An Interesting Modification to the Bresenham Algorithm for Hidden-Line Solution

Václav Skala

Department of Technical Cybernetics
Technical University, Nejedlého sady 14
306 14 PLZEŇ, CZECHOSLOVAKIA

1. Introduction

The solution of many engineering problems have as a result functions of two variables, that can be given either by an explicit function description, or by a table of the function values. The functions, that can be given either by an explicit function description, or by a table of the function values. The functions have been usually plotted with respect to visibility. The subprograms for plotting the functions of two variables were not so simple ([6]-[7]) although visibility may be achieved by the relatively simple algorithm at the physical level of the drawing, if we assume raster graphics devices are used. The Bresenham algorithm for drawing line segments can be modified in order to enable the drawing of explicit functions of two variables with respect to the visibility.

Though the order of curve drawing is essential for the method used the algorithm has not been published yet. Williamson [7] solved the problem by fixing the position of the view-point, Watkins [6] only pointed out that some rotation angles can cause wrong hidden-line elimination and Boutland's method [1] can use only one angle.

Therefore the algorithm that ensure the right order of the curve's drawing is presented here.

2. Problem specification

Let us have an explicit function of two variables x and y

$$z = f(x, y)$$

where: $x \in \langle ax, bx \rangle$ and $y \in \langle ay, by \rangle$

and we want to display that function by using the graphical raster display or plotter. For many scientific problems is enough to show the behaviour of that function by drawing the function slices according to the x and y axises, e.g. curves

$$z = f(x, y_i) \quad i=1,\ldots,n$$

NATO ASI Series, Vol. F17
Fundamental Algorithms for Computer Graphics
Edited by R. A. Earnshaw

where: $x \in \langle ax, bx \rangle$ and $ay = y_1 < y_2 < \cdots < y_n = by$
and curves:

$$z = f(x_j, y) \qquad j = 1, \ldots, m$$

where: $y \in \langle ay, by \rangle$ and $ax = x_1 < x_2 < \cdots < x_m = bx$

The given function can be represented either by a function specification or by a table of the function values for the grid points in the x-y plane. If the function is complex it can be very difficult to imagine the function behaviour because some parts are in the reality invisible. The problem has been solved by Watkins [6], Williamson [7] and Boutland [1] relatively very successfully. The principle of the solution is generally very simple. If we have drawn the first two slices parallel to the x axis we have produced two curves and the space between them is the strip of invisibility. Let us suppose that we draw the lines in the direction from foreground to background. Now if we want to draw the third curve it is obvious that those parts which are passing through the strip of invisibility are invisible and therefore ought not to be drawn, see figure 1.

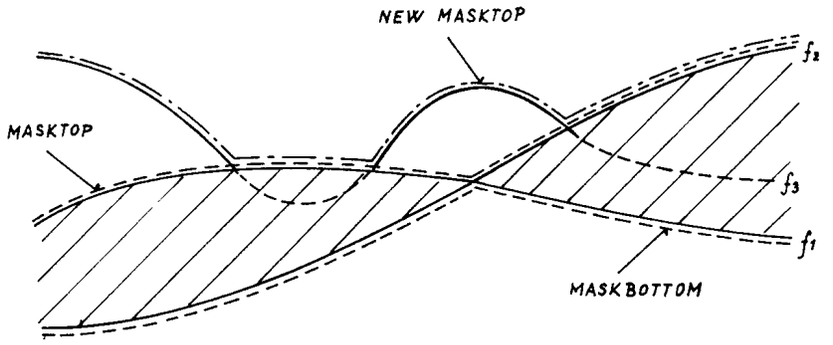
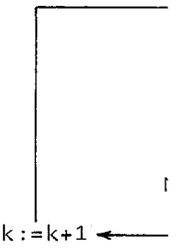


Figure 1.

