Pattern Recognition Letters 31 (2010) 28–35

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

An efficient first-scan method for label-equivalence-based labeling algorithms

Lifeng He [a,*], Yuyan Chao [b], Kenji Suzuki [c]

[a] Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute, Aichi 480-1198, Japan
[b] Graduate School of Environmental 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

ARTICLE INFO

Article history:
Received 21 January 2009
Received in revised form 20 August 2009
Available online 2 September 2009

Communicated by M. Couprie

Keywords:
Connected component
Labeling algorithm
First scan
Mask
Pattern recognition

ABSTRACT

Label-equivalence-based connected-component labeling algorithms complete labeling in two or more raster scans. In the first scan, each foreground pixel is assigned a provisional label, and label equivalences between provisional labels are recorded. For doing this task, all conventional algorithms use the same mask that consists of four processed neighbor pixels to process every foreground pixel. This paper presents a simple yet efficient first-scan method for label-equivalence-based labeling algorithms. In our method, foreground pixels following a background pixel and those following a foreground pixel are processed in a different way. By use of this idea, the pixel followed by the current foreground pixel can be removed from the mask. In other words, the mask used in our method consists of three processed neighbor pixels. Thus, for processing a foreground pixel, the average number of times for checking the processed neighbor pixels in the first scan is reduced from 2.25 to 1.75. Because the current foreground pixel following a background pixel or a foreground pixel can be known without any additional computing cost, our method is efficient for any image that contains at least one foreground pixel. Experimental results demonstrated that our method is effective for improving the efficiency of label-equivalence-based labeling algorithms.

1. Introduction

Labeling all pixels of each connected component a unique label in a binary image is indispensable for processing in pattern recognition and machine intelligence (Ronsen and Denjiver, 1984; Rosenfeld and Kak, 1982). For applications to dynamic images, e.g., computer (robot) vision, automatic detection, and automatic tracking, faster labeling algorithms are always demanded. Many labeling algorithms have been proposed for addressing this issue. For ordinary computer architectures and pixel-based representation images, there are the following two classes of labeling algorithms:

(1) Label-equivalence-based algorithms. These algorithms process an image in the raster scan direction at least twice. In the first scan, a provisional label is assigned to each foreground pixel. All provisional labels assigned to the same connected component are called *equivalent labels*, and the relationships between equivalent labels are called *label equivalences*. Any label equivalence is recorded as soon as found. After the first scan, all label equivalences are resolved, which means finding a representative label for each group of equivalent labels. Then, they relabel the pixel by the representative label of the provisional label assigned in the first scan. There are pixel-based algorithms that resolve label equivalences between pixels and run-based algorithms that resolve label equivalences between runs, each of which means a block of consecutive foreground pixels in a row.
For pixel-based algorithms, there are multi-scan (Haralick, 1981; Hashizume et al., 1990), four-scan (Suzuki et al., 2003), and two-scan (Rosenfeld and Pfalts, 1966; Rosenfeld, 1970; Lumia et al., 1983; Lumia, 1983; Shirai, 1987; Gotoh et al., 1987; Komeichi et al., 1988; Naoi, 1995; Wu et al., 2009; He et al., 2009) algorithms. On the other hand, the algorithm proposed by He et al. (2008) is a run-based one.
(2) Searching and label propagation algorithms (Ballard, 1982; Shoji and Miyamichi, 1995; Rosenfeld and Kak, 1982; Chang et al., 2004; Hu et al., 2005; Martin-Herrero, 2007). These algorithms first search an unlabeled foreground pixel, assign it a new label, and then propagate the label to all foreground pixels connected to the pixel in later processing.
Although searching and label propagation algorithms usually use the raster scan to find an unprocessed foreground pixel in an image, they process an image in an irregular way, depending on the shapes of connected components in the image. Therefore, they are not suitable for pipeline processing (Hattori, 1990), parallel implementation (Hirschberg

* Corresponding author. Fax: +81 561641108.
E-mail address: helifeng@ist.aichi-pu.ac.jp (L. He).

0167-8655/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.patrec.2009.08.012

et al., 1979; Nassimi and Sahani, 1980; Schiloach and Vishkin, 1982; Manohar and Ramapriyan, 1989; Alnuweiri and Prasanna, 1992; Wang et al., 2003), systolic-array implementation (Nicol, 1995), or hardware implementation (Yang, 1988).

In order to assign each foreground pixel a provisional label and record label equivalences between provisional labels in the first scan, all conventional label-equivalence-based algorithms for pixel-based (as opposed to run-based) images use the same mask consisting of the four processed neighbor pixels. They process every foreground pixel in the same way.

This paper presents a simple, yet efficient first-scan method for label-equivalence-based labeling algorithms. In our method, we process foreground pixels following background pixels and those following foreground pixels in a different way. The mask used in our method consists of only the three processed neighbor pixels of the current foreground pixel in the row above. Thus, the number of times for checking neighbor pixels can be reduced, which leads to more efficient processing.

The rest of this paper is organized in the following way. In the next section, we review the first-scan methods in conventional label-equivalence-based algorithms. Section 3 introduces our first-scan method. We show experimental results in Section 4 and provide a discussion in Section 5. Lastly, we give our conclusion in Section 6.

