Bull. Nov. Comp. Center, Comp. Science, 6 (1997), 87–98

Cellular algorithm for isoline extraction from a 2D image

A. Selikhov

A cellular algorithm for extraction of isolines from 2D image of 3D surface is presented. The algorithm is divided into two stages: secondary quantization and extraction of isolines. Realization of each stage as one-time iteration on the base of the Parallel Substitution Algorithm is obtained. Results of the algorithm simulation in a parallel substitution simulation system ALT are presented for an example of a 256-colours grayscale surface image processing. Cellular structures for extraction of two types (dense and minimal) isolines are also suggested.

1. Introduction

The task of isoline extraction from a 2D image is based on a representation of the image being processed as a continuous function $z = \mathbf{F}(x, y)$, where x and y are coordinates on the image plane and z represents a colour of each image point. Such representation allows to consider a 2D image as a 3D surface and to describe any of different image properties as a third coordinate z.

Up-to-date systems for graphical information processing are based on the using of a primary quantized image representation obtained from a continuous one by the well-known image quantization methods. Therefore, a quantized image characterized by a discrete function $z = F(x, y)$ is assumed to be available. An image in Windows bitmap format with 256 gradations of a gray colour is used as an example.

The solving of the isoline extraction task is based on an image analysis aimed to clarify for each image element (pixel) whether it belongs to an isoline. So, this image processing can be presented exactly as extraction of contours from the initial image but not as construction of them. Obviously, the problem is local, i.e., to make any decision about each pixel it is unnecessary to have information about all image pixels, but it is enough to have it about pixels from a limited neighbourhood.

Discreteness and locality as peculiarities of the isoline extraction task, allow to solve it by means of cellular algorithms as most appropriate ones and to use cellular automata or cellular neural networks as special structures for hardware implementation. Because these structures are initially parallel, simulation of algorithms which will be realized on these structures should

be carried out by parallel or quasi-parallel graphical processing systems. Taking into account all above peculiarities, it is possible to formulate the following requirements for the isoline extraction algorithm. This algorithm should be:

- parallel and synchronous;
- based on a graphical language with graphical representation of operations used;
- easy mapped onto architecture of a special parallel processors.

To satisfy these requirements the Parallel Substitution Algorithm (PSA) [1] is chosen as a theoretical base. Simulation of the isoline extraction algorithm is realized in the parallel substitution simulation system ALT [2].

There are three parts in this paper. Basic formal notions and definitions concerned with given images and required for a formal representation of the task and the algorithm are introduced in the first part. The second part contains the algorithm description. Some peculiarities of the algorithm and simulation results obtained by processing of a 256-colours grayscale surface image are presented in the third part.

2. Basic notions and definitions

Assume that an image to be processed is represented in a continuous colour spectrum. Being used in computer processing systems, it is primary quantized into array of discrete elements (pixels), each of them having a certain colour. A colour of a pixel may be represented by an integer value from some range, for example from 0 to 255. For the isoline extraction task such quantized image is an initial graphical representation of a discrete function $z = F(x, y)$, where a colour of each pixel is coded by the variable z. A line $z = \text{const}$ in a graphical representation of the function $z = F(x, y)$ is called an *isoline*.

The isoline procedure should be done in two stages: secondary quantization and extraction of isolines proper.

1. Secondary quantization. At this stage the image is quantized *for a second time*. A step h equal to the distance between isolines is given in units of measurements of the variable z. The result of secondary quantization may be illustrated by Figure 1. Figure 1a represents a projection of the function $z = F(x, y)$ onto the plane XZ. Figure 1b is the projection representation after a second quantization process. This quantization also results in a colour scale transformation but with a less number of colours, than that of the initial one.

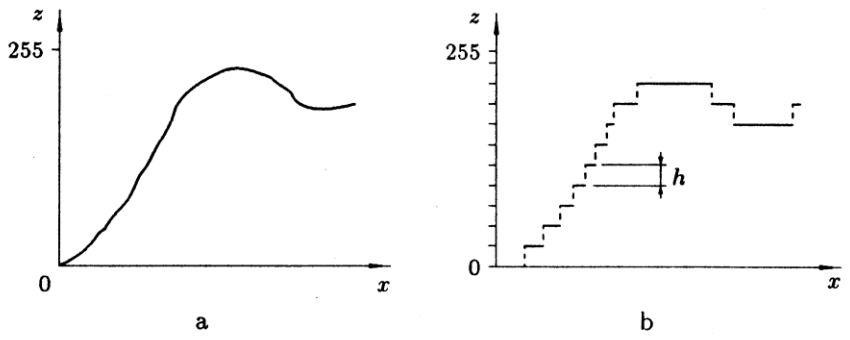


Figure 1. A representation of projections of the function F: a) initial (primary quantized) function; b) secondary quantized function

