10

Contour Extraction and Compression-Selected Topics

Andrzej Dziech
AGH Universisty of Science and Technology, Telecommunication Dept., Cracow, Poland

The subject of this chapter is to describe selected topics on contour extraction, approximation and compression in spatial domain. Contours are treated as important image structure required in many applications, for example in analysis of medical image, computer vision, robot guidance, pattern recognition etc. Contour compression plays important role in many practical tasks associated with contour and image processing.
The main approach solving the problem of contour compression is based on polygonal approximation. In this chapter the traditional and new methods for contour approximation in spatial domain are presented. These methods are often much faster than the methods of compression based on transform coding. In order to extract the contours from the image some algorithms have been developed. The selected well known methods for contour extraction and edge detection as well as the new algorithms are also presented in this chapter. The author is grateful to his Ph.D. students A. Ukasha and B. Fituri for their contribution in preparation of this chapter.

1. Contour Analysis and Extraction

Contours and line drawings have been an important area in image data processing. In many applications, e.g., weather maps and geometric shapes, it is necessary to store and transmit large amounts of contours and line drawings and process the information by computers.
Several approaches have been used to extract and encode the boundary points of contours and line drawings. The extracted data is then used for further processing and applications. Contour approximation and compression are some of the processing operations performed on contours and has been considered by several authors.
In encoding contours and line drawings, efficient data compression and good reconstruction are both usually required. Freeman proposed an eight-directional encoding scheme for contour lines. The proposed chain code is obtained by superimposing a rectangular grid on the curve and then quantizing the curve to the nearest one of the eight possible grid points. The chain encoding scheme represents contour lines by 3 bits/link, where a link is defined as one of the eight possible straight-line segments between two adjacent quantized points.
Efforts have been made to improve the coding efficiency. Freeman proposed a chain difference coding scheme which assigned variable-length codewords to the difference between two consecutive links. This coding scheme represents contour lines by about 2 to 2.1 bits/link on average.

Source: Mobile Robots: Perception & Navigation, Book edited by: Sascha Kolski, ISBN 3-86611-283-1, pp. 704, February 2007, Plv/ARS, Germany

The purpose of this subchapter is to investigate selected methods for contour extraction. At the beginning some relationship between the image and contours that can be extracted from the image, are briefly described.

In the simplest case, an image may consist of a single object or several separated objects of relatively high intensity. This allows figure/ground separation by thresholding. In order to create the two-valued binary image a simple threshold may be applied so that all the pixels in the image plane are classified into object and background pixels. A binary image function can then be constructed such that pixels above the threshold are foreground ("1") and below the threshold are background ("0").

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are used 0 for black and 1 for white. In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest". This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as *segmentation techniques* - segmenting the foreground from background.

In general there are two basic approaches for shape representation: by contours and by regions. Polygonal approximation, chain code, geometric primitives, parametric curves, Fourier descriptors and Hough transform are the examples of contour based shape representation methods. These methods share some common characteristics [1]:

(1) *Shape information extraction:* the representation would facilitate some contour characteristics comprehension.
(2) *Data compression:* data compression rates can vary in wide range depending on the method of compression and the structure of contours
(3) *Noise elimination:* digital curves can be corrupted with noise and/or undesirable details treated as redundancy elements. The method should filter the noise and redundancies.
(4) *Curvature evaluation:* this step is important in contour description. The major difficulty is due to the discrete nature of the curve, making the majority of the methods noisy and scale dependent. There are psychological results showing that curvature plays a fundamental role in human shape perception.

The most typical contour representations are illustrated below [2]:

1) Generalized representation (θ, l).

Fig. 1.1 shows the contour representation using the (θ, l) generalized chain coding scheme.

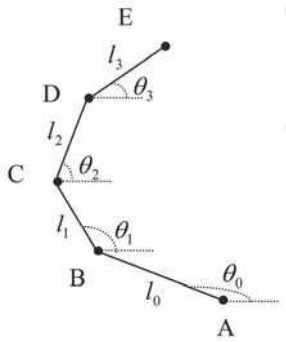


Fig. 1.1 Generalized representation of contour.

2) Polar representation (α, l).
Fig. 1.2 shows the contour representation using the (α, l) polar chain coding scheme.

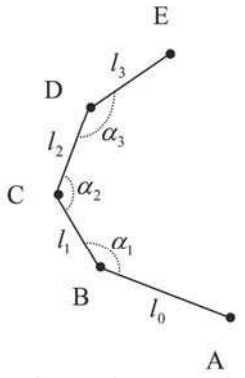


Fig. 1.2 Polar representation.

3) Cartesian representation
Cartesian representation of contour is shown in Fig. 1.3

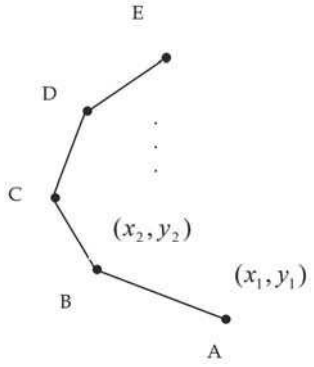


Fig. 1.3 Cartesian representation.

Cartesian representation leads to decomposition of two-dimensional contour (y, x) into two one –dimensional signals x(n) and y(n), where n is a variable representing the current length of contour, as it is shown below
One of the widely used procedures related to contour tracing is proposed by Freeman [3] . This procedure is based on an eight- or four-directional chain encoding scheme as shown in Fig. 1.5. An 8-directional chain-coding uses eight possible directions to represent all possible line segments connecting nearest neighbours according to the 8-connectivity scheme as shown in Fig. 1.5a. 4-directional chain-coding uses four possible directions to represent all possible line segments connecting nearest neighbours according to the 4-connectivity scheme as shown in Fig. 1.5b.
This scheme describes arbitrary geometric configurations in a simple and efficient method. The chain code is obtained by superimposing a rectangular grid on the curve and then quantizing the curve to the nearest grid point. The Freeman chain code may subsequently be described in cartesian or polar systems.

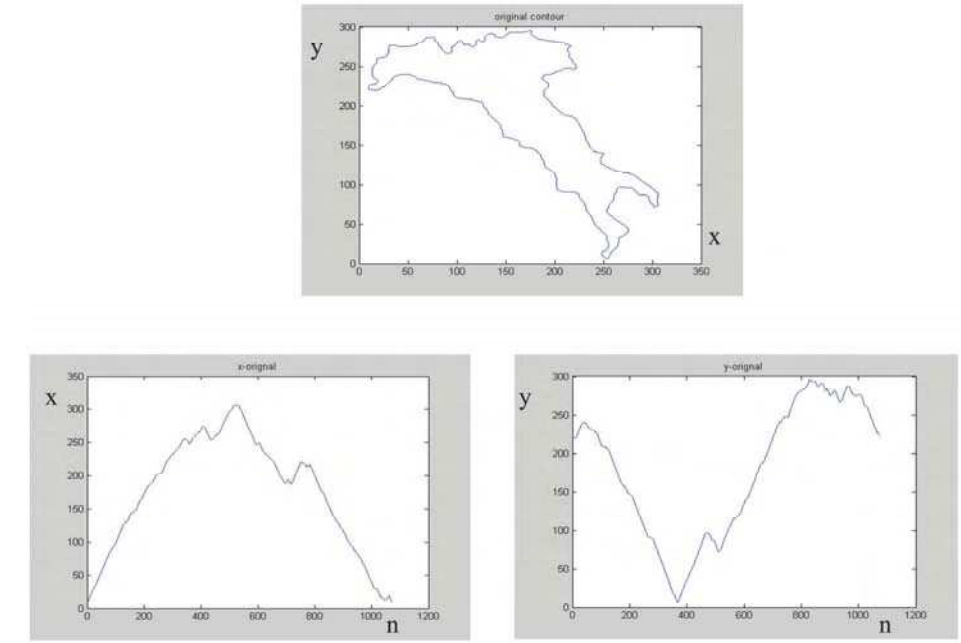
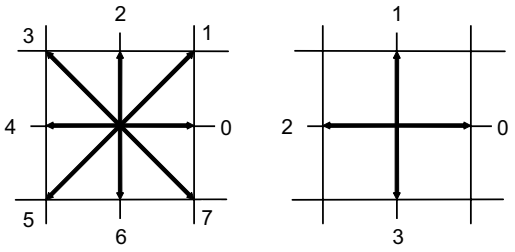


Fig. 1.4 Contour decomposition.



a) 8-directional chain code. b) 4-directional chain code.

Fig. 1.5 Freeman chain code.

Completely enclosed boundary regions can be coded with a simple modification to the basic chain coding. The outer boundary is first chain coded in a normal manner when this boundary has been closed ,a code group 0401 is inserted in the chain code, and an "invisible line" connecting the two boundaries is encoded. When the second boundary is reached, the code group 0402 is inserted in the chain code to indicate the end of the invisible line. The inner boundary is then chain coded in a normal manner. The prefix 04 of the "invisible line" code and the "visible line" code designates the rarely occurring event of a right shift followed by a left shift. This prefix is also used with other codes to indicate a variety of special cases.