2. Conventional first-scan methods in label-equivalence-based algorithms

For an $N \times M$-size binary image, we use $b(x,y)$ to denote either the pixel at (x,y) in the image or the value of the pixel when the meaning is clear from the text, and $label(x,y)$ for the label of the pixel $b(x,y)$, where $1 \leqslant x \leqslant N$ and $1 \leqslant y \leqslant M$. We assume that an image is given in the PBM format, that is, foreground pixels and background pixels in an image are represented by 1 and 0, respectively. As is the same in most labeling algorithms, we assume that all pixels on the border of an image are background pixels, and we consider only eight connectivity.

For labeling a binary image, label-equivalence-based labeling algorithms need two or more raster scans. In the first scan, they assign to each foreground pixel a provisional label and record label equivalences between provisional labels. Then, after resolving label equivalences, all equivalent labels will have a unique representative label. Finally, each provisional label is replaced by its representative label.

In the first scan, all conventional label-equivalence-based labeling algorithms use the mask shown in Fig. 1 to process pixels one by one in an image in the raster scan direction. For convenience, hereafter, we use c_1, c_2, c_3, c_4 to denote $b(x-1,y)$, $b(x-1,y-1)$, $b(x,y-1)$, $b(x+1,y-1)$, and use $l_i (1 \leqslant i \leqslant 4)$ for the value of c_i, respectively. Except for the algorithm proposed in (He et al., 2009), all such algorithms do the following processing: For each current pixel $b(x,y)$, if it is a background pixel, nothing is done. On the other hand, if it is a foreground pixel, it is assigned a new label if there is no foreground pixel in the mask. Otherwise, the current foreground pixel is assigned the minimal label in the mask. Moreover, all different provisional labels (if any) in the mask are recorded as equivalent labels.

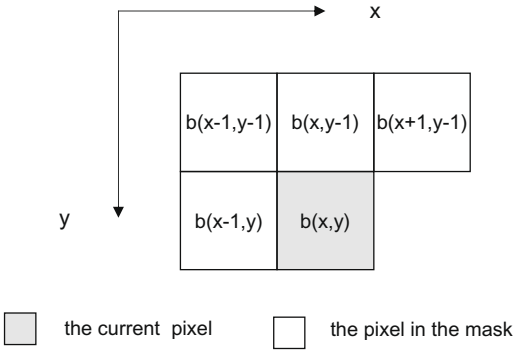

Fig. 1. The mask used in conventional label-equivalence-based algorithms.

Because such algorithms need to calculate the minimal label in the mask, they need to check every pixel in the mask. Thus, for processing a foreground pixel, the number of times for checking pixels in the mask is four.

The method proposed in (He et al., 2009) is more efficient than the algorithms mentioned above. In this algorithm, at any point in the first scan, all equivalent labels found so far are combined in a set, called *equivalent label set*, where the smallest label is referred to as the representative label. The corresponding relation of a provisional label and its representative label is recorded in a table, called *the representative table*. For convenience, we use S_t for the set of provisional labels with t as the representative label, and $r\_label[a]$ to represent the representative label of provisional label a. In this way, for any provisional label f in provisional label set S_t, we have $r\_label[f] = t$.

In the first scan, for the current pixel $b(x,y)$, if it is a background pixel, nothing needs to be done. Otherwise, i.e., $b(x,y)$ is a foreground pixel, we check whether there are foreground pixels in the mask. If there are, the current foreground pixel can be assigned any label in the mask. Moreover, when two foreground pixels with different provisional labels, say, u and v, in the mask become connected due to the existence of the current foreground pixel, the label equivalence in the mask needs to be resolved. The pseudo code for this processing, denoted as $resolve(u,v)$, is shown as follows.

```
%% resolve(u, v)
m = r_label[u];
n = r_label[v];
  if(m < n)
    S_m = S_m ∪ S_n;
    for each label w ∈ S_n
      r_label[w] = m;
    end of for
  else if(m > n)

    S_n = S_m ∪ S_n;
  for each label w ∈ S_m
    r_label[w] = n;
  end of for
end of if
```

On the other hand, if there is no foreground pixel in the mask, it means that the foreground pixel does not connect with any foreground pixel having been scanned, i.e., the foreground pixel belongs a connected component consisting of itself only, a new provisional label *ProLabel*, which is initialized to 1, is assigned to the pixel, i.e., $label(x,y) = ProLabel$. The equivalent label set corresponding to the connected component is established as $S_{ProLabel} = \{ProLabel\}$, and the representative label of *ProLabel* is set to itself, i.e., $r\_label[ProLabel] = ProLabel$. Then, *ProLabel* increases by 1 for consecutive processing.

By case analysis, an efficient procedure for processing a pixel $b(x,y)$, denoted as $Processing(x,y)$, was proposed, the flowchart of which is shown in Fig. 2. Accordingly, the first-scan procedure can be given as follows.

```
%% The first-scan procedure proposed in He et al. (2009).
ProLabel = 1;
for(y = 1; y <= M; y++)
    for(x = 1; x <= N; x++)
        Processing(x, y);
    end of for
end of for
```

By this method, we need not calculate the minimal label in the mask, and in most cases, we need not check all pixels in the mask. For example, for processing a foreground pixel $b(x,y)$ in the case where $b(x,y-1)$ is a foreground pixel, the only thing needs to be done is assigning $b(x,y)$ the same label assigned to $b(x,y-1)$. Thus, the operations for processing foreground pixels in the first scan can be reduced substantially.