2. Extraction of isolines. The object under processing is a quantized image of a rectangular form divided into many cells (pixels) whose size is determined by required precision. Suppose that the image has n pixels along the vertical coordinate and m pixels along the horizontal one. So, each pixel has its own coordinate pair and colour. Let the pixel colour be coded by an element from a finite set $C = \{c_0, c_1, c_2, \ldots, c_r\}$, $c_i \in \mathbf{N}$, and let M be a finite set of coordinate pairs

$$M = \{\langle i, j\rangle \mid 1 \le i \le n,\ 1 \le j \le m,\ n, m \in \mathbf{N}\}.$$

Then a pixel may be denoted as $p = (c, \langle i, j\rangle)$, where $c \in C$.

Let the colour $c_{\min} = \min\{C\}$ ~~in~~ the set C be referred to as a background colour, and the set $B = \{(c, \langle i, j\rangle) \mid c = c_{\min}\}$ as a background respectively. Thus, an initial quantized image is an array of all pixels p with coordinates from M. Let us define a subset $A \subset C \times M$ as a cellular array, in which there are no identical coordinate pairs from M. To avoid some difficulties associated with processing pixels on the border of the image, let us introduce the following sets:

$G_1 : \{(c, \langle i, j\rangle) \mid 0 \le i \le n+1,\ j = 0\}$ – extreme left column,

$G_2 : \{(c, \langle i, j\rangle) \mid i = 0,\ 0 \le j \le m+1\}$ – extreme top row,

$G_3 : \{(c, \langle i, j\rangle) \mid 0 \le i \le n+1,\ j = m+1\}$ – extreme right column,

$G_4 : \{(c, \langle i, j\rangle) \mid i = n+1,\ 0 \le j \le m+1\}$– extreme bottom row,

where $c = c_{\min}$.

Assume that the image always contains such a set U that

$$U = G_1 \cup G_2 \cup G_3 \cup G_4, \quad U \subseteq B, \tag{1}$$

i.e., the set U consists of only such pixels which have the background colour. It will be shown below that background pixels never change their colours during the processing.

Definition 1. A set $I = A \cup U$ is called *an image.*

It should be noticed, that Windows bitmap images widely used in computer graphic have the similar representation. This similarity allows to apply the proposed algorithms directly to bitmap representations.

Definition 2. A subset $O_l \subset \dot{I}$ of pixels with identical colours $c_l \in C$, i.e.,

$$O_l = \{(c, \langle i, j \rangle) \mid c = c_l\} \tag{2}$$

is called *a pattern.*

Let $\varphi(i, j)$ be a function defined on M which allows to obtain any coordinate pair $\langle i', j' \rangle$ in dependence on $\langle i, j \rangle$. Let $T(i, j)$ be any set of such functions. Let us distinguish a subset $T'(i, j)$ of functions $\varphi(i, j)$:

$$T'(i, j) = \{\varphi_k(i, j) \mid k = 1 \ldots 8\} \tag{3}$$

and subset

$$T''(i, j) = \{\varphi_k(i, j) \mid k = 1, 3, 5, 7\}, \tag{4}$$

where

$$\begin{aligned} &\varphi_1(i, j) = (i - 1, j), \quad \varphi_2(i, j) = (i - 1, j + 1), \\ &\varphi_3(i, j) = (i, j + 1), \quad \varphi_4(i, j) = (i + 1, j + 1), \\ &\varphi_5(i, j) = (i, j + 1), \quad \varphi_6(i, j) = (i + 1, j - 1), \\ &\varphi_7(i, j) = (i, j - 1), \quad \varphi_8(i, j) = (i - 1, j - 1). \end{aligned}$$

Definition 3. A set

$$N(p) = \{(c, \varphi(i, j)) \mid \varphi(i, j) \in T(i, j)\}, \tag{5}$$

where $c \in C$, is called *a neighbourhood* of a pixel. The neighbourhood is *dense* ($N'(p)$) when $T(i, j) = T'(i, j)$ and *minimal* ($N''(p)$) when $T(i, j) = T''(i, j)$.