-Length of a chain: The length of an 8-directional chain code may be found using the following relation:

$$L = T(n_e + n_o\sqrt{2})$$

where n_e -number of even valued elements (chains)

n_o --number of odd valued elements (chains)

T- scale factor proportional to grid spacing.

-*Inverse chain*: The inverse chain of an 8-directional chain code may be obtained using the following relation:

$$c_i^{-1} = c_i \oplus 4$$

where $\oplus$ -addition mod 8

Example:

-For the curve shown

(a) write the Freeman chain code using the 8-directional scheme.

(b) Find the length of the chain code.

(c) Find the inverse of the chain code.

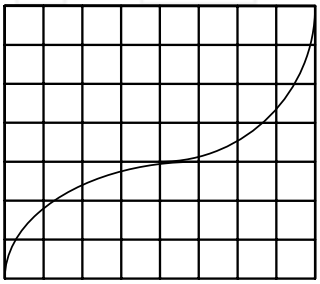
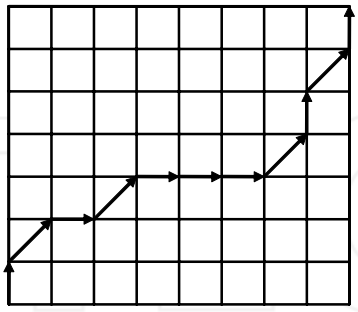


Fig. 1.6 Chain encoding example

(a) In the Figure below is shown the tracing of the curve using the 8-dierectional scheme.



The chain code is obtained as: (21010001212)

(b) The length of the chain code, assuming T=1, is :

$$L = T(n_e + n_o\sqrt{2}) = 1(7+4\sqrt{2}) = 12.657$$

(c) The inverse of the code is: . $c_i^{-1} = c_i \oplus 4$ =(65454445656)

1.1 OCF and MSCE Methods of Contour Extraction

The separation of objects from their background referred to as segmentation of gray scale images and as contour tracing (or boundary following) for binary images, often provide important features in pattern recognition and scene analysis and is used in variety of other applications, for example, recognition of human faces from essential contours.

In general contour extraction from two-dimensional images may be accomplished in two operations: 1)*Edge detection*; 2)*Contour tracing*.

1) *Edge detection*: The aim of edge detection is to identify and enhance edges (pixels) belonging to boundaries of object of interest in the original image. An edge element is defined as a picture element which lies on the boundary between objects or regions of different intensities or gray levels. Many edge detection methods have been proposed for detecting and enhancing edges in digital images. Most of these methods are implemented as some form of gradient operators. Some images can be characterized as containing some objects of interest of reasonably uniform brightness placed against a background of differing brightness. Typical examples include handwritten and typewritten text, and airplanes on the a runway. For such images, brightness is a distinguishing feature that can be utilized to locate the object. This method is termed as luminance thresholding.

2) *Contour tracing*: The contour tracing algorithm traces the contour and extracts the contour information which is then passed to subsequent processing. One of the most widely used procedures to follow contours and line drawings is that of Freeman, which provides a code that possesses some manipulative properties. The Freeman chain code can subsequently be described in cartesian or polar systems.

The problem of contour extraction from 2D-digital image has been studied by many researchers, and a large number of contour extraction methods have been developed. Most of the developed methods can be assigned to either of two major classes known as sequential methods or Object Contour Following (OCF), and parallel methods or Multiple Step Contour Extraction (MSCE). In Fig.1.6 is shown a block diagram of the contour extraction and processing from gray level images.

A brief description of the two main classes of contour extraction methods, OCF and MSCE, is given.

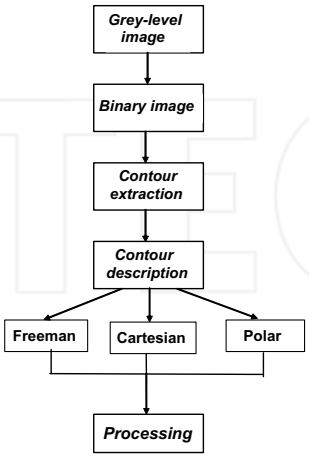


Fig. 1.6 Contour extraction and processing from gray level images.

Two main approaches can be used for the extraction of object contours. The Object Contour Following (OCF) and Multiple Step Contour Extraction (MSCE). The OCF methods, sequentially detect and extract object contour edges. By sequential methods, it is meant that the result at a point is dependent upon the result of the previously processed points. The MSCE methods are referred to as parallel schemes for object contour extraction. By parallel, it is meant that the decision of whether or not a point is on an edge is made on the basis of the gray level of the point and its neighbours. So, the edge detection operator in principle can be applied simultaneously everywhere in the picture. It should be noted that the definitions of sequential and parallel schemes are used with respect to edge detection. To produce a closed boundary, the extracted edges have to be connected together to form a closed curve.

(i) Object Contour Following(OCF)

The OCF methods, which are also called Bug Following, can be used to trace (follow) the contour edges of a 2-D digital image. The idea of these methods is illustrated in Fig.1.7. The extraction procedure consists of finding a starting point and then cross the edge between the white and black regions, record the co-ordinates of the black pixel then turn left continuously until a white pixel is found, record the black pixel co-ordinates as the next contour edge point. Start turning right until a black pixel is found. Terminate this procedure when the starting point of the contour is reached again.

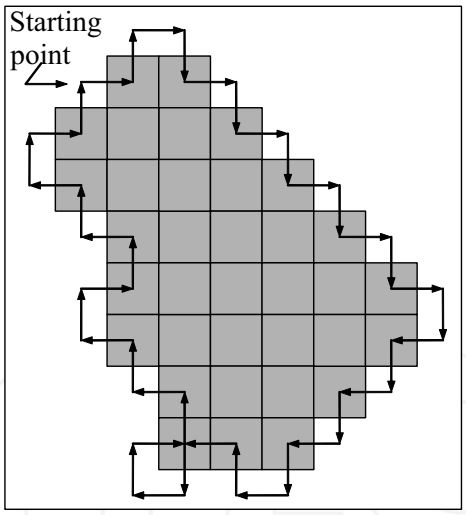


Fig. 1.7 The Object Contour Follower.

(ii) Multiple Step Contour Extraction (MSCE)

The key feature of MSCE methods is that the gradient between two pixels with different gray scale levels represents the difference between the two pixels, and the gradient will be zero for the pixels with the same gray scale level. A threshold value will determine if the gradient is interpreted as an object edge or not. An additional procedure is used to improve the overall contour structure by joining all disjoined edges and thinning the thick edges.

Fig.1.8 shows the steps required for extracting object contours by the MSCE methods. Although the method of gradient operator for generating edge elements is parallel, the method of connecting (tracing) these extracted edge elements is sequential.

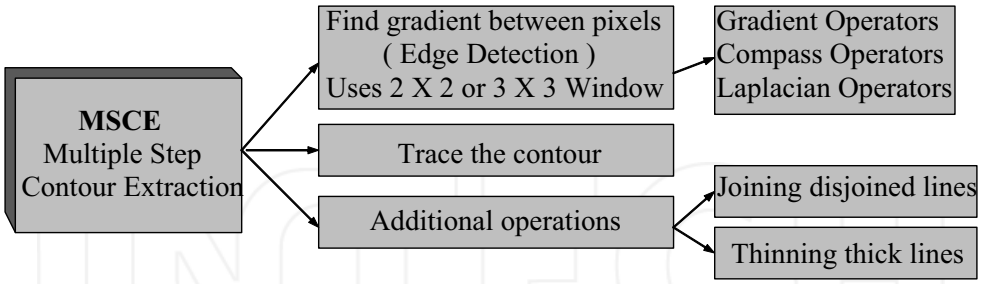


Fig. 1.8 Block Diagram of The Multiple Step Contour Extraction.

The three main steps of the MSCE methods are: edge detection, contour tracing and the additional procedures for joining disjoined lines, and thinning thick lines.

a) Edge Detection

Local discontinuities in image luminance or gray levels are called luminance edges. Global discontinuities are called boundary segments. Edges characterise object boundaries. They are used for segmentation and identification of objects in images. Edge points can be thought of as pixel locations with abrupt gray level change. For example, it is reasonable to define edge points in binary images as black pixels (object pixels) with at least one white nearest neighbour pixel (background pixel). Most techniques used for edge detection are limited to processing over the 2x2 or 3x3 windows shown in Fig. 1.9a and Fig. 1.9b respectively.

