YCVIU 1061 ARTICLE IN PRESS No. of pages: 15
DISK / 1/10/03 / Vimala(CE) / Panneer (TE) DTD 4.3.1 / SPS

ELSEVIER

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Computer Vision and Image Understanding xxx (2003) xxx–xxx

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

Notes

A linear-time component-labeling algorithm using contour tracing technique

Fu Chang,* Chun-Jen Chen, and Chi-Jen Lu

Institute of Information Science, Academia Sinica, 128 Academia Road, Section 2, Nankang, Taipei 115, Taiwan

Received 8 August 2003; accepted 8 September 2003

Abstract

A new linear-time algorithm is presented in this paper that simultaneously labels connected components (to be referred to merely as components in this paper) and their contours in binary images. The main step of this algorithm is to use a contour tracing technique to detect the external contour and possible internal contours of each component, and also to identify and label the interior area of each component. Labeling is done in a single pass over the image, while contour points are revisited more than once, but no more than a constant number of times. Moreover, no re-labeling is required throughout the entire process, as it is required by other algorithms. Experimentation on various types of images (characters, half-tone pictures, photographs, newspaper, etc.) shows that our method outperforms methods that use the equivalence technique. Our algorithm not only labels components but also extracts component contours and sequential orders of contour points, which can be useful for many applications.

Keywords: Component-labeling algorithm; Contour tracing; Linear-time algorithm

1. Introduction

Researchers often face the need to detect and classify objects in images. Technically, image objects are formed out of components that in turn are made of

* Corresponding author.
E-mail addresses: fchang@iis.sinica.edu.tw (F. Chang), dean@iis.sinica.edu.tw (C.-J. Chen), cjlu@iis.sinica.edu.tw (C.-J. Lu).

1077-3142/$ - see front matter
doi:10.1016/j.cviu.2003.09.002

connected pixels. It is thus most equitable to first detect components from images. When objects have been successfully extracted from their backgrounds, they also need to be specifically identified. For the latter purpose, component contour is often a useful resource for identifying objects. There are methods that identify objects from either chain codes [5] or Fourier descriptors [12], which are derived from object contours. There are also methods that match object contours against certain stochastic models [9]. These methods demonstrate that both component and contour labeling is an effective method for detecting and identifying two-dimensional objects.

In this paper, we present a method that simultaneously labels contours and components in binary images. This method is applicable in areas in which we must detect components and also classify them by means of certain contour features. Document analysis and recognition (DAR) in particular is an area for which our method is beneficial. High-order objects, such as half-tone pictures, characters, textlines, and text regions, need to be classified in order to effectively perform DAR [1]. Components are the basic ingredients of all high-order objects. Labeling components is therefore a commonly used technique for extracting high-order objects. The objective of DAR is not simply to extract high-order objects, but to recognize individual characters found within textual areas. There are many methods that employ certain contour features for classifying characters [2,10,16].

Our method labels each component using a contour tracing technique. This method is based on the principle that a component is fully determined by its contours, just as a polygon is fully determined by its vertices. This method also provides a procedure for finding all component pixels. We scan an image the same way as it would be encountered by a scanner, i.e., from top to bottom and from left to right per each line. When an external or internal contour is encountered, we use a contour-tracing procedure [6] to complete the contour and assign a label, say L, to all pixels on the contour. When the contour is traced back to its starting point, we resume scanning at that point. Later on, when the contour pixels labeled L are visited again, we assign the same label L to black pixels that lie next to them.

Our method has the following advantages. First, it requires only one pass over the image. Contour points are visited more than once due to the aforementioned contour tracing procedure, however no more than a constant number of times. Second, it does not require any re-labeling mechanism. Once a labeling index is assigned to a pixel, its value is unchanged. Third, we obtain as by-products all contours and sequential orders of contour pixels. Fourth, experimental results show that our algorithm is faster than traditional component-labeling algorithms.

Our paper is organized as follows. A review of five traditional component-labeling algorithms is given in Section 2. The details of our method are described in Section 3. Analysis and proof of our algorithm are provided in Section 4. The experimental results of our method as compared with the five algorithms from Section 2 are discussed in Section 5. A brief conclusion is given in Section 6.

2. Review of traditional component-labeling algorithms

In this section, we review five important methods for component labeling. One of them is the first proposed method, and the other four use varied strategies in attempt to improve on the first. They all attempt to re-label component pixels according to an equivalence relation induced by 8-connectivity. The first method proposed by Rosenfeld and Pfaltz [13] performs two passes over a binary image. Each point is encountered once in the first pass. At each black pixel P, a further examination of its four neighboring points (left, upper left, top, and upper right) is conducted. If none of these neighbors carries a label, P is assigned a new label. Otherwise, those labels carried by neighbors of P are said to be equivalent. In this case, the label of P is replaced by the minimal equivalent label. For this purpose, a pair of arrays is generated, one containing all current labels and the other the minimal equivalent labels of those current labels. In the second pass, label replacements are made.

