ballard. Computer Vision. Englewood, New Jesey: Prentice-Hall, 1982.

[2] R.C. Gonzalez and R.E. Woods. Digital Image Processing. Addison Wesley, 1992.

[3] R.M. Haralick. Some neighborhood operations. In Real Time/Parallel Computing Image Analysis, 11-35, New York, 1981. Plenum Press.

[4] A. Hashizume, R. Suzuki, H. Yokouchi, et al. An algorithm of automated {RBC} classification and its evaluation. Bio Medical Engineering, 28(1):25-32, 1990.

[5] K. Suzuki, I. Horiba, and N. Sugie. Linear-time connected-component labeling based on sequential local operations. Computer Vision and Image Understanding, 89:1-23, 2003.

[6] A. Rosenfeld and J.L. Pfalts. Sequential operations in digital picture processing. Journal of ACM, 13(4):471-494, 1966.

[7] R. Lumia, L. Shapiro, and O. Zungia. A new connected components algorithm for virtual memory computers. Computer Vision, Graphics, and Image Processing, 22(2):287-300, 1983.

[8] R.M. Haralick and L.G. Shapiro. Computer and Robot Vision I, 28-48. Addison-Wesley, Reading, MA, 1992.

[9] S. Naoi. High-speed labeling method using adaptive variable window size for character shape feature. In IEEE Asian Conf. Computer Vision, volume 1, 408-411, 1995.

[10] L. He, Y. Chao, and K. Suzuki. A Linear-Time Two-Scan Labeling Algorithm. 2007 IEEE International Conference on Image Processing (ICIP), pp.V-241-V-244, 2007, San Antonio, Texas, USA.

[11] L. He, Y. Chao, and K. Suzuki. A Run-based Two-Scan Labeling Algorithm. IEEE Transactions on Image Processing, 17(5):749-756, 2008.

[12] L. He, Y. Chao, and K. Suzuki, K. Wu. Fast Connected-Component Labeling. Pattern Recognition, 42:1977-1987, 2009.

[13] L. He, Y. Chao, and K. Suzuki. An Efficient First-Scan Method for Label-Equivalence-Based Labeling Algorithms. Pattern Recognition Letters, 31:28-35, 2010.

[14] L. He, Y. Chao, and K. Suzuki. Two Efficient Label-Equivalence-Based Connected-Component Labeling Algorithms for Three-Dimensional Binary Images. IEEE Transactions on Image Processing, . vol.20, no.8, pp.2122-2134, 2011.

[15] L. He, Y. Chao, and K. Suzuki. A Run-Based One-and-a-Half-Scan Connected-Component Labeling Algorithm. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 24, No. 4, pp.557-579, 2011.

[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. Proc. 10th Int. Conf. Pattern Recognition, pages 655-658, 1990.

[17] F. Chang, C.J. Chen, and C.J. Lu. A linear-time component-labeling algorithm using contour tracing technique. Computer Vision and Image Understanding, 93:206--220, 2004.

[18] Q. Hu, G. Qian and W.L. Nowinski. Fast connected-component labeling in three-dimensional binary images based on iterative recursion. Computer Vision and Image Understanding, 99:414-434, 2005.

[19] A. Alexey, K. Tomas, W. Florentin, and D. Babette. Real-Time Image Segmentation on a GPU. Facing the Multicore-Challenge, Lecture Notes in Computer Science, 6310:131-142, 2011.

[20] Christopher Wolfe, T. C. Nicholas Graham, and Joseph A. Pape. Seeing through the fog: an algorithm for fast and accurate touch detection in optical tabletop surfaces. In ACM International Conference on Interactive Tabletops and Surfaces (ITS '10)}. ACM, 73-82, 2010, New York, NY, USA.

[21] B. Dellen, E.A. Erdal, and F. Wrgtter. Segment Tracking via a Spatiotemporal Linking Process including Feedback Stabilization in an n-D Lattice Model. Sensors, 9(11):9355-9379, 2009.

[22] N. Otsu. A threshold selection method from gray-level histograms. IEEE Trans. Systems Man and Cybernetics, 9:62-66, 1979.

[23] J.K. Udupa and V.G. Ajjanagadde. Boundary and object labeling in three-dimensional images. Computer Vision, Graphics, and Image Processing, 51(3): 355-369, 1990.

[24] K.B. Wang, T.L. Chia, Z. Chen, Parallel execution of a connected component labeling operation on a linear array architecture, J. Inf. Sci. Eng. 19, 353–370, 2003.

[25] X. D. Yang. Design of fast connected components hardware, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Ann Arbor MI, June 1988, pp. 937–944.