Note that the same pixel numbering will be used with all edge detection operators.

	i	i+1
j	P(i,j)	P(i+1,j)
j+1	P(i,j+1)	P(i+1,j+1)

Fig. 1.9a Pixel numbering for 2x2 edge detecting operators.

		i	
j	P_0	P_1	P_2
	P_7	P(i,j)	P_3
	P_6	P_5	P_4

Fig. 1.9b Pixel numbering for 3x3 edge detecting operators.

b) Gradient Operators:

These operators measure the gradient of the image G(i, j) in two orthogonal directions, horizontal, and vertical directions. Except with the case of Roberts cross operator where the gradient is taken across the diagonals.

The gradient is given by :

$$G(i,j) = \sqrt{g_v^2(i,j) + g_h^2(i,j)} \tag{1.1}$$

where g_v : is the gradient in the vertical direction.
g_h : is the gradient in the horizontal direction.
Instead of using the square root gradient given by Eq.(1.1), the gradient is often approximated using the absolute gradient given by the following equation :

$$G(i,j) = \left| g_v(i,j) \right| + \left| g_h(i,j) \right| \quad (1.2)$$

Eq.(1.2) is easier to perform and to implement in digital hardware.
The Roberts , Sobel , Prewitt [26] operators, showing the horizontal and vertical masks (diagonal masks in case of Roberts cross) together with the necessary equations for finding the gradient, are introduced next, as an example of edge detection using gradient operators.

Roberts cross gradient operator :
Roberts has used a simple window of 2x2 to introduce the square-root difference operator given in Fig.1.10, it is often called Roberts cross gradient operator. The edge detection is carried out using Eq.(1.1), where :

$$g_v = P(i, j) - P(i+1, j+1) \quad (1.3)$$
$$g_h = P(i, j+1) - P(i+1, j) \quad (1.4)$$

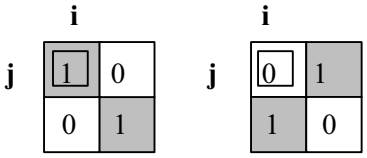
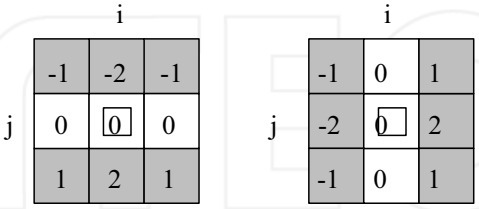


Fig. 1.10 Roberts cross gradient operator.

Sobel gradient operator :
The masks which are used by Sobel for finding the gradient of an image are shown in Fig. 1.11. The corresponding equation used for calculating the gradient is given by Eq.(1.1), where :

$$g_v = (P_2 + 2P_3 + P_4) - (P_0 + 2P_7 + P_6) \quad (1.5)$$
$$g_h = (P_0 + 2P_1 + P_2) - (P_6 + 2P_5 + P_4) \quad (1.6)$$



Horizontal mask. Vertical mask.

Fig. 1.11 Sobel gradient operator.

Laplacian operators :
Three different Laplacian operators [26][27], with the necessary equations for calculating the gradient, are shown in Fig.1.12. For images with smooth changes in gray level values the

Laplacian operators give better results than the previous operators. But it is more sensitive to noise, and produces double edges.

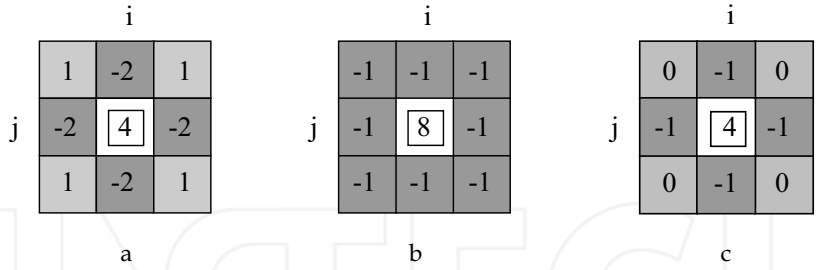


Fig. 1.12 Three different types of Laplacian operators.

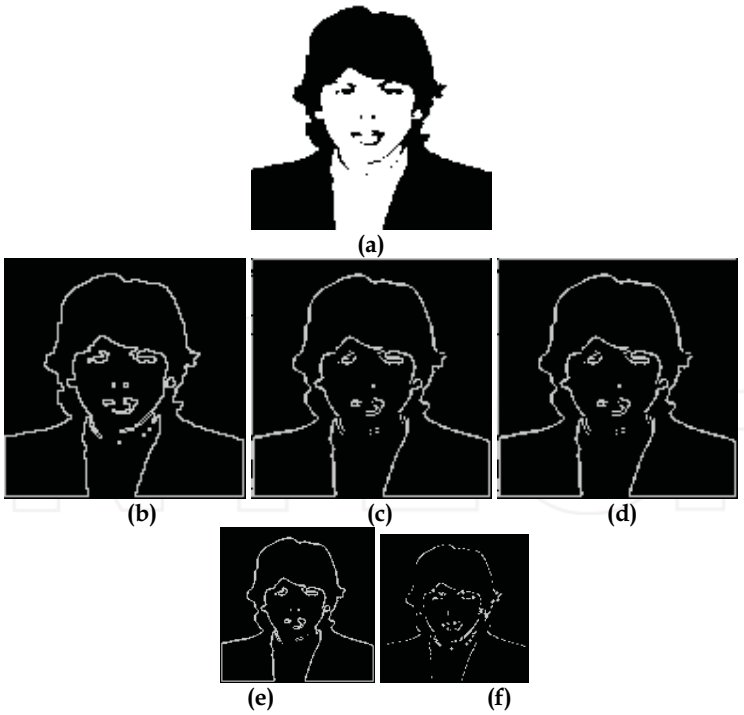
For the operator of Fig. 1.12a -c the edges are detected by calculating the gradients between pixels using the following formulas respectively :

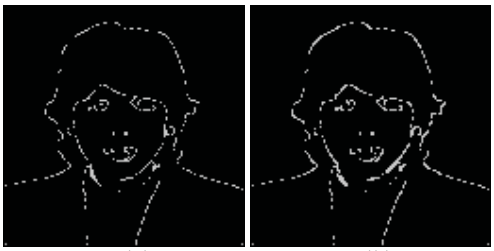
$$G(i,j) = 4F(i,j) + (P_0 + P_2 + P_4 + P_6) - (P_1 + P_3 + P_5 + P_7) \quad (1.7)$$

$$G(i,j) = 8F(i,j) - (P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7) \quad (1.8)$$

$$G(i,j) = 4F(i,j) - (P_1 + P_3 + P_5 + P_7) \quad (1.9)$$

To compare the performance of the selected gradient operators, the binary image of Fig. 1.13a is used. The detected edges obtained by applying different gradient operators are shown in Fig. 1.13b-h.





(g) **(h)**

(a) Original image, (b) Prewitt gradient Operator, (c) Roberts cross gradient operator, (d) Sobel gradient operator, (e) Prewitt compass operator, (f - h) Three types of Laplacian operators

Fig. 1.13 Different edge detection results on a real binary image.

1.2 Object-oriented Contour Extraction OCE

This algorithm is based on 4x4 pixels window structure to extract the object contours by the four central pixels which are processed simultaneously. The algorithm uses the features of both OCF and MSCE methods to overcome most of the disadvantages they have.

It features a parallel implement and an effective suppression of noises. It can be realized in real-time [18].

The following three steps are needed for the extraction procedure:

Step1: The image is framed with zeros.

Step2: Eight rules of edge extraction are applied and are coded using 8-directional chain-code as shown in Listing 1.1.

Listing 1.1

Implementation of the eight rules for contour extraction (4x4 windows)

```
a(i,j) ← 0; i = 1,2,…..,N; j = 1,2,…..,N;
for i = 2,3,…..,N-1; j = 2,3,…..,N-1;
{
if b(i,j+1) and b(i+1,j+1) and [b(i,j+2) or b(i+1,j+2)]
then a(i,j+1) ← a(i,j+1) or 2^0                    { edge 0 }
if b(i+1,j) and b(i,j+1) and b(i+1,j+1)
then a(i,j+1) ← a(i,j+1) or 2^1                    { edge 1 }
if b(i+1,j) and b(i+1,j+1) and [b(i+2,j) or b(i+2,j+1)]
then a(i+1,j+1) ← a(i+1,j+1) or 2^2                { edge 2 }
if b(i,j) and b(i+1,j) and b(i+1,j+1)
then a(i+1,j+1) ← a(i+1,j+1) or 2^3                { edge 3 }
if b(i,j) and b(i+1,j) and [b(i,j-1) or b(i+1,j-1)]
then a(i+1,j) ← a(i+1,j) or 2^4                    { edge 4 }
if b(i,j) and b(i+1,j) and b(i,j+1)
then a(i+1,j) ← a(i+1,j) or 2^5                    { edge 5 }
if b(i,j) and b(i,j+1) and [b(i-1,j) or b(i-1,j+1)]
then a(i,j) ← a(i,j) or 2^6                        { edge 6 }
if b(i,j) and b(i,j+1) and b(i+1,j+1)
then a(i,j) ← a(i,j) or 2^7                        { edge 7 }
}
```

where:

b(i,j) is the binary value of a pixel point (i,j) and 2^k (k:0-7) is the extracted edge code.