Haralick [8] designed a method to remove the extra storage required for the pair of arrays proposed in the first method. Initially, each black pixel is given a unique label. The labeled image is then processed iteratively in two directions. In the first pass, conducted from the top down, each labeled point is reassigned the smallest label among its four neighboring points. The second pass is similar to the first, except that it is conducted from the bottom up. The process goes on iteratively until no more labels change. The memory storage of this method is small, but the overall processing time varies according to the complexity of the image being processed.

The method proposed by Lumia et al. [11] compromises between the two previous methods. In the first top-down pass, labels are assigned to black pixels as in the first method. At the end of each scan line, however, the labels on this line are changed to their minimal equivalent labels. The second pass begins from the bottom and works similarly as the top-down pass. It can be proved that all components obtain a unique label after these two passes.

Fiorio and Gustedt [4] employ a special version of the union-find algorithm [15] in that it runs in linear time for the component-labeling problem (see also Dillencourt et al. [3]). This method consists of two passes. In the first pass, each set of equivalent labels is represented as a tree. In the second pass, a re-labeling procedure is performed. The operation used in the union-find technique serves to merge two trees into a single tree when a node in one tree bears an 8-connectivity relationship to a node in the other tree.

The method proposed by Shima et al. [14] is particularly suitable for compressed images in which a pre-processing procedure is required to transform image elements into runs. A searching step and a propagation step are exercised iteratively on the run data. In the searching step, the image is encountered until an unlabeled run (referred to as focal run) is found and is assigned a new label. In the propagation step, the label of each focal run is propagated to contiguous runs above or below the scan line.

3. Our method

In our method, we scan a binary image from top to bottom and from left to right per each line. We first provide an overview of this method as follows. Conceptually, we can divide the operations into four major steps that are illustrated in Figs. 1A–D.

In Fig. 1A, when an external contour point, say A, is encountered the first time, we make a complete trace of the contour until we return to A. We assign a label to A and to all points of that contour.

In Fig. 1B, when a labeled external contour point A' is encountered, we follow the scan line to find all subsequent black pixels (if they exist) and assign them the same label as A'.

In Fig. 1C, when an internal counter point, say B, is encountered the first time, we assign B the same label as the external contour of the same component. We then trace the internal contour containing B and also assign to all contour points the same label as B.

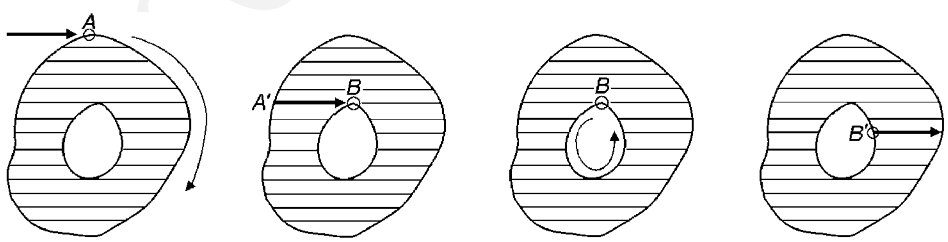
In Fig. 1D, when a labeled internal contour point, say B', is encountered, we follow the scan line to find all subsequent black pixels (if they exist) and assign them the same label as B'.

In the above procedure, we only make a single pass over the image and assign to each component point either a new label or the same label as the point preceding it on the san line. The details of this algorithm are given below.

For simplicity, we assume that the pixels in the uppermost row are all white (if they are not, we add a dummy row of white pixels). For a given document image I, we associate with I an accompanying image L, which stores the label information. Initially, all points of L are set to 0 (i.e., they are *unlabeled*). We then start to scan I to find a black pixel. Let C be the label index for components. Initially, C is set to 1. The aforementioned four conceptual steps can be reduced to three logical steps. The first step deals with a newly encountered external point and all points of that contour, the second step a newly encountered internal point and all points of that contour, and the third step all black pixels not dealt in the first two steps.

Let P be the current point that is being dealt by our algorithm.

Step 1: If P is unlabeled and the pixel above it is a white pixel (Fig. 2), P must be on an external contour of a newly encountered component. So we assign label C to P, meanwhile execute *contour tracing* (a contour tracing procedure whose details will



A **B** **C** **D**

Fig. 1. The four major steps in tracing and labeling component points.

