A LINEAR-TIME TWO-SCAN LABELING ALGORITHM*

Lifeng He[1,3]*, Yuyan Chao*[2]*, and Kenji Suzuki*[3]

[1]Aichi Prefectural University, Nagakute, Aichi 480-1198, Japan
Also with Shaanxi University of Science and Technology, China.
[2]Nagoya Sangyo University, Aichi 488-8711, Japan
Also with Shaanxi University of Science and Technology, China.
[3]Department of Radiology, Division of the Biological Sciences
The University of Chicago, Chicago, IL 60637, USA

ABSTRACT

This paper presents a fast linear-time two-scan algorithm for labeling connected components in binary images. In the first scan, provisional labels are assigned to object pixels in the same way as do most conventional labeling algorithms. To improve efficiency, we use corresponding equivalent label sets and a representative label table for resolving label equivalences. When the first scan is finished, all provisional labels belonging to each connected component in a given image are combined in the corresponding equivalent label set, and they are assigned a unique representative label with the representative label table. During the second scan, by use of the completed representative label table, all provisional labels belonging to each connected component are replaced by their representative label. Our algorithm is very simple in principle, and is easy to implement. Experimental results demonstrated that the efficiency of our algorithm is superior to that of other labeling algorithms.

Index Terms— Image labeling algorithm, image processing, image pattern recognition, connected components, label equivalence

1. INTRODUCTION

Labeling connected components in a binary image is one of the most fundamental operations in pattern recognition and computer vision. Labeling is said to be more time-consuming than any other fundamental operations such as noise reduction, interpolation, thresholding, and edge detection; therefore, it is a "bottleneck" in the entire processing procedure.

Many algorithms have been proposed for addressing this issue. For ordinary computer architectures, there are the following four classes of algorithms: (1) Multi-scan algorithms: Algorithms [1, 2] scan an image in forward and backward raster directions alternately to propagate the label equivalences until no label changes. (2) Two-scan algorithms: Algorithms [3, 4, 5] complete labeling in two scans: during the first scan, provisional labels are assigned to object pixels, and the label equivalences are stored in a one-dimensional (1D) or two-dimensional (2D) table array. After the first scan or during the second scan, the label equivalences are resolved by use of some algorithms for resolving label equivalences. The resolved results are generally stored in a 1D table. During the second scan, the provisional labels are replaced by the smallest equivalent label by use of the table. (3) Hybrid algorithm [6]: Like multi-scan algorithms, this algorithm scans an image in forward and backward raster directions alternately. During the scans, as in two-scan algorithms, a 1D table is used for recording and resolving label equivalences. Experiments showed that four-scan was efficient for completed labeling. (4) Contour-tracing algorithm [7]: This algorithm avoids analysis of label equivalences by tracing the contours of objects (connected components).

In this paper, we propose a fast linear-time two-scan algorithm for labeling connected components in binary images by use of equivalent label sets and a representative label table for solving label equivalences. Our algorithm is simpler than all conventional algorithms. By carrying out experiments, we demonstrated the efficiency of our algorithm compared with conventional labeling algorithms.

2. OUTLINE OF OUR PROPOSED ALGORITHM

2.1. First Scan

For an $N \times N$ binary image, we use $b(x, y)$ to denote the pixel value at (x, y) in the image, V_O for the pixel value for objects (*object pixels*) and V_B for that for the background (*background pixels*). We assume that V_O and V_B are larger than any possible number of provisional labels, and $V_O < V_B$

Like most conventional algorithms, by use of the mask [8] shown in Fig. 1, our algorithm scans a given image in the raster scan direction once by performing the following se-

*THIS WORK WAS PARTIALLY SUPPORTED BY THE RESEARCH ABROAD PROJECT OF AICHI PROFECTURAL UNIVERSITY, JAPAN.