If we analyze the problem in detail we will realize that we need to represent the borders of the strip of invisibility. It can be done by the MASKTOP and MASKBOTTOM functions. The real representation of the MASKTOP and MASKBOTTOM functions we will omit temporarily. Now the problem of drawing curves with respect to the visibility becomes simple, see algorithm 1., because we will draw the next function slice only if and only if the curve points are outside of the strip of invisibility.

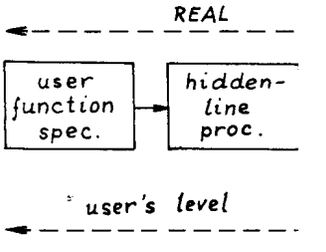
The visibility problem has been solved by Watkins [6] by introducing mask vectors for the representation of the MASKTOP and MASKBOTTOM bounds. Several problems had to be solved because all computation was done in the floating point representation:

- the first problem :
  or MASK[i+1] if th
  e.g. $i < x < i+1$
- the second problem
  have to be set up
  an interpolation p
  ble interpolation
- the third problem
  the curve is paral
  slope computation



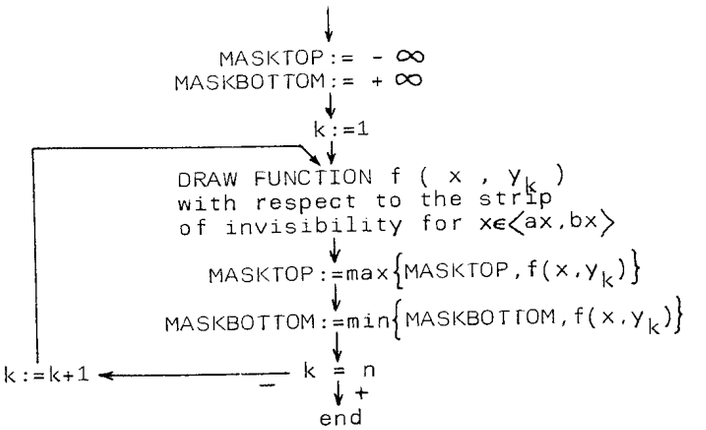
3. Proposed method

In [6] the functio
vectors with values i
gine the whole proces



Now we can ask our
ing the efficiency of

- the first problem is how to decide if we have set up MASK[i] or MASK[i+1] if the coordinate x is between values i and i+1 e.g. $i < x < i+1$
- the second problem is that the MASKTOP and MASKBOTTOM arrays have to be set up for all point of the curve. That means that an interpolation procedure has to be employed, with some suitable interpolation step length.
- the third problem is that special case has to be solved: when the curve is parallel with the z axis, the usual line segment slope computation can fail.



Algorithm 1.

3. Proposed method

In [6] the functions MASKTOP and MASKBOTTOM are represented by vectors with values in floating point representation. We can imagine the whole proces of hidden-line drawing as follows in figure 2.

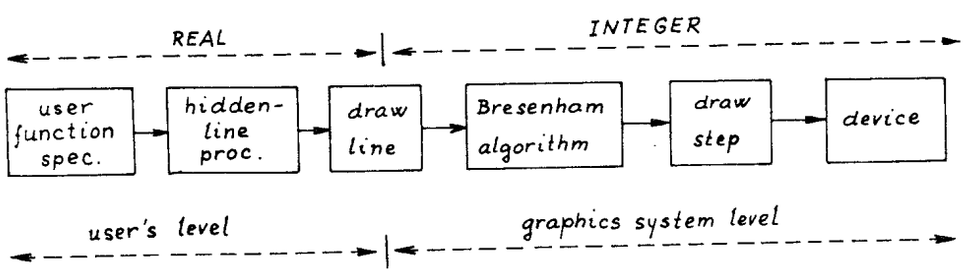


Figure 2.

Now we can ask ourselves if there is any possibility of increasing the efficiency of the hidden-line solution. One possibility is

to combine the complete Watkin s algorithm with the Bresenham algorithm directly at the physical level. Because we are dealing with the raster devices at the physical level we have got rid of all these above mentioned problems.