3. The proposed first-scan method

Because we process pixels in an image in the raster scan direction, each row in the image can be considered as alternations of contiguous background-pixel blocks and contiguous foreground-pixel blocks. For each row, after passing all pixels in the first background-pixel block, we process the first foreground pixel in the first foreground-pixel block. Such a foreground pixel $b(x,y)$ follows a background pixel, i.e., $b(x-1,y)$ is a background pixel. After processing this pixel, we process all other foreground pixels (if any) in the first foreground-pixel blocks one by one. Such a foreground pixel $b(x,y)$ follows another foreground pixel, i.e., $b(x-1,y)$ is a foreground pixel. The remaining background-pixel blocks and foreground-pixel blocks can be processed alternately in the same way.

By the above discussion, during the first scan, we can divide foreground pixels into two classes: (1) those following a background pixel, and (2) the others, i.e., those following a foreground pixel. For each foreground pixel $b(x,y)$ in class (1), we know that $b(x-1,y)$ is a background pixel. On the other hand, for each foreground pixel $b(x,y)$ in class (2), we know that $b(x-1,y)$ is a foreground pixel. Thus, for processing a foreground pixel $b(x,y)$ in the first scan, because we already know whether $b(x-1,y)$ is a foreground pixel or not, we need not check pixel $b(x-1,y)$ again. In other words, pixel $b(x-1,y)$ can be removed from the mask. Therefore, the mask used in our method consists of the remaining three pixels, $b(x-1,y-1)$, $b(x,y-1)$ and $b(x+1,y-1)$ (i.e., c_2, c_3 and c_4), as shown in Fig. 3.

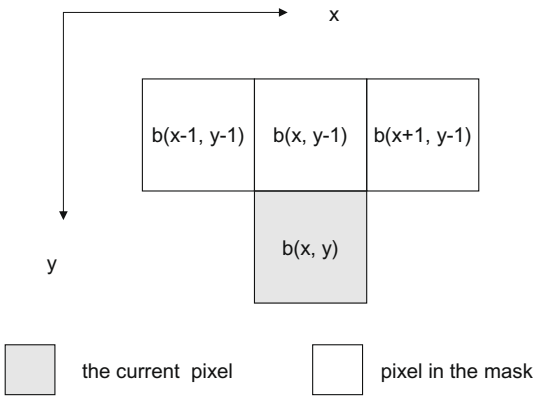

Fig. 3. The mask used in our first-scan method.

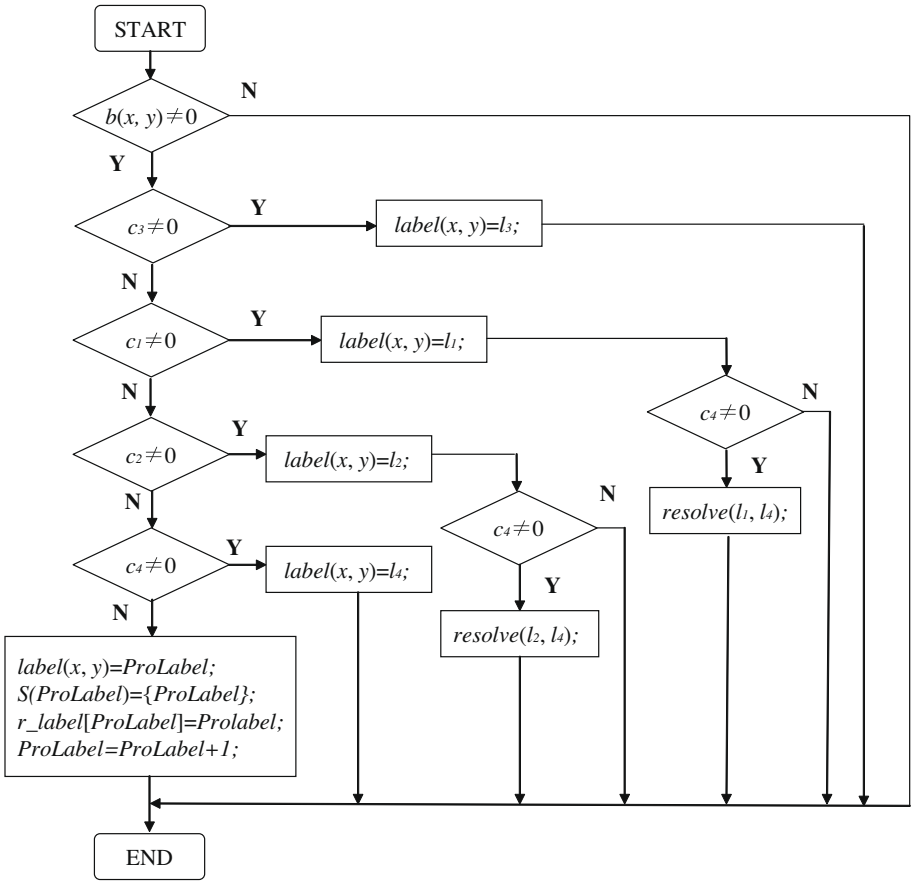

Fig. 2. The flowchart for processing a foreground pixel in the first scan in the algorithm proposed in (He et al., 2009).

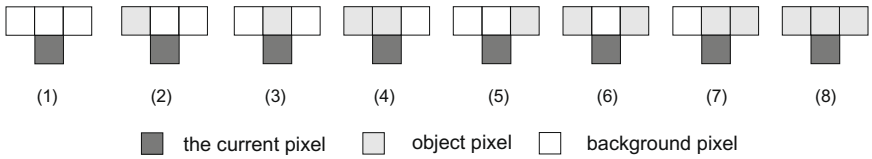

Fig. 4. Eight possible configurations for a foreground pixel in our mask.

