

# ALGORITHM FOR SEMIAUTOMATED VECTORIZATION

**Cpt. Eng. Alexei Adrian Mihai**

*Defence Research Agency*

*PO Box 51-16 Bucharest, Zip 765550, Romania*

*Fax: +40-1-4231030*

This algorithm can be used for the semiautomated raster/vector conversion (interactive). It uses a raster file as support, which is produced by scanning map originals as color map separates.

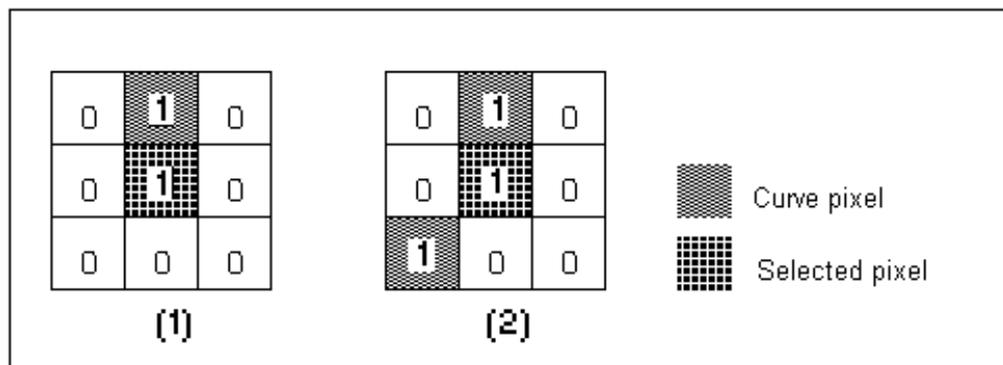
The algorithm proposed can be utilized for raster-vector conversion of linear and areal features within the map.

## 1. Digitizing the linear features

The first stage assumes the fixing of an end of the linear feature to be further digitized. The approximate determination of the starting point coordinates is made by means of a locator. In order to find the accurate coordinates of the starting pixel two methods are applied, depending on the linear feature type:

- for an open curve: it is chosen the pixel on the curve with the feature of having only a neighbouring pixel set on 1, the rest of pixels being 0 (figure 1 (1) ).

- for a closed curve: it is chosen the pixel on the curve with the feature of having only two neighbouring pixels set on 1, the rest of neighbouring pixels being 0 (figure 1. (2) ).



The second stage consists in following the linear feature based on the circle method. This method assumes the analysis of pixels found on  $C(x_0, y_0, r)$ , where  $(x_0, y_0)$  are the current vertex coordinates.

The equation of the circle is given by the relationship (1):

$$\begin{aligned} x &= x_0 + r \cdot \cos(\varphi) \\ y &= y_0 + r \cdot \sin(\varphi) \end{aligned} \quad (1)$$

Where  $r$ : represents the radius (it is constant);

$$\begin{aligned} - \varphi &= 0..360 \\ - x_0, y_0 &: \text{the coordinates of the current vertex;} \end{aligned} \quad (2)$$

After a complete exploring around the current vertex  $(x_0, y_0)$ , the following 3 situations are possible:

- 1 – There is only one pixel set on 1;
- 2 - There are at least two pixels set on 1;
- 3 – There is no pixel set on 1;

1. In this case a new vertex of the linear feature was found, of  $(x, y)$  coordinates given by the formula (1). Further on, the current point will become  $(x, y)$  ( $x_0=x, y_0=y$ ) and it goes on the application of the circle method (see figure 2. (1) ).

2. In this case an intersection of linear features was encountered. In this situation, the exploration of the linear tract stops and the program will ask the user to point the direction it will keep exploring (see figure 2. (2) ). The coordinates of the intersection points are stored on a list, in order to be able to come back in this point.