The solution of the hidden-line problem is now relatively very simple, becausy we have to change only the procedure DRAW-STEP, that generates code for the physical movement, in order to take account of the strip of invisibility. Because DRAW-STEP draws only one step we have to check only if the next end-point in the raster is inside of the strip of invisibility or not. The structure of the proposed method is shown on figure 3.

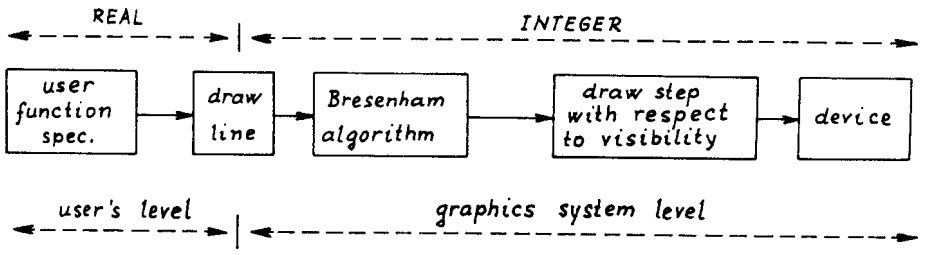


Figure 3.

It is obvious that we need only integer representation for the MASKTOP and MASKBOTTOM masking arrays. The simplified solution is shown by the algorithm 2.

It was find out that the lines (on the physical level) which are parallel to y axis cause some problems with setting of the masks arrays, see figure 4. Suppose that we have defined the strip of invisibility and we want to draw the line segment x_1x_2. The problem is that if we want to draw the segment between the points 1 and 2 we have to change the strip of invisibility so the future points 3 and 4 become inner points in the strip of invisibility; but that is not true. Therefore in the complete algorithm the content of the mask´s arrays is changed only if dx<>0. The whole algorithm can be found in [4], where the clipping is realized too.

Watkin´s original method and proposed solution have one common problem, that has not been published yet. Because of rotation sometimes the foreground and background can be altered and the order in which the curves are drawn cause a violation of the masking premises. The second problem is how to select the scales for scaling in order not to lose any part of the picture and use the full screen area. The first problem seems to be more complicated and it is more fundamental. The proposed solution is presented bellow. The second

```
problem can be s
for the screen c

{ GLOBAL VARIABL
VAR xO,yO: REAL;
    masktop,mask
PROCEDURE draw (
VAR flag5: BOOLE
BEGIN xO:=xO+dx;
      flag5:=FAL
      IF masktop
      BEGIN flag
      IF maskbot
      BEGIN flag
      IF flag5 T
                E
END;

PROCEDURE bresen
VAR j,d,a,b: INT
BEGIN a:=v+v; d:
      FOR j:=1 T
          IF d<

END;
```

problem can be solved easily by finding maximal and minimal values for the screen coordinates.

```
{ GLOBAL VARIABLES }
VAR xO,yO: REAL;
    masktop,maskbottom: ARRAY [0..1024] OF INTEGER;
PROCEDURE draw ( dx,dy: INTEGER );
VAR flag5: BOOLEAN;
BEGIN xO:=xO+dx; yO:=yO+dy;
      flag5:=FALSE;
      IF masktop[xO] <= yO THEN
      BEGIN flag5:=TRUE; masktop[xO] :=yO; END;
      IF maskbottom[xO] >= yO THEN
      BEGIN flag5:=TRUE; maskbottom[xO] :=yO; END;
      IF flag5 THEN physline(dx,dy)
               ELSE physmove(dx,dy)
END;

PROCEDURE bresenham (u,v: INTEGER );
VAR j,d,a,b: INTEGER;
BEGIN a:=v+v; d:=a-u; b:=a-u-u;
      FOR j:=1 TO u DO
          IF d<0 THEN BEGIN draw(1,0); d:=d+a; END
                 ELSE BEGIN draw(1,1); d:=d+b; END
END;
```

Algorithm 2.

