Pattern Recognition 42 (2009) 1977 – 1987

Contents lists available at ScienceDirect

Pattern Recognition

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

Fast connected-component labeling

Lifeng He[a,e,*], Yuyan Chao[b,e], Kenji Suzuki[c], Kesheng Wu[d]

[a]*Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute, Aichi 480-1198, Japan*
[b]*Graduate School of Environment Management, Nagoya Sangyo University, Aichi 488-8711, Japan*
[c]*Department of Radiology, Division of the Biological Sciences, The University of Chicago, Chicago, IL 60637, USA*
[d]*Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, USA*
[e]*ShaanXi University of Science & Technology, Xi'an Shannxi 710021, China*

A R T I C L E   I N F O

Article history:
Received 27 March 2008
Received in revised form 2 September 2008
Accepted 20 October 2008

Keywords:
Labeling algorithm
Label equivalence
Connected component
Linear-time algorithm
Pattern recognition

A B S T R A C T

Labeling of connected components in a binary image is one of the most fundamental operations in pattern recognition: labeling is required whenever a computer needs to recognize objects (connected components) in a binary image. This paper presents a fast two-scan algorithm for labeling of connected components in binary images. We propose an efficient procedure for assigning provisional labels to object pixels and checking label equivalence. Our algorithm is very simple in principle, easy to implement, and suitable for hardware and parallel implementation. We show the correctness of our algorithm, analyze its complexity, and compare it with other labeling algorithms. Experimental results demonstrated that our algorithm is superior to conventional labeling algorithms.

1. Introduction

Labeling of connected components in a binary image is one of the most fundamental operations in pattern analysis (recognition), computer (robot) vision, and machine intelligence [1–3]. 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 objects (connected components) in binary images; in other words, labeling is required in almost all image-based applications such as fingerprint identification, character recognition, automated inspection, target recognition, face identification, medical image analysis, and computer-aided diagnosis [4–6]. Especially in real-time applications such as traffic-jam detection, automated surveillance, and target tracking, faster labeling algorithms are always demanded.

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 such as the Von Neumann architecture and two-dimensional images, there are mainly the following four classes of algorithms:

(1) Multi-scan algorithms. Algorithms [7,8] scan an image in the forward and backward raster directions alternately to propagate label equivalences until no label changes.
(2) Two-scan algorithms. Algorithms [9–19] complete labeling in two scans: during the first scan, they assign provisional labels to object pixels, and record label equivalences. Label equivalences are resolved during or after the first scan. During the second scan, all equivalent labels are replaced by their representative label.
(3) Hybrid algorithms. The algorithm [20] is a hybrid between multi-scan algorithms and two-scan algorithms. Like multi-scan algorithms, the hybrid algorithm scans an image in the forward and backward raster directions alternately. During the scans, as in two-scan algorithms, a one-dimensional table is used for recording and resolving label equivalences. Experimental results demonstrated that four is the upper limit on the number of scans [20,21].
(4) Tracing-type algorithms. These algorithms [10,21–23] avoid analysis of label equivalences by tracing the contours of objects (connected components) or by use of an iterative recursion operation. Such algorithms had been considered to be efficient only for simple images, but not for complicated images, until Chang's contour-tracing algorithm [23] was proposed.

There are other labeling algorithms for special image representations and special computer architectures. Algorithms [24–31] were

* Corresponding author. Tel.: +81 561 64 1111; fax: +81 561 64 1108.
E-mail addresses: helifeng@ist.aichi-pu.ac.jp (L. He), chao@nagoya-su.ac.jp (Y. Chao), suzuki@uchicago.edu (K. Suzuki), KWu@lbl.gov (K. Wu).

0031-3203/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.10.013

developed for the images represented by hierarchical tree structures [32–34], i.e., n-ary trees such as bintree, quadtree, octree, etc. The efficiency of such algorithms may be better than that of other conventional ones, but in the worst case, it is the same as that of conventional two-scan algorithms. On the other hand, parallel algorithms [35–43] were developed for parallel machine models such as a mesh-connected massively parallel processor or systolic array processors.

This paper presents a new two-scan algorithm for labeling of connected components in binary images. We propose an efficient strategy for assigning provisional labels to object pixels and checking label equivalence by case analysis, and we use the equivalent-label sets and a representative label table for solving label equivalences [19,44]. Experimental results showed that our algorithm is efficient for connected-component labeling.

The rest of this paper is organized as follows: we review the method for assigning provisional labels to object pixels and checking label equivalence in conventional raster-scan labeling algorithms in the next section, and we introduce our algorithm in Section 3. In Section 4, we show experimental results, and in Section 5, we present a discussion. We give our concluding remarks in Section 6.

The preliminary version of this paper was presented at the 2007 IEEE International Conference on Image Processing (ICIP 2007) [44]. However, the efficient strategy for assigning provisional labels for object pixels by case analysis given in Section 3, the correctness of our algorithm given in Section 5.1, the complexity analysis, and the efficiency analysis of the proposed algorithm given in Sections 5.2 and 5.3, respectively, the comparison with other conventional labeling algorithm given in Sections 5.4, 5.5, 5.6 and 5.7, and the method for generating consecutive labels for connected components given in Section 5.8 are new.

2. Conventional two-raster-scan algorithms

For an $N \times N$ binary image,[1] we use $b(x,y)$ to denote the pixel value at (x,y) in the image, where $1 \leqslant x,y \leqslant N$, and V_O for the value of object pixels and V_B for that of background pixels. We assume that V_O and V_B are larger than the value of any provisional label, and $V_O < V_B$.

For convenience, we consider only the case for eight-connected connectivity in this paper, because our algorithm can easily be extended to that for labeling with four-connected connectivity.

By using the mask [3] shown in Fig. 1, all conventional pixel-based raster-scan algorithms except the one proposed in Ref. [44] scan a given image in the raster scan direction once (the first scan) to process pixels one by one. If the current pixel $b(x,y)$ is a background pixel, nothing needs to be done. On the other hand, if there is no object pixel in the mask other than the current pixel, it is assigned a new label. Otherwise, it is assigned the minimal label in the mask, and all different labels (if any) in the mask are recorded to be *equivalent labels*.

By the above first scan, the connected component illustrated in Fig. 2(a), for example, is provisionally labeled as the one shown in Fig. 2(b), where *label* 1 and *label* 4, *label* 2 and *label* 5, *label* 2 and *label* 3, and *label* 3 and *label* 4 are recorded to be equivalent labels.

There are several methods for recording and resolving label equivalences. One uses an $L \times L$ two-dimensional array table [9,14] to record label equivalences, where L is the number of provisional labels assigned to an image.[2] Each element X_{ij} of the table is initialized to 0. If provisional labels u and v are found to be equivalent labels, then the element X_{uv} is set to 1. After the first scan, for each provisional label m from 0 to L, it analyzes the table to find all m's equivalent labels, and m is used as their representative label.

Another method is an application of the so-called union-find algorithm [18,45–48]. It records label equivalences by use of union-find trees. When two provisional labels are found to be equivalent, the two union-find trees corresponding to the two labels are connected together. Thus, after the first scan, all equivalent provisional labels will be combined in the same union-find tree, and the label on the root is used as their representative label.

The third method was proposed in Refs. [19,44]. By this method, all equivalent labels are combined in a set (called and *equivalent label set*), and the smallest label among them is used as their *representative label*, whose relation is recorded in the so-called *representative label table*. When two provisional labels are found to be equivalent, the two equivalent label set corresponding to the two labels are combined together, and the representative label table is updated at the same time. Thus, after the first scan, all equivalent provisional labels will be combined in the equivalent label set and hold the same representative label.

