A Run-Based Two-Scan Labeling Algorithm

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

***Abstract*—We present an efficient run-based two-scan algorithm for labeling connected components in a binary image. Unlike conventional label-equivalence-based algorithms, which resolve label equivalences between provisional labels, our algorithm resolves label equivalences between provisional label sets. At any time, all provisional labels that are assigned to a connected component are combined in a set, and the smallest label is used as the representative label. The corresponding relation of a provisional label and its representative label is recorded in a table. Whenever different connected components are found to be connected, all provisional label sets concerned with these connected components are merged together, and the smallest provisional label is taken as the representative label. When the first scan is finished, all provisional labels that were assigned to each connected component in the given image will have a unique representative label. During the second scan, we need only to replace each provisional label by its representative label. Experimental results on various types of images demonstrate that our algorithm outperforms all conventional labeling algorithms.**

***Index Terms*—Connected components, labeling algorithm, linear-time algorithm, pattern recognition, run-length encoding.**

I. Introduction

LABELING connected components in a binary image is one of the most fundamental operations in pattern analysis (recognition) and computer (robot) vision [8], [32]. For example, labeling is indispensable 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. Because connected components in an image may have complicated geometric shapes and complex connectivity, labeling is said to be more time-consuming than any other fundamental operations such as noise reduction, interpolation, thresholding, and edge detection. Labeling cannot be completed by mere parallel local operation [23], but needs sequential operations [30], [32].

Manuscript received July 24, 2007; revised November 11, 2008. This work was completed while L. He was a Research Associate at The University of Chicago. This work was supported in part by the TOYOAKI Scholarship Foundation, Japan. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zhigang (Zeke) Fan.

L. He is with the Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute, Aichi 480-1198, Japan, and also with the College of Information and Computer Science, Shaanxi University of Science and Technology, Shaanxi, China (e-mail: helifeng@ist.aichi-pu.ac.jp).

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

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

Digital Object Identifier 10.1109/TIP.2008.919369

Many labeling algorithms have been proposed for addressing this issue. For ordinary computer architectures and pixel representation images, there are three kinds of labeling algorithms: 1) raster-scan and label-equivalence-resolving algorithms, 2) searching and label propagation algorithms, and 3) contour-tracing and label propagation algorithms.

Raster-scan and label-equivalence-resolving algorithms scan an image in the raster direction, and they assign a provisional label to a new pixel that is not connected to other previously scanned pixels. Provisional labels assigned to the same connected component are called equivalent labels. There are three methods for resolving label equivalences.

i) *Multiscans* [11], [13]. These algorithms scan the image in forward and backward raster directions alternately to propagate the label equivalences until no label changes. The number of scans depends on the geometrical complexity of connected components. For example, for an N-step stair-like connected component, $2(N-1)$ scans are necessary [43].

ii) *Two scans* [9], [12], [19]–[21], [24], [30], [31], [40]. These algorithms store the label equivalences that are found in the first scan in a 1-D or 2-D table array. After the first scan, the label equivalences are resolved by use of a search algorithm. The resolved results are generally stored in a 1-D table. Then, the provisional labels are replaced by the smallest equivalent label with use of the table during the second scan. To label an object pixel, these algorithms need to compute the minimal label in the mask corresponding to the pixel, and the same label equivalence may be stored multiple times [32], or a very large memory, i.e., an $L \times L$-size memory, is necessary for storing label equivalences, where L is the number of provisional labels assigned to an image [9]. Moreover, the computational complexity of the search algorithms proposed so far is usually proportional to the order of L^2, where L is also the number of provisional labels assigned to an image [10].

iii) *Four scans*. This algorithm [43] is a hybrid between multiscan algorithms and two-scan algorithms. Like multiscan algorithms, the hybrid algorithm scans an image in the forward and backward raster directions alternately. During the scans, as in two-scan algorithms, a 1-D table is used for recording and resolving label equivalences. According to the results in [43], four is the upper limit on the number of scans, and the algorithm was faster than other raster-scan and label equivalence resolving algorithms.

Searching and label propagation algorithms [32], [39] search an image until an unlabeled object is found, and they assign a new label to it. Then the label is repeatedly propagated to neighboring connected objects until the whole component is labeled. Such processings are executed iteratively until there is

no unlabeled object in the image; they process an image in an irregular way.

Contour tracing and label propagation algorithms [2], [4] are somewhat similar to searching and label propagation algorithms. They first search an unlabeled border pixel of a component, and they assign a new label to it. Then they trace the whole border of the components, and mark all pixels in the border with the same labels. Such processings are executed iteratively until there is no unlabeled border pixel in the image. Moreover, at each row, all pixels that are consecutive from a labeled border pixel are assigned the same label as the border pixel. They also process an image in an irregular way.

On the other hand, some labeling algorithms [1], [3], [5], [16], [22], [25], [28], [38], [45] have been proposed for parallel machine models, such as a mesh-connected massively parallel processor. Other algorithms [6], [7], [15], [33], [35]–[37], [44], [46] have been proposed for the images represented by hierarchical tree [42], [17], [34], i.e., *n*-ary trees such as bintree, quadtree, octree, etc. Moreover, the hardware implementation of the above raster scan label-equivalence-resolving algorithms were also studied [1], [14], [26], [27], [47].

In this paper, we propose a run-based, efficient two-scan connected-component labeling algorithm. The run data that are obtained during the first scan are recorded in a queue, and are used for detecting the connectivity in the further processing. At any time, all provisional labels that are assigned to a connected component so far during the first scan are combined in a provisional label set, and the smallest label is used as the representative label. For each current run, if it is not connected to any previous runs in the row above the scan row, the pixels in the current run are assigned a new provisional label; otherwise, i.e., if there are some runs in the row above the scan row that are connected to the current run, the pixels in the current run is assigned the same provisional label that was assigned to the leftmost one, and all provisional label sets corresponding to these runs are merged with the smallest label as the representative label. Thus, when the first scan is finished, all provisional labels that were assigned to each connected component in the given image will be combined in the same provisional label set, and will have a unique representative label. During the second scan, each provisional label is replaced by its representative label.

