*Optica Applicata, Vol. XXXI, No. 3, 2001*

# Fast contour tracing algorithm

JANUSZ KOZŁOWSKI

Associazione Istituzione Libera Università Nuorese, Dipartimento Technologie Ottiche, Viale della Resistenza, 39, I-08100 Nuoro, Italy.

KRZYSZTOF PATORSKI

Warsaw University of Technology, Faculty of Mechatronics, Institute of Micromechanics and Photonics, ul. Chodkiewicza 8, 05–252 Warszawa, Poland.

A simple and fast algorithm for contour tracing in two-level BitMap image is presented. The common corner of four neighbouring pixels, NODE, is defined as the basic element of the analysed structure. First the image is convoluted with $2 \times 2$ kernel which identifies all nodes, then the algorithm follows the line connecting contour nodes, and extracts coordinates of border pixels according to the initial conditions, *i.e.*, to the defined connectivity type and contour rotation. The main parts of this procedure (written in $C^{++}$) and results obtained with the demonstration program KRATA, based on this algorithm, are presented.

## 1. Introduction

One of the most natural features enabling 2D-object definition is its contour. It is often used in different object recognition procedures as a cloud of border pixels (Hough transform) or their ordered chain [1]. The chain coding is also one of the main instruments applied in the field of image compression [2].

The use of vectorial description of the border of the analysed object may be very convenient also for other applications. The system of shape measurement presented in [3] uses the contour vector as a register of starting points of the line-by-line analysis of the fringe pattern.

In this paper, a simple and fast procedure for finding the contour line ordered coordinates is presented. Other existing procedures, which could be applied for this purpose and described in a number of works, are in our opinion more complex. ZINGARETTI, GASPARRONI and VECCI [4] gave a short review of these methods and proposed their own algorithm based on the application of 57 templates (of $2 \times 3$-pixels) and the original idea of simultaneous definition of multilevel contours of adhering regions. The algorithm proposed in [4] gives the possibility of a complete topological analysis of the image. The method of template identification, however, is time consuming and makes the algorithm too heavy for some simple applications.

The contemporary definition of both contours, internal and external, requires virtual separation of the analysed regions along the lines passing between pixels

(through the nodes). This auxiliary interpretation, not formulated explicitly by Zingaretti and Vecci, was the starting point of our algorithm. The node recognition method described could be applied for global contour tracing, which is the case in the present paper, or, for huge images, using on-scan segment creation and their linking-up as proposed by Zingaretti.

## 2. Description of the algorithm

The immediate consequence of distinction of nodes as the main elements of the image is the observance that in the specific case of two-colour bitmap any node is formed by 4 pixels. This means that the number of the relative templates may be reduced to 16.

The automatic "recognition" of templates (attributing numbers to nodes) can be done by convoluting the two-level bitmap with the kernel **K** (Fig. 1**b**). The code numbers (CN) resulting from this convolution and the relative templates are shown in Fig. 1**a** in the first two columns.

The contour tracing procedure described below uses a set of six driving parameters. In almost all situations, schematically illustrated in the second column (see Fig. 1**a**), the assumption of the contour line type (*e.g.*, counterclockwise) is sufficient to determine where and where from it passes through the node. In two cases (CN = 6 and CN = 9) the contour line passes twice through the same node, so in order to distinguish between these cases, it is necessary to take into consideration the contour line entrance direction.

