南開學報 第四卷 第三期 民國九十六年
Journal of Nan Kai, Vol. 4, No. 3, pp.9-14 (2007)

A Binary Image Segmentation Algorithm Using Dynamic Stack

Shye-Chorng Kuo[1], Shyi-Shiun Kuo[2], Yu-Hua Yu[2], Meng-Tu Lee[2]

[1] Department of Electronic Engineering,
Nan Kai Institute of Technology
[2] Department of Computer Science and Information Engineering,
Nan Kai Institute of Technology

Abstract

Image segmentation is an essential task in image processing. Segmentation is the process that subdivides an image into its constituent parts or components. The algorithms of segmentation are generally based on discontinuity or similarity of the grey-values of pixels. The conventional methods are thresholding, edge-based approach, region growing, and region split/merge, etc. Generally, the segmentation algorithms are complicated and hard to be implemented by the computer program languages and they are often time consuming.

In this paper, we propose an algorithm to handle the process of a binary image segmentation by dynamic stack, which can successfully segment the different components within the binary image and the most important of all is that the algorithm can be easily implemented by the computer program languages because of the characteristics of operations on the stack with short time consumption.

Keywords: Image segmentation, binary image, stack, thresholding

1. Introduction

Image segmentation plays an important role in many applications of the image processing and pattern recognition whose performances are significantly influenced by the result of image segmentation. Large amounts of segmentation algorithms have been developed [4, 10]. Traditionally, these segmentation techniques can be classified into two categories: boundary representation and regional representation. The algorithms are based on one of two basic properties of grey-level values: discontinuity and similarity. In the boundary representation, the approaches are to partition an image based on abrupt changes in grey level, such as the detection of isolated points and the edges in an images [2, 5, 6, 11]. In the regional representation, the approaches are based on label region, region growing and split/merge algorithms [1, 3, 7, 8, 12]. In the field of image segmentation, the fundamental methods are applied to binary image. The effective method of segmentation of binary images is by examining the connectivity of pixels with their neighbors and labeling the connected sets. Two practical methods are Pixel labeling and Run-length connectivity analysis [9]. Often, the algorithms for segmentation are either complicated to be implemented by the computer program languages or time consuming.

In this paper, we present an algorithm that bases on the idea of the Pixel labeling and uses the stack to record the interesting pixels within the 8-neighbor of the current observed pixel by the basic operations of stack: push and pop. The

NATIONAL CENTRAL LIBRARY 國家圖書館 R.O.C.

algorithm we proposed is not time consuming since it doesn't use the arithmetic operations but only conditional operations and pushing or popping the data to or from the stack and can successfully and also efficiently segment the isolated components within the processing binary image .

We introduce the details of the algorithm in section 2. Section 3 presents the experimental results and the conclusions are drawn in section 4.

2. Method

Before we go on to discuss the algorithm, we are first to describe the hypotheses as below:

(1) Assume a binary image with *width* pixels in width and *height* pixels in height, where *width* and *height* are both integer values. The border of the image is with one pixel wide blank.

(2) Use a 2-D array to store the image and conveniently we declare the array as *image[width, height]*.

(3) If the pixel is a constituent part of a component in the original image, the pixel's color (or value) is equal to '1' , otherwise the pixel's color is equal to '0'.

(4) Define 8-neighbor of a pixel:

If a pixel is at the location (x,y) and with the color p, that is $p=$ *image[width, height]*, the locations of its 8 neighbors are defined as $(x,y-1)$, $(x+1,y-1)$, $(x+1,y)$, $(x+1,y+1)$, $(x,y+1)$, $(x-1,y+1)$, $(x-1,y)$, $(x-1,y-1)$.Then we define an array, $p[i], i=2\sim9$, to record the color of each 8-neighbor, where *p[2]=image[x,y-1]*, *p[3]= image[x+1,y-1]*, *p[4]= image[x+1,y]*, *p[5]= image[x+1,y+1]*, *p[6]= image[x,y+1]*, *p[7]= image[x-1,y+1]*, *p[8]= image[x-1,y]*, *p[9]= image[x-1,y-1]*.