Like other raster-scan labeling algorithms (multiscan algorithms, two-scan algorithms, and hybrid algorithm), our algorithm is based on label equivalence. However, unlike these conventional algorithms, which resolve the label equivalences between provisional labels, we resolve the label equivalences between provisional label sets. This leads our algorithm to be more efficient than others.

Two run-based connected-component labeling algorithms have been proposed for compressed images with run-length encoding. One, proposed in [39], is an improvement of the propagation-type algorithm proposed in [32] by use of block sorting and tracing of runs, and label propagation to the connected runs. It performs a searching step and a propagation step iteratively on the run data. In the searching step, the image is scanned until an unlabeled run is found; then the run is assigned a new label. In the propagation step, the new label propagates to neighbor runs above or below the current row until all the runs that belong to the connected component are labeled with the same label.

The other algorithm, proposed in [41], is a run-based contour-tracing algorithm. It records run data in the order of raster scanning and performs labeling by local and global run tracing (contour tracing). In the local run tracking, the next run for run tracking along the contour is determined. In global run tracing, the run is labeled by tracing the contour at each run.

The above two run-based labeling algorithms are tracing-type and need preprocessing; thus, they are not suitable for pipeline processing and parallel implementation.

In comparison with the two conventional run-based labeling algorithms, our algorithm is much simpler, is raster-scan-based (i.e., processes an image in a regular way), needs no preprocessing, and is suitable for parallel implementation.

Experimental results on various types of images, as well as various-sized noise images show that our algorithm is faster than the fastest raster-scan labeling algorithm, as found in [43], and the fastest labeling algorithm, as found in [4].

The rest of this paper is organized as follows. We introduce our algorithm in the next section and consider its efficient implementation in Section III. In Section IV, we show the experimental results. We compare our algorithm with conventional two-scan labeling algorithms and the contour-tracing algorithm in Section V and give our concluding remarks in Section VI.

II. Outline of Our Proposed Algorithm

For an $N \times M$ binary image (where the original image is binarized into the object and the background, and N is the matrix size), we use $p(y \times N + x)$ to denote the pixel value at (x, y) in the image, where $1 \leq x \leq N$ and $1 \leq y \leq M$, and V_O for the pixel value for the object (called *object pixels*) and V_B for that of the background (called *background pixels*). We assume that V_O and V_B are larger than any possible number of provisional labels (a safe method, for example, is using a number larger than the largest possible number of provisional labels, i.e., $N \times M/4$, for an $N \times M$ image), and $V_O < V_B$. Similar to most labeling algorithms, we assume that all pixels on the border of an image are background pixels.

A *run* is a block of contiguous object pixels in a row. A run from $p(s)$ to $p(e)$ in an $N \times M$ image is described by $r(s, e)$. The run data can easily be obtained and recorded during the first scan.

Let $r(s, e)$ be the current run. Then, by eight-connected connectivity, all runs in the row above the scan row such that one of their pixels occurs between $p(s - N - 1)$ and $p(e - N + 1)$ are connected to the current run, as shown in Fig. 1. In other words, a run $r(a, b)$ such that $a \leq e - N + 1$ and $b \geq s - N - 1$ is eight-connected with the current run $r(s, e)$. All such runs, including the current run itself, belong to the same connected component. On the other hand, if there is no such run existing in the previous row above the scan row, the current run is not connected to any previous run.

In our algorithm, at any point, all provisional labels that are assigned to a connected component found so far during the first scan are combined in a set $S(r)$, where r is the smallest label and is referred to as the *representative label*. We use a table R, called

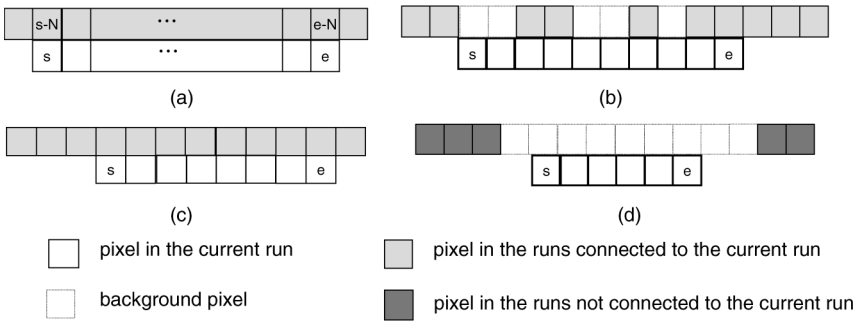


Fig. 1. Eight-connected connectivity for the current run. (a) Eight-connected pixel region for the current run; (b), (c) samples of runs eight-connected with the current run; (d) sample of no run eight-connected with the current run.

