Mezinárodní konference Algoritmy 89, JSMaF při SAV,
April 17-21, 1989, Vysoké Tatry, Štrbské Pleso (Algorithms 89)

Hidden-Line, Hidden-Surface and Hidden-Contouring Problem Solutions by the Modified Bresenham's Algorithm

Vaclav Skala
Computer Science Department, Technical University, Nejedleho sady 14, 30614 Plzen

1. Introduction

Solutions of many engineering problems result in functions of two variables which can be given either by an explicit function description or by a table of function values. Functions have usually been displayed on a display screen or have been drawn on a plotter in several manners. The known methods have been based either on drawing contours drawn in axonometric projection [3] or on drawing functions of two variables with respect to visibility ([12]-[14]). Each method of displaying must have its own specialised algorithms, which has not been simple so far. In all cases the problem of displaying has been complicated not only by an enormous volume of data but also from the point of view of programs and their structure. Many algorithms have been published in literature but they differ from each other.

In the following paragraphs a unifying approach to the Hidden-Line, Hidden-Surface and Hidden-Contouring problems will be shown. It is based on a modification of the Bresenham's algorithm for the straight line drawing. The advantage of the suggested solution is a simple hardware implementation, especially with devices controlled by microprocessors. The proposed algorithms are fast and they do not use floating-point operations at all.

2. Hidden-Line Problem Solution

Let us have an explicit function of two variables x and z $\quad y = f(x, z)$
where: $x \in \langle a_x, b_x \rangle$ and $z \in \langle a_z, b_z \rangle$
and we want to display this function by using a raster device, e.g. a raster display or a plotter. For many scientific problems it is enough to show the behaviour of the given function by drawing function slices according to the x and z axes, e.g. the curves:

$$y = f(x, z_i) \qquad i = 1, \dots, n$$

where: $x \in \langle a_x, b_x \rangle$ and $a_z = z_1 < z_2 < \dots < z_n = b_z$
and the curves:

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

where: $z \in \langle a_z, b_z \rangle$ and $a_x = x_1 < x_2 < \dots < x_m = b_x$

The given function can be represented either by an explicit function specification or by a table of the function values for grid points in the x-z plane. If the function is rather complex it can be very difficult to imagine the behaviour of the function because some parts are invisible. The problem has been very successfully solved by ([6],[12]-[14]).Sometimes the method is called the Floating Horizont Method.

The principle of the known solution is generally simple. If two slices parallel to the x-axis are drawn then the space between these two slices is a strip of invisibility. The strip is actually part of a surface of the given function. Let us suppose that curves are drawn from the foreground to the background. Now if the third curve should be drawn (far from the observer) it is obvious that those parts that pass through the strip of invisibility are invisible and therefore they should not be drawn, see fig.2.1.

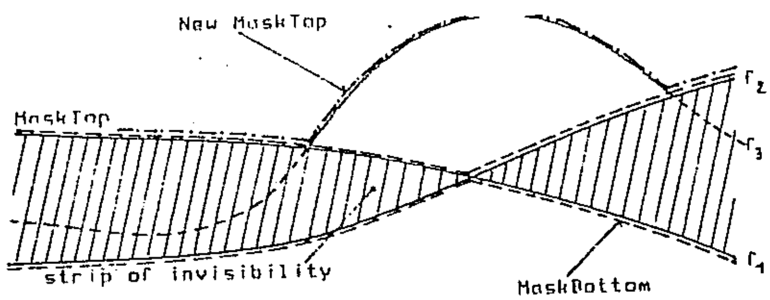


Figure 2.1.

Further the following abbreviations will be used:

MaskTop	MT	MaskBottom	MB

If the problem is analyzed in detail, it is obvious that there is a necessity to represent borders of the strip of invisibility. It can be made by introducing MaskTop and MaskBottom functions. The real representation of these functions can be temporarily omitted. Now, the problem of drawing the curves with respect to visibility becomes very simple as it is shown in algorithm 2.1., because only those parts of the curves, the points of which are lying outside the strip of invisibility, are drawn. In [6] the problem is straightly solved and the description of the method is easy to understand.

start

MT := $-\infty$
MB := $+\infty$

k := 1

DRAW FUNCTION f (x , z_k) with respect to the visibility for x∈<ax,bx>

MT:=max (MT , f (x , z)) for all x
MB:=min (MB , f (x , z)) for all x

k = n — (−) → k = k + 1

(+)

end

Algorithm 2.1.

where: n is the number of slices of the function f
min, max are functions whose two arguments are functions and the results are functions too

The visibility problem has been solved by Watkins [12] introducing mask vectors for MaskTop and MaskBottom boundary representations. Several problems had to be solved because all computations were done in the floating point representation.