YCVIU 1061 ARTICLE IN PRESS No. of pages: 15
DISK / 1/10/03 / Vimala(CE) / Panneer (TE) DTD 4.3.1 / SPS

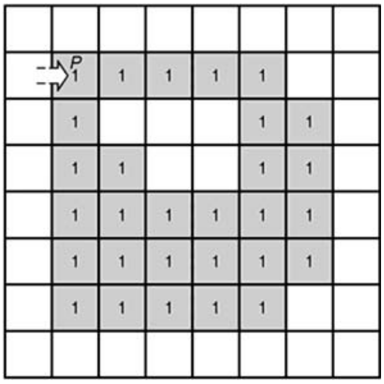
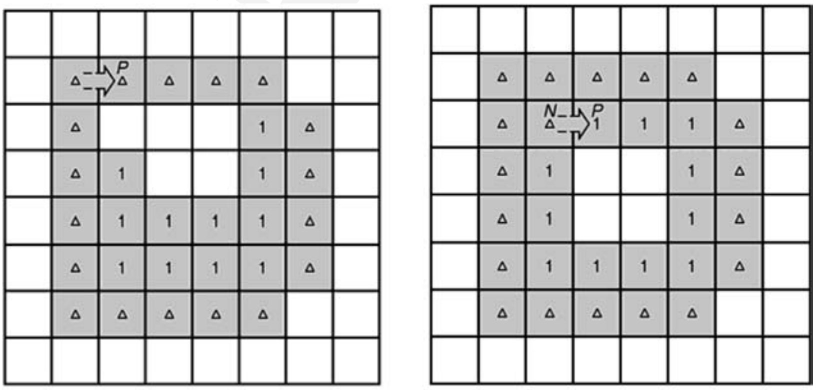


Fig. 2. P is the starting point of an external contour. 1, unlabeled black pixels.

be given later) to find that external contour, and assign label C to all the contour pixels. We then increase the value of C by 1.

Step 2: If the pixel below P is an *unmarked* white pixel (the meaning of 'unmarked' will be given in a moment), P must be on a newly encountered internal contour. There are two possibilities. First, P is already labeled (Fig. 3A). In this case, P is also an external contour pixel. Second, P is unlabeled (Fig. 3B). In this case, the preceding point N on the scan line (the left neighbor of P) must be labeled. We then assign P the same label as N. In either case, we proceed to execute *contour tracing* to find the internal contour containing P, and assign the same label to all the contour pixels.

Step 3: If P is not a point dealt in Step 1 or Step 2 (i.e., P is not a contour point), then the left neighbor N of P must be a labeled pixel (Fig. 4). We assign P the same label as N.



A B

Fig. 3. (A) P is the starting point of an internal contour. P also lies on an external contour. (b) P is the starting point of an internal contour, but it is not on an external contour. 1, unlabeled black pixels; △, labeled black pixels.

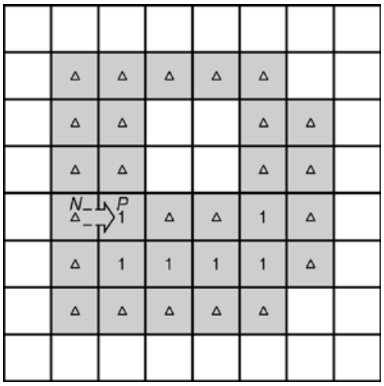


Fig. 4. *P* is an unlabeled point and its left neighbor *N* is already labeled. 1, unlabeled black pixels; △, labeled black pixels.

As is illustrated in Fig. 5, in order to avoid executing *Counter Tracing* at the point *Q*, we *mark* surrounding white pixels of a component with a negative integer. Thus, at the time the scan line sweeps *Q*, the pixel below *Q* is no longer an *unmarked* white pixel. On the other hand, the neighbor below a first encountered internal contour pixel *P* (Fig. 5) is still unmarked since the internal contour containing that pixel has not been traced yet.

By marking surrounding white pixels, we also ensure that each internal contour is traced only once. As illustrated in Fig. 6, when the internal contour has been traced, the neighbor below *R* is no longer an unmarked pixel and we thus avoid tracing the internal contour once again when *R* is encountered by the scan line (we need to trace the internal contour at a point only when the white pixel below that point is unmarked).

The operation of marking surrounding white pixels with a negative integer is included in the procedure *Tracer*, which is called forth by the procedure *contour tracing*. Both procedures will be described below.

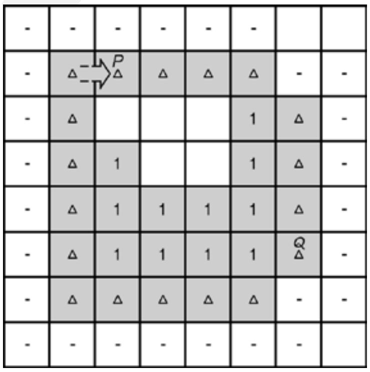


Fig. 5. Surrounding white pixels are marked with a native integer when a contour has been traced. 1, unlabeled black pixels; △, labeled black pixels; -, marked white pixels.