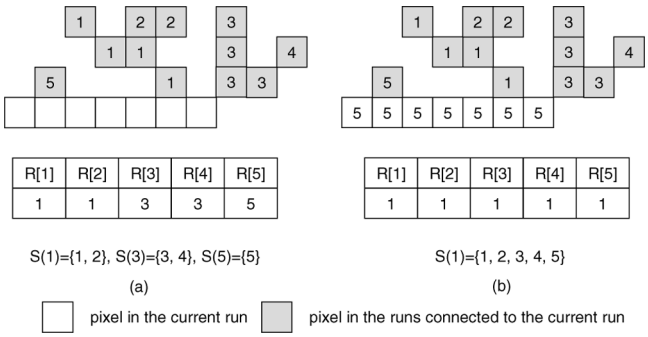


Fig. 2. Operations for label equivalence resolving: (a) before processing the current run; (b) after processing the current run.

the representative label table, to record the relation between a label in $S(r)$ and its representative label as follows:

$$\forall l \in S(r), R(l) = r.$$

In the first scan, from $i = 0$, our algorithm scans pixel $p(i)$ one by one in the given image in the raster scan direction. When a new run $r(s, e)$ is found, the run data are recorded. At the same time, the eight-connected area with the current run in the above row is detected. If there is no run eight-connected with the current run in the row above the scan row, the current run belongs to a new connected component that was not found so far. All pixels in the current run are assigned a new label l, the provisional label set corresponding to the connected component, i.e., the current run, is established as $S(l) = \{l\}$, and the representative label of l is set to l, i.e., $R(l) = l$.

On the other hand, if there are some runs eight-connected with the current run in the row above the scan row. Suppose that $S(r_1)$, ..., and $S(r_n)$ are the provisional label sets corresponding to those runs from left to right. Because, $S(r_1), \ldots, S(r_n)$, and the current run belong to the same connected component, $S(r_1), \ldots$, and $S(r_n)$ are merged to $S(r)$, where $r = min\{r_1, \ldots, l_n\}$, and all pixels in the current run are assigned the same provisional label as the upper left-most run, i.e., r_1.

For example, Fig. 2(a) shows a case before processing the current run and Fig. 2(b) shows the case after processing the current run.

Because whenever temporary connected components are found to be connected, all corresponding provisional label sets are merged with a single representative label, when the first scan finished, all provisional labels that belong to a connected component in a given image are merged in a corresponding set, and they have the same representative label. Thus, during the second scan, we need only to replace each provisional label with its representative label.

The correctness of our algorithm is quite straightforward from the above explanation.

III. Implementation

In our algorithm, we need to merge provisional label sets. By use of the connection-list data structure, we can complete the mergence of two sets in constant steps.

Because, in our algorithm, every provisional label belongs to only one provisional label set, and for an $N \times M$-size image, the maximum number of provisional labels is $N \times M/4$, we can use two $N \times M/4$-size 1-D arrays to realize connection lists for all provisional label sets. For convenience, hereafter we will use the term *provisional label lists* instead of provisional label sets.

One array, denoted $\mathrm{next}[\,]$, is used to represent the label next to the previous label in the list. $\mathrm{next}[i] = j$ means that the label next to label i is j. In particular, we use $\mathrm{next}[i] = -1$ for indicating that label i is the tail label of the list, i.e., there is no label next to label i. The other array, denoted $\mathrm{tail}[\,]$, is used for indicating the tail label of a provisional label set. $\mathrm{tail}[u] = v$ means that the tail label of the provisional label list $\mathcal{S}(u)$ is v.

On the other hand, the representative label table can also be implemented easily by use of an $N \times M/4$-size 1-D array, $\mathrm{rtable}[\,]$. Setting the representative label of a provisional label l to c can be made easily by the operation $\mathrm{rtable}[l] = c$, and for any provisional label u, its representative label v can be found simply by the operation $v = \mathrm{rtable}[u]$.

When we connect two provisional label lists $\mathcal{S}(m)$ and $\mathcal{S}(n)$, the head label of $\mathcal{S}(n)$ is connected to the tail label of $\mathcal{S}(m)$, and the resulting provisional label list is $\mathcal{S}(m) = \mathcal{S}(m) \cup \mathcal{S}(n)$; otherwise, i.e., if $m > n$, the head label of $\mathcal{S}(m)$ is connected to the tail of $\mathcal{S}(n)$. Therefore, for any provisional label list $\mathcal{S}(u)$, the representative label u certainly occurs as its head label.

In this way, with any member x of a provisional label list $\mathcal{S}(h)$, we can simply find its head label (the representative label) h by $h = \mathrm{rtable}[x]$, and its tail label t by $t = \mathrm{tail}[\mathrm{rtable}[x]]$.

Moreover, from the head label h, by using $\mathrm{next}[\,]$, we can find all of the members of the list one by one until we reach -1.

With $\mathrm{rtable}[\,]$, $\mathrm{next}[\,]$, and $\mathrm{tail}[\,]$, the creation and connecting of equivalent label lists can be made easily. The creation of a new equivalent label list $\mathcal{S}(m) = \{m\}$ can be made by the following operations:

$$\mathrm{rtable}[m] = m;$$
$$\mathrm{next}[m] = -1;$$
$$\mathrm{tail}[m] = m.$$

On the other hand, to connect two provisional label lists, say, $\mathcal{S}(u)$ and $\mathcal{S}(v)$, without loss of generality, suppose that $u < v$, i.e., $\mathcal{S}(v)$ is merged into $\mathcal{S}(u)$, we should connect the head of $\mathcal{S}(v)$ to the tail of $\mathcal{S}(u)$, set the tail of $\mathcal{S}(u)$ as the tail of $\mathcal{S}(v)$, and set u as the representative label for every label in list $\mathcal{S}(v)$. For doing this, the following simple operations, denoted *merge-operation*(u, v), are performed:

$i = v;$
while $(i \neq -1)$ do
  $\mathrm{rtable}[i] = u;$
  $i = \mathrm{next}[i];$
end of while
$\mathrm{next}[\mathrm{tail}[u]] = v;$
$\mathrm{tail}[u] = \mathrm{tail}[v].$

Let x and y be the provisional labels assigned to pixels in two runs r_1 and r_2, respectively. If r_1 and r_2 are found to be connected. The operations for resolving the equivalence of two provisional label sets that contain x and y, respectively, denoted to *resolve*(x, y), are as follows:

$u = \mathrm{rtable}[x], v = \mathrm{rtable}[y];$
if$(u < v)$
  merge-operation(u, v);
else if $(u > v)$
  merge-operation(v, u);
end of if

Notice that if $u = v$, which means that the two provisional labels x and y have the same representative label, i.e., they belong the same provisional label set, nothing needs to be done.

