Pattern Recognition 70 (2017) 25–43

Contents lists available at ScienceDirect

Pattern Recognition

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

The connected-component labeling problem: A review of state-of-the-art algorithms

Lifeng He[a,b,*], Xiwei Ren[a], Qihang Gao[a], Xiao Zhao[a], Bin Yao[a], Yuyan Chao[c]

[a] Artificial Intelligence Institute, College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Shaanxi 710021, PR China
[b] Faculty of Information Science and Technology, Aichi Prefectural University, Aichi 4801198, Japan
[c] Faculty of Environment, Information and Business, Nagoya Sangyo University, Aichi 4888711, Japan

ARTICLE INFO

Article history:
Received 10 June 2016
Revised 7 March 2017
Accepted 15 April 2017
Available online 22 April 2017

Keywords:
Connected-component labeling
Shape feature
Image analysis
Image understanding
Pattern recognition
Computer vision

ABSTRACT

This article addresses the connected-component labeling problem which consists in assigning a unique label to all pixels of each connected component (i.e., each object) in a binary image. Connected-component labeling is indispensable for distinguishing different objects in a binary image, and prerequisite for image analysis and object recognition in the image. Therefore, connected-component labeling is one of the most important processes for image analysis, image understanding, pattern recognition, and computer vision. In this article, we review state-of-the-art connected-component labeling algorithms presented in the last decade, explain the main strategies and algorithms, present their pseudo codes, and give experimental results in order to bring order of the algorithms. Moreover, we will also discuss parallel implementation and hardware implementation of connected-component labeling algorithms, extension for n-D images, and try to indicate future work on the connected component labeling problem.

1. Introduction

Following introduction in famous textbooks on digital image processing [1–3], for an $N \times N$-sized binary image,[1] the pixel at the coordinate (x, y), where $0 \le x \le N-1$ and $0 \le y \le N-1$, in the image is denoted as $b(x, y)$. When it is clear from the context, we also use $b(x, y)$ to denote the value of itself. Foreground pixels are also called object pixels. While not stated otherwise, we assume that the values of object pixels and background pixels are 1 and 0, respectively. Moreover, for convenience, we assume all pixels in the border of an image are background pixels.

For pixel $b(x, y)$, the four pixels $b(x-1, y)$, $b(x, y-1)$, $b(x+1, y)$, and $b(x, y+1)$ are called the *4-neighbors* of the pixel; the four-neighbors together with the four pixels $b(x-1, y-1)$, $b(x+1, y-1)$, $b(x-1, y+1)$, and $b(x+1, y+1)$ are called the *8-neighbors* of the pixel. Two object pixels p and q are said to be *8-connected* (*4-connected*) if there is a path which consists of object pixels a_1, a_2, ..., a_n such that $a_1 = p$ and $a_n = q$, and for all $1 \le i \le n-1$, a_i and a_{i+1} are 8-neighbor (4-neighbor) for each other. For example, object pixels p and q in Fig. 1(a) are 8-connected. An *8-connected (4-connected) component* in a binary image is the maximum set of object pixels in the image such that any of two pixels in the set are 8-connected (4-connected). A connected component is also called an *object*. For convenience, in this article, we will use connected component and object in exactly the same meaning. Moreover, because objects with 8-connectivity are more complicated than those with 4-connectivity, we will only consider 8-connectivity for objects. For example, there are four objects in Fig. 1(a).

For image analysis, image understanding, pattern recognition, and computer vision, we often change an image into a corresponding binary image, where pixels belonging to objects which we want to recognize are transfer to foreground pixels (object pixels) and all other pixels are transfer to background pixels. In order to distinguish different objects in a binary image, connected-component labeling is an indispensable operation, which consists in assigning a unique label to all pixels of each object in the image. After labeling, a binary image will be transferred to a labeled image. For example, Fig. 1(b) is a labeled image of the image shown in Fig. 1(a). Thus, after connected-component labeling, we can extract each object in the (labeled) image by its label, and then, further calculate its shape features such as area, perimeter, circularity, centroid etc. Because connected components in an image may have complicated geometric shapes and complex connectivity, connected-component labeling is said to be more time-consuming than any other fundamental operations on binary images such as

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

[1] For convenience, in this article we only consider $N \times N$-sized binary images. All algorithms can be easily extended to $N \times M$-sized binary images.

http://dx.doi.org/10.1016/j.patcog.2017.04.018

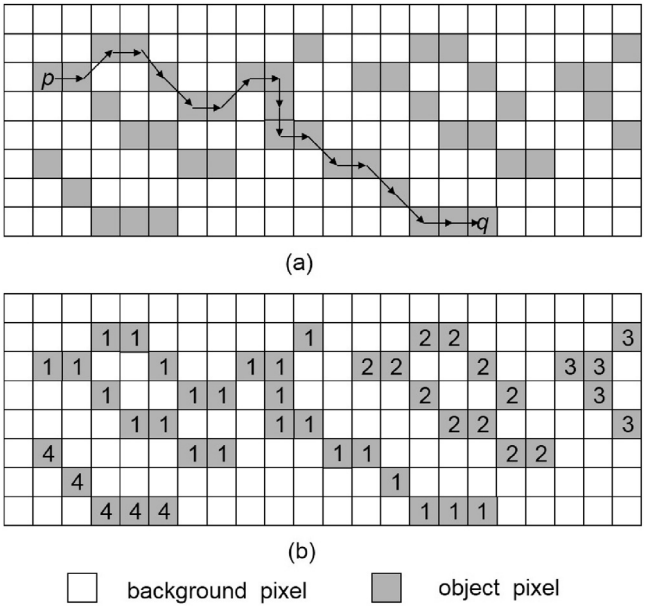

Fig. 1. 8-connected object pixels and connected components.

noise reduction, interpolation, thresholding, and edge detection. Especially, labeling cannot be completed by mere parallel local operation, but needs sequential operations [4].