After label equivalences are resolved, the second scan is executed by replacement of all equivalent labels with their representative label.

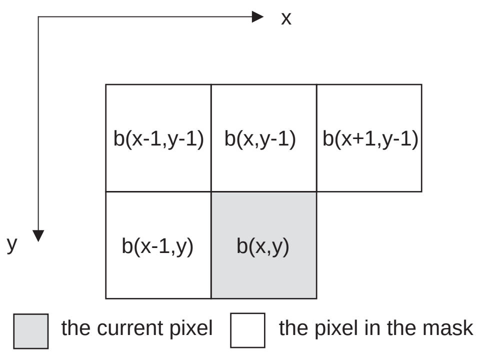


Fig. 1. Mask for eight-connected connectivity.

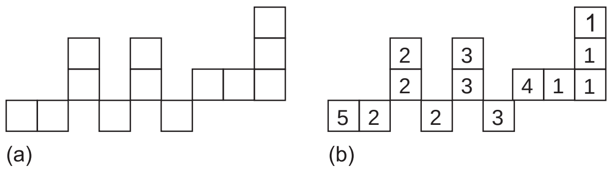


Fig. 2. Illustration of provisional labeling by the first scan. (a) Connected-component example. (b) Provisional labels assigned after the first scan.

3. Outline of our proposed algorithm

Because the method by using equivalent label sets and the representative label table for resolving label equivalences is simple and efficient, we also use the method for resolving label equivalence in our algorithm.

By this method, in the first scan, all provisional labels that belong to a connected component found at this point will be combined in the same equivalent label set and hold the same representative label. That is, all labels in an equivalent label set are equivalent. Therefore, when processing an object pixel, in the case where there is at least one object pixel in the mask, instead of assigning the minimum label in the mask to the object pixel, we can assign an arbitrary provisional label in the mask to it. This simplifies the labeling operation by avoiding calculation of the minimum label in the mask.

[1] For convenience, in this paper, we consider only $N \times N$ images. However, our algorithm works for any $N \times M$ images.

[2] For an $N \times N$ image, L has the order $O(N)$.

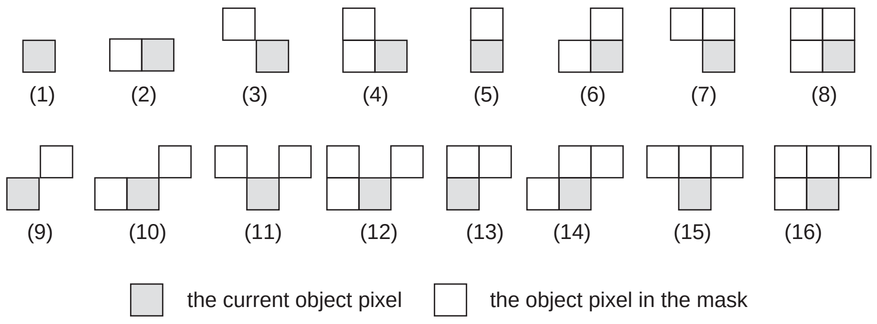


Fig. 3. Sixteen possible cases for the current object pixel in the mask for eight-connected connectivity.

On the other hand, with the mask for eight-connected connectivity, for an object pixel $b(x, y)$, there are 16 possible cases in the mask, as shown in Fig. 3.

Case (1) is the only case necessary for creating a new equivalent label set.

In cases (2)–(9) and (13)–(16), $b(x, y)$ can take on any existing provisional label in the mask without the need for consideration of label equivalences. In our algorithm, because any label equivalence is resolved immediately whenever it is found, if there is any label equivalence in the mask, it should have been resolved already. Considering, for example, case (16), by use of our algorithm, the order of processing of the object pixels in the mask is $b(x-1, y-1) \rightarrow b(x, y-1) \rightarrow b(x+1, y-1) \rightarrow b(x-1, y)$. After $b(x, y-1)$ is processed, the provisional labels for $b(x-1, y-1)$ and $b(x, y-1)$ should have been combined in the same equivalent label set and have the same representative label. In the same way, after $b(x+1, y-1)$ is processed, the provisional labels for $b(x-1, y-1)$, $b(x, y-1)$, and $b(x+1, y-1)$ should have been combined in the same equivalent label set and have the same representative label. Finally, after $b(x-1, y)$ is processed, the provisional labels for $b(x-1, y-1)$, $b(x, y-1)$, $b(x+1, y-1)$, and $b(x-1, y)$ should have been combined in the same equivalent label set and had the same representative label. Because all provisional labels (if any) in the mask have the same representative label, $b(x, y)$ can take on any of the labels without consideration of any label equivalence.

Case (12) can be classified into either case (10) or (11), because $b(x-1, y)$ and $b(x-1, y-1)$ should have belonged to the same equivalent label set and already have had the same representative label.

Cases (10) and (11) are the only two cases necessary for considering label equivalence, where the two object pixels in the mask would not be "connected" if $b(x, y)$ was not an object pixel. In each case, $b(x, y)$ can take either of the labels of the two object pixels. If the two object pixels have the same representative label, nothing needs to be done. Otherwise, i.e., if they have different representative labels, the equivalence between the two labels should be resolved.

The operations for all cases in our proposed algorithm are summarized in Table 1, where c_1, c_2, c_3, and c_4 denote pixels $b(x-1, y)$, $b(x-1, y-1)$, $b(x, y-1)$, and $b(x+1, y-1)$, respectively, in the mask shown in Fig. 1, 0 denotes the background pixel, 1 denotes the object pixel, and *resolve* (u, v) denotes the operation for resolving the equal equivalence of labels u and v.

Because the operation *resolve* takes much time compared to the other operations, we should avoid it as much as possible. The Karnaugh map [49] for the operation *resolve* is shown in Fig. 4. The condition under which *resolve* does not take place can be derived as

$$c_3 + \overline{c_4} + \overline{c_1} \cdot \overline{c_2},$$

where c_i ($\overline{c_i}$) is true if c_i is (not) an object pixel, and $(\cdot)$ and $(+)$ denote the logical multiplication ('AND') and the logical summation ('OR'), respectively.

That is, the conditions where *resolve* does not happen are: c_3 is an object pixel, or c_4 is not an object pixel, or neither c_1 nor c_2 is an object pixel. Considering that we need to assign a provisional label to $b(x, y)$, the most efficient way is to check first whether c_3 is an object pixel or not, because if it is an object pixel, we do not need to consider label equivalences, and we can assign c_3's label to $b(x, y)$ without checking other pixels in the mask. Otherwise, for example, suppose that we check c_4 first. If c_4 is not an object pixel, we do not need to consider label equivalences, but we should check other pixels in the mask further to determine the provisional label for $b(x, y)$. Obviously, we need to do more work in the latter case.