After the first scan, each object pixel $p(i)$ is labeled with a provisional label l, i.e., $p(i) = l$. If we set $\mathrm{rtable}[V_B] = V_B$ in advance, the process of replacing a provisional label with its representative label in the second scan can be completed easily by the following operation:

$$p(i) = \mathrm{rtable}[p(i)].$$

TABLE I
ALGORITHMS USED IN THE COMPARISONS

Algorithm	Type	Reference
Lumia	Multi-scan	Lumia et al. [20]
Shirai	Two-scan	Shirai [40]
Hybrid	Four-scan	Suzuki et al. [43]
Contour	Contour tracing	Chang et al. [4]
Ours	Run-based two-scan	Our algorithm

Now we consider how to record and use run data. In our algorithm, because we scan an image in the raster scan direction, the run that is found first will be used first. Therefore, we use queue data structures $s\_\mathrm{queue}[\,]$ (called *start-queue*) and $e\_\mathrm{queue}[\,]$ (called *end-queue*) to record the starting pixel number and ending pixel number, respectively.

Moreover, for the current run $r(s, e)$, any run $r(m, n)$ such that it ends before/at $p(e - N)$, i.e., $n <= e - N$, is impossible to be connected with any coming run, the related data can be deleted from the start-queue and end-queue after the current run is processed. In other words, after the current run $r(s, e)$ is processed, only the runs that end after/at $p(e - N + 1)$, need be recorded. Therefore, both the start-queue and end-queue can be realized by an $(N/2 + 1)$-size 1-D-array circular buffer.

The operations for processing the current run $r(s, e)$ in an image can be summaried as follows: we first record s and e in the start-queue and the end-queue, respectively. Then we check the runs that are recorded in the start-queue and end-queue one by one until a run $r(h, t)$ that ends after/at $s - N - 1$ is found (at least the current run $r(s, e)$ satisfies the condition, and all runs that end before $s - N - 1$ do not connect with the current run). If run $r(h, t)$ starts before/at $e - N + 1$, it connects to the current run $r(s, e)$, we label every pixel in $r(s, e)$ with the same label as that with which the pixel $p(h)$ was labeled. Moreover, because all following runs such that they end before/at $e - N$ connect to the current run, we resolve the label equivalences between them and the current run (such runs do not connect to the next current run and, therefore, are no longer necessary). Last, we check the next run, which ends after/at $e - N + 1$ (again, at least, the current run is such a run). If it starts before/at $e - N + 1$, it connects the current run; we resolve the label equivalence between it and the current run. Moreover, this run might connect to the next current run; its data h and t are kept in the program for consecutive processing.

On the other hand, if the run $r(h, t)$ starts after $e - N + 1$, it does not connect to the current run $r(s, e)$. In other words, the current run does not connect to any run that was found before. We label every pixel in the current run with a new label m, and increase m by one for the consecutive processing. Because run $r(h, t)$ might connect to the next current run, its data h and t are kept in the program for the consecutive processing.

IV. EXPERIMENTAL RESULTS

We implemented our algorithm with the C language on a PC-based workstation (Intel Pentium D 3.0 GHz + 3.0 GHz CPUs, 2-GB Memory, Mandriva Linux OS). All data in this section were obtained by averaging of the execution time for 10 000 runs.

TABLE II
COMPARISON OF VARIOUS EXECUTION TIMES [*msec*] ON VARIOUS KIND IMAGES

Image type		#CC	Lumia	Shirai	Hybrid	Contour	Ours
natural	max.	2660	2045.9	52.4	18.4	3.8	2.9
	mean	847	173.3	40.0	13.7	2.4	1.8
	min.	19	11.7	33.9	9.6	1.2	1.0
medical	max.	1525	807.7	41.7	14.9	2.6	1.8
	mean	281	192.1	38.8	13.4	1.9	1.5
	min.	20	15.3	37.2	11.4	1.5	1.2
textural	max.	372	72.7	42.8	28.5	3.7	2.5
	mean	83	35.2	39.3	27.4	2.7	1.8
	min.	1	15.6	38.5	26.4	1.5	1.1
special	max.	372	748.9	39.3	15.8	7.4	2.5
	mean	83	164.9	22.5	7.9	3.9	1.5
	min.	1	2.5	0.96	2.2	1.1	0.8

#CC, number of connected components

Test images are composed of 50 natural images (including landscape images, aerial images, fingerprint images, portrait images, still-life images, snap shot images, and text images, obtained from the Standard Image Database (SIDBA) developed by the University of Tokyo[1] and the image database of the University of Southern California[2] were used for realistic testing of labeling algorithms. In addition, seven texture images, which were downloaded from the Columbia-Utrecht Reflectance and Texture Database,[3] and 25 medical images obtained from a medical image database of The University of Chicago were used for testing. All of these images were transformed into binary images by means of Otsu's threshold selection method [29]. Moreover, specialized pattern (stair-like, spiral-like, saw-tooth-like, checker-board-like, and honeycomb-like connected components) images are used for testing. All of these images are 512×512 in size.

The algorithms used for comparison are listed in Table I. The comparison results are listed in Table II.

The results for six representative images are illustrated in Fig. 3, where the object pixels are displayed in black.