3. In this case the endpoint of the linear feature was found and the analysis is over (see figure 2 (3) ).

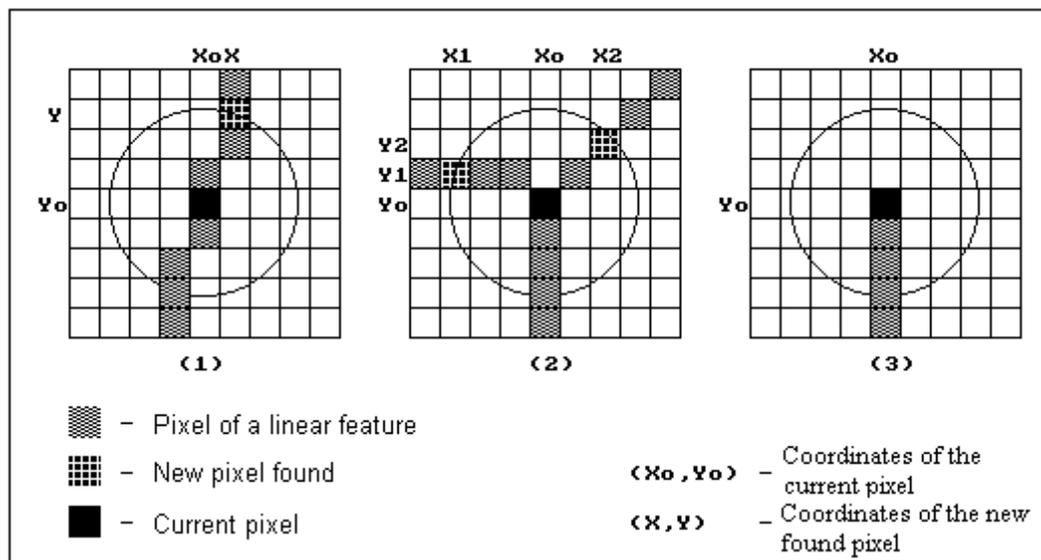


Figure 2. Possible cases.

## 2. Digitizing the areal features

The first stage assumes the determination of a pixel on the outline of the areal to be further digitized. The areal must be pointed by means of a locator and automatically is searched an feature belonging to the outline. An outline pixel has the property that not all the neighbouring pixels are set on 1 (see figure 3).

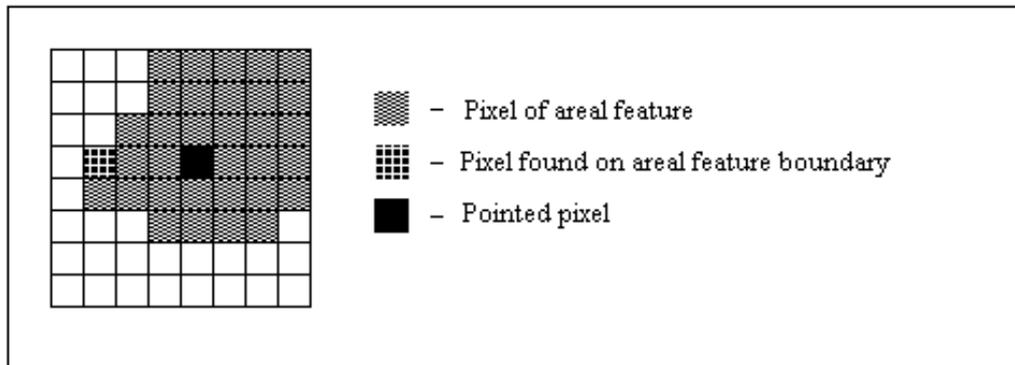


Figura 3. First pixel on an areal feature.

The second stage consists in following of the outline of the areal feature. This one goes on as well as in the digitizing of the linear features until the pixel on the outline is found (see figure 4). The following of the areal feature is over when it is reached the initial point.