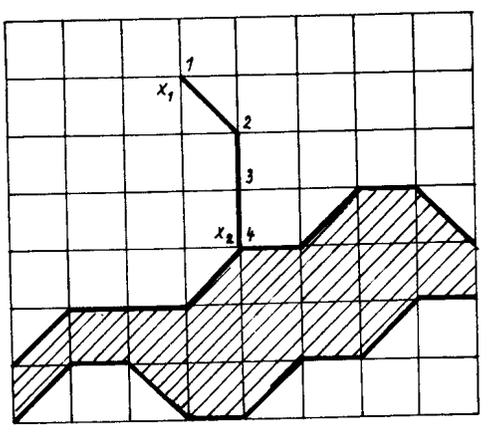


Figure 4.

4.Design of the drawing order

If we rotate the function or have a look at the function from different points, we have to keep the basic rule of the drawing. We have to draw at first the function slices that are nearer to us. Watkins [6] pointed out this problem, but the problem solution has not been published yet and many users have real difficulties to ensure that. Therefore when the function is rotated many pictures are drawn wrong. Let us try to find the solution.

Assume that the points:

$$X_1=(ax,ay,0) \qquad X_3=(bx,by,0)$$
$$X_2=(bx,ay,0) \qquad X_4=(ax,by,0)$$

are the corner-points of the grid in the x-y plane. We want to know the order of the drawing of the drawing slices.

Assumes that the points:

$$X_1' = T(X_1) \qquad X_3' = T(X_3)$$
$$X_2' = T(X_2) \qquad X_4' = T(X_4)$$

are the corner points of the grid after the rotation transformation. Now we have to pick up two margines from which we will start to draw the picture. We have to select the end-points of these margines that has the smallest z' coordinate. We will mark that point by the index r. In general there are two basic possibilities that are shown on figure 5.

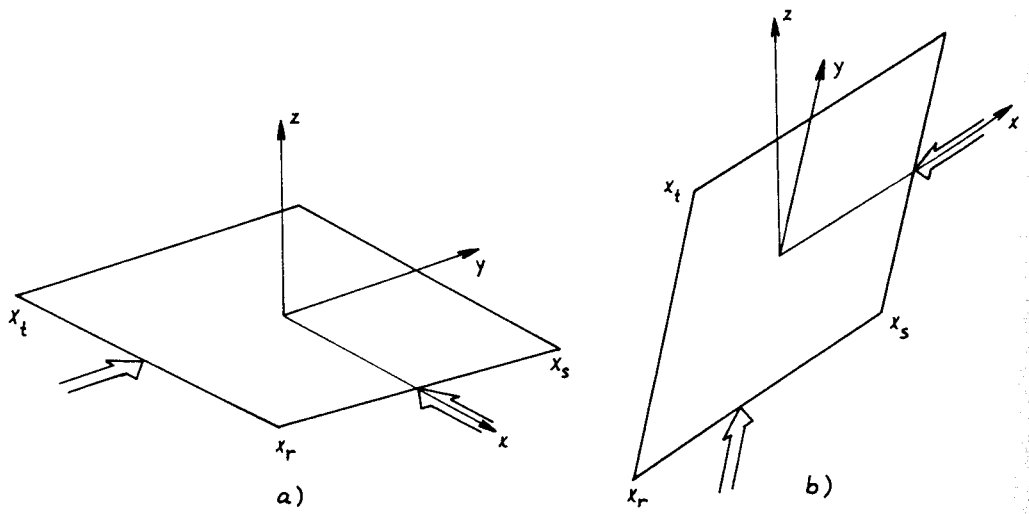


Figure 5.

The direction in which the slices are to be drawn are marked by ⇐.