On the other hand, 41 noise images of each of six sizes (32×32, 64×64, 128×128, 256×256, 400×400, and 512×512 pixels) (a total of 246 images) were used for further comparing our algorithm and the fastest conventional raster-scan algorithm, i.e., the hybrid algorithm, as well as the fastest conventional algorithm, i.e., the contour-tracing labeling algorithm. For each size, the 41 noise images were generated by thresholding the images containing uniform random noise with 41 different threshold values from 0 to 1000 with a step of 25. Because connected components in such noise images have a complicated geometrical shape and complex connectivity, serious evaluation of labeling algorithms can be performed with these images.

Noise images of various sizes were used for testing the performance of the execution time versus the image size. As shown in Fig. 4, all three algorithms have the ideal linear characteristics which are plotted against the image size. It should be noted that, for all images, the maximum running time of our algorithm was smaller than the average time of the hybrid algorithm and the contour-tracing algorithm.

[1]http://sampl.ece.ohio- state.edu/data/stills/sidba/index.htm

[2]http://sipi.usc.edu/database/

[3]http://www1.cs.columbia.edu/CAVE/software/curet/index.php

Noise images with a size of 512×512 pixels were used for testing the execution time versus the number of object pixels in a given image. The results are shown in Fig. 5. The results show that our algorithm was at least four times faster than the the hybrid algorithm for any number of object pixels in an image, and was faster than the contour-tracing algorithm for all images, and in the worst case (maximum running time), our algorithm was 1.8 times faster than the contour-tracing algorithm.

V. COMPARISONS

We compared our algorithm with conventional two-scan labeling algorithms and the fastest conventional labeling algorithm, i.e., the contour-tracing labeling algorithm proposed in [4].

A. Comparisons With Conventional Two-Scan Labeling Algorithms

All conventional two-scan labeling algorithms are pixel-based (as opposed to run-based). In the first scan of conventional algorithms, the minimum provisional label in the mask shown in Fig. 6 is assigned to the object pixel; thus, calculation of the minimum provisional label in the mask is needed, whereas our algorithm is run-based; thus, all pixels in a run are assigned the same label without calculation of the minimum provisional label. Suppose that there are P object pixels and L runs in an image, where $P \geq L$. For an image that contains large connected components, i.e., $P >> L$, our algorithm should be much faster than conventional two-scan labeling algorithms. The results shown in Fig. 3 confirm this analysis, i.e., for the complicated image shown in Fig. 3(a), our algorithm is 20 times faster than the Shirai algorithm, whereas for the simple image shown in Fig. 3(b), which contains longer runs, our algorithm is 30 times faster than the same conventional two-scan algorithm.

In conventional two-scan labeling algorithms, an object pixel is assigned a new provisional label if there is not any object pixel in the mask, whereas in our method, any object pixel in the current run is not assigned a new provisional label only if there is not any run connected to the current run in the previous row. Obviously, the condition for assigning a new label in our algorithm is more strict than that in conventional two-scan labeling algorithms; thus, in general, the number of times for assigning provisional labels in our algorithm should be smaller than that in

