6. Border Tracing Algorithm

6.1. Objectives

The purposes of this laboratory session are:

- to extract the objects' contours using a border tracing algorithm;
- to represent efficiently each extracted contour using chain codes;
- to take advantage of using chain codes in representing the objects' contours (border reconstruction, matching, merging etc.).

6.2. Theoretical Background

6.2.1. Border Tracing Algorithm

The border tracing algorithm is used to extract the contours of the objects (regions) from an image. When applying this algorithm it is assumed that the image with regions is either binary or those regions have been previously labeled.

Algorithm's steps:

1. Search the image from top left until a pixel of a new region is found; this pixel P_0 is the starting pixel of the region border. Define a variable *dir* which stores the direction of the previous move along the border from the previous border element to the current border element. Assign
   (a) *dir* = 0 if the border is detected in 4-connectivity (Fig. 6.1a)
   (b) *dir* = 7 if the border is detected in 8-connectivity (Fig. 6.1b)
2. Search the 3x3 neighborhood of the current pixel in an anti-clockwise direction, beginning the neighborhood search at the pixel positioned in the direction
   (a) (*dir* + 3) *mod* 4 (Fig. 6.1c)
   (b) (*dir* + 7) *mod* 8 if *dir* is even (Fig. 6.1d)
   (*dir* + 6) *mod* 8 if *dir* is odd (Fig. 6.1e)
   The first pixel found with the same value as the current pixel is a new boundary element P_n. Update the *dir* value.
3. If the current boundary element P_n is equal to the second border element P_1 and if the previous border element P_{n-1} is equal to P_0, stop. Otherwise repeat step (2).
4. The detected border is represented by pixels $P_0 \ldots P_{n-2}$.

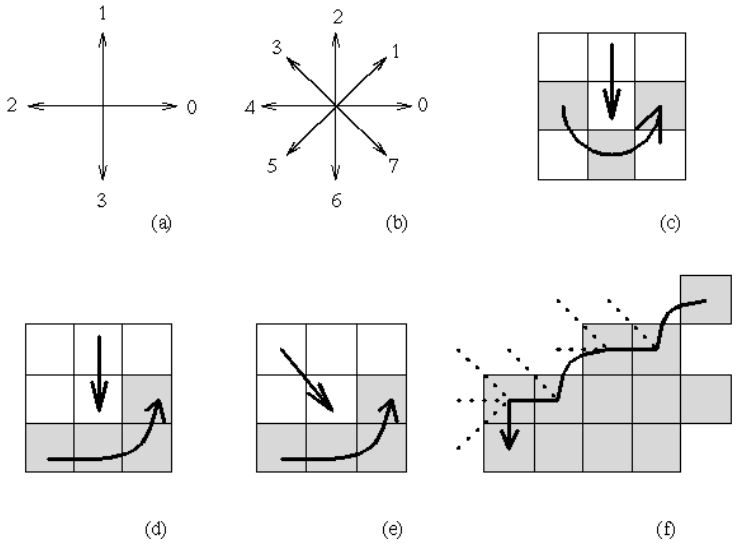

Fig. 6.1 (a) Direction notation, 4-connectivity, (b) 8-connectivity, (c) pixel neighborhood search sequence is 4-connectivity, (d),(e) search sequence in 8-connectivity, (f) boundary tracing in 8-connectivity (dashed lines show pixels tested during the border tracing).

Remarks:

- The above algorithm works for all regions larger than one pixel.
- Looking for the border of a single-pixel region is a trivial problem.
- This algorithm is able to find region borders but does not find borders of region holes.
- To search for the object's holes' borders as well, the border must be traced starting in each region or hole border element if this element has never been a member of any border previously traced.
- Note that if objects are of unit width, more conditions must be added.

6.2.2. Chain Codes Extraction

The *chain code* provides a storage-efficient representation for the boundary of an object in a binary image. The chain code representation incorporates such pertinent information as the length of the boundary of the encoded object, its area, and moments. Chain codes lend to efficient calculation of certain curve parameters. Additionally, chain codes are invertible in that an object can be reconstructed from its chain code representation.

The basic idea behind the chain code is that each boundary pixel of an object has an adjacent boundary pixel neighbor whose direction from the given boundary pixel can be specified by a unique number between 0 and 7 (8-connectivity neighborhood). Chain codes could also be defined using a 4-connectivity neighborhood. A 4-connecivity neighborhood chain codes example it is presented in Fig. 6.4.

In the following we discussion we use the 8-connectivity neighborhood. Given a pixel, consider its eight neighboring pixels. Each 8-neighbor can be assigned a number from 0 to 7 representing one of eight possible directions from the given pixel (see Fig. 6.2). This is done with the same orientation throughout the entire image.

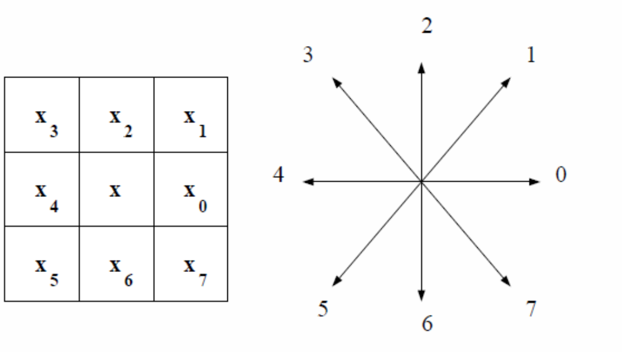

Fig. 6.2 The 8-neighborhood and the associated eight directions

The chain code for the boundary of a binary image is a sequence of integers $c=\{c_0, c_1, \ldots, c_{n-1}\}$, having each c_i from the set $\{0,1, \ldots ,7\}$ for i=0, 1, … , n-1. The number of elements in the sequence c is called the length of the chain code. The elements c_0 and c_{n-1} are called the *initial* and *terminal point* of the code, respectively. Starting at a given base point, the boundary of an object in a binary image can be traced out using the head-to-tail directions that the chain code provides.