YCVIU 1061 ARTICLE IN PRESS No. of pages: 15
DISK / 1/10/03 / Vimala(CE) / Panneer (TE) DTD 4.3.1 / SPS

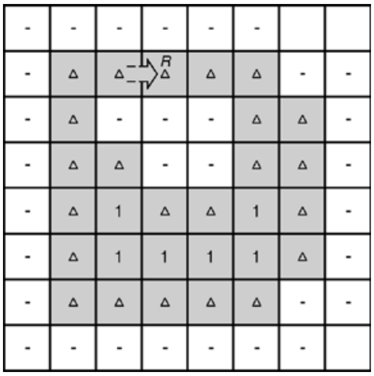


Fig. 6. Surrounding white pixels are marked with a negative integer when an internal contour has been traced. 1, unlabeled black pixels; △, labeled black pixels; -, marked white pixels.

3.1. Contour tracing

The goal of the procedure *contour tracing* is to find an external or internal contour at a given point, sat S. At point S, we first execute a procedure called *Tracer*. If *Tracer* identifies S as an isolated point, we reach the end of *contour tracing*. Otherwise, *Tracer* will output the contour point following S. Let this point be T. We then continue to execute *Tracer* to find the contour point following T and so on, until the following two conditions hold: (1) *Tracer* outputs S and (2) the contour point following S is T. The procedure stops only when both conditions hold. As shown in Fig. 7, when S is the starting point and T is the next contour point, the path traced by *Tracer* is $STUTSVWVS$.

3.2. Tracer

For a given contour point P, the goal of *Tracer* is to find among P's eight neighboring points for the contour point following P. The position of each neighboring point of P is assigned an index as shown in Fig. 8A. The search proceeds in a clockwise direction from the *initial* position that is determined in the following way.

If P is the starting point of an external contour, the initial position is set to 7 (upper right) since the point above P is known to be a white pixel and the next point

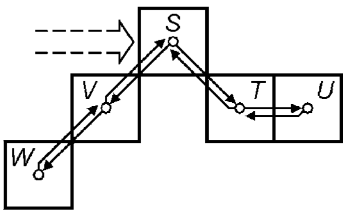


Fig. 7. Tracing the contour of a stripe-shaped component.

YCVIU 1061 ARTICLE IN PRESS No. of pages: 15
DISK / 1/10/03 / Vimala(CE) / Panneer (TE) DTD 4.3.1 / SPS

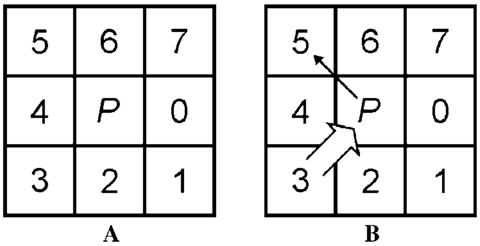


Fig. 8. (A) The neighboring points of P are indexed from 0 to 7. (B) If the previous contour point lies at 3, the next search direction is set to be 5.

in the clockwise direction is at position 7. If, however, P is the starting point of an internal contour, the initial position is set to 3 (lower left) since the point below P is also known to be a white pixel and the next point in the clockwise direction is at position 3. On the other hand, if the previous contour point exists and lies at position 3 (lower left), for example, then the initial position is set to 5 since the pixel at position 4 must have been visited already (Fig. 8B). In general, when P is not the starting point of a contour, irrespective of whether the contour is external or internal, its initial search position is set to $d + 2 \pmod 8$, where d is the position of the previous contour point.

Once the initial position is determined, we proceed in a clockwise direction to locate the first black pixel. This pixel is the contour point following P. If no black pixel is found in the whole circle, P is identified to be an isolated point.

Marking surrounding white pixels (with a negative integer) can be done as we search for the following contour point. As illustrated in Fig. 9, A is the current point and C is the following contour point. When tracing from A to C, we mark the white pixel B with a negative integer.

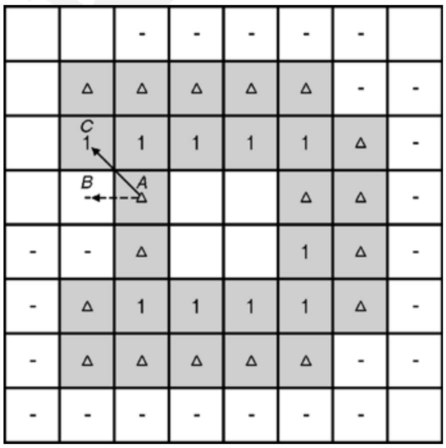


Fig. 9. When tracing an external or internal contour, we also mark the surrounding white points. 1, unlabeled black pixels; △, labeled black pixels; -, marked white pixels.

YCVIU 1061 ARTICLE IN PRESS No. of pages: 15
DISK / 1/10/03 / Vimala(CE) / Panneer (TE) DTD 4.3.1 / SPS

4. Complexity and efficacy

We first analyze the time complexity of our algorithm. The key lemma is as follows:

Lemma 1. *Our algorithm visits each pixel a constant number of times.*

Proof. Since the image is encountered only once, all non-contour pixels are visited exactly once. On the other hand, the number of times *contour tracing* visits a pixel is equal to the number of contours containing the pixel. A contour pixel can lie on at most four contours (Fig. 10). Thus, our algorithm scans each non-contour pixel only once and traces a contour pixel no more than four times. □

Since each pixel, when visited, takes a constant amount of time for processing, Lemma 1 immediately implies the following.