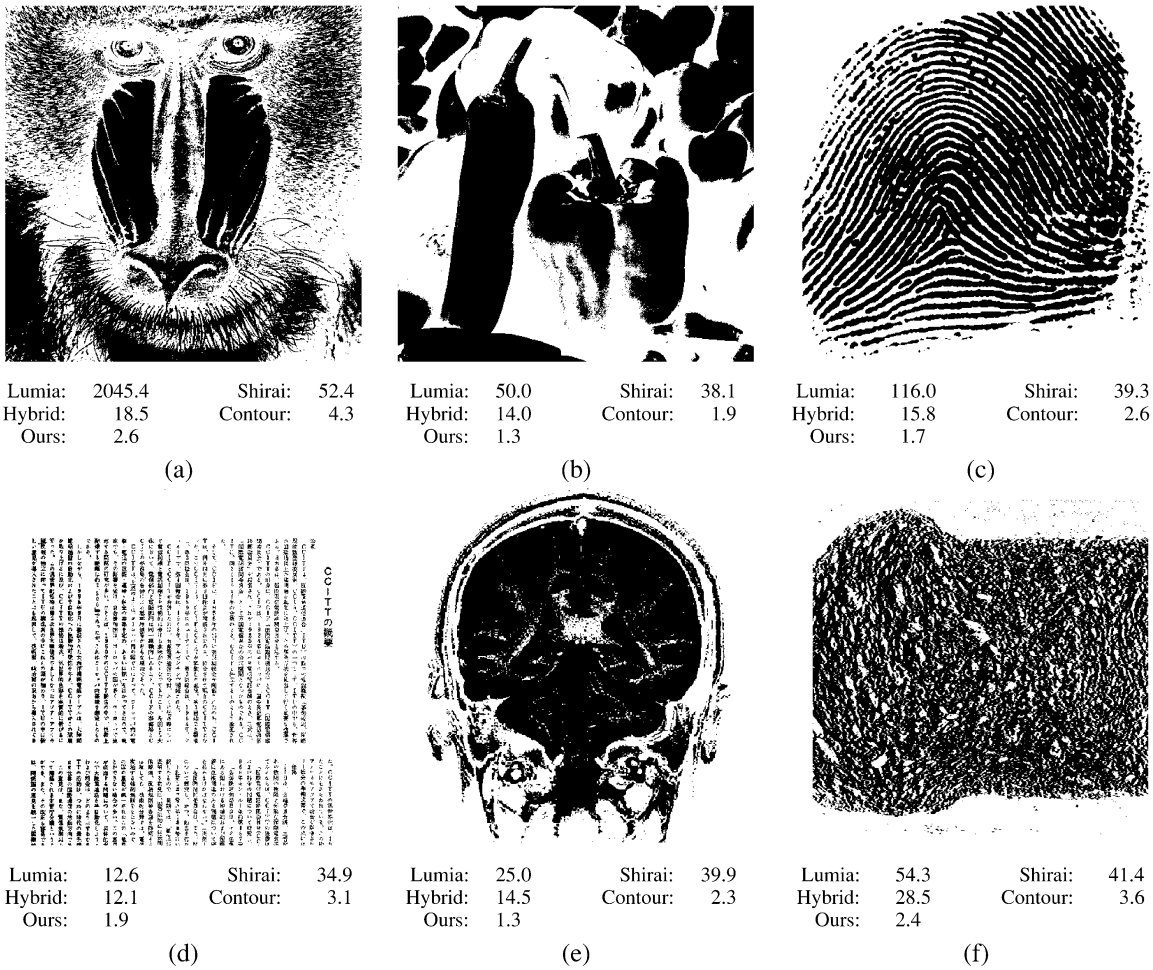


Fig. 3. Execution time [*msec*] of labeling algorithms for six representative images: (a) an animal image; (b) a still-life image; (c) a fingerprint image; (d) a Japanese text image; (e) a medical image; (f) a texture image.

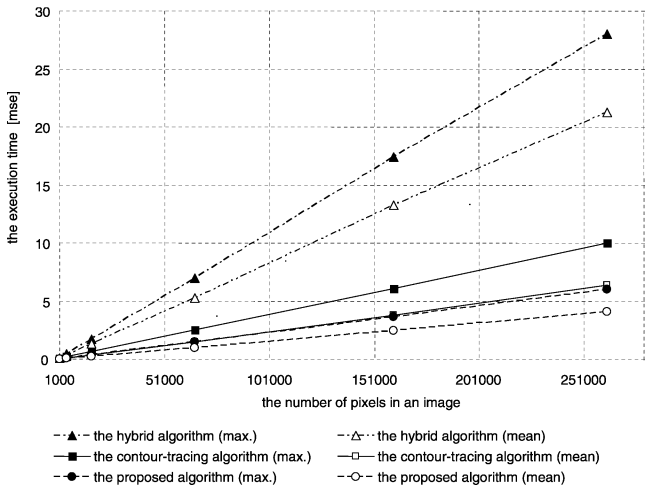


Fig. 4. Execution time versus image size.

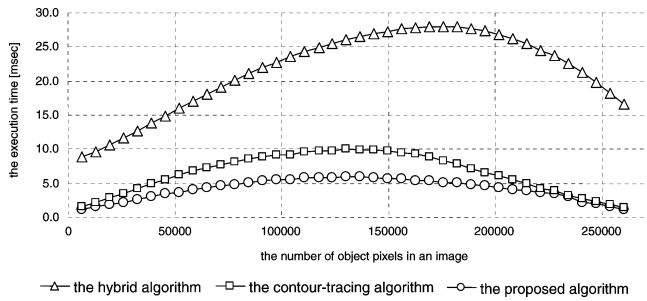


Fig. 5. Execution time versus the number of object pixels in an image.

conventional two-scan labeling algorithms. A simple example is shown in Fig. 7. Fig. 7(a) and (b) shows the provisional labels assigned to a connected component by a conventional two-scan algorithm and those assigned to the same connected component by our algorithm. The number of provisional labels assigned by the conventional raster-scan algorithm is larger than that by our algorithm.

Fig. 8 shows the number of provisional labels assigned by the conventional two-scan algorithm and our algorithm for different 512×512-pixel noise images. The number of provisional labels assigned by our algorithm is smaller than that by the conventional two-scan algorithm for any images.

The reduction in the number of provisional labels reduces not only the time for marking new labels, but also that for resolving label equivalences and, thus, leads to faster labeling.

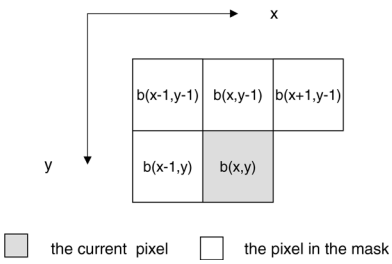


Fig. 6. Mask for the eight-connected connectivity.

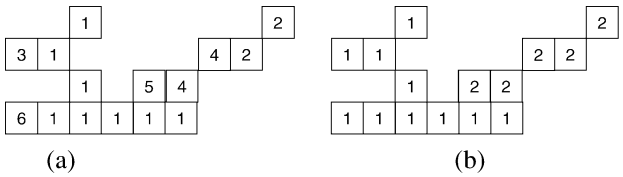


Fig. 7. Provisional labels assigned to a connected component after first scan (a) by a conventional two-scan algorithm and (b) by our algorithm.

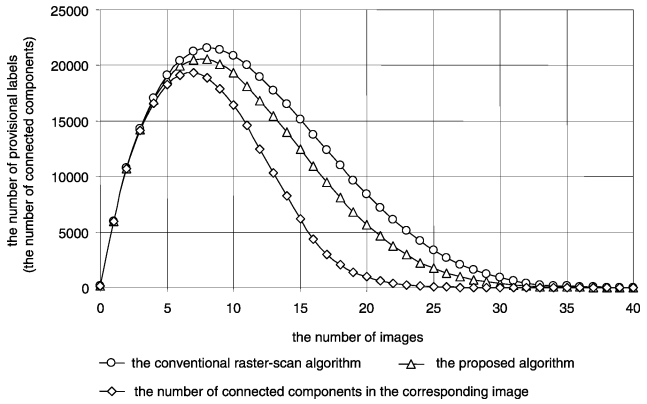


Fig. 8. Number of provisional labels assigned by different algorithms.

Moreover, conventional two-scan labeling algorithms record label equivalences during the the first scan, and then, i.e., after completion of the first scan, they resolve the label equivalences. After completion of resolving process for all label equivalences, the second scan can be started.