Based upon the above discussion, the efficient procedure for labeling the current object pixel and resolving label equivalences in the mask can be summarized as follows:

```
if ($c3 \neq V_B$)
  $b(x, y) = c3$;
else if ($c1 \neq V_B$)
  $b(x, y) = c1$;
  if ($c4 \neq V_B$)
    $resolve(c4, c1)$;
else if ($c2 \neq V_B$)
  $b(x, y) = c2$;
  if ($c4 \neq V_B$)
    $resolve(c2, c4)$;
else if ($c4$)
  $b(x, y) = c4$;
else
  $b(x, y) = m$, $m = m + 1$.
```

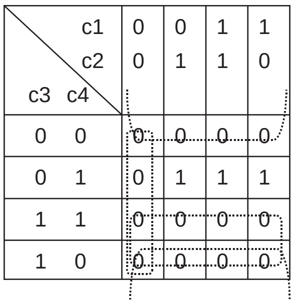
After the first scan, the provisional labels that are assigned to a connected component in the given image will be combined in an equivalent label set and hold the same representative label. Then, by the second scan, each provisional label is replaced by its representative label. Thus, after the second scan, all pixels in a connected component will be assigned a unique label.

4. Comparative evaluation

All algorithms used for our comparison were implemented in the C language and compiled under the same condition, and all experimental results presented in this section were obtained by averaging of the execution time for 5000 runs on a PC-based workstation (Intel Pentium D 930 3.0 GHz + 3.0 GHz CPUs, 2 GB Memory, Mandriva Linux OS) by use of one core.

Table 1
Operations in the sixteen cases

Case	c_4	c_3	c_2	c_1	$b(x,y)$	Operations
(1)	0	0	0	0	m	$m=m+1$
(2)	0	0	0	1	c_1	No operation
(3)	0	0	1	0	c_2	No operation
(4)	0	0	1	1	c_1 or c_2	No operation
(5)	0	1	0	0	c_3	No operation
(6)	0	1	0	1	c_1 or c_3	No operation
(7)	0	1	1	0	c_2 or c_3	No operation
(8)	0	1	1	1	c_1, c_2, or c_3	No operation
(9)	1	0	0	0	c_4	No operation
(10)	1	0	0	1	c_1 or c_4	$resolve(c_1, c_4)$
(11)	1	0	1	0	c_2 or c_4	$resolve(c_2, c_4)$
(12)	1	0	1	1	c_1, c_2, or c_4	$resolve(c_1, c_4)$ or $resolve(c_2, c_4)$
(13)	1	1	0	0	c_3 or c_4	No operation
(14)	1	1	0	1	c_1, c_3, or c_4	No operation
(15)	1	1	1	0	c_2, c_3, or c_4	No operation
(16)	1	1	1	1	c_1, c_2,c_3, or c_4	No operation



c3 c4 \ c1 c2	0 0	0 1	1 1	1 0
0 0	0	0	0	0
0 1	0	1	1	1
1 1	0	0	0	0
1 0	0	0	0	0

Fig. 4. The Karnaugh map for the operation *resolve*.

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) [20] and noise images. 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 of the images containing uniform random noise with 41 different threshold values from 0 to 1000 in steps of 25. 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[3] and the image database of the University of Southern California,[4] were used for realistic testing of labeling algorithms. In addition, seven texture images, which were downloaded from the Columbia-Utrecht Reflectance and Texture Database,[5] 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 [50].

We mainly compared our algorithm with the following three labeling algorithms: (1) the contour-tracing connected-component labeling (CT) algorithm proposed in Ref. [23]; (2) the scan plus array-based union-find (SAUF) algorithm proposed in Ref. [18]; (3) the run-based two-scan (RTS) algorithm proposed in Ref. [19]. The programs of the SAUF algorithm and the RTS algorithm were provided by the original authors, and that of the CT algorithm was downloaded from the original authors' website.[6]

4.1. Execution time versus the number of object pixels and the number of pixels in an image

We used all noise images to test the linearity of the execution time versus image size. As shown in Fig. 5, all of the three algorithms have the ideal linear characteristics versus image size.

Noise images with a size of 512×512 pixels were used for testing the execution time versus the density of an image. As shown in Fig. 6, our algorithm is superior to the CT algorithm, the SAUF algorithm, and the RTS algorithm for all images.

4.2. Comparisons in terms of the maximum, mean, and minimum execution times

Natural images, medical images, texture images, and artificial images with specialized shape patterns were used for this test. We also compared our algorithm with one representative multi-scan algorithm, i.e., Lumia's algorithm [11], and two representative two-scan algorithms, i.e., Rosenfeld's algorithm [3] and Shirai's algorithm [13]. The results of the comparisons are shown in Table 2. The results for the selected six images are illustrated in Fig. 7, where the object pixels are displayed in black. The results demonstrated that our algorithm was the fastest of all of the algorithms for all images.

5. Discussion

In this section, we analyze our algorithm and make some qualitative comparisons of our algorithm with representative conventional labeling algorithms, including the SAUF algorithm, the RTS algorithm, and the CT algorithm [23].

5.1. Correctness of our algorithm

The correctness of our algorithm is quite straightforward: (1) every object pixel is assigned a provisional label[7]; (2) every provisional label belongs to only one equivalent label set; (3) each equivalent label set has only one representative label;

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

[6] http://dar.iis.sinica.deu.tw/Download
[7] If there is no object pixel in the mask, the object pixel is assigned a new provisional label. Otherwise, it is assigned a label in the mask.

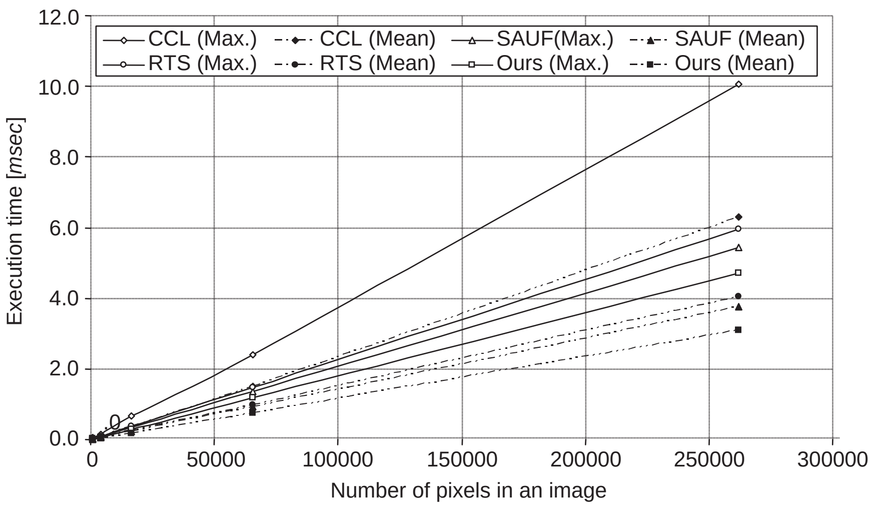


Fig. 5. Execution time versus the number of pixels in an image.

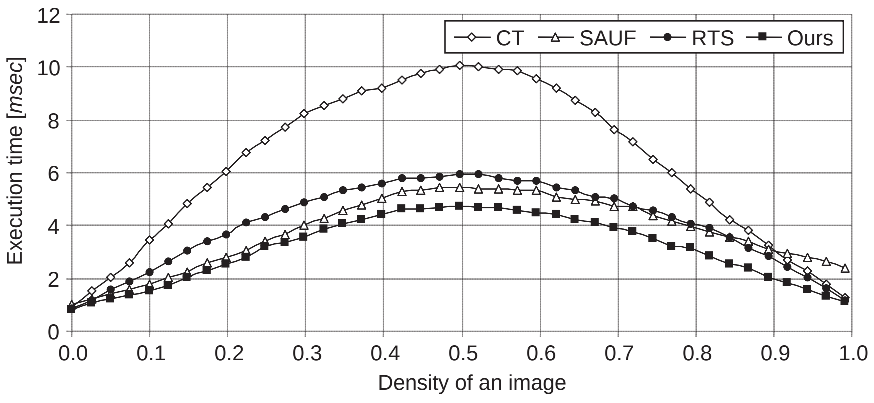


Fig. 6. Execution time versus the density of object pixels in an image.