Step3: The extracted contour edges are sorted and stored or optimized according to the application requirements. The extraction procedure is shown in Fig. 1.14.

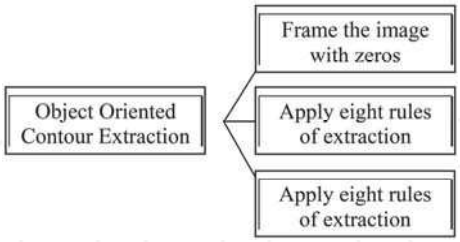


Fig. 1.14 Object-oriented Contour Extraction (OCE).

The problem in the OCE procedure is that contours extracted from the objects near the image boundary, i.e. objects within one pixel distance from the image border, are not closed and that is why the image should be framed with two background pixels to ensure the closure of the contours. Fig. (1.15a) shows that the extracted edges do not form closed contours; while Fig. (1.15b) shows that after framing the image with at least two underground pixels all extracted contours are closed.

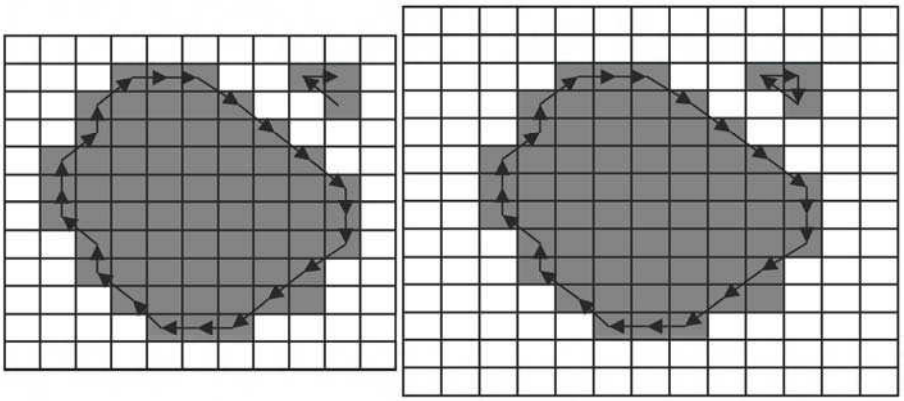


Fig. 1.15 OCE procedure (a) Without correcting the first step and (b) After correcting the first step.

1.3 Single Step Parallel Contour Extraction 'SSPCE' (3x3 windows)

There are two algorithms; the first one [35] [35] uses 8-connectivity scheme between pixels, and 8-Directional Freeman chain coding [3] scheme is used to distinguish all eight possible line segments connecting nearest neighbors. This algorithm uses the same principle of extraction rules as in the OCE algorithm. The second algorithm [35] uses the 4-connectivity scheme between pixels, and 4-Directional Freeman chain coding scheme is used to distinguish all four possible line segments. Both algorithms use an 3x3 pixels window structure to extract the object contours by using the central pixel to find the possible edge direction which connect the central pixel with one of the remaining pixels surrounding it.

The first algorithm gives exactly the same extracted contours as the OCE algorithms but is much faster ; while the second algorithm gives similar contours, but not identical and is also faster . Consider now the first algorithm in details.
The edges can be extracted by applying the definition that an object contour edge is a straight line connecting two neighboring pixels which have both a common neighboring object pixel and a common neighboring underground pixel [33]. By this definition, no edges can be extracted from the three following cases:

1- If all nine pixels are object pixels; i.e. the window is inside an object region.
2- If all nine pixels are background pixels; i.e. the window is inside a background region.

If the center pixel is an object pixel surrounded by background pixels; i.e. it is most probable that the center pixel in this case is a point noise caused by image digitalization.
So, this algorithm uses the same principle and steps of extraction rules as the OCE algorithm using 3x3 window. The eight rules of edge extraction are applied and are coded using 8-directional chain-code as shown in Listing 1.2.

Listing 1.2 (3x3 windows)
Implementation of the eight rules for contour extraction (3x3 windows)

```
a(i,j) ← 0; i = 1,2,.....,N; j = 1,2,.....,N;
for i = 2,3,.....,N-1; j = 2,3,.....,N-1;
{
if b(i,j) and b(i+1,j) and [b(i,j+1) or b(i+1,j+1)] and  [ not [b(i,j-1) or b(i+1,j-1)]]
then a(i,j) ← a(i,j) or 2^0                                          { edge 0 }
if b(i,j) and b(i+1,j) and  b(i+1,j-1) and [ not [b(i,j-1)]]
then a(i,j) ← a(i,j) or 2^1                                          { edge 1 }
if b(i,j) and b(i,j-1) and [b(i+1,j) or b(i+1,j-1)] and  [ not [b(i-1,j) or b(i-1,j-1)]]
then a(i,j) ← a(i,j) or 2^2                                          { edge 2 }
if b(i,j) and b(i,j-1) and  b(i-1,j-1) and [ not [b(i-1,j)]]
then a(i,j) ← a(i,j) or 2^3                                          { edge 3 }
if b(i,j) and b(i-1,j) and [b(i,j-1) or b(i-1,j-1)] and  [ not [b(i,j+1) or b(i-1,j+1)]]
then a(i,j) ← a(i,j) or 2^4                                          { edge 4 }
if b(i,j) and b(i-1,j) and  b(i-1,j+1) and [ not [b(i,j+1)]]
then a(i,j) ← a(i,j) or 2^5                                          { edge 5 }
if b(i,j) and b(i,j+1) and [b(i-1,j) or b(i-1,j+1)] and  [ not [b(i+1,j) or b(i+1,j+1)]]
then a(i,j) ← a(i,j) or 2^6                                          { edge 6 }
if b(i,j) and b(i,j+1) and  b(i+1,j+1) and [ not [b(i+1,j)]]
then a(i,j) ← a(i,j) or 2^7                                          { edge 7 }
}
```

1.4 Contour Extraction Based on 2x2 Windows

This algorithm is mainly used for gray scale images . It uses a smaller window for contour extraction than its predecessors, i.e. 2x2 window shown in Fig. 1.16.

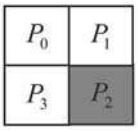


Fig. 1.16 Pixel numbering for 2x2 windows.

The processed pixel is the darker one. Two buffers are required for a real time contour extraction system. First buffer is used for the storage of a previously processed image line and the second one keeps pixel values of the currently processed image line.
The algorithm uses the 8-connectivity scheme, and the extracted edges are coded by using of the 8-directional chain coding. It does not require any storage of the scanned (processed) image.
The three main steps of the algorithm are:

- Frame the image with zeros,
- Extract contour edges using the eight rules
- Sort the extracted contour edges.

The eight rules of edge extraction are applied and are coded using 8-directional chain-code as shown in Listing 1.3.

Listing 1.3
Implementation of the eight rules for contour extraction (2x2 windows)

```
a(i,j) ← 0; I = 1,2,…..,N; j = 1,2,…..,N;
for i = 2,3,…..,N-1; j = 2,3,…..,N-1;
{
if ( b(i-1,j) = b(i,j) ) ⋂ ( b(i-1,j) ≠ b(i-1,j-1) ) ⋂ (b(i-1,j) ≠ b(i,j-1) )
then a(i-1,j) ← a(i-1,j) ⋃ b(i-1,j) ⋃ 2^0 { edge 0 }
if ( b(i-1,j) = b(i,j-1) ) ⋂ ( b(i-1,j) ≠ b(i-1,j-1) )
then a(i-1,j) ← a(i-1,j) ⋃ b(i-1,j) ⋃ 2^1 { edge 1 }
if ( b(i,j-1) = b(i,j) ) ⋂ ( b(i,j) ≠ b(i-1,j) ) ⋂ (b(i,j) ≠ b(i-1,j-1) )
then a(i,j) ← a(i,j) ⋃ b(i,j) ⋃ 2^2 { edge 2 }
if ( b(i-1,j-1) = b(i,j) ) ⋂ ( b(i,j) ≠ b(i-1,j) )
then a(i,j) ← a(i,j) ⋃ b(i,j) ⋃ 2^3 { edge 3 }
if ( b(i,j-1) = b(i-1,j-1 ) ) ⋂ (b(i,j-1) ≠ b(i,j) ) ⋂ (b(i,j-1) ≠ b(i-1,j) )
then a(i,j-1) ← a(i,j-1) ⋃ b(i,j-1) ⋃ 2^4 { edge 4 }
if ( b(i,j-1) = b(i-1,j) ) ⋂ ( b(i,j-1) ≠ b(i,j) )
then a(i,j-1) ← a(i,j-1) ⋃ b(i,j-1) ⋃ 2^5 { edge 5 }
if ( b(i-1,j-1) = b(i-1,j) ) ⋂ ( b(i-1,j-1) ≠ b(i,j-1) ) ⋂ (b(i-1,j-1) ≠ b(i,j))
then a(i-1,j-1) ← a(i-1,j-1) ∪ b(i-1,j-1) ∪ 2^6 { edge 6 }
if ( b(i-1,j-1) = b(i,j) ) ⋂ ( b(i-1,j-1) ≠ b(i,j-1) )
then a(i-1,j-1) ← a(i-1,j-1) ∪ b(i-1,j-1) ∪ 2^7 { edge 7}
}
```

The algorithm does not require the storage of the scanned image, i.e. it can be used for real time applications.