In comparison, our algorithm resolves any label equivalence as soon as it is found in the first scan. The label-equivalence resolving processing is independent from the provisional label assignment. At the same time of completion of the first scan, all label equivalences have been resolved; thus, the second scan can be started immediately after the first scan. Therefore, our algorithm is more suitable for pipeline processing and/or parallel implementation by use of the systolic array architecture [46].

B. Comparisons With the Contour-Tracing Labeling Algorithm

The contour-tracing labeling algorithm proposed in [4] is the fastest and newest conventional algorithm. It completes labeling by one scan (i.e., no re-labeling is required). Moreover, during labeling, it can extract connected-component contours and the sequential orders of contour points, which are also important for some image processing and pattern recognition. However, it need put an image in a larger array to avoid checking whether a pixel is on the boundary of the image. Moreover, because it accesses an image in an irregular way, it is not suitable for pipeline processing or parallel implementation. In comparison, our algorithm processes an image in the raster-scan order; therefore, it is suitable for pipeline processing and parallel implementation. Moreover, our algorithm is faster than the contour-tracing algorithm. On the other hand, for an $N \times M$-size image, our algorithm requires three $N \times M/4$-size 1-D arrays and two $(N/2+1)$-size 1-D arrays for resolving label equivalences. In addition, our algorithm does not provide connected-component contours and the sequential orders of contour points.

VI. Conclusion

In this paper, we proposed a run-based two-scan connected-component-labeling algorithm. Unlike conventional label-equivalence-resolving algorithms, which resolve label equivalences between provisional labels, we resolve label equivalences between provisional label sets. For resolving label equivalences, only three $N \times M/4$-size and two $(N/2+1)$-size 1-D arrays are needed. Our algorithm is very simple, and experimental results demonstrated that our algorithm is faster than all conventional labeling algorithms.

References

[1] H. M. Alnuweiri and V. K. Prasanna, "Parallel architectures and algorithms for image component labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 10, pp. 1024–1034, Oct. 1992.

[2] D. H. Ballard, *Computer Vision*. Englewood, NJ: Prentice-Hall, 1982.

[3] P. Bhattacharya, "Connected component labeling for binary images on a reconfigurable mesh architectures," *J. Syst. Arch.*, vol. 42, no. 4, pp. 309–313, 1996.

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

[5] A. Choudhary and R. Thakur, "Connected component labeling on coarse grain parallel computers: An experimental study," *J. Parallel Distrib. Comput.*, vol. 20, pp. 78–83, Jan. 1994.

[6] M. B. Dillencourt, H. Samet, and M. Tamminen, "A general approach to connected-component labeling for arbitrary image representations," *J. ACM*, vol. 39, no. 2, pp. 253–280, Apr. 1992.

[7] I. Gargantini, "Separation of connected component using linear quad- and oct-trees," in *Proc. 12th Conf. Numer. Math. Comput.*, Winnipeg, MB, Canada, 1982, vol. 37, pp. 257–276.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison Wesley, 1992.

[9] T. Gotoh, Y. Ohta, M. Yoshida, and Y. Shirai, "Component labeling algorithm for video rate processing," in *Proc. SPIE*, Apr. 1987, vol. 804, pp. 217–224.

[10] T. Goto, Y. Ohta, M. Yoshida, and Y. Shirai, "High speed algorithm for component labeling," *Trans. IEICE*, vol. J72-D-II, no. 2, pp. 247–255, 1989, (in Japanese).

[11] R. M. Haralick, "Some neighborhood operations," in *Real Time/Parallel Computing Image Analysis*. New York: Plenum, 1981, pp. 11–35.

[12] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Reading, MA: Addison-Wesley, 1992, pp. 28–48, volume I.

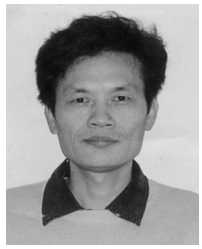
[13] A. Hashizume *et al.*, "An algorithm of automated RBC classification and its evaluation," *Bio Med. Eng.*, vol. 28, no. 1, pp. 25–32, 1990.

[14] T. Hattori, "A high-speed pipeline processor for regional labeling based on a new algorithm," in *Proc. Int. Conf. Pattern Recognition*, Jun. 1990, pp. 494–496.

[15] J. Hecquard and R. Acharya, "Connected component labeling with linear octree," *Pattern Recognit.*, vol. 24, no. 6, pp. 515–531, 1991.

[16] D. S. Hirschberg, A. K. Chandra, and D. V. Sarwate, "Computing connected components on parallel computers," *Commun. ACM*, vol. 22, no. 8, pp. 461–464, 1979.