Table 2
Comparison of various execution times (ms) on various kinds of images

Image type	Rosenfeld	Lumia	Shirai	Hybrid	CT	SAUF	RTS	Ours
Natural								
Max.	2185.2	2045.9	52.4	18.4	3.8	3.2	2.9	2.3
Mean	524.6	173.3	40.0	13.7	2.4	2.1	1.8	1.5
Min.	22.7	11.7	33.9	9.6	1.2	1.3	1.0	1.0
Medical								
Max.	650.2	807.7	41.7	14.9	2.6	2.3	1.8	1.5
Mean	378.9	192.1	38.8	13.4	1.9	1.7	1.5	1.2
Min.	75.5	15.3	37.2	11.4	1.5	1.4	1.2	0.9
Textural								
Max.	1068.7	72.7	42.8	28.5	3.7	2.9	2.5	2.0
Mean	296.7	35.2	39.3	27.4	2.7	2.5	1.8	1.6
Min.	13.8	15.6	38.5	26.4	1.5	2.3	1.1	1.1
Special								
Max.	1061.3	748.9	39.3	15.8	7.4	2.2	2.5	1.8
Mean	316.8	164.9	22.5	7.9	3.9	1.2	1.5	0.8
Min.	2.0	2.5	0.96	2.2	1.1	0.3	0.8	0.3

(4) whenever two different parts of a single connected component are found to be connected in the first scan, all provisional labels belonging to the two parts are combined, while conditions (2) and (3) are preserved at all times. In this way, at any time in the first scan, all provisional labels for each part of a single connected component found so far are combined with a corresponding equivalent label set.

Thus, when the first scan is finished, all provisional labels assigned to a connected component are combined in a corresponding equivalent label set, and all of them have the same representative

Rosenfeld:	147.5	Lumia:	17.5
Shirai:	45.2	Hybrid:	12.9
CT:	3.4	SAUF:	2.1
RTS:	2.0	Ours:	1.7

(a)

Rosenfeld:	541.9	Lumia:	154.8
Shirai:	40.5	Hybrid:	17.6
CT:	3.3	SAUF:	2.5
RTS:	2.2	Ours:	1.7

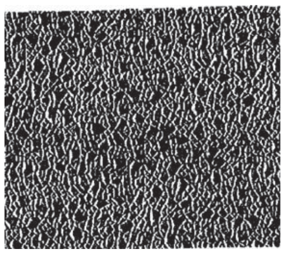
(b)

Rosenfeld:	583.9	Lumia:	174.2
Shirai:	38.8	Hybrid:	15.1
CT:	2.2	SAUF:	2.2
RTS:	1.6	Ours:	1.5

(c)

Rosenfeld:	36.8	Lumia:	13.2
Shirai:	34.3	Hybrid:	10.1
CT:	1.2	SAUF:	1.4
RTS:	1.0	Ours:	1.0

(d)



Rosenfeld:	339.8	Lumia:	38.5
Shirai:	42.8	Hybrid:	28.4
CT:	3.7	SAUF:	2.9
RTS:	2.5	Ours:	2.0

(e)

Rosenfeld:	545.2	Lumia:	203.7
Shirai:	39.8	Hybrid:	14.8
CT:	2.0	SAUF:	1.9
RTS:	1.4	Ours:	1.4

(f)

Fig. 7. Execution time (ms) of labeling algorithms for the selected six images: (a) a text image; (b) a fingerprint image; (c) a portrait image; (d) a snapshot image; (e) a texture image; (f) a medical image.

label. On the second scan, all object pixels having the corresponding provisional labels in a connected component are assigned their unique representative label. Therefore, every connected component is assigned a unique label by our algorithm.

5.2. Maximum execution time of our algorithm

To complete labeling, our algorithm performs the following procedures:

(1) Assigning provisional labels to object pixels during the first scan.

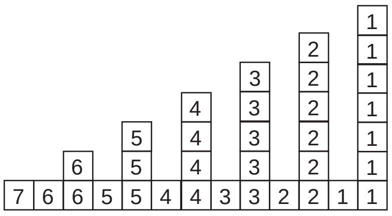


Fig. 8. A connected component with seven provisional labels.

(2) Creating equivalent label sets and setting representative labels for all new provisional labels.
(3) Resolving label equivalences.
(4) Replacing the provisional labels of object pixels with their representative labels during the second scan.

For an $N \times N$-pixel image, both the maximum number of provisional labels and the maximum number of connected components have an order of $\mathcal{O}(N^2)$. Accordingly, the order of the maximum number of label equivalences among provisional labels is also $\mathcal{O}(N^2)$.

By our algorithm, it is obvious that the orders of procedures (1) and (4) are $\mathcal{O}(N^2)$. In procedure (2), only constant operations are executed for each new provisional label. Because the order of the maximum number of provisional labels is $\mathcal{O}(N^2)$, the order of procedure (2) in the worst case is also $\mathcal{O}(N^2)$.

Now we consider procedure (3). For resolving a label equivalence between two provisional labels, there are two operations: one is combining two equivalent label sets; the other is updating the representative label table. By the method, the operation for combining two equivalent label sets can be made in constant time, i.e., the order is $\mathcal{O}(1)$. Because the maximum number of label equivalences occurring in an image has the order $\mathcal{O}(N^2)$, the order of all such operations for labeling an image is also $\mathcal{O}(N^2)$.

The only remaining operation is updating the representative label table. For convenience, we use $\mathcal{S}(i)$ to denote an equivalent label set with i as the representative label. By combining $\mathcal{S}(u_1) = \{u_1, \ldots, u_m\}$ into $\mathcal{S}(v_1) = \{v_1, \ldots, v_m\}$, where $v_1 \leqslant u_1$, the combined equivalent label set will be $\mathcal{S}(v_1) = \{v_1, \ldots, v_m, u_1, \ldots, u_m\}$. To realize this, for each label $x \in \{u_1, \ldots, u_m\}$, we need to set the representative label of X as v_1. The execution time of this process depends on the number of members in $\mathcal{S}(u_1)$, i.e., m.

For a W-pixel connected component with M provisional labels, we consider the following two special cases: (1) when the maximum time that the operation is executed for updating the representative label table happens; (2) when the connected component has the maximum number of provisional labels, i.e., when M is the maximum.

In the first case, the maximum times that the operation is executed for updating the representative label table should be $1 + 2 + 3 + \cdots + (M - 1)$, the order of which is $\mathcal{O}(M^2)$. A connected component with seven provisional labels is illustrated in Fig. 8. In such a case, the number of pixels in the connected component has an order of $\mathcal{O}(M^2)$. Thus, $O(M^2) = O(W)$.

In the second case, the maximum number of provisional labels for a W-pixel connected component has the order $\mathcal{O}(W)$. A nine-pixel connected component is illustrated in Fig. 9. In this case, the times that the operation is executed for updating the representative label table is $1 + 1 + \cdots + 1 = (W - 1)/2$, the order of which is $\mathcal{O}(W)$.