1.5 Comparison of Contour Extraction Algorithms (Different Windows)

The comparison is made between the following three algorithms:

- Contour extraction CE referred to as the third algorithm (or 2x2 windows).
- SSPCE method; it will be referred to as the second algorithm (or 3x3 windows).
- OCE method; it will be referred to as the third algorithm (or 4x4 windows).

The comparison is performed with respect to the number of operation and number of contour edges. The binary test images are illustrated in Fig. 1.17.

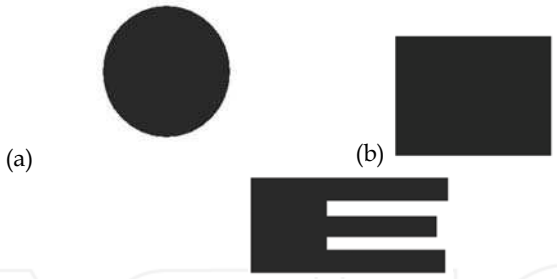


Fig. 1.17 Binary images (a) Circle (b) Square (c) E letter.

The comparison between the three algorithms with respect to the number of operations versus the number of edges for Circle, Square and E letter contours respectively is shown in Tab. 1.1, Tab. 1.2 and Tab. 1.3

NE	20	30	40	50	60	**AE**
1^{st} algor. (*NO*)	*276497*	*276581*	*276666*	*276748*	*276835*	*277060*
2^{nd} algor. (*NO*)	*89810*	*89894*	*89979*	*90061*	*90148*	*90373*
3^{rd} algor. (*NO*)	*1,065018*	*1,065092*	*1,065167*	*1,065239*	*1,065316*	*1,065514*

NE – Number of Edges, AE – All Edges and *NO* is the Number of Operations,

Table 3.1 Comparison between the algorithms for Circle image.

NE	50	100	150	200	**AE**
1^{st} algor. (*NO*)	*447287*	*447687*	*448087*	*448487*	*450351*
2^{nd} algor. (*NO*)	*446898*	*447298*	*447698*	*448098*	*448442*
3^{rd} algor. (*NO*)	*1, 726850*	*1, 727200*	*1, 727550*	*1, 727900*	*1, 728201*

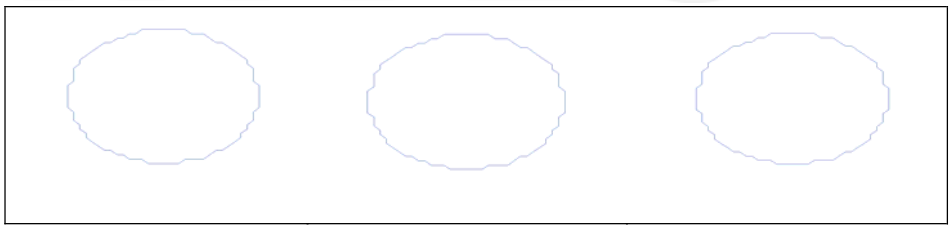
NE – Number of Edges, AE – All Edges and *NO* is the Number of Operations,

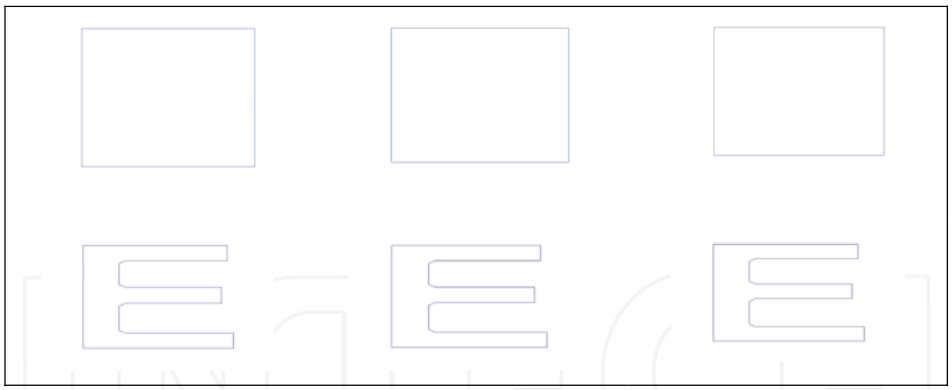
Table3.2 Comparison between the algorithms for Square image

NE	20	60	100	125	150	**AE**
1^{st} algor. (*NO*)	*109410*	*109732*	*110053*	*110254*	*110454*	*110718*
2^{nd} algor. (*NO*)	*56629*	*56951*	*57272*	*57473*	*57673*	*57937*
3^{rd} algor. (*NO*)	*407648*	*407930*	*408211*	*408387*	*408562*	*408793*

NE – Number of Edges, AE – All Edges and *NO* is the Number of Operations,

Table 3.3 Comparison between the algorithms for E letter image.

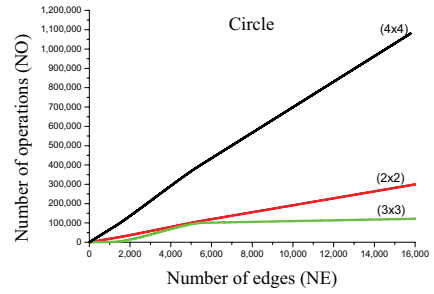




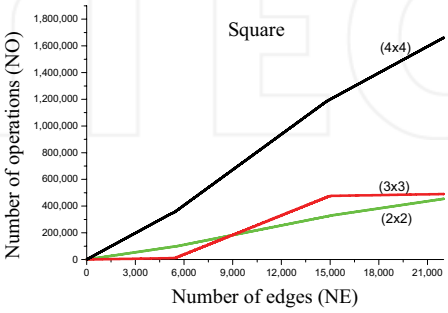
Extracted contours using first algorithm | Extracted contours using second algorithm | Extracted contours using third algorithm

Fig. 1.18 Extracted contours using the three different algorithms.

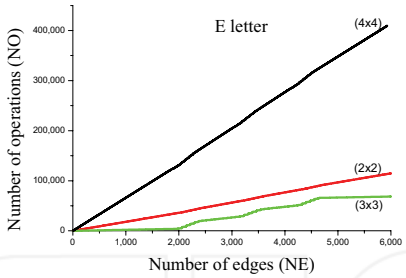
The first column of Fig. 1.18 shows the extracted contours by the first algorithm. The second column shows the extracted contours by the second algorithm and the third one- the extracted contours by the third algorithm.
The comparison between the three algorithms with respect to the number of operations versus the number of edges for the binary images is illustrated in Fig. 1.19



(a)



(b)



(c)

Fig. 1.19 Number of operations versus number of edges for the all algorithms for the shapes of (a) Circle (b) Square (c) E letter.

The results presented in Fig.1.19 show that the fastest algorithm is dependent on the structure of contour.

2. Methods of Contour Approximation and Compression in Spatial Domain

In many applications of contour processing and analysis it is desirable to obtain a polygonal approximation of an object under consideration. In this chapter we briefly consider the algorithms that have been introduced for polygonal approximation of extracted contours. The algorithm presented by Ramer uses the maximum distance of the curve from the approximating polygon as a fit criterion [45]. There exist algorithms referred to as the Triangle family of contour approximation . The first algorithm is based on the ratio between the height and length triangle distances for each segment, and this ratio is used as the fit criterion of the algorithm which is referred to as height over length triangle ratios algorithm [46] and [47]. The second algorithm is based on the height triangle distance for each segment as the fit criterion of the algorithm which is referred to as height length triangle algorithm. The third algorithms is related to the square of the height triangle distance for each segment as the fit criterion of the algorithm which is referred to as height square length triangle algorithm. The fourth algorithm is associated with the area for each triangle segment as the fit criterion of the algorithm which is referred to as the area triangle algorithm.

2.1 Polygonal approximation

A digitized picture in a 2D array of points is often desired to be approximated by polygonal lines with the smallest number of sides, under the given error tolerance E.