Theorem 2. *Our algorithm runs in linear time.*

We proceed to prove the efficacy of our algorithm. The contour tracing procedure is a well-known technique whose proof can be found in Haig et al. [7]. The pixel on a contour first encountered in the scanning process must be, due to the scanning direction, on the leftmost point on the uppermost row of that contour. We refer to this point as the *opening pixel* of the contour. Moreover, the opening pixel of the external contour of a component is also called the *opening pixel* of that component. Note that our algorithm ensures that each component is first encountered at its opening pixel and each contour is traced from its opening pixel.

It is also clear that all black pixels are labeled with a certain index by our algorithm. To prove the correctness of our algorithm, we need to show that all pixels

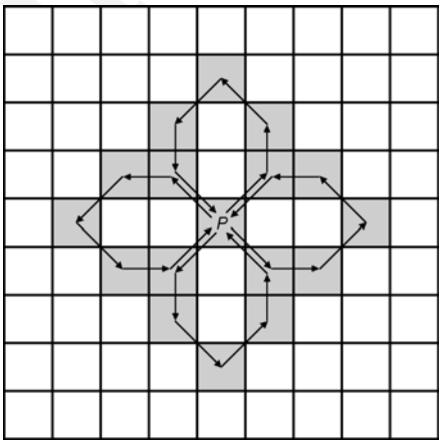


Fig. 10. An example in which a contour pixel *P* lies on four contours.

of the same component are assigned the same label and that all pixels of different components are assigned different labels.

Lemma 3. *All pixels in the same component are assigned the same label.*

Proof. Suppose that the opening pixel of the component is labeled C. According to Step 1 of our algorithm, all pixels on that external contour are also labeled C. To prove that all the remaining pixels of the same component are assigned the same label, we apply induction in the same order as they are encountered by the scan line. Suppose that the current pixel encountered is P. We assume that any component pixel encountered before P is labeled C. We then have to show that P is also labeled C. Assuming, without loss of generality, that P is *not* an external contour point (we already know that external contour pixels are labeled C), we consider the following three cases.

Case 1: P is not on any internal contour. In this case, P must be an interior point since P is not an external contour point either. It follows that the left neighbor Q of P is a black pixel. Since Q is encountered before P, Q is labeled C by our inductive hypothesis. So P is also labeled C, according to Step 3 of our algorithm.

Case 2: P is on an internal contour Φ but P is not the opening pixel of Φ. Let Q be the opening pixel of Φ. Q must be encountered before P, by the definition of an opening pixel. Q is labeled C by our inductive hypothesis. It follows that P is assigned the same label as Q, according to Step 2 of our algorithm. Thus, the label of P is C.

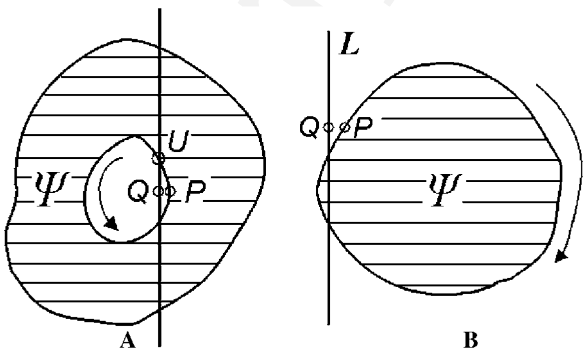


Fig. 11. (A) If P lies on an internal contour Ψ and its left neighbor Q is a white pixel, then Ψ contains a point U lying above Q. (B) Otherwise, the vertical line L that passes Q does not intersect with Ψ above Q. This implies that Q lies outside the component containing P, and that P is an external contour point.

Table 1
Six types of methods, including ours, that are being compared

A	Rosenfeld et al.
B	Haralick [8]
C	Lumia et al. [11]
D	Fiorio et al.
E	Shima et al. [14]
F	Our method

Case 3: P is the opening pixel of any internal contour containing P. In this case, the left neighbor Q of P is a black pixel for the following reason: if Q is a white pixel, P must lie on some internal contour Ψ, and Ψ contains a pixel U that lies on a row above Q (Fig. 11A) and also above P. This contradicts the fact that P is an opening pixel. We thus prove that Q is a black pixel. Since Q is encountered before P, Q is labeled C by our inductive hypothesis. P is thus labeled C, according to Step 2 of our algorithm.