Table 1
Operations in the eight configurations where the current foreground pixel follows a background pixel.

	c_2	c_3	c_4	Operations
(1)	0	0	0	$label(x,y) = ProLabel$, $S_{ProLabel} = \{ProLabel\}$, $r\_label[ProLabel] = ProLabel$, $ProLabel = ProLabel + 1$
(2)	1	0	0	$label(x,y) = l_2$
(3)	0	1	0	$label(x,y) = l_3$
(4)	1	1	0	$label(x,y) = \{l_2 \text{ or } l_3\}$
(5)	0	0	1	$label(x,y) = \{l_4\}$
(6)	1	0	1	$label(x,y) = \{l_2 \text{ or } l_4\}, resolve(l_2, l_4)$
(7)	0	1	1	$label(x,y) = \{l_3 \text{ or } l_4\}$
(8)	1	1	1	$label(x,y) = \{l_2, l_3, \text{ or } l_4\}$

For a foreground pixel, there are eight configurations in our mask, as shown in Fig. 4.

For a foreground pixel following a background pixel, we can summarize the operations for each configuration in Table 1. Because the operation *resolve* takes much time compared with other operations, we should avoid it as much as possible. The Karnaugh map (Karnaugh, 1953) for the operation *resolve* is shown in Fig. 5. The condition under which operation *resolve* takes place can be derived as

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

where $c_i(\overline{c_i})$ is true if c_i is (not) a foreground pixel, and $(\cdot)$ denotes the logical multiplication ('AND').

By use of the same method proposed in (He et al., 2009), the first-scan procedure for processing a foreground pixel following a background pixel, denoted as *procedure 1*, can be derived as follows:

```
%% Procedure 1
if(c3≠0)
    label(x,y) = l3;
else if(c1≠0)
    label(x,y) = l1;
    if(c4≠0)
        resolve(l1,l4);
    end of if
else if(c4≠0)
    label(x,y) = c4;
else
    label(x,y) = ProLabel,   S_ProLabel = {ProLabel};
    r_label[ProLabel] = ProLabel, ProLabel = ProLabel + 1;
end of if
```

On the other hand, for a foreground pixel $b(x,y)$ following another foreground pixel, i.e., where $b(x-1,y)$ is a processed foreground pixel, we can assign the pixel's label to $b(x,y)$, i.e., $label(x,y) = l_1$, directly without checking other pixels in the mask, and then consider whether operation *resolve* is necessary. The operations for each configuration shown in Fig. 4 can be summarized in Table 2. The Karnaugh map for the operation *resolve* is shown in Fig. 6. The condition under which operation *resolve* is necessary can be derived as

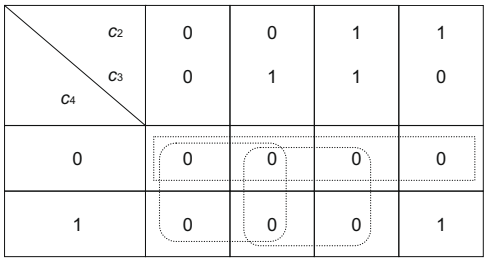

Fig. 5. Karnaugh map for the operation *resolve* for processing a foreground pixel following a background pixel.

Table 2
Operations in the eight configurations where the current foreground pixel follows a foreground pixel.

	c_2	c_3	c_4	Operations
(1)	0	0	0	$label(x,y) = l_1$
(2)	1	0	0	$label(x,y) = l_1$
(3)	0	1	0	$label(x,y) = l_1$
(4)	1	1	0	$label(x,y) = l_1$
(5)	0	0	1	$label(x,y) = l_1, resolve(l_1, l_4)$
(6)	1	0	1	$label(x,y) = l_1, resolve(l_1, l_4)$
(7)	0	1	1	$label(x,y) = l_1$
(8)	1	1	1	$label(x,y) = l_1$

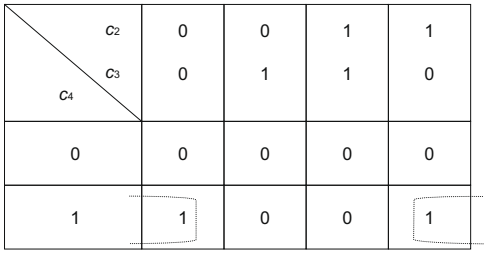

Fig. 6. Karnaugh map for the operation *resolve* for processing a foreground pixel following a foreground pixel.

$$\overline{c_3} \cdot c_4.$$

Thus, the procedure for processing a foreground pixel following a foreground pixel in the first-scan, denoted as *procedure 2*, can be summarized as follows:

```
%% Procedure 2
label(x,y) = l1;
if(c3 == 0 and c4≠0)
    resolve(l1,l4);
end of if
```

Notice that in *procedure 2* we do not check pixel c_2, i.e., $b(x-1,y-1)$. In other words, in this case, we need not consider pixel $b(x-1,y-1)$.