(5) Define a dynamic stack：*Stack*

Stack records the x-y coordinates of a pixel and can dynamically change the length.

(6) Define the three operations on the stack:

IsEmpty(Stack): Reports whether the *Stack* is empty or not.

PUSH(Stack,x,y): Places the x-y coordinates of the current observed pixel on the top of the *Stack*.

POP(Stack,x,y): Removes the x-y coordinates of the pixel from the top of the *Stack*.

Now we describe the algorithm as follows:

The procedure begins to scan the pixels one by one within the 2-D binary image from left to right and top to down, that is, to scan the pixels row by row from top to down.

While the pixel's color is equal to 1, the algorithm chooses this pixel as a seed and marks this pixel with the successive color that it often increases the current value by one and creates a dynamic stack to push the x-y coordinates of the pixel whose color is equal to 1 among 8 neighbors, meanwhile, interrupts the scanning.

Then, pop the stack to obtain one neighbor as a seed and executes the previous steps until the stack is empty. When the stack is empty, one component is segmented from the image and the scanning continues from the previous interrupted pixel to search another new seed and the previous steps are repeatedly.

The procedure stops until the last pixel is scanned in the image, then all the components within the image are found and the pixel's colors of the different components are marked with the different ones.

In order to expound clearly, the explicit instructions of the algorithm are presented as follows:

```
color = 1
For y = 0 To height - 1
 For x = 0 To width - 1
  If (image(x, y) is equal to 1) Then
   color = color + 1
   image(x, y) = color
    Store the pixel color of each 8-neighbor to p[i], i=2~9.
   For i = 2 To 9
    If (p[i] is equal to 1) Then
       Call PUSH(Stack, xc, yc)
       /* xc: the x-coordinate of p[i],
        yc: the y-coordinate of p[i]
     End If
    End For
    While (IsEmpty(Stack) is equal to False)
      Call Pop(Stack, xc, yc)
      image(xc, yc) = color
       Store the pixel color of each 8-neighbor to p[i], i=2~9.
      For i = 2 To 9
       If (p[i] is equal to 1) Then
         Call PUSH(Stack, xc, yc)
         /* xc: the x-coordinate of p[i],
           yc: the y-coordinate of p[i]
       End If
      End For
```

NATIONAL CENTRAL LIBRARY R.O.C.

```
            End While /* End of while
        End if
    End For /* End of x for- loop
End For /* End of y for- loop
```

3. Experimental Results

In this section, three images are used to demonstrate the validity and the efficiency of our proposed algorithm. The first image is Target as shown in Fig. 1(a) and its split result is shown in Fig. 1(b), where the separated segments are spread with the different colors.

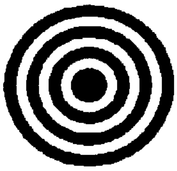

Fig. 1(a) Target

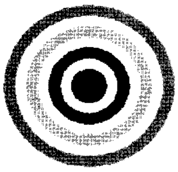

Fig. 1(b) The split result of Target

Second, a complicated texture image but is only with two separated components indeed and the split result are shown in Fig. 2(a) and Fig. 2(b) respectively.

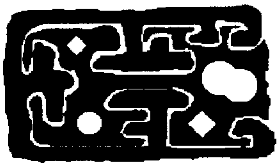

Fig. 2(a) A complicated texture image

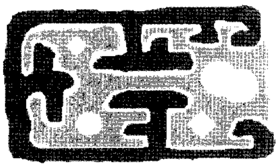

Fig. 2(b) The split result of the complicated texture image

Third, a Chinese and numeric character image is shown in Fig. 3(a) and its split result is shown in Fig. 3(b).

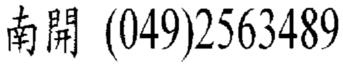
南開 (049)2563489

Fig. 3(a) A Chinese and numeric character image

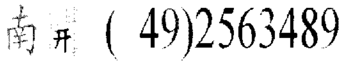
南 开 (49)2563489

Fig. 3(b) The split result of the Chinese and numeric character image

The first and the second test images are of the sizes of 320x 200 and the last is 796x108.