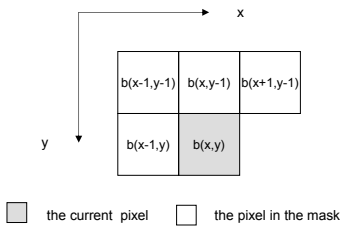

Fig. 1. The mask for the eight-connected connectivity

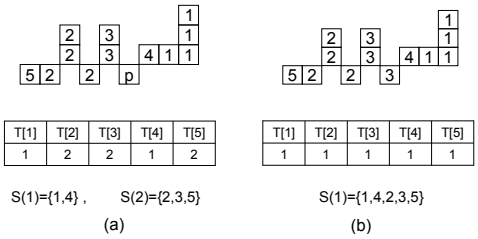

Fig. 2. (a) Before, (b) after processing of pixel p.

quential local operations for each current pixel $b(x, y)$:

if $(b(x, y) = V_B)$	non operation
else if $(b_{\min}(x, y) = V_B)$	$b(x, y) = m, m = m + 1$
otherwise	$b(x, y) = b_{\min(x,y)}$

$b_{\min}(x, y) = \min[\{b(i, j) \mid \forall b(i, j) \in M_S\}]$,

where m (which is initialized to 1) is a provisional label, (m=m+1) is an increment of m, $min(\cdot)$ is an operator for calculating the minimum value, and M_S is the region of the mask, i.e., $b(x$-1,y-1), $b(x, y$-1), $b(x$+1,y-1) and $b(x$-1,$y)$.

2.2. Label Equivalence Resolving

In our algorithm, *equivalent label sets* and a *representative label table* are used for resolving label equivalences.

Suppose that provisional labels $l_1, \cdots, l_n$ found so far in the first scan belong to a certain connected component. We call the provisional labels $l_1, \cdots, l_n$ *equivalent labels*, and the set $\{l_1, \ldots, l_n\}$ an *equivalent label set*.

The smallest label i in an equivalent label set is called the *representative label* of the set. For convenience, we denote such an equivalent label set by $\mathcal{S}(i)$. We also refer the smallest label i to the representative label of all labels in $\mathcal{S}(i)$. For example, for an equivalent label set $\{2, 5, 4, 3\}$, the representative label of the set is the smallest label, 2; thus, the set is denoted by $\mathcal{S}(2)$=$\{2, 5, 4, 3\}$, and 2 is the representative label for all labels in $\mathcal{S}(2)$.

The information that the representative label of a provisional label j is i is recorded in the *representative label table* $\mathcal{T}$ by use of $\mathcal{T}[j]$=i. For example, if there is an equivalent label set $\mathcal{S}(2) = \{2, 5, 4, 3\}$, then $\mathcal{T}[k]$=2 for all $k \in \{2, 5, 4, 3\}$.

When the current object pixel in the mask is assigned a new provisional label i, only the current object pixel is known to belong to the connected component found there because there is no other object pixel in the mask. Therefore, we create a new equivalent label set for this one-pixel connected component as $\mathcal{S}(i)$=$\{i\}$ and initialize the representative table for the corresponding label i as $\mathcal{T}[i]$=i.

On the other hand, if the current pixel is assigned an existing provisional label j, where we assume that label j's representative label is u, i.e., $\mathcal{T}[j]$=u (then $j \in \mathcal{S}(u)$), and we further assume that there are some object pixels in the mask, then, the current pixel and these object pixels belong to the same connected component, and they are equivalent labels. For each of such object pixels in the mask with a provisional label k, where $\mathcal{T}[k]$=v ($k \in \mathcal{S}[v]$), there are three cases: (a) j=k. This means that they are marked with the same provisional label, and obviously nothing needs to be done. (b) $\mathcal{T}[j]$=$\mathcal{T}[k]$, i.e., u=v. This means that labels j and k have the same representative label, and thus, they already belong to the same equivalent label set. That is, the label equivalence between labels j and k has been resolved before; therefore, nothing further needs to be done. (c) $\mathcal{T}[j] \neq \mathcal{T}[k]$, i.e., $u \neq v$. This means that labels j and k belong to different equivalent label sets, i.e., $\mathcal{S}(u)$ and $\mathcal{S}(v)$, respectively; therefore, the label equivalence of labels j and k should be resolved.