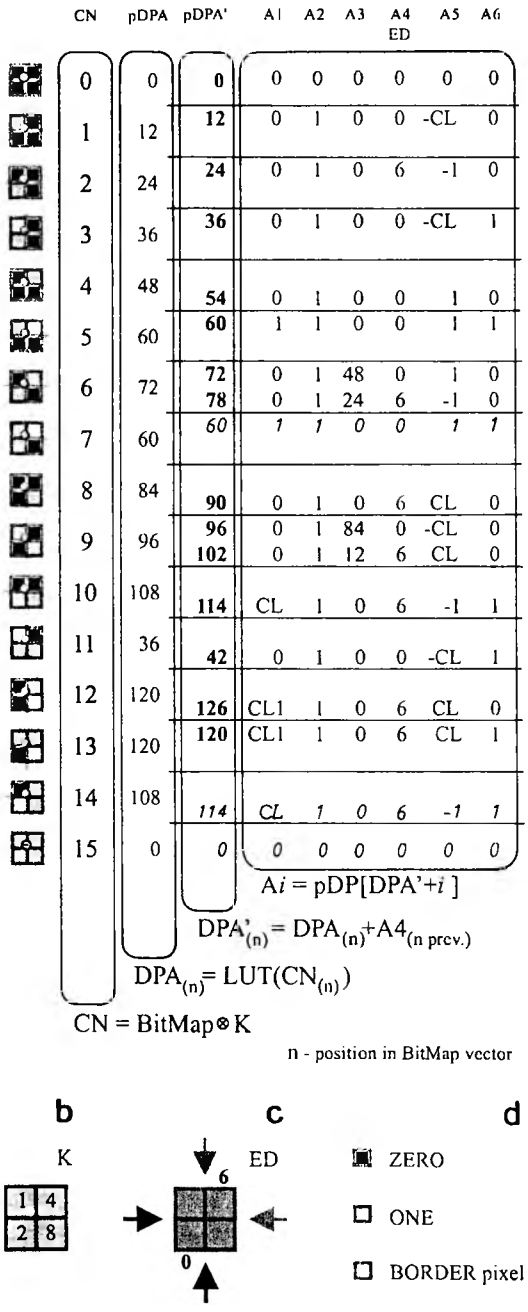
The efficiency of our algorithm demonstrates that it is sufficient to define only two cases of entrance direction (ED) type, first when the contour line is entering from left or down, and the second – from the right or up.

Sixteen code numbers have been associated with 11 basic node types. Two possible ED types increase the effective number of node types (however, it cannot be grater than 22).

The contour tracing action to be completed in each node is driven in our procedure by 6 driving parameters (DP) so the total number of DP cannot be greater than 132. We have defined all possible DP values as elements of the pDP matrix. To access the required set of 6 driving parameters it is sufficient to pass the address of the first element of this set (driving parameter address, DPA).

The association between CN and DPA is obtained by placing all DPA values in the pDPA matrix, so that their positions correspond to the CN value (sixteen element matrix).

In practice, in the first step of the procedure the DPA values are directly associated with nodes of the analysed BitMap – see the third column in Fig. 1**a** and the algorithm of convolution reported below.




| CN | pDPA | pDPA' | A1 | A2 | A3 | A4 ED | A5 | A6 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 12 | **12** | 0 | 1 | 0 | 0 | -CL | 0 |
| 2 | 24 | **24** | 0 | 1 | 0 | 6 | -1 | 0 |
| 3 | 36 | **36** | 0 | 1 | 0 | 0 | -CL | 1 |
| 4 | 48 | **54** | 0 | 1 | 0 | 0 | 1 | 0 |
| 5 | 60 | **60** | 1 | 1 | 0 | 0 | 1 | 1 |
| 6 | 72 | **72** | 0 | 1 | 48 | 0 | 1 | 0 |
| | | **78** | 0 | 1 | 24 | 6 | -1 | 0 |
| 7 | 60 | *60* | *1* | *1* | *0* | *0* | *1* | *1* |
| 8 | 84 | **90** | 0 | 1 | 0 | 6 | CL | 0 |
| 9 | 96 | **96** | 0 | 1 | 84 | 0 | -CL | 0 |
| | | **102** | 0 | 1 | 12 | 6 | CL | 0 |
| 10 | 108 | **114** | CL | 1 | 0 | 6 | -1 | 1 |
| 11 | 36 | **42** | 0 | 1 | 0 | 0 | -CL | 1 |
| 12 | 120 | **126** | CL1 | 1 | 0 | 6 | CL | 0 |
| 13 | 120 | **120** | CL1 | 1 | 0 | 6 | CL | 1 |
| 14 | 108 | *114* | *CL* | *1* | *0* | *6* | *-1* | *1* |
| 15 | 0 | *0* | *0* | *0* | *0* | *0* | *0* | *0* |

Fig. 1. Schema of the processing principle applied (**a**), coding kernel **K** (**b**), entrance direction coefficient coding (**c**), colour code of bitmap values (**d**).

The fourth column in Fig. 1**a** represents the specific-case DPA values obtained by adding the parameter relevant to the entrance direction coefficient to the DPA value.