It is found from the above discussion that, in both cases, the order of the operation for updating the representative label table for a W-pixel connected component becomes $\mathcal{O}(W)$. Thus, the order of the operations for updating the representative label table for labeling an $N \times N$ image should be $\mathcal{O}(N^2)$.

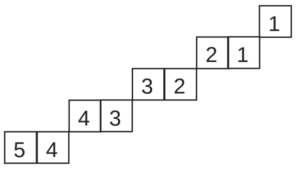


Fig. 9. The maximum number of provisional labels for a nine-pixel connected component.

Therefore, for an $N \times N$ image, the order for resolving label equivalences is also $\mathcal{O}(N^2)$.

Because the order of every procedure for labeling an $N \times N$ image in our algorithm is $\mathcal{O}(N^2)$, the order of our algorithm should be $\mathcal{O}(N^2)$.

5.3. *Efficiency analysis of our algorithm*

We analyzed each step of our algorithm in terms of efficiency; in other words, we analyzed the reasons for the efficiency of our algorithm. Because processing for background pixels in our algorithm is the same as that in other raster-scan algorithms, we focus on an analysis of processing for object pixels. For convenience, we assume that each of the 16 possible cases for the current object pixel shown in Fig. 3 occurs with the same probability in this analysis.

The efficiency of our algorithm is achieved by the following three improvements: (1) simplification of the labeling operation by avoiding the calculation of the minimum label in the mask (we call this efficiency improvement 1 hereafter); (2) checking c_3 first in the mask by the case analysis introduced in Section 3 (efficiency improvement 2); (3) using the label-equivalence-resolving method and the relative data structures (efficiency improvement 3). We describe the analysis of each efficiency improvement in detail below.

Conventional raster-scan labeling algorithms assign the minimal provisional label in the mask to the current object pixel. For deriving the minimal provisional label, a comparison operation is required four times in any cases in the conventional algorithms, whereas in our algorithm shown in Section 3, a comparison operation is required once for cases (5)–(8), (13)–(16); twice for cases (2), (4), (10), and (12); three times for cases (3), (11), and (15); and four times for cases (1) and (9). The average time required for comparison operations by our algorithm is $(1 \times 8 + 2 \times 4 + 3 \times 3 + 2 \times 4)/16 \approx 2.06$, which is less than four times that are needed by conventional raster-scan algorithms.

On the other hand, for processing an object pixel, our algorithm checks whether a pixel in the mask is an object pixel in the order $c_3 \to c_1 \to c_2 \to c_4$. By the procedure shown in Section 3, the checking operation is required once for cases (5)–(8), (13)–(16); three times for cases (2), (4), (10), and (12); and four times for cases (1), (3), (9), and (11). The average number of checking operations is $(1 \times 8 + 3 \times 4 + 4 \times 4)/16 = 2.25$. In comparison, conventional two-scan labeling algorithms check pixels in the mask in the order $c_1 \to c_2 \to c_3 \to c_4$. The checking operation is required three times for cases (5)–(8), (13)–(16); and four times for cases (2), (4), (9), (10), (11), and (12). The average number of checking operations is $(3 \times 8 + 4 \times 8)/16 = 3.5$. Thus, our algorithm is more efficient by approximately 35% than are conventional two-scan algorithms.

We performed an experimental test to demonstrate the efficiency achieved by each efficiency improvement by using 512×512 noise images. The results are shown in Fig. 10, where *algorithm A*, *algorithm B*, *algorithm C*, and *algorithm D* indicate our algorithm without efficiency improvements 1 and 2, that without efficiency improvement 1, that without efficiency improvement 2, and that with all three efficiency improvements (i.e., our complete algorithm), respectively.

5.4. *Comparison with multi-scan algorithms*

In multi-scan algorithms, because, at every scan, the label equivalences propagate only in the neighboring object pixels (thus, the label equivalences will be resolved slowly), they need to scan an image at least twice (usually, many times), depending on the complexity of the connected components in an image. The worst case is $N/2$ times, where N is the image matrix size [20]. Thus, the order of the maximum time for labeling an $N \times N$ image is $\mathcal{O}(N^3)$. At each scan except the first one, the provisional labels of object pixels are rewritten in order to propagate label equivalences. Moreover, at each scan and for each object pixel, the algorithms need to calculate the minimum label in the mask.

In comparison with multi-scan algorithms, our algorithm resolves each label equivalence completely as soon as it is found (thus, the label equivalences will be resolved very quickly), it scans an image only twice, and the labels of object pixels need to be rewritten only once. Moreover, our algorithm does not need to calculate the minimum label in the mask. On the other hand, we need three additional L-sized one-dimensional arrays for resolving label equivalences, where L is the largest provisional label in an image.

5.5. *Comparison with conventional two-scan labeling algorithms*

The main problems with the methods for resolving label equivalences in conventional two-scan labeling algorithms are complicated, and they usually need a large memory space. For example, for an L-provisional label image I_L, Shirai's algorithm [14], which is an improved version of that proposed in Ref. [13], first uses a two-dimensional $2L^2/(3d)$ bit-size table to remove the duplicate equivalences, where d is the number of sub-images, which are divided from I_L for reducing the memory size. Then, it sets an $L \times L$ size connection table $\mathcal{T}$, each of whose elements X_{ij} is initialized to 0. If provisional labels u and v are found to be equivalent labels, then the element X_{uv} is set to 1.

After the first scan, for each provisional label m from 0 to L, which is initially marked to be *unprocessed*, it finds all m's equivalent labels by analyzing the table $\mathcal{T}$ as follows: (1) if m is not *unprocessed*, do nothing; (2) if m is *unprocessed*, put m into a queue $\mathcal{Q}$; (3) while Q is not empty, for the first element p of $\mathcal{Q}$, if p is not *unprocessed*, do nothing, otherwise, change p's status to *processed*, and for each element X_{pq} of $\mathcal{T}$ such that $X_{pq} = 1$, the provisional label v is m's equivalent label, and put q into $\mathcal{Q}$.

The efficiency of this algorithm depends on the size of L, i.e., the number of provisional labels. Moreover, it has no linearity property versus image sizes [20,23].

Another representative two-scan labeling algorithm [11], i.e., Lumia's algorithm, employs a small local table to store label equivalences at each row of an image to improve the memory space problem. This algorithm was improved in Ref. [16] by implementation with *run-length coding*. Because searching of the table and replacing of provisional labels are performed after scanning of every row, this algorithm is not more efficient than other conventional two-scan labeling algorithms.

On the other hand, because the union-find algorithm [45–48] is a general method for resolving the *disjoint set union problem* or the *equivalence problem*, it was used for resolving label equivalences in two-scan algorithms [18,51].

The union-find algorithm makes use of a tree to represent each set and consists of three basic operations:

(1) *MakeSet*(E) creates a new tree only with the root E.
(2) *Find*(E) finds the root of the node E.
(3) *Union*(A, B) combines two trees with the roots A and B into a single tree by linking one of the roots to the other.