In our algorithm, the label equivalence in case (c) is resolved easily. Because labels j and k belong to the same connected component, all provisional labels in $\mathcal{S}(u)$ and $\mathcal{S}(v)$ should be equivalent. Therefore, the equivalent label sets $\mathcal{S}(u)$ and $\mathcal{S}(v)$ should be combined into a single equivalent label set. According to the definition of the representative label, if $u < v$, $\mathcal{S}(v)$ is combined into $\mathcal{S}(u)$, i.e., $\mathcal{S}(u)$=$\mathcal{S}(u)\cup\mathcal{S}(v)$. Because all labels in $\mathcal{S}(u)$ should have the representative label u, for every label l in the equivalent label set $\mathcal{S}(v)$, its representative label should be changed from v to u. This procedure can be made simply by setting $\mathcal{T}[l]$ equal to u. On the other hand, if $u > v$, then S_u is combined into $\mathcal{S}(v)$, i.e., $\mathcal{S}(v)$=$\mathcal{S}(u) \cup \mathcal{S}(v)$, and for all $l \in \mathcal{S}(u)$, set $\mathcal{T}[l]$ equal to v.

For example, in Fig. 2 (a), at the time before processing the pixel p, there are two equivalent label sets, i.e., $\mathcal{S}(1)$=$\{1, 4\}$ and $\mathcal{S}(2)$=$\{2, 3, 5\}$, and $\mathcal{T}[1]$=1, $\mathcal{T}[4]$=1, $\mathcal{T}[2]$=2, $\mathcal{T}[3]$=2, and $\mathcal{T}[5]$=2. When we process pixel p, p is assigned label 3, and we find that there is a label equivalence between label 3 and label 4. Because $\mathcal{T}[3]$=2 and $\mathcal{T}[4]$=1, label 3 and label 4 belong to different equivalent label sets, i.e., $\mathcal{S}(2)$ and $\mathcal{S}(1)$, respectively. Therefore, we should combine $\mathcal{S}(2)$ with $\mathcal{S}(1)$. As a result, we have $\mathcal{S}(1)$=$\{1, 2, 3, 4, 5\}$, and $\mathcal{T}[2]$, $\mathcal{T}[3]$ and $\mathcal{T}[5]$ are set to 1, as shown in Fig. 2 (b).

2.3. Second Scan

Because any label equivalence is resolved immediately whenever it is found, all label equivalences should be resolved as soon as the first scan is finished. Thus, after the first scan, all provisional labels belonging to each connected component in a given image have a unique representative label. Therefore, for each object pixel $b(x, y)$, in the second scan, we merely need to replace the provisional labels with their representative label. If we set $\mathcal{T}[V_B]$=V_B in advance, this process can

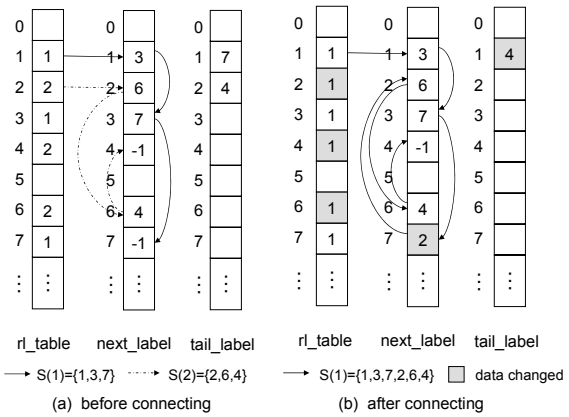

Fig. 3. Operations for label equivalence resolving

be completed easily by the following operation:

$$b(x,y)=\mathcal{T}[b(x,y)]\ .$$

3. IMPLEMENTATION OF OUR ALGORITHM

In our algorithm, for assigning the current object pixel a provisional label, instead of taking the minimum provisional label in the mask, we can take an arbitrary provisional label in the mask, if any, because every label in the mask will have the same representative label after label equivalences (if any) in the mask are resolved. This simplifies the labeling task by avoiding the operation for calculating the minimum label in the mask. For label equivalence resolving, we need to consider only the case where two different equivalent label sets are connected on the current object pixel.