3. Proposed Method for Hidden-Line Solution

In [11] the functions MaskTop and MaskBottom are represented by vectors with values in the floating point representation. The whole process of drawing with respect to visibility can be imagined as follows in fig.3.1.

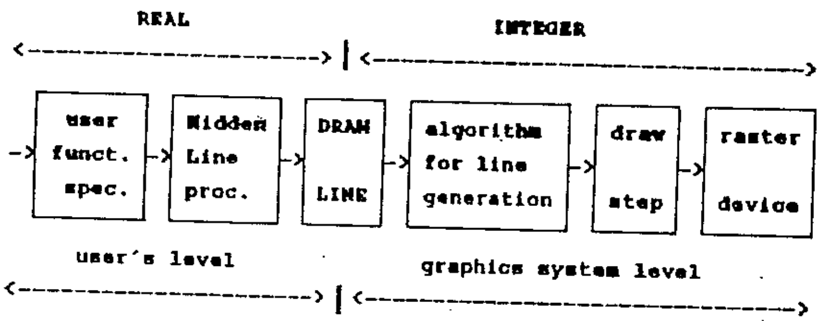


Figure 3.1.

Now a question might arise if there are any possibilities of increasing the efficiency of the hidden-line problem solution. One possibility is to combine the known Watkins's algorithm with the Bresenham's algorithm for drawing straight lines direct on the physical level.

A solution of the hidden-line problem becomes relatively very simple now because only the DRAW STEP procedure must be changed. This procedure must generate code for physical movement in order to take the invisibility into account. Because the DRAW STEP procedure draws one physical step it must only be checked whether the next end-point in the raster is inside or outside of the strip of invisibility. The structure of the proposed method is shown in fig.3.2.

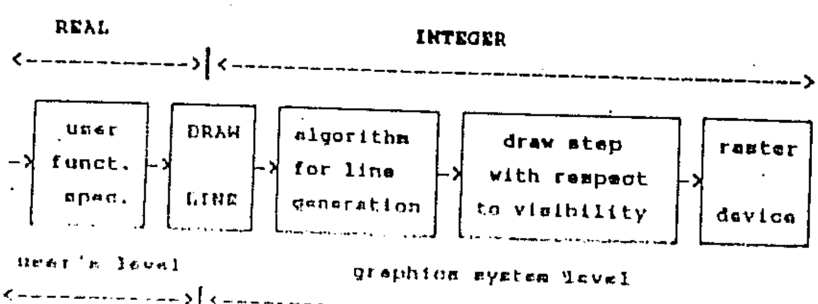


Figure 3.2.

It is obvious that only integer representation for MaskTop and MaskBottom arrays are acually needed. The simplified solution is shown in algorithm 3.1. There are two temporary arrays M1T and M1B for new masks in order to get rid of all very special problems with near horizontal lines. Spaces were included into identifiers for legibility and in the actual implementation will be omitted.

```
{ Global variables }
CONST res=1023;
VAR xp,yp:INTEGER; { absolute coordinates }
    mt,mb:ARRAY [0..res] OF INTEGER;{ MaskTop & MaskBottom }
    mlt,mlb:ARRAY [0..res] OF INTEGER; { temporary mt & mb }
    u,v,ix,iy:INTEGER;
{ Procedure for the first initialization of masks }
PROCEDURE initialize;
VAR i: INTEGER;
BEGIN FOR i:=0 TO res DO
      BEGIN mlt[i]:=-maxint;mlb[i]:=maxint END
END;
{ Procedure for masks setting after each slice has }
{ been drawn }
PROCEDURE set up masks;
BEGIN mt:=mlt;mb:=mlb END;
{ Draw step with respect to visibility of the given point }
PROCEDURE draw step;
BEGIN IF yp < mb[xp] THEN switch on (xp,yp);
      IF yp > mt[xp] THEN switch on (xp,yp);
      IF yp > mlt[xp] THEN mlt[xp]:=yp;
      IF yp < mlb[xp] THEN mlb[xp]:=yp
END;
{ Procedure for drawing straight line 0<=abs(v)<=abs(u) }
PROCEDURE bresenham;
VAR j,d,a,b: INTEGER;
BEGIN IF u >= v THEN
      BEGIN { 1.,4.,5.,8. octant }
            a:=v+v;d:=a-u;b:=a-u-u;
            FOR j:=1 TO u DO
            BEGIN IF d < 0 THEN d:=d+a
                           ELSE BEGIN yp:=yp+iy;d:=d+b END;
                  xp:=xp+ix;
                  draw step
            END
      END
      ELSE
      BEGIN { 2.,3.,6.,7. octant }
            a:=u+u;d:=a-v;b:=a-v-v;
            FOR j:=1 TO v DO
            BEGIN IF d < 0 THEN d:=d+a
                           ELSE BEGIN xp:=xp+ix;d:=d+b END;
                  yp:=yp+iy;
                  draw step
            END
      END
END;
PROCEDURE draw to ( x,y: INTEGER );
{ draw line with respect to visibility (xp,yp)->(x,y) }
BEGIN ix:=sign (x-xp);iy:=sign(y-yp);
```

```
      u:=abs(x-xp);    v:=abs(y-yp);
      bresenham
END;
BEGIN ( now the drawing itself - body of the main program )
      initialize;
9:    set up masks; ( masks must be set up for each slice )
      .....
      draw to (x,y); ( draw the function slice )
      .....
      goto 9;
      .....
END.
```