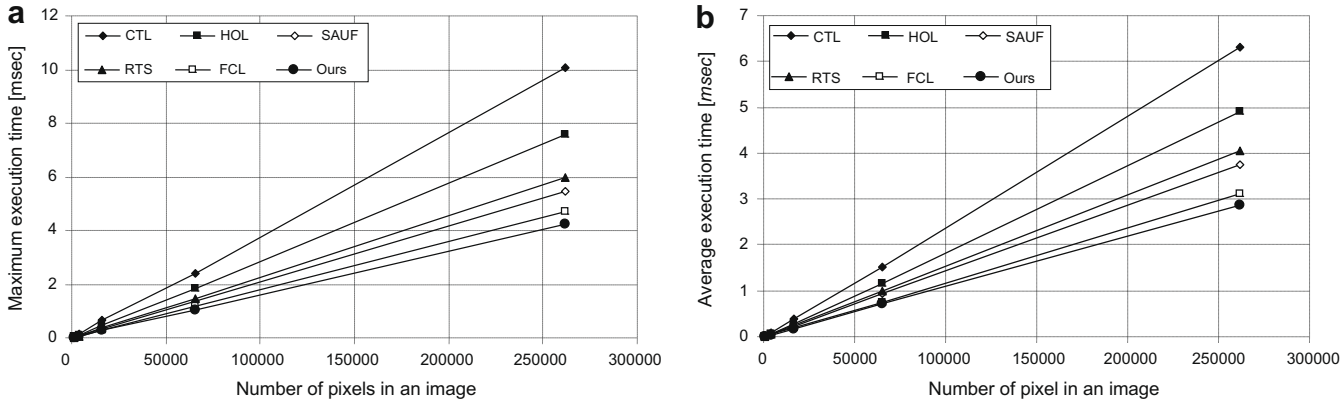

Fig. 7. Execution time versus the number of pixels in an image: (a) maximum execution time and (b) average execution time.

Finally, our first-scan procedure can be summarized as follows:

```
ProLabel = 1;
for(y = 1; y <= M; y++)
    for(x = 1; x <= N; x++)
        if(b(x,y)≠0)
            procedure1;
            x++;
            while(b(x,y)≠0)
                procedure2;
                x++;
            end of while
        end of if
    end of for
end of for
```

It is obvious that the current foreground pixel following a background pixel or a foreground pixel is found without any additional computing cost.

4. Comparative evaluation

By use of our proposed first-scan algorithm, after the first scan, every foreground pixel is assigned a provisional label and all provisional labels assigned to a connected component are combined in an equivalent label set with a unique representative label. Thus, during the second scan, for a background pixel, we assign 0 to it, and for a foreground pixel $b(x,y)$, we replace the provisional label assigned to it, i.e., $label(x,y)$ with the representative label of $label(x,y)$, i.e., $r\_label[label(x,y)]$. Thus, all pixels belonging to a connected component will be assigned a unique label.

For convenience to explain, we have used $b(x,y)$ for denoting the input image and $label(x,y)$ for output image. However, because each foreground pixel is assigned a provisional label larger than 1, and all background pixels are assigned 0, for checking whether a pixel $b(x,y)$ is a foreground pixel or not, we can use the value of either $b(x,y)$ or its label $label(x,y)$. That is, if the value of either $b(x,y)$ or its label $label(x,y)$ is not 0, $b(x,y)$ is a foreground pixel, otherwise a background pixel. In this way, for recording provisional labels, instead of using an additional 2D array $label(\_,\_)$ we can reuse $b(\_,\_)$, i.e., we can use $b(\_,\_)$ for both the input image and output image. This allows us to save the memory for $label(x,y)$ and to do the second scan more efficiently.

The data structures for equivalent label sets and the representative label table are exactly the same as shown in (He et al., 2008, 2009), and our algorithm can be implemented in a similar way.

We mainly compared our algorithm (the in-place version) with the following five recently proposed labeling algorithms: (1) the Contour-Tracing connected-component Labeling (CTL) algorithm proposed in (Chang et al., 2004); (2) the Hybrid Object Labeling (HOL) algorithm proposed in (Martin-Herrero, 2007); (3) the Scan plus Array-based Union-Find (SAUF) algorithm proposed in (Wu et al., 2009); (4) the Run-based Two-Scan (RTS) algorithm proposed in (He et al., 2008); and (5) the Fast Connected-component Labeling (FCL) algorithm proposed in (He et al., 2009). The program of the CTL algorithm was downloaded from the authors' web site,[1] and the others were provided by the authors.

All algorithms used for our comparison were implemented in the C language and compiled with the same command, 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 CPU core.

Images used for testing were the same as used in (He et al., 2008, 2009), which are 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) (Suzuki et al., 2003) 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.

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 (Otsu, 1979).

Because connected components in the noise images described above have complicated geometrical shapes and complex connectivity, we use these images to make severe evaluations of labeling algorithms. The maximum and average execution times of the six algorithms are shown in Fig. 7a and b, respectively.

[1] http://ocrlnx03.iis.sinica.edu.tw/dar/Download%20area/ccl.php3.
[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.