[17] C. L. Jackins and S. L. Tanimoto, "Octrees and their use in representing 3D objects," *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 249–270, Nov. 1980.
[18] S. D. Kim, J. H. Lee, and J. K. Kim, "A new chain-coding algorithm for binary images using run- length codes," *Comput. Vis., Graph., Image Process.*, vol. 41, no. 1, pp. 114–128, 1988.
[19] M. Komeichi, Y. Ohta, T. Gotoh, T. Mima, and M. Yoshida, "Video-rate labeling processor," in *Proc. SPIE*, Sep. 1988, vol. 1027, pp. 69–76.
[20] R. Lumia, L. Shapiro, and O. Zungia, "A new connected components algorithm for virtual memory computers," *Comput. Vis., Graph., Image Process.*, vol. 22, no. 2, pp. 287–300, 1983.
[21] R. Lumia, "A new three-dimensional connected components algorithm," *Comput. Vis., Graph., Image Process.*, vol. 23, no. 2, pp. 207–217, 1983.
[22] M. Manohar and H. K. Ramapriyan, "Connected component labeling of binary images on a mesh connected massively parallel processor," *Comput. Vis., Graph., Image Process.*, vol. 45, no. 2, pp. 133–149, 1989.
[23] M. Minsky and S. Papert, *Perceptron*. Cambridge, MA: MIT Press, 1969.
[24] S. Naoi, "High-speed labeling method using adaptive variable window size for character shape feature," in *Proc. IEEE Asian Conf. Computer Vision*, Dec. 1995, vol. 1, pp. 408–411.
[25] D. Nassimi and S. Sahani, "Finding connected components and connected ones on a mesh connected parallel compute," *SIAM J. Comput.*, vol. 9, no. 4, pp. 744–757, 1980.
[26] C. J. Nicol, "Design of a connected component labeling chip for real time image processing," in *Proc. IEEE Asia-Pacific Conf. Circuits and Systems*, Sydney, Australia, Dec. 1992, pp. 142–147.
[27] C. J. Nicol, "A systolic approach for real time connected component labeling," *Comput. Vis. Image Understand.*, vol. 61, no. 1, pp. 17–31, 1995.
[28] S. Olariu, J. L. Schwing, and J. Zhang, "Fast component libeling and convex hull computation on reconfigurable meshes," *Image Vis. Comput.*, vol. 11, no. 7, pp. 447–455, 1993.
[29] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
[30] A. Rosenfeld and J. L. Pfalts, "Sequential operations in digital picture processing," *J. ACM*, vol. 13, no. 4, pp. 471–494, Oct. 1966.
[31] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, Jan. 1970.
[32] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, 2nd ed. San Diego, CA: Academic, 1982, vol. 2.
[33] H. Samet, "Connected component labeling using quadtrees," *J. ACM*, vol. 28, no. 3, pp. 487–501, Jul. 1981.
[34] H. Samet, "The quadtree and related hierarchical data structures," *Comput. Surv.*, vol. 16, no. 2, 1984.
[35] H. Samet, "Computing geometric properties of images represented by linear quadtrees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 2, pp. 229–240, Jul. 1985.
[36] H. Samet and M. Tamminen, "An improved approach to connected component labeling of images," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami, FL, 1986, pp. 312–318.
[37] H. Samet and M. Taminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-10, no. 4, pp. 579–586, Jul. 1988.
[38] Y. Schiloach and U. Vishkin, "An o(log n) parallel connectivity algorithm," *J. Algorithms*, vol. 3, pp. 57–67, 1982.
[39] Y. Shima, T. Murakami, M. Koga, H. Yashiro, and H. Fujisawa, "A high-speed algorithm for propagation-type labeling based on block sorting of runs in binary images," in *Proc. 10th Int. Conf. Pattern Recognition*, 1990, pp. 655–658.
[40] Y. Shirai, "Labeling connected regions," in *Three-Dimensional Computer Vision*. New York: Springer-Verlag, 1987, pp. 86–89.
[41] K. Shoji and J. Miyamichi, "Connected component labeling in binary images by run-based contour tracing," *Trans. Inst. Electron., Inf. Commun. Eng.*, vol. J83-D-II, no. 4, pp. 1131–1139.
[42] S. N. Srihari, "Hierarchical representations for serial section images," in *Proc. 5th Int. Conf. Pattern Recognition*, 1980, pp. 1075–1080.
[43] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Comput. Vis. Image Understand.*, vol. 89, pp. 1–23, 2003.
[44] M. Taminen and H. Samet, "Efficient octree conversion by connectivity labeling," in *Proc. SIGGRAPH Conf.*, Minneapolis, MN, 1984, pp. 43–51.
[45] L. W. Tucker, "Labeling connected components on a massively parallel tree machine," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami, FL, 1986, pp. 124–129.
[46] K. B. Wang, T. L. Chia, and Z. Chen, "Parallel execution of a connected component labeling operation on a linear array architecture," *J. Inf. Sci. Eng.*, vol. 19, pp. 353–370, 2003.
[47] X. D. Yang, "Design of fast connected components hardware," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ann Arbor, MI, Jun. 1988, pp. 937–944.

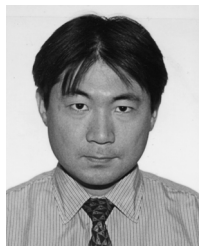


Lifeng He received the B.E. degree from the Northwest Institute of Light Industry, China, in 1982, the B.E. degree from Xian Jiaotong University, China, in 1986, and the M.S. and the Ph.D. degrees in AI and computer science from Nagoya Institute of Technology, Japan, in 1994 and 1997, respectively.

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 was with The University of Chicago, Chicago, IL, as a Research Associate. His research interests include image processing, automated reasoning, and artificial intelligence.

Yuyan Chao received the B.E. degree from the Northwest Institute of Light Industry, China, in 1984, and the M.S. and the Ph.D. degrees from Nagoya University, Japan, in 1997 and 2000, respectively.

From 2000 to 2002, she was a special foreign researcher with the Japan Society for the Promotion of Science at the Nagoya Institute of Technology. She is an Associate Professor at 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.



Kenji Suzuki (SM'04) received the 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 the Ph.D. degree (by published work) in information engineering from Nagoya University in 2001.

From 1993 to 1997, he was a Researcher with the Research and Development Center, Hitachi Medical Corporation. From 1997 to 2001, he was a faculty member with the Faculty of Information Science and Technology, Aichi Prefectural University, Japan. In 2001, he joined the Kurt Rossmann Laboratories for Radiologic Image Research as a Research Associate in the Department of Radiology, Division of the Biological Sciences, The University of Chicago, Chicago, IL. 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. He has published more than 100 scientific papers (including 45 peer-reviewed journal papers) in the fields of medical image analysis, machine learning, computer vision, and pattern recognition.

Dr. Suzuki has served as a referee for more than 15 journals, including the 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 RSNA in 2006. He is a member of IEICE, IEEJ, IPSJ, JNNS, and JCS.