Algorithm 3.1.

In the above presented algorithm it is assumed that the order of the drawing is from the foreground to the background. Sometimes the foreground and the background are altered in the published algorithms, e.g. in [12], and the results are wrong, because the order in which the curves are drawn cause a violation of the given premises. Therefore the problem is how to select the order of the drawing if some transformations are made. It is also assumed that we use only paralel projection and that vertical lines remain vertical after all transformations too.

4. Hidden-Surface Problem Specification

The problem of the hidden-surface elimination is very often solved in a quite different ways in which many tricks are employed. Because all the known methods (for drawing functions of two variables) do not use the advantage of the solution in the raster environment a very simple algorithm will be described. The problem of the hidden-surface elimination will be solved in a very similar manner to the hidden-line elimination shown above. The slices will be drawn again from the foreground to the background and after drawing the first two slices two masks for borders and two masks for intensity levels will be set up, see fig.4.1.

Further the following abbreviations will be used:

Mask Current	MC	Intensity Current	IC
Mask Previous	MP	Intensity Previous	IP
Mask Top	MT	Mask Bottom	MB

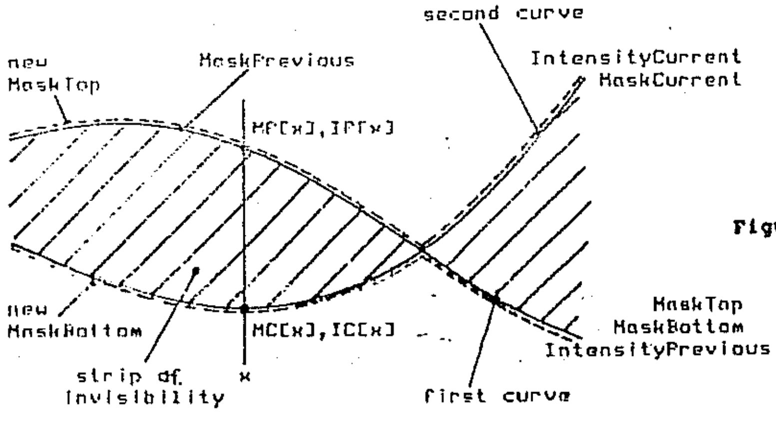
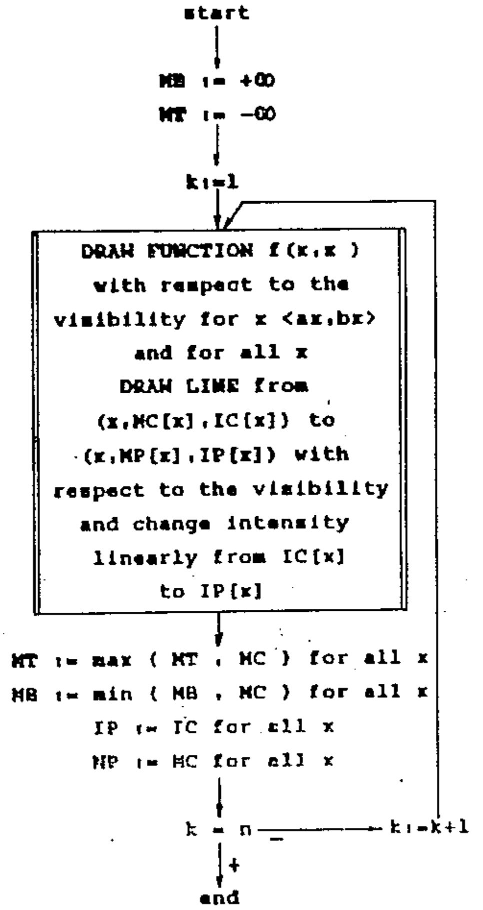
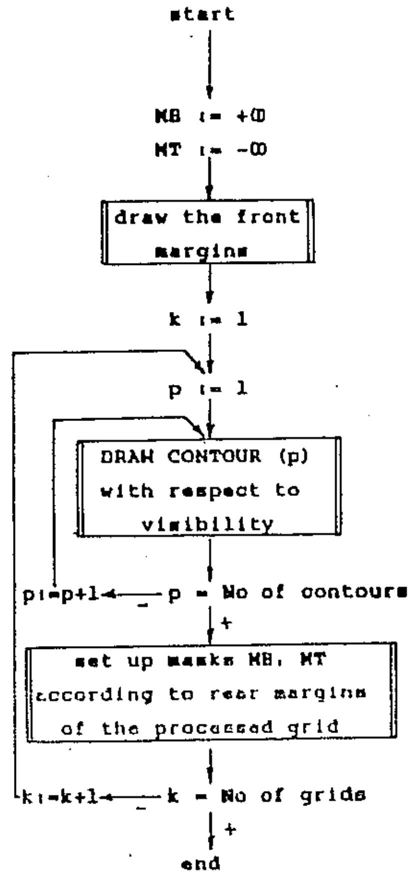