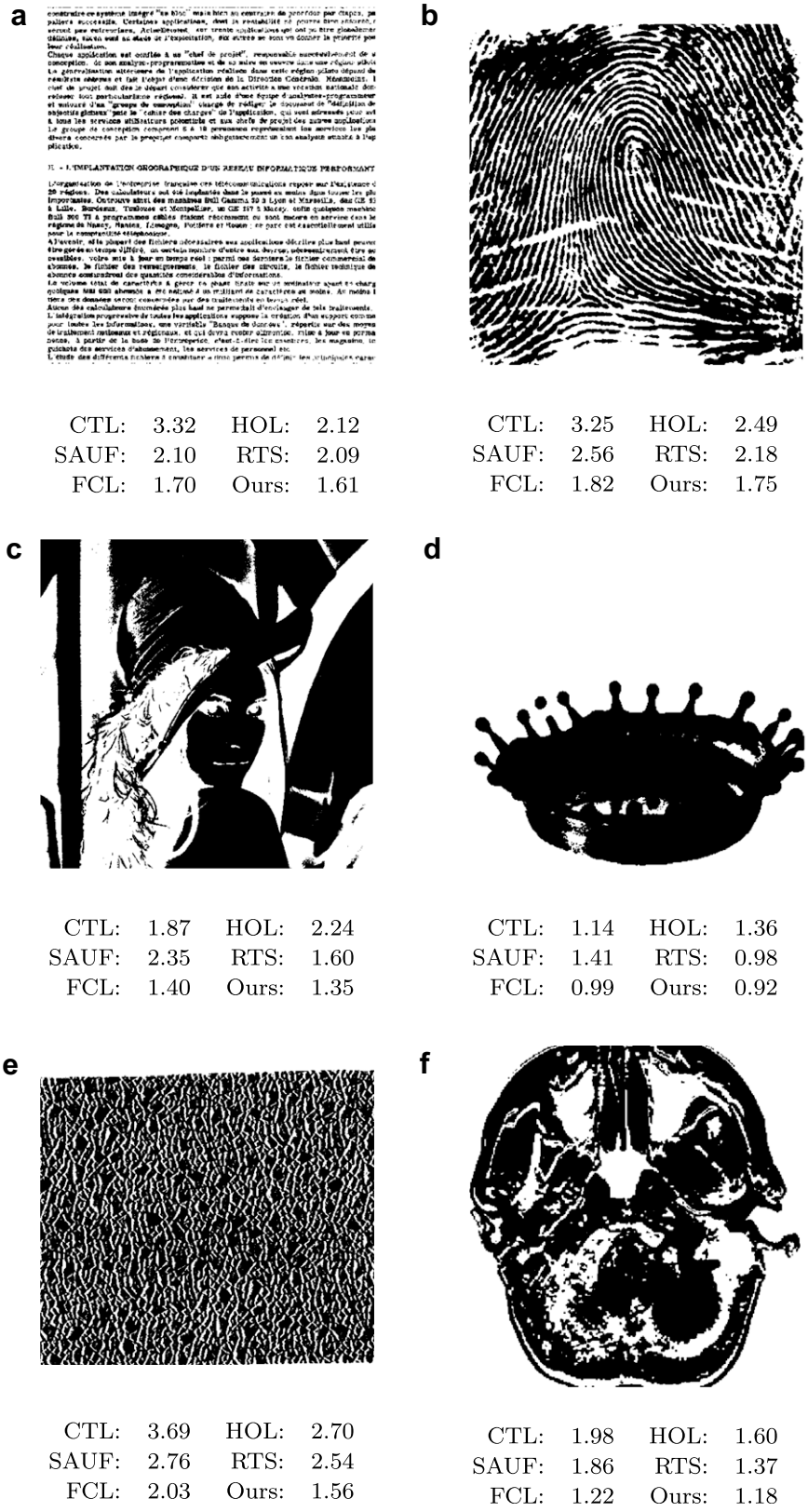

Fig. 8. Execution time [ms] of labeling algorithms for the selected six images: (a) a text image; (b) a fingerprint image; (c) a portrait image; (d) a snapshot image; (e) a texture image; (f) a medical image.

From Fig. 7, we know that either the maximum or minimum execution time of every algorithm is linear versus image size, and that our algorithm is superior to others.

Natural images, medical images, texture images, and artificial images with specialized shape patterns were used for comparisons in terms of the maximum, mean, and minimum execution times.

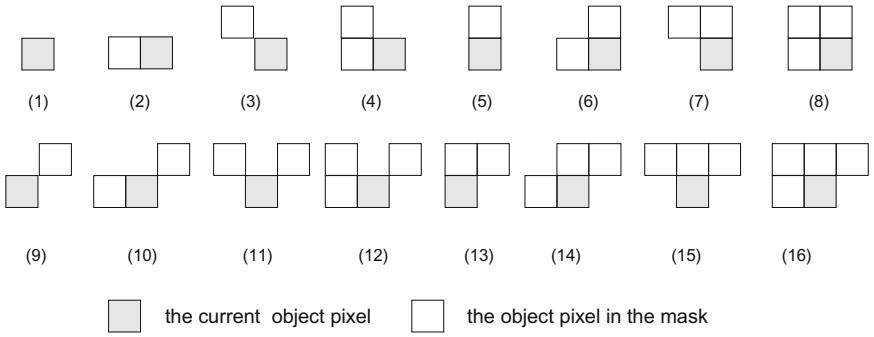

Fig. 9. Sixteen possible configurations for a foreground pixel.

Table 3
Comparison of various execution times [ms] on various kinds of images.