Example of subsets $T'(i, j)$ and $T''(i, j)$ is given in Figure 2.

Definition 4. A set $K_l \subseteq O_l$ $(l = 1, \ldots, r)$ in which for any pixel $p \in K_l$ there exists at least one pixel $q \in N(p)$, such that $q \notin O_l$, i.e.,

$$K_l = \{p \mid (\exists q \in N(p)) \wedge (q \notin O_l)\} \tag{6}$$

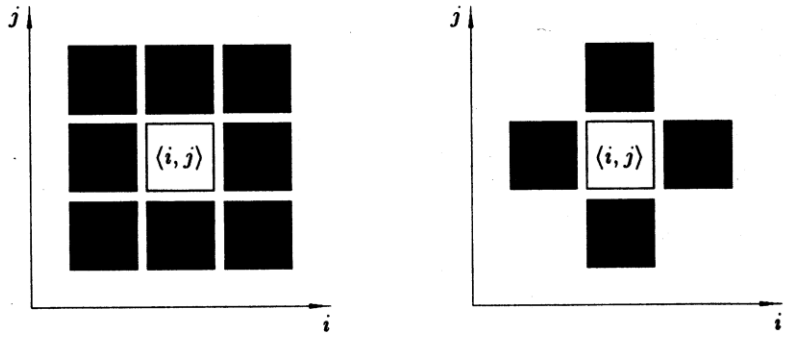


Figure 2. Subsets $T'(i,j)$ (a) and $T''(i,j)$ (b) corresponding to dense and minimal neighbourhoods of the pixel with coordinates $\langle i,j\rangle$

is called *a contour* of the pattern O_l. According to dense and minimal neighbourhoods there are *dense* K_l' and *minimal* K_l'' contours.

Remark. If the pattern consists of one pixel, the contour is that pixel itself.

It is easy to notice that any pixel deleted from the contour K_l'' makes a respective contour disconnected. That is why such contour is called "minimal". We can also write: $K_l'' \subseteq K_l'$.

It also follows from the definition of a contour that any image includes both the pixels which belong to the contour and the pixels which do not belong to it and are located inside the pattern. The latters are called *internal* pixels. A set $\bar{K}_l$ of internal pixels of a pattern O_l can be defined as follows:

$$\bar{K}_l = \{p \mid N(p) \subset O_l\}, \tag{7}$$

i.e., for any pixel p from $\bar{K}_l$ all pixels q from its neighbourhood belong to the pattern O_l.

Let us consider a certain part of the image being processed and any given pattern $O_l \subset I$. In general it contains (Figure 3):

1) a set of pixels $p = (c, \langle i,j\rangle)$, $c = lh$, which belong to a contour K_l and have at least one pixel $q = (c', \langle i,j\rangle)$, $c' = (l-1)h$ in its (dense or minimal) neighbourhood:

$$K_l^{>} = \{p \mid (\exists q \in N(p)) \wedge (c > c')\}, \tag{8}$$

where c and c' are colours of pixels p and q respectively;

2) a set of internal pixels $p = (c, \langle i,j\rangle)$, $c = lh$, for which all pixels $q = (c', \langle i,j\rangle)$ from its (dense or minimal) neighbourhood have the colour $c' = lh$:

$$\bar{K}_l = \{p \mid (\exists q \in N(p)) \wedge (c = c')\}; \tag{9}$$

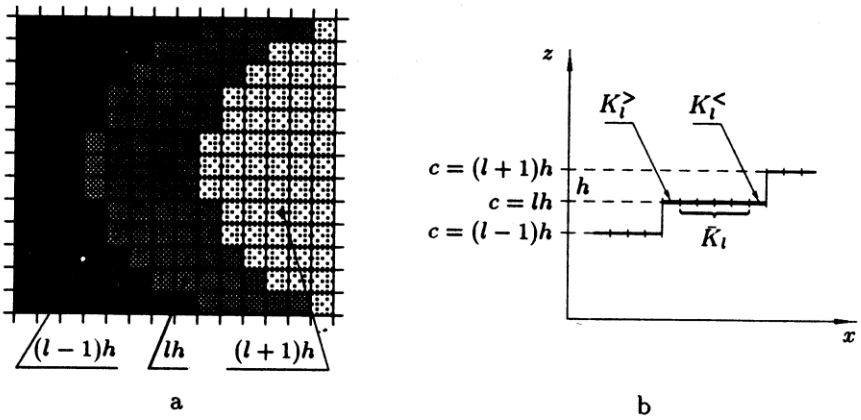


Figure 3. a) a part of image; b) a pixel sets representation for a pattern

3) a set of pixels $p = (c, \langle i, j \rangle)$, $c = lh$, which belongs to a contour K_l and have at least one pixel $q = (c', \langle i, j \rangle)$, $c' = (l+1)h$ in its (dense or minimal) neighbourhood:

$$K_l^< = \{p \mid (\exists q \in N(p)) \wedge (c < c')\}. \tag{10}$$

Obviously, the sets $K_l^>$ and $K_l^<$ are subsets of K_l. As it may be seen from Figures 1 and 3 only the pixels of sets $K_l^>$ have colours corresponding to the values of variable z in the initial function $z = F(x, y)$. Therefore it is possible to make the following

Definition 5. A set of contour pixels $K_l^>$ is called an *isoline*.

The extraction of isolines is the procedure, which transforms all pixel colours not equal to colours of isolines to the background colour and leaves isoline pixels with the initial colour values for all l. Thus, the procedure of isoline extraction is reduced to that of contours $K_l^>$ extraction. In the next section the algorithm of such image transformation will be considered.

3. Isoline extraction algorithm

Let us use the PSA [1] terms to describe the algorithm stages and define the neighbourhood function

$$S(i, j) = \{(x_k, \varphi(i, j)) \mid \varphi(i, j) \in T(i, j)\}, \tag{11}$$

which sets a neighbourhood $N(p)$ $(p = (c, \langle i, j \rangle))$ into correspondence to each pair of coordinates $\langle i, j \rangle \in M$, $T(i, j) = T'(i, j)$ or $T(i, j) = T''(i, j)$ for a dense or minimal neighbourhood respectively.

When the numeration of pixels in $N(p)$ is fixed, it is possible to consider the set of variables x_k from (11) as a vector $X^r(i,j) = (x_1, \ldots, x_r)$, and a set of their values – as a vector $C^r(i,j) = (c_{l_1}, \ldots, c_{l_r})$, where c_{l_k} is a colour of k-th pixel in the neighbourhood $N(p)$, $c_{l_k} \in C$.

Array transformation is represented by a set of parallel substitutions which have the following form:

$$\Theta : (c, \langle i,j \rangle) * S(i,j) \to (\Phi(X^r, c), \langle i,j \rangle), \tag{12}$$

where $S(i,j)$ plays the role of a context.

This parallel substitution is an elementary parallel action aimed to transform the image. The function $\Phi(X^r, c)$ transforms the colour of the pixel and will be given below.

On the base of parallel substitutions it is possible to represent the isoline extraction algorithm at each stage as the following

Procedure.

1. Parallel substitution (12) is applied to all pixels p ($p \notin U$) of the image.
2. If there are no changes of pixel colours then we have got the result, else go to 1.

In other words, each of two stages of the isoline extraction algorithm can be described as a procedure presented above.

The first stage of the isoline extraction algorithm performing secondary quantization can be described as one-time transformation of pixel colours by using the quantization function which can be written as follows:

$$d(c) = lh, \quad lh \le c < (l+1)h, \tag{13}$$

for all $(c, \langle i,j \rangle) \in O_l$.

Since $d(c)$ does not depend on the colours of neighbourhood pixels, $S(i,j)$ is empty. So, the substitution (12) should be rewritten as follows:

$$\Theta_1 : (c, \langle i,j \rangle) \to (d(c), \langle i,j \rangle). \tag{14}$$

The second stage of the algorithm (proper isoline extraction) requires to define such a function (let it be f) instead of Φ in (12) which would transform colours of all those pixels which do not belong to isolines. Let $p = (c, \langle i,j \rangle)$ be a pixel of a pattern O_l, $c \neq c_{\min}$. The condition that the pixel p belongs either to the set $\bar{K}_l$ or to the set $K_l^<$ according to (9) and (10) is

$$\bigwedge_{k=1}^{r} (c \le x_k), \tag{15}$$

where $r = 4$ or $r = 8$, $x_k \in X^r$, $\bigwedge$ is the logical "and".