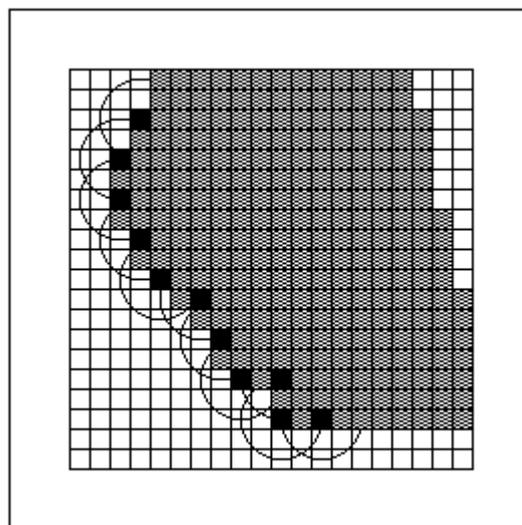


Figura 4. Tracking boundary.

Pseudocode for this algorithm:

\* read image

```

* select first point
Gata = False;
while ( Gata = False ) do
  NrPuncte = 0;
  for  $\varphi$  de la 0 la 360 do
    x =  $x_0 + r \cdot \cos(\varphi)$ ;
    y =  $y_0 + r \cdot \sin(\varphi)$ ;
    * read value for a pixel at coordinates (x,y);
    if (valoare pixel = 1) then
      NrPuncte = NrPuncte + 1;
  □
□
if ( NrPuncte = 0 ) then
  Gata = True;
□
if ( NrPuncte = 1 ) then
  x =  $x_0$  ;
  y =  $y_0$  ;
□
if (NrPuncte  $\geq$  2) then
  * pointing the track to follow;
  x =  $x_0$ ;
  y =  $y_0$ ;
□
□

```

In figure 5 it is presented a part of a layer with the representation of the relief. This raster image was vectorised by means of the proposed algorithm, and a vector image was obtained (see figure 6).

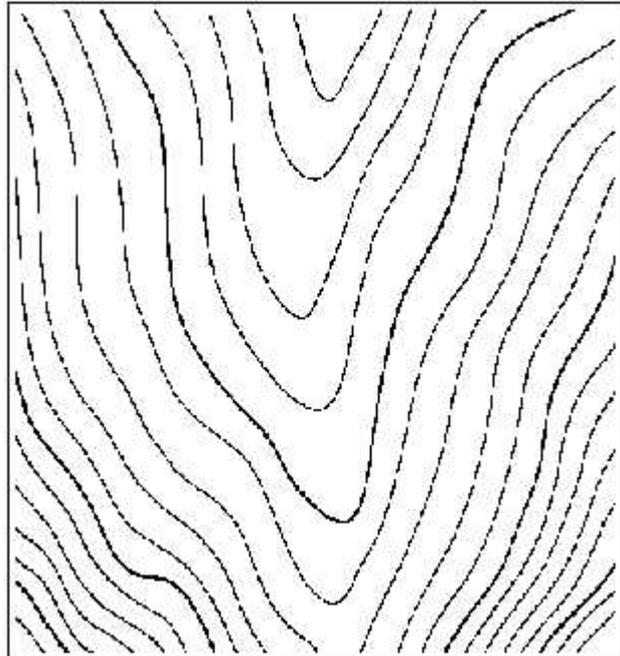


Figure 5. Raster image of the contour lines.