Image type		CTL	HOL	SAUF	RTS	FCL	Ours
Natural	max.	3.84	3.37	3.23	2.90	2.33	2.18
	mean	2.34	2.04	2.07	1.71	1.49	1.32
	min.	1.13	1.13	1.29	0.95	0.95	0.87
Medical	max.	2.59	1.63	2.28	1.71	1.47	1.31
	mean	1.92	1.25	1.89	1.37	1.21	1.09
	min.	1.52	0.79	1.54	1.19	0.93	0.90
Textural	max.	3.69	2.70	2.87	2.54	2.03	1.80
	mean	2.66	2.10	2.57	1.67	1.56	1.38
	min.	1.58	1.56	2.37	1.17	1.14	1.05
Special	max.	7.42	4.80	2.24	2.48	1.78	1.53
	mean	3.87	2.07	1.21	1.53	0.90	0.83
	min.	1.13	0.46	0.33	0.76	0.29	0.26

The results for six selected images are illustrated in Fig. 8, where the foreground pixels are displayed in black. Based on our experimental results, our algorithm was the fastest of all algorithms for all of the test images. The results are shown in Table 3.

5. Comparison with conventional label-equivalence-based algorithms

For a foreground pixel $b(x,y)$, there are 16 possible configurations for the four processed neighbor pixels, as shown in Fig. 9.

As described in Section 2, for processing a foreground pixel in the first scan, except for the algorithm proposed in (He et al., 2009), all label-equivalence-based labeling algorithms need to calculate the minimal label in the mask consisting of the four processed neighbor pixels, as shown in Fig. 1; thus, the number of times for checking pixels in the mask is four.

For the algorithm proposed in (He et al., 2009), by its first-scan procedure described in Section 2, the number of times for checking pixels in the mask is one for configurations (5), (6), (7), (8), (13), (14), (15), and (16); three for configurations (2), (4), (10), and (12); and four for configurations (1), (3), (9), and (11). The average number of times is $(1 \times 8 + 3 \times 4 + 4 \times 4)/16 = 2.25$.

In comparison, our mask, as shown in Fig. 3, consists of three processed neighbor pixels in the row above the foreground pixel. For a foreground pixel following a background pixel, by *procedure 1*, the number of times for checking pixels in the mask is one for configurations (5), (7), (13), and (15), and three for configurations (1), (3), (9), and (11). On the other hand, for a foreground pixel following another foreground pixel, by *procedure 2*, the number of times for checking pixels in the mask is one for configurations (6), (8), (14), and (16), and two for configurations (2), (4), (10), and (12). The average number of times is $(1 \times 8 + 3 \times 4 + 2 \times 4)/16 = 1.75$.

Finally, for each of the 16 configurations shown in Fig. 9, for processing a foreground pixel in the first scan, the number of times for checking the neighbor pixels in the FCL algorithm, our algorithm and other conventional label-equivalence-based labeling algorithms, is shown in Table 4. Note that, for any configuration, the number of times for checking neighbor pixels in our algorithm is smaller than or equal to those in the other two methods. Moreover, for an image with at least one foreground pixel, configuration (1) will occur at least once; thus, our method should be effective for all images with more than one foreground pixel.

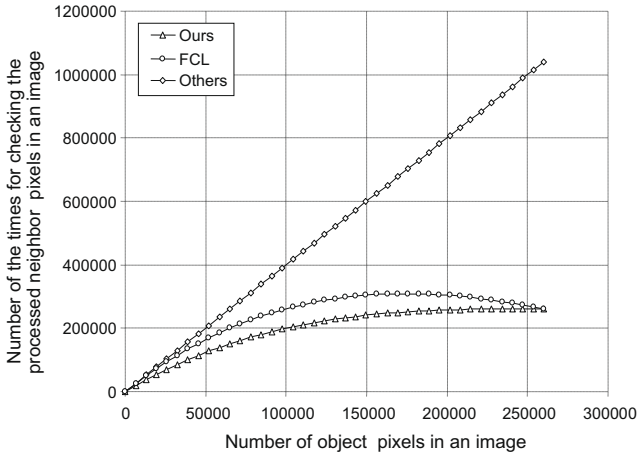

Fig. 10. The number of times for checking neighbor pixels in the first scan.

Table 4
Number of times for checking the processed neighbor pixels.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	Avg.
FCL	4	3	4	3	1	1	1	1	4	3	4	3	1	1	1	1	2.25
Ours	3	2	3	2	1	1	1	1	3	2	3	2	1	1	1	1	1.75
Others	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

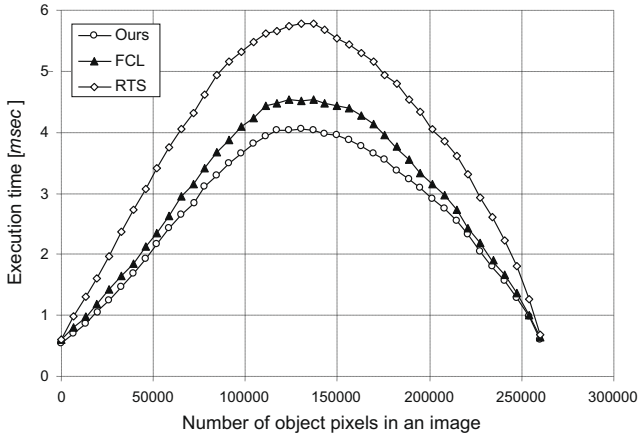

Fig. 11. Execution times of the first scan.

We used the 512×512 noise images to investigate the number of times needed for checking the processed neighbor pixels in the first scan for the three methods. The results are shown in Fig. 10.

On the other hand, the run-based label-equivalence-based labeling algorithm proposed in (He et al., 2008), i.e., the RTS algorithm, does not use a mask for assigning provisional label and record label equivalences in the first scan. However, it needs to find and record run data in an image. Because the process of the second scan in the FCL algorithm, the RTS algorithm and our algorithm are exactly the same, we use the 512×512 noise images to compare the execution times of the first scan for the three methods. The results, shown in Fig. 11, demonstrate that our algorithm is superior to the other two algorithms.