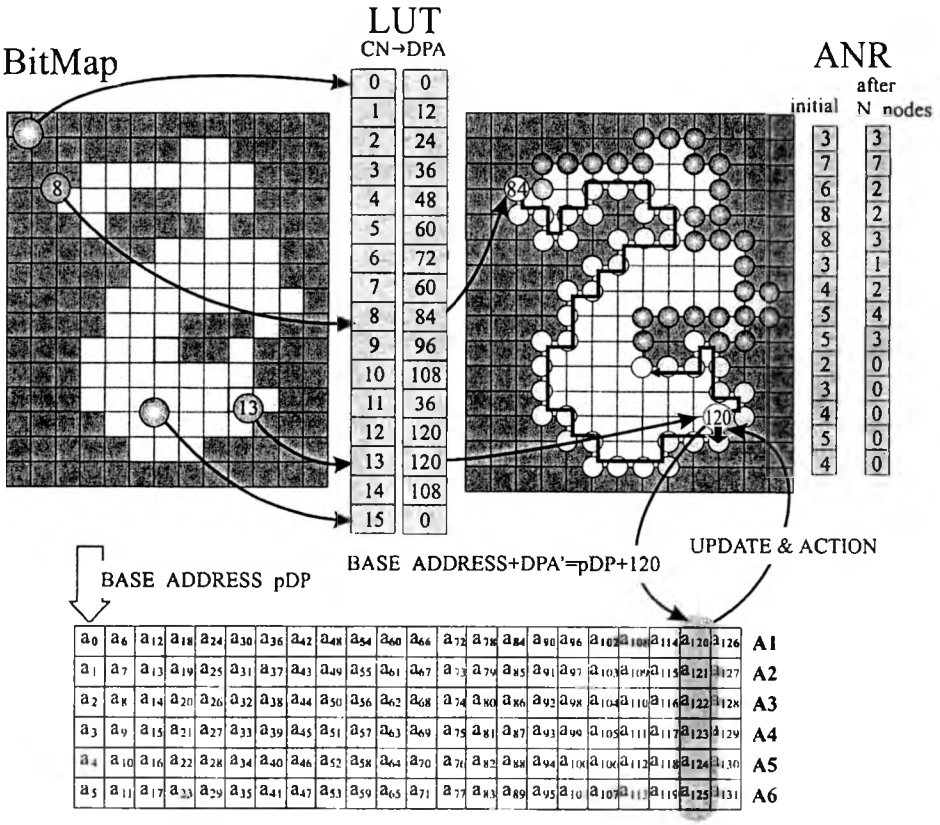



Fig. 2. Schema of example contour tracing.

The specific functions of each of the six driving parameters (denoted by **A1** through **A6** in Fig. 1**a**) are described below.

1. *Associating the border pixel with the node type*

The idea of defining the contour line with the use of nodes instead of pixels does not change the fact that what has to be really found is the sequence of border pixels (pBord vector). The proposed system of association between node type and border pixel is schematically explained in Fig. 1**a** (the first column) and in Fig. 1**d**. The corresponding driving parameter A1 (0; 1; CL; CL1) defines position of the border pixel with respect to the current node.

2. *Finding the first point of the contour loop and the problem of completeness of the analysis*

To minimise the number of accesses to nodes we propose to create the active-nodes register (ANR) enabling external control of the state of the analysis. The

ANR is a vector of the length equal to the number of rows minus 1, and contains the values related to the number of active nodes in each row. Passing through the node reduces the proper ANR value by 1 (except nodes 6 and 9). After having closed the contour loop, the values remaining in the ANR correspond to other loops (not analysed yet). Position of the first non zero value of DPA in the row for which ANR is not ZERO indicates the first node of the next contour loop. The above counting action within ANR is controlled by the driving parameter **A2** (0; 1).

3. *Internal updating of pDPA.*

The driving parameter addresses, corresponding to the nodes passed, are modified in the cases of nodes 6 and 9, and erased in other cases. This makes possible to visit again the nodes in the first case, and to exclude them from further analysis in the second case. The new value of DPA constitutes a driving parameter **A3**.

4. *Definition of the contour line entrance direction.*

Depending on the contour entrance direction (exit from the previous node) the driving parameter address has to be shifted by 6 (to the next DP set) or not. The driving parameter **A4** (0; 6) corresponds to this shift.

5. *Direction indicating the next node position* can be defined by one of four possible values: DOWN, LEFT, RIGHT and UP. These values are defined by the fifth driving parameter **A5** (-CL; -1; 1; CL). The parameter A4 could be easily associated with A5, however their separate definition saves processing time.

6. *As mentioned in point 1, not every node is associated with a border pixel;* so in general, the counting of nodes and construction of the border pixel sequence (addition of new pBord vector elements) have to be controlled independently. This function is governed by the sixth driving parameter **A6** (0; 1). The pBord vector contains, as even elements, position of the border pixel in sequence of the current contour loop, and as odd elements, position of the border pixel in the analysed BitMap.