Since every provisional label belongs to only one equivalent label set, we can use two V_B-sized 1D arrays to realize connection lists for equivalent label sets. One array, denoted $n\_label[\ ]$, is used to represent the label next to the previous label, e.g., $n\_label[i]$=j means that the label next to label i is j. In particular, we use $n\_label[i]$=-1 for indicating that label i is the tail label, i.e., there is no label next to label i. The other array, denoted $t\_label[\ ]$, is used for indicating the tail label of an equivalent label set, e.g., $t\_label[u]$=v means that the tail label of $\mathcal{S}(u)$ is v.

Creation of a new equivalent label set $\mathcal{S}(m)$=$\{m\}$ can be made by the following operations:

$rl\_table[m]=m, n\_label[m]=-1, t\_label[m]=m.$

On the other hand, to resolve the label equivalence between provisional labels x and y that belong to different provisional equivalent label sets, e.g., $\mathcal{S}(u)$ and $\mathcal{S}(v)$, where u=$rl\_table[x]$ and v=$rl\_table[y]$, without loss of generality, suppose that $u < v$, i.e., $\mathcal{S}(v)$ is combined into $\mathcal{S}(u)$, we should connect the head of $\mathcal{S}(v)$ to the tail of $\mathcal{S}(u)$, set the tail of $\mathcal{S}(u)$ as the tail of $\mathcal{S}(v)$, and set u as the representative label for every label in $\mathcal{S}(v)$. To do this, the following simple operations are performed:

$u=rl\_table[x], v=rl\_table[y];$
$i=v;$
while $i \neq -1$ do
  $rl\_table[i]=u;$
  $i=n\_label[i];$
end while
$n\_label[t\_label[u]]=v;$
$t\_label[u]=t\_label[v].$

For example, $rl\_table[\ ]$, $n\_label[\ ]$ and $t\_label[\ ]$ for the equivalent label sets $\mathcal{S}(1)$=$\{1,3,7\}$ and $\mathcal{S}(2)$=$\{2,6,4\}$ are as shown in Fig. 3 (a). If provisional labels 6 and 3 are equivalent labels, then the results of the above operations are as shown in Fig. 3 (b).

4. EXPERIMENTAL RESULTS

We implemented our algorithm with the C language on a PC-based workstation (Intel Pentium D 3.0GHz + 3.0GHz CPUs, 2GB Memory, Mandriva Linux OS). All data were obtained by averaging of the execution time for 5,000 runs.

Forty-one noise images of each of five sizes (32×32, 64×64, 128×128, 256×256 and 512×512 pixels) were used for testing (a total of 205 images). For each size, the 41 noise images were generated by thresholding the images containing uniform random noise with 41 different threshold values from 0 to 1,000 with a step of 25. Because connected components in such noise images have a complicated geometrical shape and complex connectivity, severe evaluation of labeling algorithms can be performed with these images.

On the other hand, 50 natural images, including landscape images, aerial images, fingerprint images, portrait images, and text images, obtained from the Standard Image Database (SIDBA) and the image database of the University of Southern California were used for realistic testing of labeling algorithms. In addition, seven texture images and 25 medical images were used for testing. All of these images were 512×512 pixels in size, and were transformed into binary images by use of Otsu's threshold selection method [9].

Because it was reported in [6] that the hybrid algorithm is superior to other conventional multi-scan and two-scan labeling algorithms, and the contour-tracing algorithm is the most recent algorithm, we mainly compared our algorithm with these two algorithms. The program of the hybrid algorithm was provided by the authors and that of the contour-tracing algorithm was downloaded from the authors' web site at http://dar.iis.sinica.deu.tw/Download.

We use all noise images to test linearity of the execution time versus image size. As shown in Fig. 4, all of the three algorithms have linear characteristics. The maximum running time of our algorithm was smaller than the average time of the hybrid algorithm and the contour-tracing algorithm; thus, our algorithm is the fastest.

We tested the three algorithms on all realistic images. The results, shown in Table 1, also demonstrated that our algorithm is the fastest algorithm for all images.