In the case ac
start to draw lir
In the case ad b
to test if

$$x_t' <=$$

If the boolean e
second point whic
from the origina
index r.

The whole proc

1. Find the inde

$$z_r' = r$$

2. Find the indic
3. If condition

$$x_t' <=$$

has value FALS
begin
Find in
$z_$
r := u
Find th
end

Now we can dr
the selected mar
end-points X_rX_t
But if we dra
we receive a pic
more convenient
and we will then
The Zig-zag m

1. Initialize th
2. Draw the marg
$X_r'X_t'$ (steps
3. Draw the func
to the direct

In the case ad a) we can see that the margins from which we will start to draw line segments belongs to the end-points $X_r X_s$ and $X_t X_r$. In the case ad b) we can see that we will fail. Therefore we have to test if

$$x'_t <= x'_r <= x'_s \quad \text{or} \quad x'_s <= x'_r <= x'_t$$

If the boolean expression has value false then we have to find the second point which has minimal z' coordinate and which is different from the original point. The new point will be remarked by the index r.

The whole procedure can be described by the algorithm 3.

1. Find the index $r \in \langle 1,4 \rangle$ so that
   $$z'_r = \min \{ z'_i \} \quad i=1,\ldots,4$$
2. Find the indices of neighbours and mark them by indices t,s
3. If condition
   $$x'_t <= x'_r <= x'_s \quad \text{OR} \quad x'_s <= x'_r <= x'_t$$
   has value FALSE then
   begin
   Find index $u \in \langle 1,4 \rangle$ so that
   $$z'_u = \min \{ z'_i \} \quad i=1,\ldots,4 \text{ and } i \neq r$$
   r := u
   Find the indices of neighbours and mark them by indices t,s
   end

Algorithm 3.

Now we can draw the function by drawing the slices according to the selected margins, which are defined by line segments with the end-points $X_r X_t$ and $X_r X_s$.

But if we draw a function whose behaviour is wild enough then we receive a picture which is wrong, see figure 6 It seems to be more convenient in this situation to apply the Zig-zag method 1 and we will then obtain the correct results, see figure 7

The Zig-zag method can be described by:

1. Initialize the mask's arrays
2. Draw the margins that are defined by the end-points $X'_r X'_s$ and $X'_r X'_t$ (steps 1,2)
3. Draw the function values according to the grid and according to the directions on figure 8 (steps 3-12)

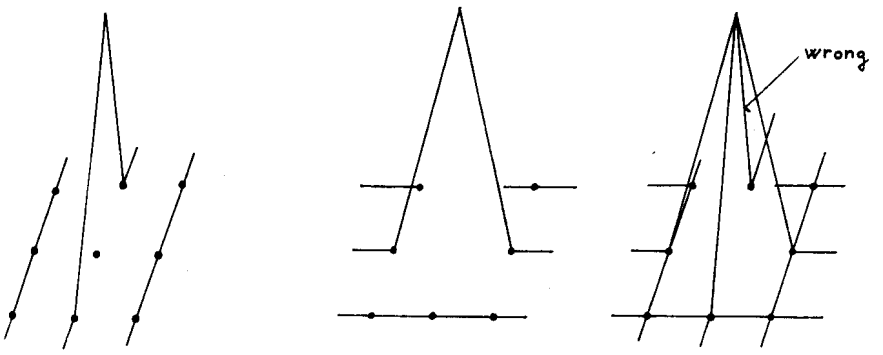


Figure 6.

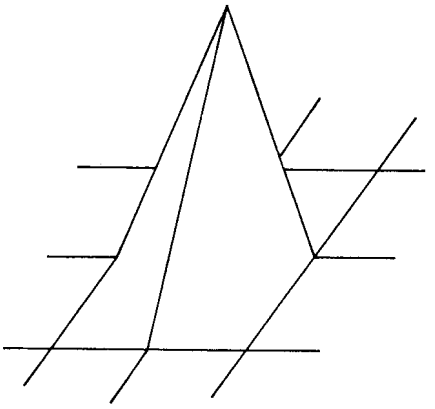


Figure 7.