In other words, a colour of the given pixel should be less than or equal to *any* colour of pixels from its dense or minimal neighbourhood. Thus, the function f looks like:

$$f(X^r, c) = \begin{cases} c_{\min}, & \bigwedge_{i=1}^{r} (c \le x_k), \\ c, & \text{otherwise.} \end{cases} \tag{16}$$

So, the function f transforms a colour of the given pixel to the background colour if this pixel does not belong to an isoline and keeps it unchanged otherwise. It is easy to notice now, that if a pixel belongs to the background, the function f does not change its colour. Using the function f, the parallel substitution (12) can be rewritten as follows:

$$\Theta_2 : (c, \langle i, j \rangle) * S(i, j) \to (f(X^r, c)\langle i, j \rangle). \tag{17}$$

Theorem. *The above procedure with substitution* (17) *being applied to any image results in the isolines extraction in one parallel iteration.*

Proof. Since any pattern O_l includes a contour $K_l^>$ by definition and the substitution (17) keeps unchanged colours of those and only those pattern pixels which belong to a contour $K_l^>$, the set $K_l^>$ can be extracted from any pattern.

Let us consider a sequence consisting of n applications of the substitution: $O_l^0 \to O_l^1 \to \ldots \to O_l^n$, where O_l^i is the result of the n-th application of substitution (17). Let us assume that before the first application the pattern O_l contains at least one pixel s, which meets condition (15) and, therefore, does not belong to the contour $K_l^>$. The next application gives a new pattern O_l^1, which does not contain this pixel s. Since f either changes a colour c for $c_{\min}$, or does not make it, the number of pixels in the pattern O_l either is decreased or is kept unchanged, i.e.,

$$|O_l^0| \ge |O_l^1| \ge |O_l^2| \ge \ldots |O_l^k| \ge \ldots, \tag{18}$$

where $|O_l^i|$ is the number of pixels in the pattern O_l after i-th application of f. Thus, after the first application of (17) the pattern O_l^1 contains no pixels which do not belong to the contour $K_l^>$ i.e., $K_l^> = O_l$. Obviously, if all pixels of the pattern O_l are contour pixels, the substitution (17) does not change this pattern by any next application of the function.

Since substitution is applied to each pixel in the entire image and at the same time, similar assertions are valid for any number of pixels s (meeting to condition (15)) in the pattern O_l. Thus the function f allows to extract the contour $K_l^>$ and the corresponding isoline, by its one-time application.

□

To summarize, it is possible to present

Isoline Extraction Algorithm.

1. One-time application of parallel substitution Θ_1 (14) using the quantizing function d.
2. One-time application of parallel substitution Θ_2 (17) using the isoline extraction function f.

So, the application of the isoline extraction algorithm consists of two iterations in which the first one quantizes initial image according to the function d and the second one changes all pixel colours by application of the function f. Besides, the algorithm allows to extract isolines out of any image having peculiarities described above.

4. Simulation of the algorithm in the ALT system

The examination of obtained theoretical results has been performed with the substitutions of the simulation system ALT [2]. The ALT (Animated Language Tools) system is intended for development and debugging of microlevel parallel algorithms (fine-grained parallel algorithms) on the base of special object-oriented technology of combined textual and graphical forms of parallel program representation and simulation of parallel computations in the single-processor operating system MS-DOS with ability of dynamical estimation of their performance. The ALT system permits to realize any PSA. The graphical representation form allows to manipulate immediately with a picture of a cellular array under processing and with templates (like template images presented in Figure 2) used in the substitutions. The text form allows to present functions by a subset of C programming language operators. The set of such operators defines the functional implementation of a model and is realized as a separate text module.

In the ALT system there is a special high-level language for compact representation of computation distribution in time and space. The program written in this language and presented as a separate text module allows to observe a sequence of the parallel substitutions execution immediately in a program running.

Thus, a parallel computation model in the ALT system consists of a parallel program, functional implementation and graphical representation of the computational environment.

A text of the program of isoline extraction looks like this:

```
;the secondary quantization stage
ch                ; make one transformation
  in I            ; in array I
    ab N          ; in dependance on the neighbourhood N
      do discr    ; by executing of a function "discr"
;the isolines (contours) extraction stage
ch                ; make one transformation
  in I            ; in array I
    ab N          ; in dependance on the neighbourhood N
      do isoline; by executing of a function "isoline"
```

Comments. Here the variable I denotes the array representing the image being processed. The variable N denotes the neighbourhood of the pixel affecting on its colour (in the quantization stage it is reduced to the pixel itself).