A lot of connected-component labeling (CCL) algorithms have been proposed since the 1960s. According to the computer architecture and/or data structure being used, CCL algorithms can be divided into five classes: (1) algorithms for the images represented by special structures, for example, run-length structure and hierarchical tree structures, i.e., n-ary trees such as bintree, quadtree, octree, etc. [5–15]; (2) algorithms for parallel machine models such as a mesh-connected massively parallel processors or systolic array processors [16–31]; (3) algorithms for hardware implementation [29–38]; (4) algorithms for 3D and/or n-D images [39–42,98]; (5) algorithms for ordinary computer architectures such as the Von Neumann architecture and two-dimensional images.

Because images represented by special structures are rarely used in practice, we will not discuss CCL algorithms for such images especially. Moreover, because algorithms for ordinary computer architectures and two-dimensional images, i.e., algorithms in the class (5) are the base of algorithms in the other classes, we will mainly focus on the algorithms in class (5), and then discuss their parallel implementation, hardware implementation, and extension for n-D images.

For ordinary computer architectures and two-dimensional images, there are mainly two types of connected component labeling algorithms: algorithms based on label-propagation, and algorithms based on label-equivalence-resolving. In this article we review the methods and strategies of these algorithms proposed in the last decade, and present experimental results in order to bring order to these algorithms. Among others, the state-of-the-art algorithms shown in the following list will be especially reviewed in this paper:

(1) The Contour Tracing Labeling (CTL) algorithm proposed by F. Chang et al. in 2004 [43];
(2) The Hybrid Object Labeling (HOL) algorithm proposed by Herrero in 2007 [44];
(3) The Optimizing Connected-connected Labeling (OCL) algorithm proposed by Wu et al. in 2009 [46];
(4) The Improved Run-based Connected-component Labeling (IRCL) algorithm proposed by He et al. in 2010 [49];
(5) The Block based Connected-component Labeling (BCL) algorithm proposed by Grana et al. in 2010 [50];
(6) The Improved Block based Connected-component Labeling (IBCL) algorithm proposed by H. Chang et al. in 2015 [52];
(7) The Improved Configuration-Transition-based Connected-component Labeling (ICTCL) algorithm proposed by Zhao et al. in 2015 [53].

Among the above algorithms, the first two algorithms are label-propagation ones, and the others are label-equivalence ones.

There were some papers on object labeling comparisons. The paper presented in Ref. [54] reviewed some popular CCL algorithms presented in Refs. [4,43,60] in that time and mainly addressed the capability for real-time video processing, hardware implementation in FPGA for embedded systems, and memory requirements. Especially, this paper also reviewed a one-scan connected-component analysis (CCA) algorithm for objects' features by combined subsequent data analysis step into the first scan of a two-scan labeling algorithm. Without the need for buffering image data, it is very suitable for hardware implementation. The paper presented in Ref. [55] mainly compared two label-equivalence-resolving strategies used in some two-scan CCL algorithms. Moreover, the paper presented in Ref. [56] reviewed some main algorithms presented in Refs. [4,43,46–48,50,65], and provided benchmarks on different processor architectures.

However, in these papers, (1) the principles of corresponding algorithms were not discussed in detail; (2) the state-of-the-art labeling algorithms proposed in Refs. [44,49,52,53] mentioned above were not reviewed in these papers; (3) the strategies for resolving label equivalence for two-scan labeling algorithms were not explained in detail; (4) the pseudo codes of the reviewed algorithms are not given; (5) the single pass connected-component analysis algorithm discussed in Ref. [54] does not generate a labeled image, thus, as mentioned in Ref. [54], the algorithm is not suitable for applications where a labeled image is required; (6) As we will discuss in Section 5.2, any two-scan CCL algorithm reviewed in this paper can be modified as one-scan CCA algorithm in a similar way.

The rest of this paper is organized as follows: we review algorithms based on label propagation in the next section, and algorithms based on label-equivalent resolving in Section III. Comparison and experimental results on various images of state-of-the-art algorithms reviewed in this paper are presented and compared in Section IV. In Section V, we discuss parallel implementation, hardware implementation, and extension for n-D images. We give our concluding remarks and future work in Section 6.

2. Algorithms based on label-propagation

These CCL algorithms [42–44,57,58] first search an unlabeled object pixel, label the pixel with a new label; then, in the later processing, they propagate the same label to all object pixels that are connected to the pixel. Because the labels assigned to object pixels will never change, these algorithms can be easily extended to calculate the shape features of objects, such as area, centroid, perimeter etc., during the labeling.

Although these algorithms usually use the raster scan to find an unlabeled object pixel, for labeling, all of them access pixels in an image in an irregular way, depending on the shapes of connected components in the image. Therefore, they are essentially not a raster-scan-type algorithm; thus, they are not suitable for pipeline processing, parallel implementation, systolic-array implementation, and hardware implementation. Moreover, although the algorithms based on label-propagation are often called one-scan algorithms, many object pixels will be scanned more than twice, therefore, they are actually not one-scan algorithms.