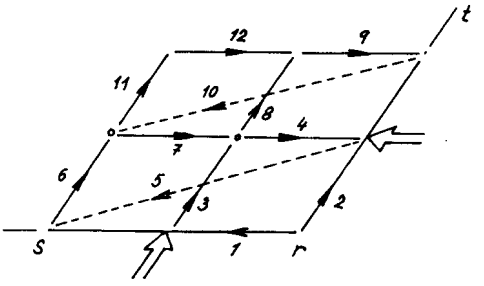


Figure 8.

Let us suppose that the function is given by the values

$f[i,j]$ $i=1,\ldots,n$ and $j=1,\ldots,m$

in the grid - points those coordinates are given by values

$x[j]$ $j=1,\ldots,m$

$y[i]$ $i=1,\ldots,n$

Then after transformation (rotation, translation) we receive values

$x'[j]$, $y'[i]$, $f'[i,j]$ $i=1,\ldots,n$ and $j=1,\ldots,m$

Now the whole process of drawing can be made in the integer representation without using a floating point processor.

5.Conclusion

The algorithm
with respect to v
puters. Because t
be realized by ab
venient to build
straight lines. N
be extended by th

- initialize mas
- draw line with

that means, that
and significantly
into the algorith
with grey scale :
using algorithm [

We can ask our
drawing lines wh:
of any basic grap

6.Acknowledgemen

I would like
for their many h
to finish this p

7.Literature

[1] Boutland J.
Design 11(1
[2] Bresenham J
Plotter, IB
[3] Pitteway M.
Scale, Comm
[4] Skala V.: H
Dept., Brun
[5] Sowerbutts
Microcomput
pp.324-327
[6] Watkins S.L
rotation, C
[7] Williamson
15(2), Febr

5.Conclusion

The algorithm presented for drawing functions of two variables with respect to visibility is intended for the use with microcomputers. Because the basic algorithm for visibility respectation can be realized by about ten assembly instructions it seems to be convenient to build it directly into the algorithm for a drawing straight lines. Now we can see that the basic graphics menu can be extended by the operations:

- initialize mask´s arrays
- draw line with respect to the visibility,

that means, that the intelligence of graphic devices can be easily and significantly improved by adding several assembly instructions into the algorithm for the drawing lines. If the graphics display with grey scale is used the algorithm can be easily improved by using algorithm [3] for drawing straight lines.

We can ask ourselves whether primitive-level instruction for drawing lines which takes account of visibility, should be a part of any basic graphics software system, e.g. GKS.

6.Acknowledgement

I would like to thank Prof. L.M.V. Pitteway and Dr.J.P.A. Race for their many helpful discussions and suggestions that enabled me to finish this project succesfully.

7.Literature

[1] Boutland J.: Surface Drawing Made Simple, Computer Aided Design 11(1) January 1979, pp.19-22

[2] Bresenham J.E.: Algorithm for Computer Control of Digital Plotter, IBM Syst. J. 4(1) 1965, pp.25-30

[3] Pitteway M.,Watkinson D.: Bresenham´s Algorithm with Grey Scale, Comm. of ACM 23(11), November 1980, pp.625-626

[4] Skala V.: Hidden-Line Processor, CSTR/29, Computer Science Dept., Brunel University, Uxbridge, Middlesex, 1984

[5] Sowerbutts W.T.: A Surface-Plotting Program Suitable for Microcomputers, Computer Aided Design 15(6), November 1983, pp.324-327

[6] Watkins S.L.: Masked Three-Dimensional Plot Program with rotation, Comm. of ACM 17(9), September 1974, pp.520-523

[7] Williamson H.: Hidden-Line Plotting Program, Comm of ACM 15(2), February 1972, pp.100-103