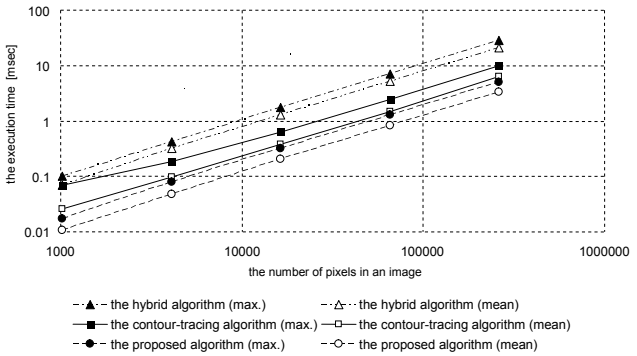

Fig. 4. Linearity of the execution time versus image size

Table 1. Comparison of the execution time [*msec*]

Image type		Hybrid	Contour	Ours
natural	max.	18.4	3.8	2.3
	mean	13.7	2.4	1.5
	min.	9.6	1.2	1.0
medical	max.	14.9	2.6	1.5
	mean	13.4	1.9	1.2
	min.	11.4	1.5	0.9
textural	max.	28.5	3.7	2.0
	mean	27.4	2.7	1.6
	min.	26.4	1.5	1.1

We also compared the execution time of our algorithm with that of other fundamental image-processing methods: thresholding, edge detection, and noise reduction. We used Otsu's threshold selection method [9] for thresholding, Marr-Hildreth edge detector [10] for edge detection, and a median filter with a kernel size of three by three pixels (the smallest one) [8] for noise reduction. The execution times for all realistic images are shown in Table 2. The results illustrate that our algorithm takes the least time among all fundamental image-processing methods.

Although the basic operations of our algorithm can be realized by the union-find algorithm [11], as reported in [7], the union-find algorithm based labeling algorithm [12] was slower than the contour-tracing algorithm. By using the simple data structure and combining two equivalent label sets as soon as they are found to be equivalent, our algorithm achieved a much better performance.

5. CONCLUDING REMARKS

We presented a fast two-scan labeling algorithm. Our strategy for solving label equivalences is much simpler than conventional two-scan algorithms. The experimental results demonstrated that our algorithm is superior to other labeling algorithms. By use of our algorithm, the bottleneck problem with labeling would be resolved in most image-processing/pattern-recognition systems.

Due to the limitation of space, an efficient implementation of our algorithm is omitted in this proceedings, but will be presented at the conference, and in a full paper in a journal.

Table 2. Comparison of our algorithm with other fundamental image-processing methods

Algorithm	Execution time [*msec*]
our algorithm (max.)	2.3
our algorithm (mean)	1.5
thresholding	4.6
edge detection	9.6
noise reduction	51.5

6. REFERENCES

[1] R. M. Haralick, *Some neighborhood operations*, pp. 11–35, Plenum Press, Addison-Wesley, Reading, MA, 1981.

[2] A. Hashizume, R. Suzuki, H. Yokouchi, and *et al.*, "An algorithm of automated rbc classification and its evaluation," *Bio Medical Engineering*, vol. 28(1), pp. 25–32, 1990.

[3] A. Rosenfeld and J. L. Pfalts, "Sequential operations in digital picture processing," *Journal of ACM*, vol. 13(4), pp. 471–494, October 1966.

[4] R. Lumia, "A new three-dimensional connected components algorithm," *Computer Vision, Graphics, and Image Processing*, vol. 23(2), pp. 207–217, 1983.

[5] Y. Shirai, *Labeling connected regions*, Springer-Verlag, 1987.

[6] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, vol. 89, pp. 1–23, 2003.

[7] F. Chang, C. J. Chen, and C. J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding*, vol. 93, pp. 206–220, 2004.

[8] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, vol. 2, 1982.

[9] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Systems Man and Cybernetics*, vol. 9, pp. 62–66, January 1979.

[10] D. Marr and E. Hildreth, "Theory of edge detection," in *Proc. Royal Soc.* 1980, vol. B207, pp. 187–217, London.

[11] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[12] C. Fiorio and J. Gustedt, "Two linear time union-find strategies for image processing," *Theoretical Computer Science*, vol. 154(2), pp. 165–181, February 1996.