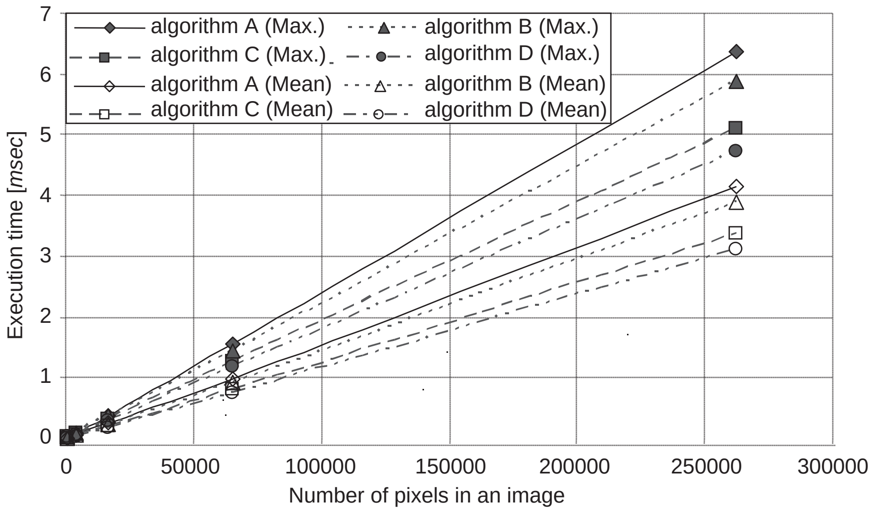


Fig. 10. Analysis of efficiency improvements in our algorithm.

Two optimizations were proposed for improving the performance of the algorithm. One is called *path compression*: whenever *Find*(E) is executed, all nodes on the path from node E to the root of the tree (including E, but not the root) are reset to point the root. Another is called *weight balancing*: when *Union*(A, B) is performed, the root of the smaller tree is linked to the root of the larger one.

The labeling algorithm employing the union-find algorithm proposed in Ref. [51] was slower than the CT algorithm [23]. However, the SAUF algorithm proposed in Ref. [18], which is implemented by use of a single L-sized one-dimensional array with path compression only (i.e., without weight-balancing), is superior to the CT algorithm for most cases.

All of the above conventional two-scan algorithms complete labeling in three phases: a first scanning phase (assigning a provisional label to each object pixel and storing label equivalences), an analysis phase (solving label equivalences), and a relabeling phase (assigning a final label to each object pixel). Moreover, for each object pixel, they need to calculate the minimum provisional label in the mask.

In comparison with these conventional two-scan algorithms: (1) Because our algorithm resolves the label equivalences immediately whenever they are found during the first scan, all label equivalences are resolved at the completion of the first scan; thus, the second scan can start immediately. In other words, resolving label equivalences is combined into the first scanning phase. (2) Our algorithm does not require the calculation of the minimum provisional label in the mask; therefore it saves a great deal of time. (3) The processing for resolving label equivalences does not depend on the provisional label assignment. Thus, our algorithm is more suitable than other two-scan algorithms for parallel implementation.

It should be noted that the optimizations of the union-find algorithm can be achieved naturally in our algorithm. Because two equivalent label sets are combined whenever they are found to be equivalent, all labels in an equivalent label set are always at the same level. In other words, there is no path that needs to be compressed further. Thus, the optimal effect of the *path-compression* operation is achieved naturally in our algorithm.

Moreover, when combining two equivalent label sets, we combine the equivalent label set $\mathcal{L}$ with a larger representative label into the other set $\mathcal{S}$. This combining operation has an effect similar to *weight balancing*. Because $\mathcal{S}$ (with a smaller representative label) is created earlier than $\mathcal{L}$ in our algorithm, with use of our combining operation, the length of $\mathcal{S}$ is usually greater than that of $\mathcal{L}$. Although it is not guaranteed that the shorter set is always combined into the longer set, our combining operation has the great advantages that we do not need to track the length of a set and to compare the lengths of two sets.

In order to confirm the above analysis, we compared our method to the method with the weight-balancing optimization on all 292 test images. The experimental results demonstrated that our algorithm was faster by 18.3% on average than the method with the weight-balancing optimization (for example, the execution times (ms) of our algorithm and the method with the weight-balancing optimization for Fig. 7(a), (b), (c), (d), (e), and (f) are (1.7, 1.9), (1.7, 2.0), (1.5, 1, 6), (1.0, 1.1), (2.0, 2.3), and (1.4, 1.6), respectively. Moreover, we investigated the proportion of sets with a smaller length which were eventually combined into sets with a greater length. We found that the proportion was 98%. This fact agreed with the above analysis and with the experimental results.

In addition, our strategy is more suitable for generating consecutive labels for connected components (see Section 5.8).

Recently, a run-based two-scan labeling algorithm was proposed [19]. Unlike other conventional two-scan labeling algorithms, which process label equivalences among pixels, this algorithm processes label equivalences among runs. It also uses equivalent label sets and the representative label table for resolving label equivalences. Because this algorithm considers the connectivity between runs, the number of provisional labels assigned to an image is usually smaller than that assigned by other conventional provisional labels. The reduction in the number of provisional labels reduces not only the time for marking new labels, but also that for resolving label equivalences.

This algorithm is very efficient for run-length-encoded images. However, for general images, because it needs to find and record run data in the first scan, it is not as efficient as our algorithm.

It is worth mentioning that, if we use only the equivalent label sets and the representative label table for resolving label equivalences, the efficiency of our algorithm will decline a lot (see Fig. 10). In this case, our algorithm was inferior to the RTS algorithm.

5.6. Comparison with the hybrid algorithm

Similar to multi-scan algorithms, the hybrid algorithm [20] is based on label equivalence propagation. Although it uses a one-dimensional table to speed up the propagation of label equivalences, to complete labeling, it still needs to scan an image four times in the forward and backward raster-scan directions alternately for propagating label equivalences. Similar to multi-scan algorithms, at every scan except the first one, the labels for object pixels are rewritten. Moreover, like other raster-scan algorithms, at each scan and for each object pixel, it needs to calculate the minimum label in the mask.

In comparison with the hybrid algorithm, as mentioned above, our algorithm resolves each label equivalence completely as soon as it is found, i.e., the label equivalence propagation is complete at each time. Our algorithm scans an image only twice, and the labels of object pixels need to be rewritten only once. Again, we do not need to calculate the minimum label in the mask. However, we need two additional L-sized one-dimensional arrays.

5.7. Comparison with the CT algorithm

The CT algorithm proposed in Ref. [23] completes labeling by one scan (i.e., no re-labeling is required), and it needs no additional memory. During labeling, the algorithm can extract connected-component contours and the sequential orders of contour points, which are also important for some image processing and pattern recognition. However, (1) because the algorithm accesses an image in an irregular way, it is not suitable for pipeline processing [55], parallel implementation [56], or systolic-array implementations. (2) For labeling, it marks some background pixels as -1 to indicate that they have been visited. For recovering the value -1 to 0, the algorithm becomes a two-scan algorithm. (3) In cases in which the value of the background pixels in a given image is not 0, and we need to set them to 0 in the output image, it also becomes a two-scan algorithm.

In comparison, our algorithm processes an image in the raster-scan order; therefore, it is suitable for hardware implementation [57], pipeline processing, and parallel implementation. For example, we can design a chip to detect the status of the mask in a single step. Moreover, our algorithm is faster than the CT algorithm for all images used in our test. On the other hand, for an $N \times N$-size image, our algorithm requires three $N \times N/4$-sized one-dimensional arrays for resolving label equivalences. In addition, our algorithm does not provide connected-component contours or the sequential orders of contour points.