The contour tracing procedure is schematically illustrated in Fig. 2.

The test program ran on 350 MHz PC analyses the contour vector of the length of over 2900 points (within the 8 Bit grey level, $512 \times 340$ pixel BitMap) in 53 ms (less than 20 ns per node). The $[X(n),Y(n)]$ graphs of the exemple contour and the form of the analysed mask are shown in Fig. 3 (the execution window of the test-program KRATA).

To demonstrate the simplicity of the algorithm applied two principal parts of the $C^{++}$ program written by the authors with Builder 3.0 are presented below (reflecting the schema shown in Figs. 1a and 2).

In the first part of the algorithm the pDPA look up Table (DPA vs. CN) and non-zero values of the driving parameters vector are declared, then the way in which these parameters have been used to calculate the BitMap-associated matrix of DPA values and the active-nodes register are presented.

```
//_____DRIVING VALUES_____
//=============== pDPA: Driving Parameters group Addresses vs. Code Number LUT
pDPA[1] =12; pDPA[2] = 24; pDPA[3] = 36; pDPA[4] = 48; pDPA[5] = 60;
pDPA[6] =72; pDPA[7] = 60; pDPA[8] = 84; pDPA[9] = 96; pDPA[10] = 108;
pDPA[11] =36; pDPA[12] = 120; pDPA[13] =120; pDPA[14] = 108;
//=============== Non-zero components of the 132-element "Driving Parameters" vector
pDP[13] = 1; pDP[61] = 1; pDP[127] = 1; pDP[117] = 6; pDP[126] =CL1;
pDP[25] = 1; pDP[64] = 1; pDP[131] = 1; pDP[123] = 6; pDP[16] =□CL;
pDP[37] = 1; pDP[65] = 1, pDP[28] = □1; pDP[129] = 6; pDP[40] =□CL;
pDP[41] = 1; pDP[82] = 1; pDP[76] = □1; pDP[94] = CL; pDP[46] =□CL;
pDP[43] = 1; pDP[91] = 1; pDP[118] = □1; pDP[106] = CL; pDP[100] =□CL;
pDP[47] = 1; pDP[119] = 1; pDP[27] = 6; pDP[124] = CL; pDP[104] =12;
pDP[55] = 1; pDP[115] = 1; pDP[75] = 6; pDP[114] = CL; pDP[80] =24;
pDP[58] = 1; pDP[121] = 1; pDP[93] = 6; pDP[130] = CL; pDP[74] =48;
pDP[60] = 1; pDP[125] = 1; pDP[105] = 6; pDP[120] = CL1;pDP[98] =84;
// CL1 is a number of columns of the BitMap plus 1,
// CL is a number of columns of the BitMap

//_____CONVOLUTION and "ANR" CREATION_____
for (i = 0; i < ((LinesN □1)*CL □1); i++)
{ pDPA[i] = *(pDrv + pBitMap[i] + pBitMap[i+CL]<<1+ pBitMap[i+1]<<2 + pBitMap[i+CL1]<<3);
                                        // ↑ convolution ↑
  contat1 += (bool)pDPA[i];             // counting active points in the current row
  if ((i+2)%CL == 0)                    // end of the row
  { *pANR= contat1;
    contat1 = 0;
    pANR++;
  }
}
//=============================================

//=============================================
while(FindNextPoint(&Node)) // Till not all values in ANR are zeros
{ EntrDirCoef = 6;  // Initial value of the Entrance Direction Coefficient
  pBord[0]    = 1;  // Initialization of border-pixels counter in the new loop
  while (pDPA[Node]) // Till the first node is not encountered for the second time (as empty)
  { LocPointer = pDP + pDPA[Node] + EntrDirCoef;  // Which group of Driving Parameters
                                                  // has to be used?
    pBord[1] = Node + *(LocPointer++);      // *A1, BorderPixel inserted in the sequence
                                            //      (Odd pos. in the vector)
    pANR[Node/CL] -= *(LocPointer++);       // *A2, Updating Active Nodes Register
    pDPA[Node]     = *(LocPointer++);       //*A3, New Driv. Param. group Address value
    EntrDirCoef    = *(LocPointer++);       // *A4, Entrance Dir. Coef. for the next node
    Node           += *(LocPointer++);      // *A5, Next Node coordinate
    pBord[2]       = pBord[0] + *(LocPointer);  // *A6, Even pos. in the pBord vector is
                                                //its number in the loop
    pBord += (*LocPointer)<<1;  // *A6, Pointer to the next two "BorderPixel description cells"
  }
}
//=============================================
```

The second part (below) illustrates the main tracing-procedure updating, node -by-node, all the relative parameters and calculation of the border pixels vector (pBord). The *FindNextPoint* function operating on the ANR vector is elementary simple and its description has been skipped. No other user-defined functions have been used in the demonstrative program.