Fig. 6.3 illustrates the process of tracing out the boundary of a triangle by following direction vectors. Suppose we choose the topmost left feature pixel of Fig. 6.3 as the base point (x=109, y=61) for the boundary encoding. The chain code for the boundary of the triangle is the following sequence of 245 codes: 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2.

Given the base point and the chain code, the boundary of the triangle can be completely reconstructed. The chain code is an efficient way of storing boundary information because it requires only three bits ($2^3 = 8$) to determine any one of the eight directions.

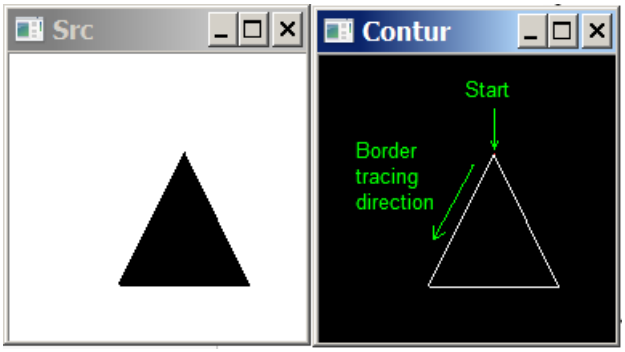

Fig. 6.3 Chain code directions with associated direction numbers

Chain codes may be made position-independent by ignoring the "start point". If they represent closed boundaries they may be "start point normalized" by choosing the start point so that the resulting sequence of direction codes forms an integer of minimum magnitude.

The "derivative" of the chain code is useful because it is invariant under boundary rotation. The derivative (really a first difference mod 4 or 8) is simply another sequence of numbers indicating the relative direction of chain code segments; the number of left hand turns of $\pi/2$ or $\pi/4$ needed to achieve the direction of the next chain segment. A *mod* 4 or *mod* 8 difference is called a chain code *derivative* (see Fig. 6.4!).

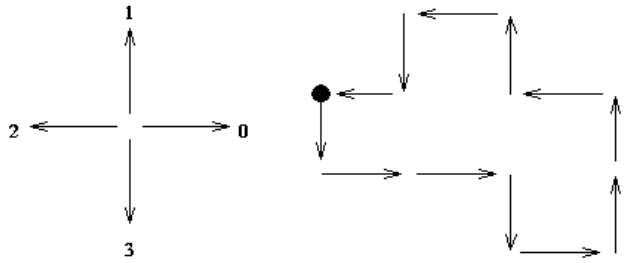

Fig. 6.4 Chain code in 4-connectivity and its derivative.

Code: 3, 0, 0, 3, 0, 1, 1, 2, 1, 2, 3, 2
Derivative: 1, 0, 3, 1, 1, 0, 1, 3, 1, 1, 3, 1

Chain codes properties:

- Chain codes describe an object by a sequence of unit-size (4-connectivity) line segments with a given orientation.
- The first element of such a sequence must bear information about its position to allow reconstruction of the region.
- Even codes {0, 2, 4, 6} correspond to horizontal and vertical directions; odd codes {1, 3, 5, 7} correspond to the diagonal directions.
- Each code can be considered as the angular direction, in multiples of 45 degrees that we must move to go from one contour pixel to the next.
- The absolute coordinates of the first contour pixel (e.g. top, leftmost) together with the chain code of the contour represent a complete description of the discrete region contour.
- When there is a change between two consecutive chain codes, then the contour has changed direction. This point is defined as a *corner*.

6.3. Practical Work

Using the *OpenCVApplication* framework and the laboratory's additional images and files:

1. Implement the border tracing algorithm and draw the object contour on an image having a single object.
2. Starting from the border tracing algorithm write the algorithm that builds the chain code and derivative chain code for an object. Compute and display (command line or output text file) both codes (chain code and derivative chain code) for an image with a single object.
3. Implement a function that reconstructs (draws) the border of an object over an image having as inputs the start point coordinates and the chain code in 8-neighborhood *(reconstruct.txt*). Load the image *gray_background.bmp* and apply the function that reconstructs the border. You should obtain the contour of the word "EXCELLENT" (having all the letters connected).
4. **Save your work. Use the same application in the next laboratories. At the end of the image processing laboratory you should present your own application with the implemented algorithms!!!**

Additional info:

The test images with a single object have:

- 8 bits/pixel
- index 0 for object's pixels (black pixels)
- other index value for background pixels (white pixels)

The file *reconstruct.txt* is a text file having:

- on the first line the start point coordinates (row column) separated with a space;
- on the second line the number of chain codes;
- on the third line the chain codes (sequence of directions in 8-connectivity) separated with a space.

6.4. Bibliography

[1] R.C.Gonzales, R.E.Woods, Digital Image Processing, 2-nd Edition, Prentice Hall, 2002.
[2] Umbaugh Scot E, Computer Vision and Image Processing, Prentice Hall, NJ, 1998, ISBN 0-13-264599-8.
[3] G.X. Ritter, J.N. Wilson – Handbook of Computer Vision Algorithms in Image Algebra Second Edition – Chapter 10.4 Chain Code Extraction and Correlation, CRC Press, New York 2001.
[4] Representation of Two-Dimensional Geometric Structures, http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/LIB/bandb8_12.pdf