6. Conclusion

In this paper, we proposed a simple, yet efficient first-scan method for label-equivalence-based labeling algorithms. The experimental results demonstrated that our method is efficient for various types of images. Like other conventional label-equivalence-based labeling algorithms, our method is suitable for pipeline processing, parallel implementation, systolic-array implementation, and/or hardware implementation.

According to the experimental results, our algorithm is more efficient than other labeling algorithms if we only do the labeling task. However, in the case where the contours of connected components are necessary, the CTL algorithm (Chang et al., 2004) should be selected, because it can provides such a work with almost no additional cost.

Acknowledgements

We thank the anonymous referees for their valuable comments that improved this paper greatly. This paper was partially supported by the TOYOAKI Schoolship Foundment, Japan.

References

Alnuweiri, H.M., Prasanna, V.K., 1992. Parallel architectures and algorithms for image component labeling. IEEE Trans. Pattern Anal. Machine Intell. 14 (10), 1024–1034.

Ballard, D.H., 1982. Computer Vision. Prentice-Hall, Englewood, New Jesey.

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

Gotoh, T., Ohta, Y., Yoshida, M., Shirai Y., 1987. Component labeling algorithm for video rate processing. In: Proc. SPIE. Advances in Image Processing, vol. 804, pp. 217–224.

Haralick, R.M., 1981. Some neighborhood operations. In: Real Time/Parallel Computing Image Analysis. Plenum Press, New York, pp. 11–35.

Hashizume, A., Suzuki, R., Yokouchi, H., et al., 1990. An algorithm of automated RBC classification and its evaluation. Bio Med. Eng. 28 (1), 25–32.

Hattori, T., 1990. A high-speed pipeline processor for regional labeling based on a new algorithm. In: Proc. Internat. Conf. on Pattern Recognition, NJ, pp. 494–496.

He, L., Chao, Y., Suzuki, K., 2008. A run-based two-scan labeling algorithm. IEEE Trans. Image Process. 17 (5), 749–756.

He, L., Chao, Y., Suzuki, K., Wu, K., 2009. Fast connected-component labeling. Pattern Recognition 42 (9), 1977–1987. doi:10.1016/j.patcog.2008.10.013.

Hirschberg, D.S., Chandra, A.K., Sarwate, D.V., 1979. Computing connected components on parallel computers. Comm. ACM 22 (8), 461–464.

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

Karnaugh, M., 1953. The map method for synthesis of combinational logic circuits. Trans. AIEE. pt I 72 (9), 593–599.

Komeichi, M., Ohta, Y., Gotoh, T., Mima, T., Yoshida, M., 1988. Video-rate labeling processor. In: Proc. SPIE. Image Processing II, vol. 1027, pp. 69–76.

Lumia, R., 1983. A new three-dimensional connected components algorithm. Comput. Vision, Graphics, Image Process. 23 (2), 207–217.

Lumia, R., Shapiro, L., Zungia, O., 1983. A new connected components algorithm for virtual memory computers. Comput. Vision, Graphics, Image Process. 22 (2), 287–300.

Manohar, M., Ramapriyan, H.K., 1989. Connected-component labeling of binary images on a mesh connected massively parallel processor. Comput. Vision, Graphics, Image Process. 45 (2), 133–149.

Martin-Herrero, J., 2007. Hybrid object labelling in digital images. Machine Vision Appl. 18 (1), 1–15.

Naoi, S., 1995. High-speed labeling method using adaptive variable window size for character shape feature. In: IEEE Asian Conf. on Computer Vision, vol. 1, pp. 408–411.

Nassimi, D., Sahani, S., 1980. Finding connected components and connected ones on a mesh connected parallel computer. SIAM J. Comput. 9 (4), 744–757.

Nicol, C.J., 1995. A systolic approach for real time connected-component labeling. Computer Vision and Image Understanding 61 (1), 17–31.

Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Trans. Systems Man Cybernet. 9, 62–66.

Ronsen, C., Denjiver, P.A., 1984. Connected Components in Binary Images: The Detection Problem. Research Studies Press.

Rosenfeld, A., 1970. Connectivity in digital pictures. J. ACM 17 (1), 146–160.

Rosenfeld, A., Kak, A.C., 1982, second ed. Digital Picture Processing, vol. 2 Academic Press, San Diego, CA.

Rosenfeld, A., Pfalts, J.L., 1966. Sequential operations in digital picture processing. J. ACM 13 (4), 471–494.

Schiloach, Y., Vishkin, U., 1982. An O(log n) parallel connectivity algorithm. J. Algorithms 3, 57–67.

Shirai, Y., 1987. Labeling connected regions. In: Three-Dimensional Computer Vision. Springer-Verlag, pp. 86–89.

Shoji, K., Miyamichi, J., 1995. Connected component labeling in binary images by run-based contour tracing. Trans. Institute Electron., Information Comm. Eng. D-II J83-D-II (4), 1131–1139 (in Japanese).

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

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

Wu, K., Otoo, E., Suzuki, K., 2009. Optimizing two-pass connected-component labeling algorithms. Pattern Anal. Appl. 12, 117–135.

Yang, X.D, 1988. Design of fast connected components hardware. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Ann Arbor, MI, pp. 937–944.