In either case, P is labeled C, which completes our inductive step. □

Lemma 4. *Pixels in different components are assigned different labels.*

Proof. From the previous lemma, all pixels in a component are assigned the same label as the opening pixel of the component. On the other hand, Step 1 of our algorithm ensures that the opening pixel of a component is assigned a new label. So, different components get different labels. □

Table 2
Performances of the six methods being compared

Document type	Image size (M pixels)	#CC	Methods					
			A	B	C	D	E	F
			Average processing time (s)					
Legacy documents	2.16	741	0.50	0.41	0.70	0.10	0.07	0.06
	4.33	1668	1.95	1.02	1.21	0.19	0.15	0.13
	8.69	3708	7.03	3.27	4.08	0.38	0.30	0.27
	17.39	6570	29.95	6.54	8.08	0.74	0.55	0.49
Headlines	2.16	439	0.70	0.79	0.76	0.12	0.10	0.07
	4.33	916	2.35	1.79	1.56	0.24	0.19	0.14
	8.69	1577	7.48	3.91	3.31	0.47	0.37	0.30
	17.39	3145	39.20	9.20	6.11	0.89	0.71	0.58
Textual content	2.16	1808	1.78	1.02	0.76	0.12	0.12	0.08
	4.33	3509	6.45	2.50	1.53	0.23	0.24	0.15
	8.69	6825	25.29	6.30	3.10	0.47	0.45	0.31
	17.39	13,157	370.71	15.31	7.37	0.92	0.92	0.66
Halftone pictures	2.16	14,823	3.18	3.86	2.77	0.18	0.19	0.09
	4.33	28,793	14.51	10.09	5.08	0.37	0.42	0.19
	8.69	52,087	59.57	25.76	13.05	0.71	0.75	0.36
	17.39	131,628	773.93	100.83	28.13	1.41	1.68	0.76
Newspaper	2.16	1408	1.65	1.08	0.83	0.12	0.11	0.07
	4.33	4828	6.61	3.51	4.20	0.24	0.24	0.15
	8.69	12,024	25.99	7.94	5.05	0.51	0.52	0.32
	17.39	16,680	287.31	17.24	21.10	0.98	0.93	0.67
Photographs	1.92	4196	2.18	4.49	2.02	0.16	0.15	0.09
	3.14	2018	1.72	2.88	2.54	0.23	0.18	0.11
	3.87	3206	3.22	3.13	5.78	0.26	0.20	0.13
	4.91	1416	2.41	2.75	2.96	0.35	0.25	0.15

#CC, number of connected components.

The following theorem is a consequence of Lemmas 3 and 4.

Theorem 5. *Our algorithm produces a correct labeling for the components.*

5. Experimental results

Our methods are compared with the five other component-labeling methods discussed in Section 2. We use six types of test images: legacy documents, headlines, textual contents, half-tone pictures, newspaper, and photographs. The test environment is an Intel Pentium III 1 GHz personal computer with 384MB SDRAM. Each document type has four sets of images. Each set, in turn, consists of four images whose sizes correspond to four paper sizes: A3, A4, A5, and A6.

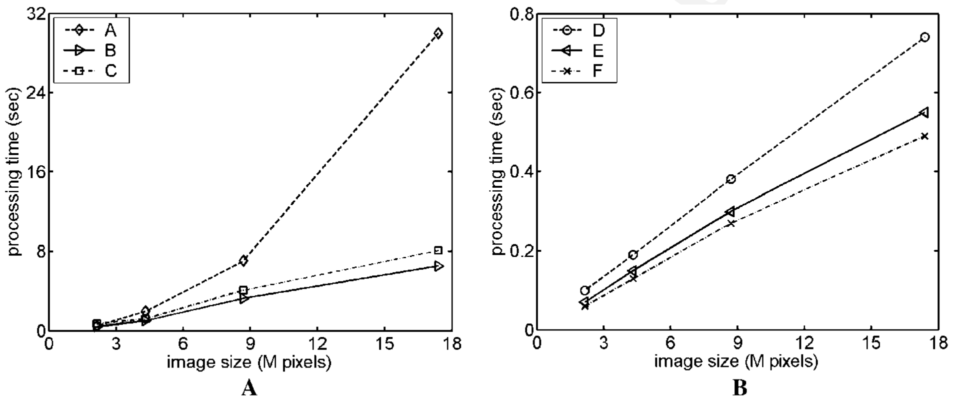


Fig. 12. Performances of the six methods for legacy documents.