Functions. For the secondary quantization step the function "discr" in the ALT system looks like this:

```
discr()
{
  (c>0?) c=(c/h)*h+1: ;
}
```

This function assigns the colour $l \times h$ to all pixels which belong to an interval from $l \times h$ to $(l+1)h$ (where l is integer value) by using the cutting of the fraction part when operations with integer numbers are applied. The value h defines the step of quantization. Addition of 1 is required to save the pattern outline when it is assumed that the pattern outline has a colour coded by 1.

For extraction of isolines as dense contours the function "isoline" is the following:

```
isoline()
{
  (c>0? && (x1>=c && x2>=c && x3>=c && x4>=c
            x5>=c && x6>=c && x7>=c && x8>=c)
  ) c=0: ;
}
```

For extraction of isolines as minimal contours:

```
isoline()
{
  (c>0? && (x1>=c && x2>=c && x3>=c && x4>=c
  ) c=0: ;
}
```

Comments. The condition $c > 0$ allows to avoid processing of the neighbourhood pixel colours for those pixels which have a null-value colour (in assumption that the background colour is coded by 0). This condition ensues from the above conclusion that only those pixel colours can be changed which have non-zero value. This condition permits to decrease the image processing time.

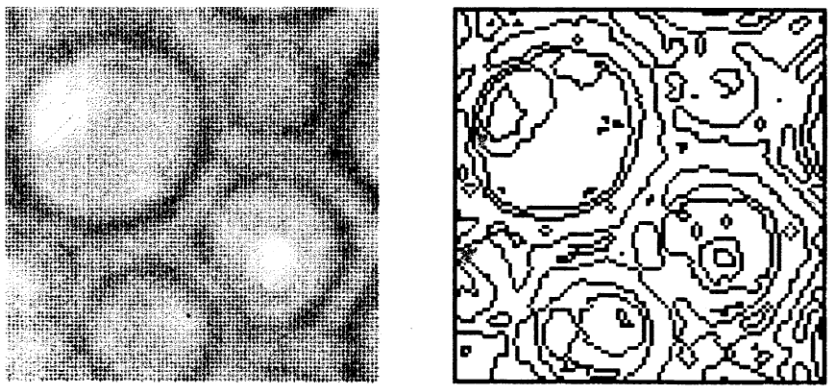
In the construction

$$\langle \text{Condition} \rangle \ ? \ \langle \text{Expression1} \rangle : \langle \text{Expression2} \rangle$$

the absence of ⟨Expression2⟩ means that no actions are performed with the value c, because it must be kept unchanged when ⟨Condition⟩ is false. It also permits to decrease the image processing time in realization of the algorithm.

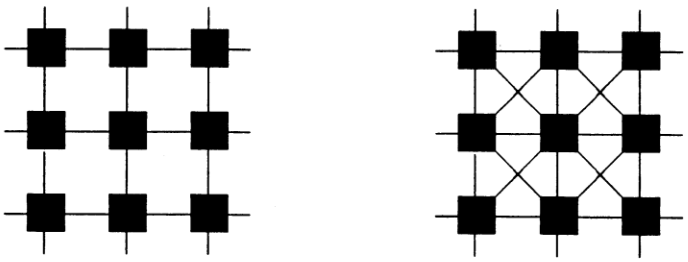
Example of surface image processing is given in Figure 4.

The presented algorithm allows to determine the architecture of a special-purpose device for extraction of isolines. This device should have a cellular



a b

Figure 4. a) initial surface image; b) the result of application of isoline extraction algorithm (colours are inverted for better viewing)



a b

Figure 5. The intercell connection topology: a) for extraction of minimal isolines, b) for extraction of dense isolines

structure in which all cells perform the described functions d and f, and the intercell connection topology is determined by the neighbourhood used. Intercell connection topologies for extraction of minimal and dense isolines are given in Figure 5.

5. Conclusion

The proposed approach to isoline extraction follows the general tendency towards parallel information processing. Therefore, main advantages of the approach are parallelism, relative realization simplicity and two-step processing. The above algorithm may be used both for processing of images given in the bitmap format, and for VLSI chip design to be implemented as a special-purpose device.

References

[1] S. Achasova, O. Bandman, V. Markova, S. Piskunov, *Parallel Substitution Algorithm: Theory and Application*, World Scientific Publishing, Singapore, 1994.

[2] Yu. Pogudin, *Simulation of fine-grained parallel algorithms with the ALT (Animated Language Tools) system*, Proceedings of First International Workshop on Distributed Interactive Simulation and Real Time Applications, January 9–10, Eilat, Israel, 1997.