5.8. Generating consecutive labels

With our proposed algorithm, each connected component in a given image will be assigned a unique label. Generally, those labels are not consecutive. In the case where we prefer consecutive labels, we only need to sort the representative table *rtable*[ ] before the second scan as follows, where L is the number of provisional labels assigned to the given image:

```
$j = 1;$
for $(i = 1; i \leqslant L; i++)$
    if $(rtable[i] == i)$
        $rtable[i] = j;$
        $j++;$
    else
        $rtable[i] = rtable[rtable[i]];$
    end if
end for
```

This process should not take much time. According to our experimental results, for all test images, it took only 2.6% additional time on average.

Note that: (1) in order to generate consecutive labels, all equivalent-label-based labeling algorithms require a similar procedure; (2) when combining two equivalent sets, if we used a weight-balancing strategy to combine a set with a larger length into one with a smaller length, then it was not guaranteed that the representative label of an equivalent set was the smallest in the set. In this case, a procedure for generating consecutive labels requires more steps than ours, as follows:

```
$j = 1;$
for $(i = 1; i \leqslant L; i++)$
    if $(rtable[i] == i)$
        $rtable[i] = j;$
        $j++;$
    end if
end for
for $(i = 1; i \leqslant L; i++)$
    $rtable[i] = rtable[rtable[i]];$
end for
```

6. Concluding remarks

In this paper, we present a fast two-scan algorithm for labeling of connected components in binary images. The advantages of our algorithm are: (1) it is simple in principle; (2) it is easy to implement (less than 50 lines in the C language); (3) it is efficient for various types of images (overall, it is faster than all conventional labeling algorithms that we could find); (4) it has the ideal linearity property versus image size (i.e., for $N \times N$ images, its complexity is $O(N^2)$); (5) it is suitable for hardware implementation (because it processes an image in a sequential order); (6) it is suitable for parallel implementation (because in our algorithm, the processing for resolving label equivalences is independent of the provisional label assignment, the two processes can be made by parallel processing). The main weaknesses of our algorithm are: (1) in comparison with the SAUF algorithm and the CT algorithm, we need more memory space for implementation; (2) for labeling, our algorithm needs to process each pixel at least twice, whereas the tracing-type labeling algorithms usually process most background pixels once.

For future work, we plan to implement our algorithm in hardware [52,53], to extend it to include three-dimensional connected-component labeling [21,54], and to develop algorithms for parallel architectures.

Acknowledgments

We thank the anonymous referees for their valuable comments that improved this paper greatly. We are grateful to the editor for his/her kind cooperation and help. We also thank Ms. E.F. Lanzl for proofreading this paper. This work was partially supported by the TOYOAKI Scholarship foundation, Japan.

References

[1] C. Ronsen, P.A. Denjiver, Connected Components in Binary Images: The Detection Problem, Research Studies Press, 1984.
[2] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1992.
[3] A. Rosenfeld, A.C. Kak, Digital Picture Processing, second ed., vol. 2, Academic Press, San Diego, CA, 1982.
[4] K. Suzuki, S.G. Armato, F. Li, S. Sone, K. Doi, Massive training artificial neural network (MTANN) for reduction of false positives in computerized detection of lung nodules in low-dose CT, Med. Phys. 30 (7) (2003) 1602–1617.
[5] K. Suzuki, H. Yoshida, J. Nappi, S.G. Armato, A.H. Dachman, Mixture of expert 3D massive-training ANNs for reduction of multiple types of false positives in CAD for detection of polyps in CT colonography, Med. Phys. 35 (2) (2008) 694–703.
[6] K. Suzuki, F. Li, S. Sone, K. Doi, Computer-aided diagnostic scheme for distinction between benign and malignant nodules in thoracic low-dose CT by use of massive training artificial neural network, IEEE Trans. Med. Imaging 24 (9) (2005) 1138–1150.
[7] R.M. Haralick, Some neighborhood operations, in: Real Time/Parallel Computing Image Analysis, Plenum Press, New York, 1981, pp. 11–35.