There are several algorithms available for determining the number and location of the vertices and also to compute the polygonal approximation of a contour. The Ramer method is based on the polygonal approximation scheme. The simplest approach for the polygonal approximation is a recursive process (Splitting methods). Splitting methods is performed by first drawing a line from one point on the boundary to another. Then, we compute the perpendicular distance from each point along the segment to the line. If this exceeds some threshold, we break the line at the point of greatest error. We then repeat the process recursively for each of the two new lines until we don't need to break any

more. For a closed contour, we can find the two points that lie farthest apart and fit two lines between them, one for one side and one for the other. Then, we can apply the recursive splitting procedure to each side. First, use a single straight line to connect the end points. Then find the edge point with the greatest distance from this straight line. Then split the straight line in two straight lines that meet at this point. Repeat this process with each of the two new lines. Recursively repeat this process until the maximum distance of any point to the poly-line falls below a certain threshold. Finally draw the lines between the vertices of an edge of the reconstructed contour to obtain the polygonal approximating contour.

The approximation of arbitrary two-dimensional curves by polygons is an important technique in image processing. For many applications, the apparent ideal procedure is to represent lines and boundaries by means of polygons with a minimum number of vertices and satisfying a given fit criterion. An approximation algorithm is presented which uses an iterative method to produce a small - but not minimum - number of vertices that lie on the given curve. The maximum distance of the curve from the approximated polygon is chosen as the fit criterion.

Analysis of multiple views of the same scene is an area of active research in computer vision. The study of the structure of points and lines in two views received much attention in the eighties and early nineties [38], [39] and [40]. Studies on the constraints existent in three and more views have followed since then [41], [42], [43] and [44]. These multiview studies have concentrated on how geometric primitives like points, lines and planes are related across views. Specifically, the algebraic constraints satisfied by the projections of such primitives in different views have been a focus of intense studies.
Polygonal approximation is illustrated in Fig. 2.1

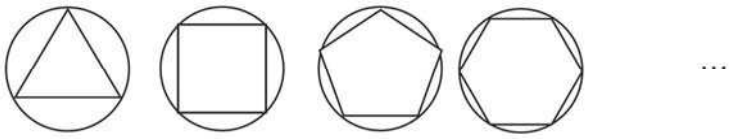


Fig. 2.1 Polygonal approximation.

Some practical methods for contour approximation are analyzed below.

2.2 Ramer Method

The algorithm is based on the maximum distance of the curve from the approximating polygon, and this distance is used as the fit criterion. The algorithm produces a polygon with a small number of edges for arbitrary two-dimensional digitized curves. The segment of the curve is approximated by a straight-line segment connecting its initial and terminus. If the fit is not fulfilling, the curve segment is terminated into two segments at the curve point most distant from the straight-line segment. This loop is repeated until each curve segment can be approximated by a straight-line segment through its endpoints. The termini of all these curve segments then are the vertices of a polygon that satisfies the given maximum-distance approximation criterion.

This type of polygonal curve representation exhibits two important disadvantages. First the polygons contain a very large number of edges and, therefore, are not in as compact a

form as possible. Second, the length of the edges is comparable in size to the noise introduced by quantization.

The idea is illustrated in the following Figures (see Fig. 2.2 to Fig. 2.11).

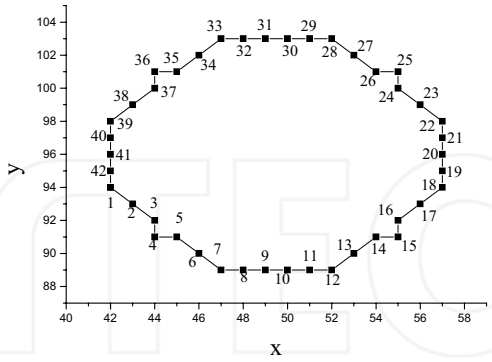


Fig. 2.2 The original contour.

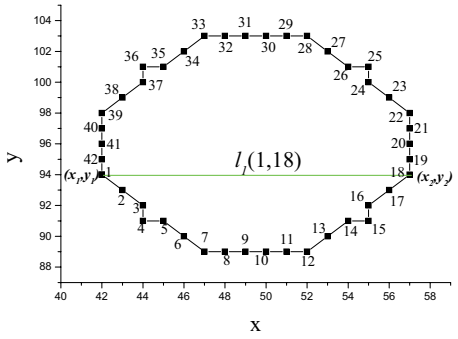


Fig. 2.3 The curve segment of straight line l_1.

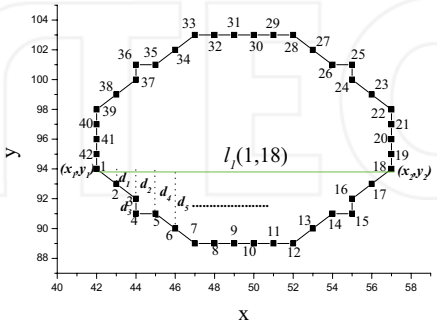


Fig. 2.4 Computation of the perpendicular distance between points in the curve segment and a line segment l_1.

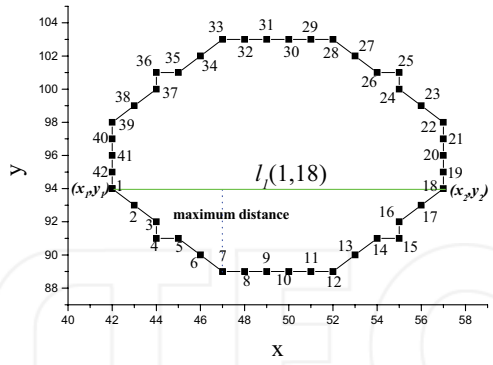


Fig. 2. 5 The maximum distance for the segment curve segment from the straight line l_1.

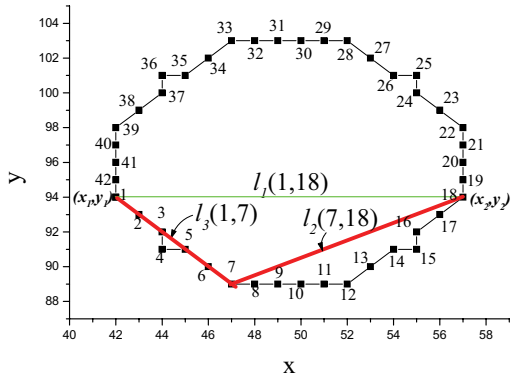


Fig. 2.6 The curve segments for the two new straight lines l_2 and l_3.

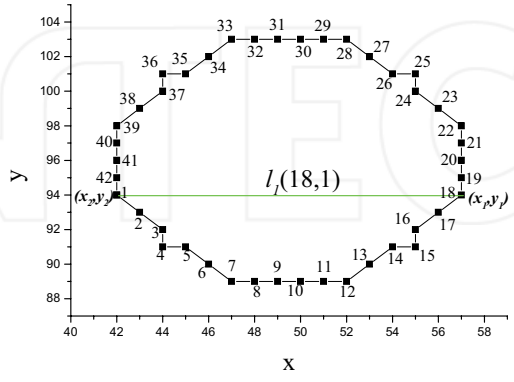


Fig. 2.7 The curve segment of straight line l_1.

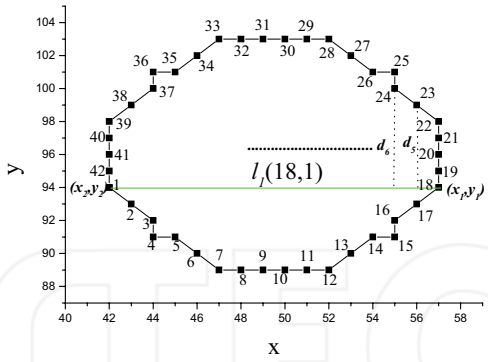


Fig. 2.8 Computation of the perpendicular distance between points in the curve segment and a line l_1.

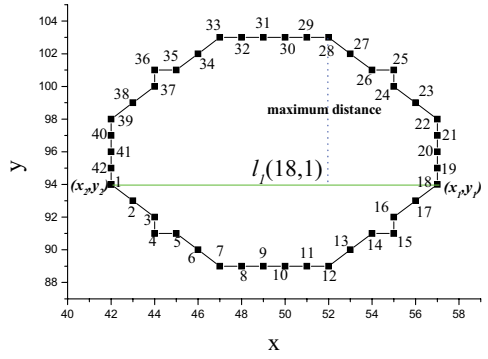


Fig. 2.9 The maximum distance for the curve segment from the straight line l_1.

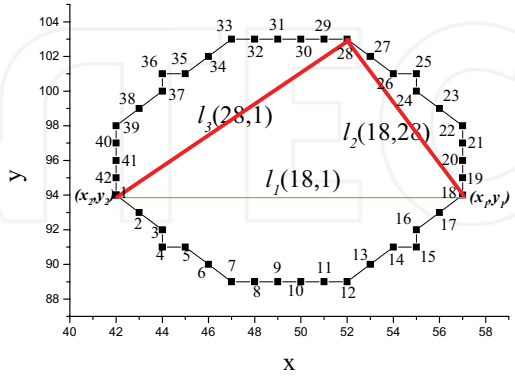


Fig. 2.10 The curve segments for the two new straight lines l_2 and l_3.