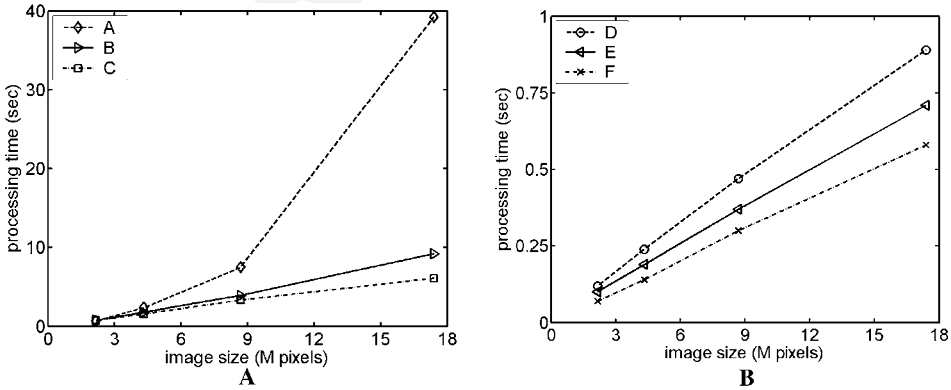


Fig. 13. Performances of the six methods for headlines.

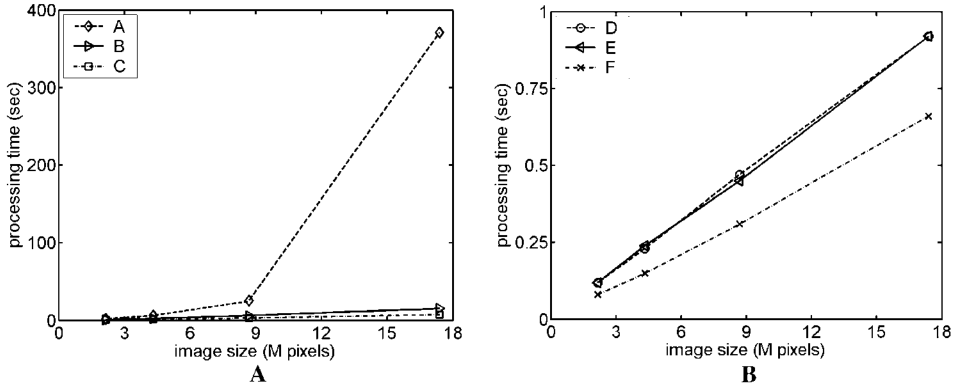


Fig. 14. Performances of the six methods for textual contents.

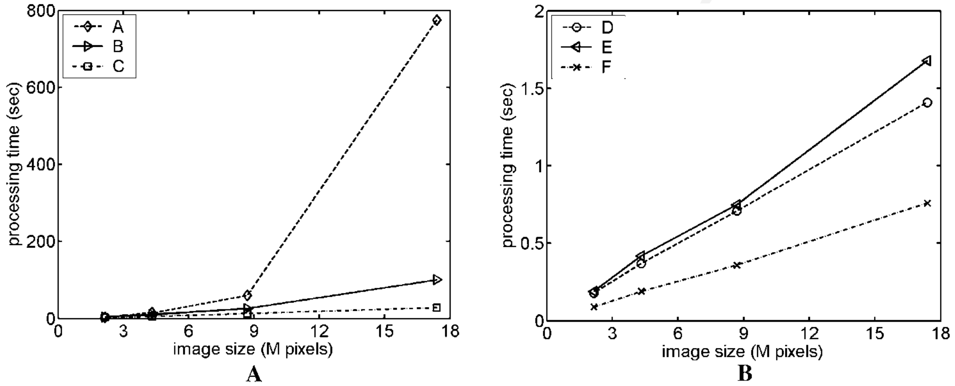


Fig. 15. Performances of the six methods for half-tone pictures.

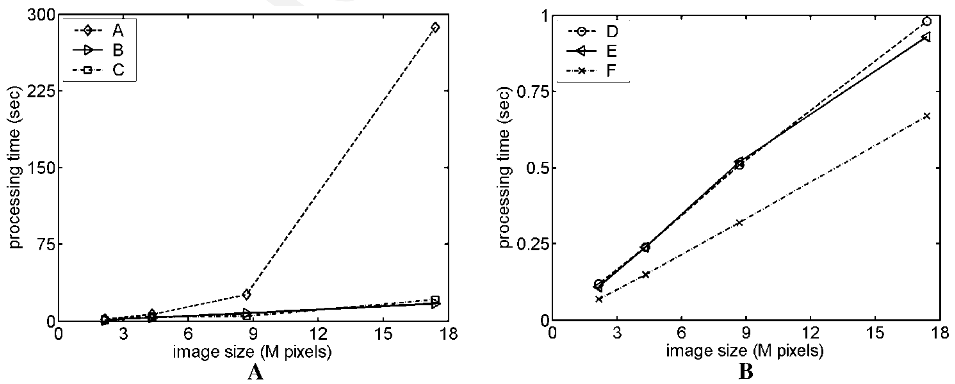
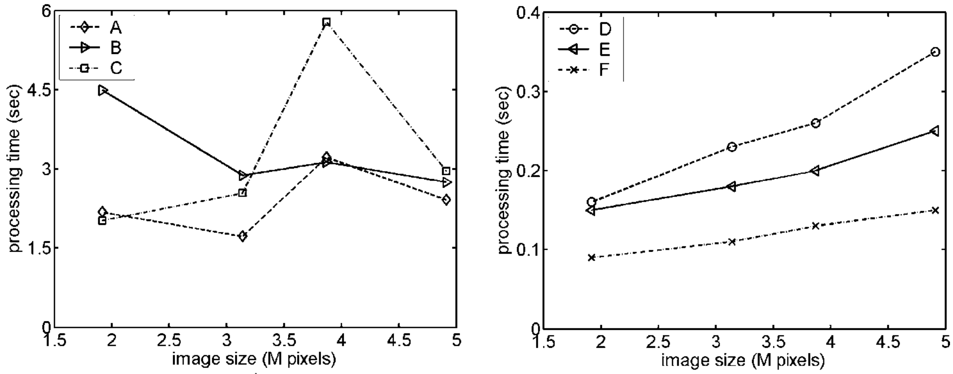