[8] A. Hashizume, R. Suzuki, H. Yokouchi, et al., An algorithm of automated RBC classification and its evaluation, Bio Med. Eng. 28 (1) (1990) 25–32.
[9] A. Rosenfeld, J.L. Pfalts, Sequential operations in digital picture processing, J. ACM 13 (4) (1966) 471–494.
[10] A. Rosenfeld, Connectivity in digital pictures, J. ACM 17 (1) (1970) 146–160.
[11] R. Lumia, L. Shapiro, O. Zungia, A new connected components algorithm for virtual memory computers, Comput. Vision Graphics Image Process. 22 (2) (1983) 287–300.
[12] R. Lumia, A new three-dimensional connected components algorithm, Comput. Vision Graphics Image Process. 23 (2) (1983) 207–217.
[13] Y. Shirai, Labeling connected regions, in: Three-Dimensional Computer Vision, Springer, Berlin, 1987, pp. 86–89.
[14] T. Gotoh, Y. Ohta, M. Yoshida, Y. Shirai, Component labeling algorithm for video rate processing, in: Proceeding of the SPIE, Advances in Image Processing, vol. 804, April 1987, pp. 217–224.
[15] M. Komeichi, Y. Ohta, T. Gotoh, T. Mima, M. Yoshida, Video-rate labeling processor, in: Proceedings of the SPIE, Image Processing II, vol. 1027, September 1988, pp. 69–76.
[16] R.M. Haralick, L.G. Shapiro, Computer and Robot Vision, vol. I, Addison-Wesley, Reading, MA, 1992, pp. 28–48.
[17] S. Naoi, High-speed labeling method using adaptive variable window size for character shape feature, in: IEEE Asian Conference on Computer Vision, vol. 1, December 1995, pp. 408–411.
[18] K. Wu, E. Otoo, K. Suzuki, Optimizing two-pass connected-component labeling algorithms, Pattern Anal. Appl. 2008, in press, doi: 10.1007/s10044-008-0109-y.
[19] L. He, Y. Chao, K. Suzuki, A run-based two-scan labeling algorithm, IEEE Trans. Image Process. 17 (5) (2008) 749–756.
[20] K. Suzuki, I. Horiba, N. Sugie, Linear-time connected-component labeling based on sequential local operations, Comput. Vision Image Understanding 89 (2003) 1–23.
[21] Q. Hu, G. Qian, W.L. Nowinski, Fast connected-component labeling in three-dimensional binary images based on iterative recursion, Comput. Vision Image Understanding 99 (2005) 414–434.
[22] D.H. Ballard, Computer Vision, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[23] F. Chang, C.J. Chen, C.J. Lu, A linear-time component-labeling algorithm using contour tracing technique, Comput. Vision Image Understanding 93 (2004) 206–220.
[24] H. Samet, Connected component labeling using quadtrees, J. ACM 28 (3) (1981) 487–501.
[25] I. Gargantini, Separation of connected component using linear quad- and oct-trees, in: Proceedings of the 12th Conference on Numerical Mathematics and Computing, vol. 37, University of Manitoba, Winnipeg, 1982, pp. 257–276.
[26] M. Taminen, H. Samet, Efficient octree conversion by connectivity labeling, in: Proceedings of the SIGGRAPH 84 Conference, ACM, Minneapolis, MN, 1984, pp. 43–51.
[27] H. Samet, Computing geometric properties of images represented by linear quadtrees, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-7 (2) (1985) 229–240.
[28] H. Samet, M. Tamminen, An improved approach to connected component labeling of images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, Florida, 1986, pp. 312–318.
[29] H. Samet, M. Taminen, Efficient component labeling of images of arbitrary dimension represented by linear bintrees, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-10 (4) (1988) 579–586.
[30] J. Hecquard, R. Acharya, Connected component labeling with linear octree, Pattern Recognition 24 (6) (1991) 515–531.
[31] M.B. Dillencourt, H. Samet, M. Tamminen, A general approach to connected-component labeling for arbitrary image representations, J. ACM 39 (2) (1992) 253–280.
[32] S.N. Srihari, Hierarchical representations for serial section images, in: Proceedings of the 5th International Conference on Pattern Recognition, 1980, pp. 1075–1080.
[33] C.L. Jackins, S.L. Tanimoto, Octrees and their use in representing 3D objects, Comput. Graphics Image Process. 14 (3) (1980) 249–270.
[34] H. Samet, The quadtree and related hierarchical data structures, Comput. Surv. 16 (2) (1984).
[35] D.S. Hirschberg, A.K. Chandra, D.V. Sarwate, Computing connected components on parallel computers, Commun. ACM 22 (8) (1979) 461–464.
[36] D. Nassimi, S. Sahani, Finding connected components and connected ones on a mesh connected parallel compute, SIAM J. Comput. 9 (4) (1980) 744–757.
[37] Y. Schiloach, U. Vishkin, An o(log n) parallel connectivity algorithm, J. Algorithms 3 (1982) 57–67.
[38] L.W. Tucker, Labeling connected components on a massively parallel tree machine, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, Florida, 1986, pp. 124–129.
[39] M. Manohar, H.K. Ramapriyan, Connected component labeling of binary images on a mesh connected massively parallel processor, Computer Vision Graphics Image Process. 45 (2) (1989) 133–149.
[40] H.M. Alnuweiri, V.K. Prasanna, Parallel architectures and algorithms for image component labeling, IEEE Trans. Pattern Anal. Mach. Intell. 14 (10) (1992) 1024–1034.
[41] S. Olariu, J.L. Schwing, J. Zhang, Fast component libeling and convex hull computation on reconfigurable meshes, Image Vision Comput. 11 (7) (1993) 447–455.
[42] A. Choudhary, R. Thakur, Connected component labeling on coarse grain parallel computers: an experimental study, J. Parallel Distributed Comput. 20 (1994) 78–83.
[43] P. Bhattacharya, Connected component labeling for binary images on a reconfigurable mesh architectures, J. Syst. Archit. 42 (4) (1996) 309–313.
[44] L. He, Y. Chao, K. Suzuki, A linear-time two-scan labeling algorithm, in: 2007 IEEE International Conference on Image Processing (ICIP), San Antonio, Texas, USA, September 2007, pp. V-241–V-244.
[45] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.
[46] R.E. Tarjan, Efficiency of a good but not linear set union algorithm, J. ACM 22 (2) (1975) 215–225.
[47] K. Mehlhorn, Data Structures and Algorithm 1: Sorting and Searching, Springer, Berlin, 1984.
[48] R.E. Tarjan, J.V. Leeuwen, Worst-case analysis of set union algorithms, J. ACM 31 (2) (1984) 245–281.
[49] M. Karnaugh, The map method for synthesis of combinational logic circuits, Trans. AIEE. pt I 72 (9) (1953) 593–599.
[50] N. Otsu, A threshold selection method from gray-level histograms, IEEE Trans. Syst. Man Cybernet. 9 (1979) 62–66.
[51] C. Fiorio, J. Gustedt, Two linear time union-find strategies for image processing, Theoret. Comput. Sci. 154 (2) (1996) 165–181.
[52] C.J. Nicol, Design of a connected component labeling chip for real time image processing, in: Proceedings of the IEEE Asia–Pacific Conference on Circuits and Systems, Sydney, Australia, December 1992, pp. 142–147.
[53] C.J. Nicol, A systolic approach for real time connected component labeling, Comput. Vision Image Understanding 61 (1) (1995) 17–31.
[54] J.K. Udupa, V.G. Ajjanagadde, Boundary and object labeling in three-dimensional images, Comput. Vision, Graphics, Image Process. 51 (3) (1990) 355–369.
[55] T. Hattori, A high-speed pipeline processor for regional labeling based on a new algorithm, in: Proceedings of the International Conference on Pattern Recognition, NJ, June 1990, pp. 494–496.
[56] 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 (2003) 353–370.
[57] 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.

About the Author—LIFENG HE 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 computer science from Nagoya Institute of Technology, Japan, in 1994 and 1997, respectively. He is an associate professor at the Aichi Prefectural University, Japan, and a guest professor at the Shaanxi University of Science and Technology, China. From September 2006 to May 2007, he worked at the University of Chicago (USA) as a research associate. His research interests include image processing, automated reasoning, and artificial intelligence.

About the Author—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 the 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 at the Nagoya Institute of Technology. She is an associate professor at the Nagoya Sangyo University, Japan, and a guest professor at the Shaanxi University of Science and Technology, China. Her research interests include image processing, graphic understanding, CAD, and automated reasoning.

About the Author—KENJI SUZUKI received his B.S. (Magna Cum Laude) and M.S. (Summa Cum Laude) degrees in electrical and electronic engineering from Meijo University, Nagoya, Japan, in 1991 and 1993, respectively, and his Ph.D. degree (by Published Work) in information engineering from Nagoya University, Nagoya, Japan, in 2001. From 1993 to 1997, he worked in the Research and Development Center at Hitachi Medical Corporation as a researcher. From 1997 to 2001, he worked in the Faculty of Information Science and Technology at Aichi Prefectural University, Japan, as a faculty member. In 2001, he joined the Kurt Rossmann Laboratories for Radiologic Image Research in the Department of Radiology, Division of the Biological Sciences at The University of Chicago, as Research Associate. He was promoted to Research Associate (Instructor) in 2003, and to Research Associate (Assistant Professor) in 2004. Since 2006, he has been Assistant Professor in the Department of Radiology, the Committee of Medical Physics, and the Cancer Research Center.

Dr. Suzuki has published more than 120 scientific papers (including 55 peer-reviewed journal papers) in the fields of medical image analysis, machine learning, computer vision, and pattern recognition. He has been serving as a referee for more than 25 journals in these fields, including IEEE Transactions on Medical Imaging, IEEE Transactions on Biomedical Engineering, IEEE Transactions on Information Technology in Biomedicine, IEEE Transactions on Image Processing, IEEE Transactions on Signal Processing, IEEE Transactions on Systems, Man and Cybernetics, and Image and Vision Computing. He has received awards for his research, including a Paul C. Hodges Award from The University of Chicago in 2002, a Certificate of Merit Award from the RSNA in 2003, a Research Trainee Prize from the RSNA in 2004, a Young Investigator Award from the Cancer Research Foundation in 2005, an Honorable Mention Poster Award at the SPIE International Symposium on Medical Imaging in 2006, and a Certificate of Merit Award from the RSNA in 2006. He was elected as a Senior Member of the IEEE in 2004. He is a member of IEICE, IEEJ, IPSJ, JNNS, and JCS.

About the Author—KESHENG WU is a Staff Computer Scientist at Lawrence Berkeley National Lab. He is the key developer of a number of software packages including FastBit and TRLan. The work on FastBit bitmap indexes has received a patent on the compression method, a best paper award from International Supercomputer Conference 2005, and an R&D 100 Award in 2008. Kesheng owned his Ph.D. in Computer Science from University of Minnesota. His research interests include data management, data analyses, visualization, scientific computing, and performance tuning. He has published more than 50 articles in scientific journals, conferences and books.