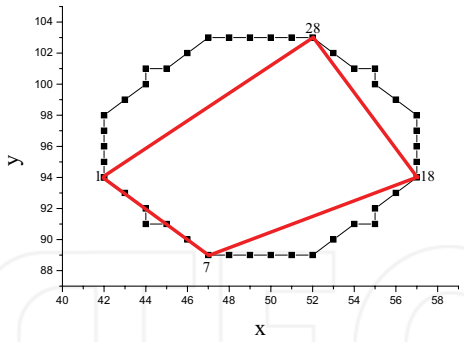


Fig. 2.11 The Original and approximated contours.

2.3 Triangle Methods of Contour Compression

The proposed algorithms belong to a family of polygonal methods of approximation. An input contour for the algorithms is extracted from 256 x 256 grey-scale images using Single Step Parallel Contour Extraction (SSPCE) method [34].

The approximation procedure starts at the time, when the first and last points of a segment are determined. The proposed criterion can be modified depending on contour representation methods. The most popular contour description methods are Freeman's chain coding, polar and Cartesian descriptions. Freeman chain coding can be used to distinguish all possible connections for both 8-connectivity and 4-connectivity schemes. A commonly used chain coding representation is the 8-Directional chain coding which uses eight possible directions to present all possible line segments connecting the nearest neighbors according to the 8-connectivity scheme. The contour extraction by these algorithms is based on (3 x 3) pixels window.

The Triangle family contains four methods of contour compression which are very similar to each other and the first method will be described in details in the following section.

A) Height Over Length Triangle Ratio Method

The algorithm refers to a quite new polygonal approximating method called the height over length ratio triangle method [46] and [47].

The idea of this method consists in segmentation of the contour points to get a triangle shape. The ratio of the height of the triangle (h) and the length of the base of the triangle (b) is then compared with the given threshold value as follows:

$$(h/b) < th \tag{4.1}$$

Where:

th - given threshold value.

The first point of each segment is called the starting point (SP) and the last one is called the ending point (EP). To calculate these values a simple trigonometric formula is used.

If the ratio value is smaller than the threshold according to Eqs. (4.1) the EP of the triangle is stored and SP is shifted to the EP, then a new segment is drawn. Otherwise the second point

(B) is stored and the SP is shifted to the B point of the triangle. Then a new segment is drawn. The stored points determine the vertices of an edge of the approximating polygon.
The algorithm scans contour points only once i.e. it does not require the storage of the analysed contour points. The original points of the contour are discarded as soon as they are processed. Only the co-ordinates of the starting point of the contour segment, and the last processed point are stored. The idea of the proposed algorithm is illustrated in Fig. 2.12. A flowchart of the proposed algorithm is depicted in Fig. 2.13.

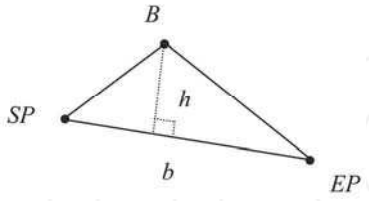


Fig. 2.12 Illustration of the basic triangle for the proposed algorithm where *h* and *b* are height and length of the triangle respectively.

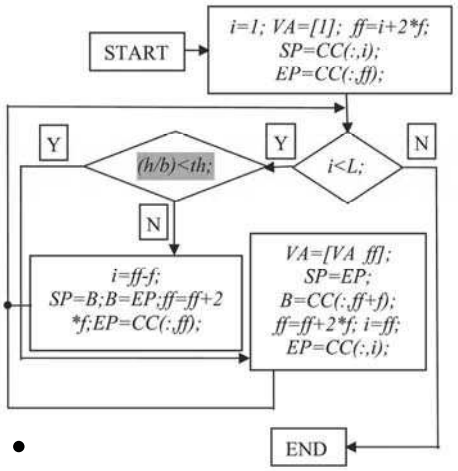


Fig. 2.13 Flowchart of the proposed algorithm.

where:
VA - sequence of indices of the final vertices;
CC - sequence of the input for the contour;
SP - starting point;
EP - ending point;
h ,*b* and *th* - as mentioned before (see Fig.4.13 and Eqs.4.1);
f - length between each two points of the triangle.

B) Height Triangle Method

The second algorithm refers to a recent polygonal approximating method called the height triangle method. The idea of this method is very similar to the previous algorithm. The only

difference is that the threshold is compared with the height of the triangle (shadow region in Fig. 2.13).
The third algorithm refers to a polygonal approximating method called the height square triangle method. The idea of this method is very similar to the previous algorithm. The difference is that the threshold is compared with the square height of the triangle (shadow region in Fig. 2.13).
The fourth algorithm refers to a recent polygonal approximating method called the triangle area method. In this case the threshold is compared with the area of the triangle (shadow region in Fig. 2.13).

2.4 Comparison Between the Triangle Family and Ramer Algorithms

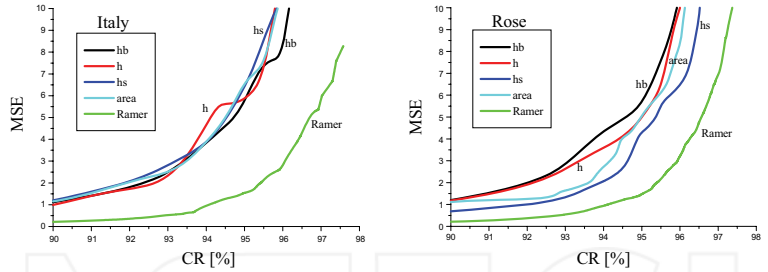
The computational complexity is one of the most important factors in evaluating a given method of approximation. High computational complexity leads to high implementation cost. The MSE (Mean Square Error) and SNR (Signal to Noise Ratio) criterions versus compression ratio are also used to evaluate the distortion.
The comparison is done for some test contours (Italy & Rose) which was extracted by using the "SSPCE" (Single Step Parallel Contour Extraction). The comparison is made between the following five algorithms:

- Height over length triangle (hb) method; it will be referred to as the first algorithm.
- Height triangle (h) method; it will be referred as the second algorithm.
- Height square triangle (hs) method; it will be referred as the third algorithm.
- Area triangle (area) method; it will be referred as the fourth algorithm.
- Ramer method; it will be referred as the fifth algorithm.

To visualise the experimental results a set of two test contours was selected. Selected contours are shown in Fig. 2.14.

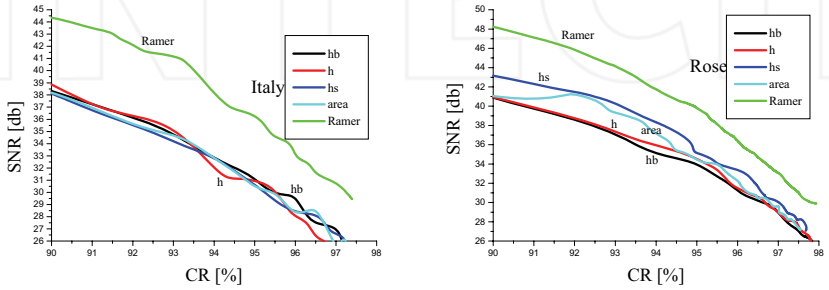
Fig. 2.14 Test contours: a) Italy b) Rose.

The comparison of the compression abilities versus the MSE & SNR are shown in the Fig. 2.15 & Fig. 2.16 respectively.



(a) (b)

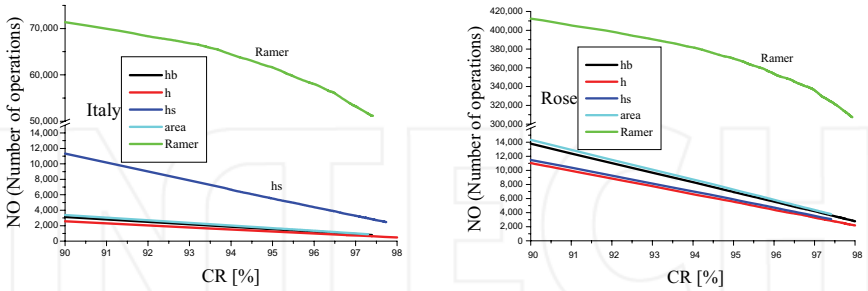
Fig. 2.15 MSE versus compression ratio for (a) Italy contour (b) Rose contour.



(a) (b)

Fig. 2.16 SNR versus compression ratio for (a) Italy contour (b) Rose contour.

Comparison of the compression abilities versus the number of operations is presented in Fig. 2.17.



(a) (b)

Fig. 2.17 NO versus compression ratio for (a) Italy contour (b) Rose contour.