Fig. 16. Performances of the six methods for newspaper.



A B

Fig. 17. Performances of the six methods for photographs.

We have to make certain test images by cutting relevant objects from various sources and pasting them onto a blank canvas of a specified size. The reason we need to make a collage out of small images is because we are not able to obtain a document that consists of only a single type of specified objects (e.g., headlines). Some test images, however, can be directly segmented from their sources (e.g., photographs, newspaper, and legacy documents) without being made into collages. All six algorithms being compared are listed in Table 1. The comparison results are listed in Table 2. The performances of all the algorithms are shown in Figs. 12–17. In these figures, the size of the test image is plotted along the horizontal axis, and the average processing time of each method is plotted along the vertical axis.

6. Conclusion

We have presented a new component-labeling algorithm that employs contour tracing technique. This method scans a binary image only once and traces each contour pixel no more than a constant number of times. It is thus computationally effective in labeling connected components and also finding all contours and sequential orders of contour pixels. In experiments on six types of images of various sizes, we compare our method with five other algorithms. The results show that our algorithm outperforms all of them in terms of computational speed.

References

[1] F. Chang, Retrieving information from document images: problems and solutions, Internat. J. Document Anal. Recogn., Special Issues Document Anal. Office Syst. 4 (2001) 46–55.

[2] F. Chang, Y.C. Lu, T. Pavlidis, Feature analysis using line sweep thinning algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) 145–158.

[3] M.B. Dillencourt, H. Samet, M. Tamminen, A general approach to connected-component labeling for arbitrary image representations, J. Assoc. Comput. Mach. 39 (1992) 253–280.

YCVIU 1061 ARTICLE IN PRESS No. of pages: 15
DISK / 1/10/03 / Vimala(CE) / Panneer (TE) DTD 4.3.1 / SPS

[4] C. Fiorio, J. Gustedt, Two linear time Union-Find strategies for image processing, Theor. Comput. Sci. 154 (1996) 165–181.
[5] H. Freeman, Techniques for the digital computer analysis of chain-encoded arbitrary plane curves, Proc. Natl. Electron. Conf. (1961) 421–432.
[6] T.D. Haig, Y. Attikiouzel, An improved algorithm for border following of binary images, IEE Eur. Conf. Circuit Theory Design (1989) 118–122.
[7] T.D. Haig, Y. Attikiouzel, M.D. Alder, Border following: new definition gives improved borders, IEE Proc.-I 139 (1992) 206–211.
[8] R.H. Haralick, Some neighborhood operations, in: M. Onoe, K. Preston, A. Rosenfeld (Eds.), Real Time/Parallel Computing Image Analysis, Plenum Press, New York, 1981.
[9] Y. He, A. Kundu, 2-D shape classification using hidden Markov model, IEEE Trans. Pattern Anal. Mach. Intell. 13 (1991) 1172–1184.
[10] A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 4–37.
[11] R. Lumia, L. Shapiro, O. Zuniga, A new connected components algorithm for virtual memory computers, Comput. Vision Graphics Image Process. 22 (1983) 287–300.
[12] E. Persoon, K.S. Fu, Shape discriminations using fourier descriptors, IEEE Trans. Syst. Man Cybernet. 7 (1977) 170–179.
[13] A. Rosenfeld, P. Pfaltz, Sequential operations in digital picture processing, J. Assoc. Comput. Mach. 12 (1966) 471–494.
[14] Y. Shima, T. Murakami, M. Koga, H. Yashiro, H. Fujisawa, A high speed algorithm for propagation-type labeling based on block sorting of runs in binary images, Proc. 10th Internat. Conf. Pattern Recogn. (1990) 655–658.
[15] R.E. Tarjan, Efficiency of a good but not linear set union algorithm, J. Assoc. Comput. Mach. 22 (1975) 215–225.
[16] O.D. Trier, A.K. Jain, T. Taxt, Feature extraction methods for character recognition—a survey, Pattern Recogn. 29 (1996) 641–662.