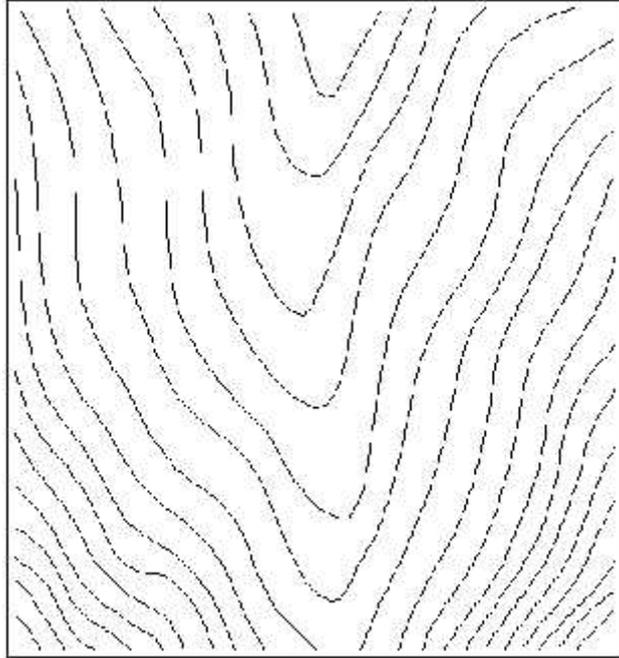


Figure 6. Vector image of the contour lines.

### 3. Conclusions

The proposed algorithm is very easy to implement and provides a satisfying speed of execution. Depending on the loading level of the image, there is the possibility of modifying the exploration radius  $r$ . If this radius increases, the vectorisation speed of the features increases, but some details are lost. It has the advantage that it can find and/or passes over the breaks within the linear features, depending on  $r$  value. Another major advantage is the fact that it doesn't need the preliminary squeletting of the image to be further digitized (see figure 7).

Pixels of  $(x_1, y_1)$  and  $(x_2, y_2)$  coordinates are determined following the exploration process. The coordinates of the pixel on the axis of the linear feature is determined with the formulae (3).

$$\begin{aligned}x_c &= (x_1 + x_2)/2; \\y_c &= (y_1 + y_2)/2;\end{aligned}\tag{3}$$

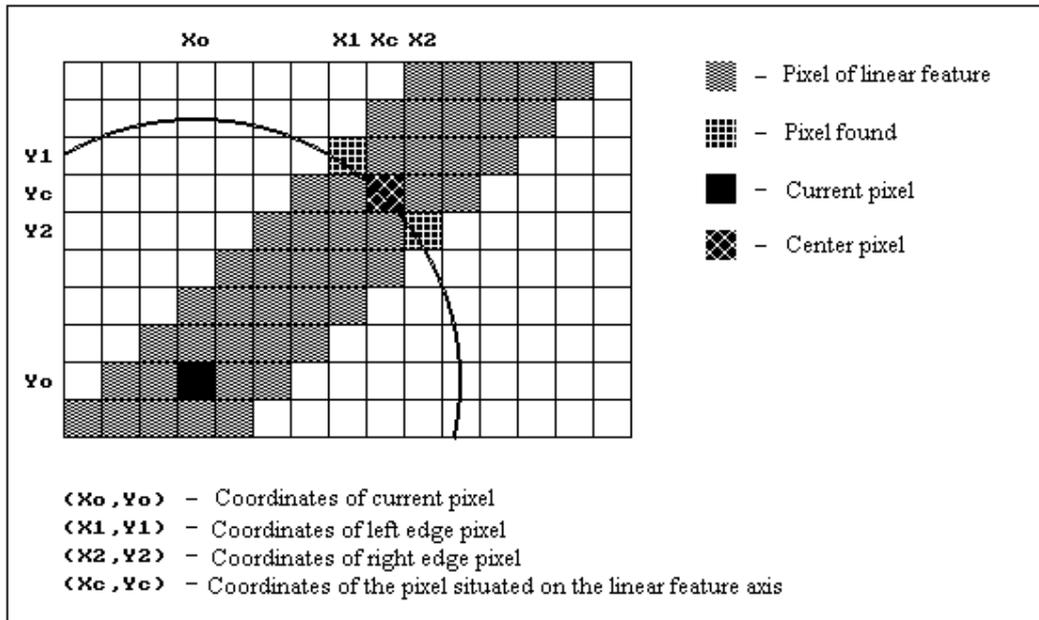


Figure 7. Tracking thick linear features.

#### 4. References

Nițu, C.(1995), Mathematical Cartography. Military Technical Academy Press, Bucharest.

\*\*\* ,(1973) Geodetic Engineer Handbook. Technical Press, Bucharest.