The plots show that SNR using the Ramer algorithm is close to the triangle family methods for the rose contour; the reconstruction quality by the triangle family algorithms are very similar but the (hs) method is much better for complicated contour as in Rose contour. The number of operations is very similar between the triangle family algorithms at high compression. The triangle family algorithms are many times faster than that of Ramer

method. The compression ratio using triangle family methods can be even greater than 97% without significant lose of quality of compressed contour, but the complexity is much less than that of the Ramer algorithm.

3. References

[1] Luciano da Fontoura Costa and Roberto M. Junior, "*Shape Analysis and Classification: Theory and Practice", CRC Pr., 2001.*

[2] A. K. Jain, "Fundamentals of Digital Image Processing", New Jersey: Prentice Hall International, 1989.

[3] H. Freeman, "Application of the Generalized Chain Coding Scheme to Map Data Processing", Proc. of the IEEE Conference on Pattern Recognition, Image Proc..,Chicago,May,1987

[4] Milan Sonka, Vaclav Hlavac, and Roger Boyle, "Image processing, Analysis, and Machine Vision", Books and Cole Publishing 1998.

[5] Clarke R. J., "Transform Coding of Images", Academic Press, 1985.

[6] Brigham E.O., "The Fast Fourier Transform", Prentice-Hall, Englewood Cliffs, 1974.

[7] Walsh, J. L. "A Closed Set of Normal Orthogonal Functions", *Amer. J. Math.* 45, 5-24, 1923.

[8] Wolfram, S., "A New kind of Science", Champaign, IL: Wolfram Media, pp. 573 and 1072-1073, 2002.

[9] A. Dziech, F. Belgassem & H. J. Nern, "Image data compression using zonal sampling and piecewise-linear transforms," *Journal of Intelligent And Robotic Systems. Theory & Applications,* 28(1-2), Kluwer Academic Publishers, June 2000, 61-68.

[10] Kunt, M., *"TRAITEMENT NUMERIQUE DES SIGNAUX"*, Editions Presses polytechniques romandes, 1981 ISBN 2-88074-060-6.

[11] D. H. Ballard and C. M. Brown, "Computer Vision", Prentice Hall, Englewood Cliffs, NJ, 1982.

[12] R. M. Haralick and L.G. Shapiro, "Computer and Robot Vision", Addison-Wesley Publishing Co., 1992.

[13] B.K.P. Horn, "Robot Vision", The MIT Press, Cambridge, MA, 1986.

[14] W.K. Pratt, "Digital Image Processing", Wiley, New York, 1991.

[15] Capson D., Kitai R., "Performance Comparisons of Contour Extraction Algorithm", IEEE Instrumentation and Measurement Technology Conf., Boulder, USA, pp.99-103, March 1986.

[16] Fu. K. S., Mui J. K., "A Survey on Image Segmentation", Pattern Recognition, Vol.13, pp.3-16, 1981.

[17] Prager J. M., "Extracting and Labelling Boundary Segments in Natural Scenes", IEEE Transactions on PAMI, Vol. PAM1-2, No.1, pp.16-27, Jan. 1980.

[18] A. Nabout and H. A. Nour Eldin, "The Topological Contour Closure Requirement for Object Extraction from 2D-Digital Images", *Proc. 1993 IEEE International Conference on Systems, Man and Cybernetics (Le Touquet, France), 1993,* pp. 120-125.

[19] Canny, J., "A Computational Approach To Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

[20] Lindeberg, Tony, "Edge detection and ridge detection with automatic scale selection", International Journal of Computer Vision, 30, 2, pp. 117-154, 1998.
[21] Rhys Lewis, "*Practical Digital Image Processing",* Ellis Horwood, New York ,1990.
[22] J. Canny, "A Computational Approach to Edge Detection", *IEEE Trans. on Pattern* Analysis and Machine Intelligence **8:6** (1986) 679-698.
[23] J. S. Chen and G. Medioni, "Detection, Localization, and Estimation of Edges", *IEEE Trans. On Pattern Analysis and Machine Intelligence* **11:2** (February 1996) 191-198.
[24] W. E. L. Grimson and T. Pavlidis, "Discontinuity detection for visual surface Reconstruction", *Computer Vision, Graphics, and Image Processing* **30** (1985) 316-330.
[25] W. H. H. J. Lunscher and M. P. Beddoes, "Optimal edge detector design : parameter selection and noise effects", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **8:2** (March 1986) 164-177.
[26] D. Marr, E. Hildreth, "Theory of edge detection", *Proceedings of Royal Society of London,* **B:207** (1980) 187-217.
[27] L. S. Davis, "A survey of edge detection techniques", *Computer Vision, Graphics, and image processing 4 (1975),* pp. 248-270.
[28] R. M. Haralick, "Edge and region analysis for digital image data", *Computer Vision, Graphics and Image Processing (1980),* pp. 60-73.
[29] Sadayuki Hongo, Mitsuo Kawato, Toshi Inui and Sei Miyake, "Contour extraction of images on parallel computer-local, Parallel and stochastic algorithm which learns energy parameters", *International Joint Conference on Neural Networks (Washington D.C.), 1989,* pp. 161-168.
[30] P. K. Sahoo, S. Soltani, A. K.C. Wong and Y.C. Chen, "A survey of thresholding techniques", *Computer Vision, Graphics and Image Processing (1988),* pp. 233-260.
[31] Duda R. O., and Hart P. E., "Pattern Classification and Scene Analysis", John Wiley & Sons, New York, 1973.
[32] Heijden F., "Image Based Measurement Systems", John Wiley, 1995.
[33]] Nabout A., Su B., Nour Eldin H. A., "A Novel Closed Contour Extractor, Principle and Algorithm", *Proc. Of the International IEEE/ISCAS Conf. On Circuits and Systems, April 29 – May 3, Seattle, USA, Vol.1,* pp. 445-448, 1995.
[34] A. Dziech, W. S. Besbas, "Fast Algorithm for Closed Contour Extraction", *Proc. Of the Int. Workshop on Systems, Signals and Image Processing*, Poznań, Poland,1997, pp. 203-206.
[35] W. Besbas, "Contour Extraction, Processing and Recognition", Poznan University of Technology, Ph. D. Thesis, 1998.
[36] M. Bennamoun, B. Boashash, and A. Bower, "Object contour Extraction from motion parameters", *Proc. 2nd Australian and New Zealand Conference on intelligent Information Systems (Brisbane, Australia),* pp. 337- 341.
[37] M. Fleck, "Some defects in finite-difference edge finders", *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992),* pp. 337-345.
[38] O. Faugeras and Q. Luong, "The Geometry of Multiple Images", MIT Press, 2001.
[39] R. Hartley and A. Zisserman, "Multiple View Gemoetry in Computer Vision", Cambridge University Press, 2000.

[40] H. C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from two Projections", *Nature*, 293:133-135, 1981.

[41] R. Hartley, "Lines and points in three views: An integrated approach", *Proc. ARPA Image Understanding Workshop*, 1994.

[42] A. Shashua, "Algebraic Functions for Recognition", *IEEE Tran. Pattern Anal. Machine Intelligence*, 16:778-790, 1995.

[43] A. Shashua, "Trilinear tensor: The Fundamental Construct of Multiple-view Geometry and its applications", *Int. Worskshop on AFPAC*, 1997.

[44] B. Triggs, "Matching Constraints and the Joint Image", *International Conference on Computer Vision*, pages 338-343, 1995.

[45] U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves", *Computer Graphics and Image Proc., Academic Press*, 1972, pp.244-256.

[46] Andrzej Dziech, Ali Ukasha and Remigiusz Baran, "A New Method For Contour Compression", *WSEAS Int. Conf. on Signal, Speech and Signal Processing (SSIP 2005),* Corfu Island, Greece, August 17-19, 2005, pp.282-286.

[47] Andrzej Dziech, Ali Ukasha and Remigiusz Baran, "Fast Method For Contour Approximation And Compression", *WSEAS Transactions on Communications Issue 1, Volume 5,* January 2006, pp. 49 - 56.

Mobile Robots: Perception & Navigation

Edited by Sascha Kolski

ISBN 3-86611-283-1

Hard cover, 704 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, February, 2007

Published in print edition February, 2007

Today robots navigate autonomously in office environments as well as outdoors. They show their ability to beside mechanical and electronic barriers in building mobile platforms, perceiving the environment and deciding on how to act in a given situation are crucial problems. In this book we focused on these two areas of mobile robotics, Perception and Navigation. This book gives a wide overview over different navigation techniques describing both navigation techniques dealing with local and control aspects of navigation as well es those handling global navigation aspects of a single robot and even for a group of robots.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Andrzej Dziech (2007). Contour Extraction and Compression-Selected Topics, Mobile Robots: Perception & Navigation, Sascha Kolski (Ed.), ISBN: 3-86611-283-1, InTech, Available from: http://www.intechopen.com/books/mobile_robots_perception_navigation/contour_extraction_and_compression-selected_topics

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821