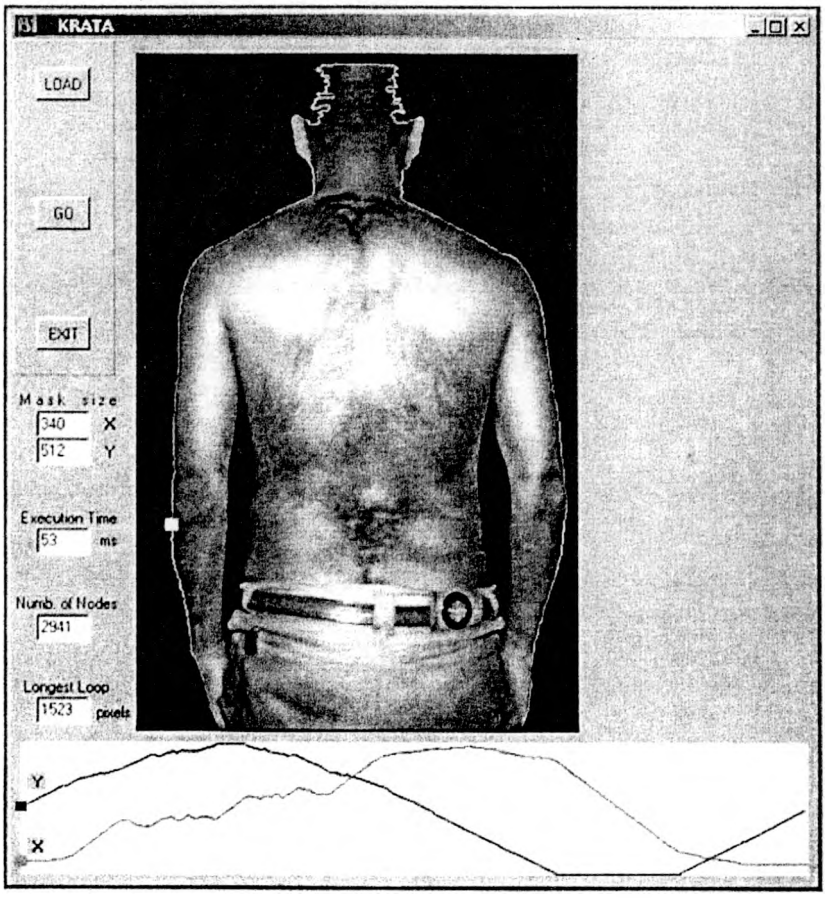



Fig. 3. The execution window of the demonstrative program KRATA written by the authors.

The graph in Figure 3 illustrates the longest loop of the traced contour. Even if all the loops are traced the extracted longest one defines, in the best way, the region of interest in our application.

## 3. Comments

For application purposes we assumed four-connectivity of the contoured objects (pixels are side-connected only with the object area under consideration). For other purposes however, assuming eight-connectivity (pixels are vertex node or side-connected with the object area under consideration), different values of the pDPA vector and pDP matrix could be defined, and similar algorithm could be applied as well. In spite of the overall size of the pDP matrix – 22 DP sets, only fifteen of them are significant in the present version of the algorithm, and only 45 DP values had to be defined (not equal to ZERO).

Adding other driving parameters (above 6) could enable extraction of some other features of the contours, *i.e.*, their length, encircled areas, chain-code representation, *etc.*

## References

[1] ZAMPERONI P., *Metodi dell'elaborazione digitale di immagini.* Masson, Milano 1990.
[2] CEDERBERG R., Comp. Graph. Image Proc. **10** (1979), 224.
[3] KOZLOWSKI J., SERRA G., Appl. Opt. **38** (1999), 2256.
[4] ZINGARETTI P., GASPARRONI M., VECCI L., IEEE Trans. Pattern Analysis and Machine Intelligence, **20** (1998), 407.

*Received February 5, 2001*