All the test images are in black and white BMP format and the resolutions are all in 300DPI. The proposed algorithm is implemented by using programming language Basic on Intel Celeron 2.8 GHz.

4. Conclusions

In this paper, we have proposed an efficient binary image segmentation algorithm that can partition the image into the significant segments robustly as showing in section 3 and is easily implemented by computer program languages and is also with short computation time.

Now, the proposed algorithm is only applied to binary image, but the applications of the binary image are not extensively used, where they are often used in OCR. Besides, the 3D images such as MR images are prevalent recently. Therefore, we will evolve the proposed algorithm to deal with the segmentation of color images and further the segmentation of 3D images in the future research.

References

[1] Adams, R., and Bischof, L., “Seeded region growing,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 6, pp. 641-647(1994).

[2] Canny, J. F., “A computational approach to edge detection,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 8, pp. 679-698(1986).

[3] Chang, Y. L., and Li, X., “Adaptive image

NATIONAL CENTRAL LIBRARY 國家圖書館 ROC

region-growing," *IEEE Trans. on Image Processing*, Vol. 3, No. 6, pp. 868-872(1994).

[4] Fu, K. S., and Mei, J. K., "A survey on image segmentation," *Pattern Recognition*, Vol. 13, pp. 3-16(1981).

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

[6] Helterbrand, J. D., "One-pixel-wide closed boundary identification," *IEEE Trans. on Image Processing*, Vol. 5, No. 5, pp.780-783(1996).

[7] Hojjatoleslami, S. A., and Kittler, J., "Region growing: a new approach," *IEEE Trans. on Image Processing*, Vol. 7, No. 7, pp. 1079-1084(1998).

[8] Hong, T. H., and Rusenfeld, A., "Compact Region Extraction Using Weighted Pixel Linking in a Pyramid," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 2, pp.222-229(1984).

[9] Jain, A.K., Fundamentals of Digital Image Processing, *Prentice-Hall*, pp.409-410(1989)

[10] Pal, R., and Pal, S. K., "A review in image segmentation techniques," *Pattern Recognition*, Vol. 26, pp. 1277-1294(1993).

[11] Sahoo, P. K., Soltani, S., and Wong, A. K. C., "A survey of thresholding technique," *CVGIP 41*, pp. 233-260(1988).

[12] Tremeau, A., and Colantoni, P., "Regions adjacency graph applied to color image segmentation," *IEEE Trans. on Image Processing*, Vol. 9, No. 4, pp. 735-744(2000).

收稿：95/03/05
初審：95/03/08
複審：95/04/15
接受：95/06/10

NATIONAL CENTRAL LIBRARY 國家圖書館 ROC

南開學報　第四卷　第三期　民國九十六年

使用動態堆疊處理二值影像切割的演算法

郭世崇 [1], 郭錫勳 [2], 俞有華 [2], 李孟度 [2]

1 南開技術學院電子工程系

2 南開技術學院資訊工程系

摘　要

影像切割在影像處理中是一項很重要的工作，影像切割是將一影像分離成其所組成的份子或物件。一般影像切割的演算法都基於影像中像素點的灰階值的連續性或相似性來進行處理，傳統的方法都採用門檻值、基於邊緣的切割方法、區域成長法及區域分割/合併等等。一般影像切割的演算法都較複雜且不容易以程式語言來實現，而且執行費時。

在本篇論文中，我們提出藉由堆疊操作的特性，來對二值影像做切割的演算法，它能成功的將二值影像中的組成物件切割出來，最重要的是我們所提出的演算法，由於使用堆疊操作的特性，因此能輕易的以程式語言來完成，而且執行不耗時。

關鍵詞：影像切割、二值影像、堆疊、門檻值

NATIONAL CENTRAL LIBRARY R.O.C. 國家圖書館