Figure 4.1.

The problem is that the invisible parts of the surface must be deleted and an appropriate visible part of the surface must be filled by an appropriate grey intensity level. To do this for all points of each curve an intensity level is needed. The intensity level can be computed as a function of the normal vector of the given surface. In the given coordinate system the z part of the normal vector must be taken, for details see [6].



Algorithm 4.1.



Algorithm 5.1.

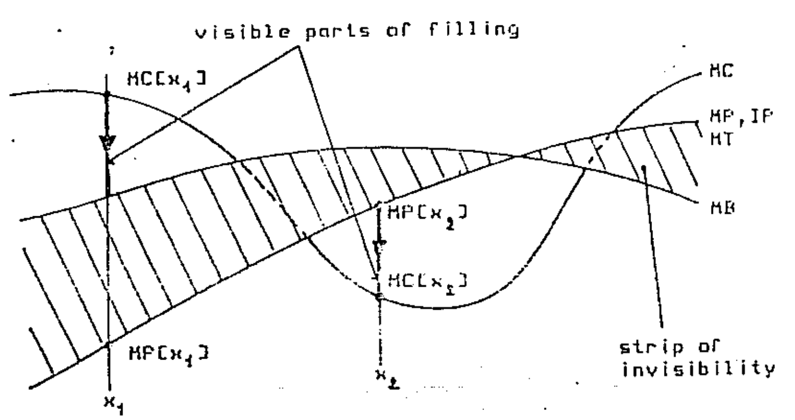


Figure 4.2.

It means that after drawing the first slice the functions MaskTop, MaskBottom (Maskbottom is equal to the MaskTop after the first slice was drawn), MaskPrevious and IntensityPrevious are known. If the second slice is drawn then the functions MaskCurrent and IntesityCurrent are known too. Now it is necessary to fill the known part of the surface that is defined by the functions MaskPrevious and MaskCurrent with an appropriate level of grey. Because all functions will be represented again by vectors for all points in the x-axis a modified Bresenham's algorithm can be employed in order to get proper intensity levels for all points. It means that for the given x the intensity level will be approximated for all possible y, e.g. a line will be drawn from the point (x,MC[x]) to the point (x,MP[x]) and the intensity level will be slowly changing from IC[x] value to IP[x] value. This must be performed for all x. In this way we will get the strip of invisibility. After that MaskTop, MaskBottom are redefined again, see fig.4.2.

If a third curve is drawn then it will be drawn only outside the strip of invisibility. But for the hidden surface it is necessary to fill the area outside the strip of invisibility between the previous curve represented by MP and the current curve represented by MC, e.g. it is necessary to fill in only a visible part of the line segment from the point (x,MC[x]) to the point (x,MP[x]) with a proper intensity level from intensity IC[x] to intensity IP[x]. The filling can be done by a modified Bresenham's algorithm again in order to ensure that the intensity level will be properly changed. It can be seen that the proposed algorithm does need all the above mentioned vectors. The whole algorithm (schematically) is shown by algorithm 4.1.

If we omit for this moment all problems with the initilization then we can use all the procedures shown above but the procedures DRAW STEP and SET UP MASKS must be changed as it is shown in algorithm 4.2. The actual implementation is slightly more complex.

```
PROCEDURE draw step;
BEGIN { draw step to the point (xp,yp) with intensity i}
      flag:=x0=xp; xo:=xp;
      IF yp > mlt[xp] THEN mlt[xp]:=yp;
      IF yp < mlb[xp] THEN mlb[xp]:=yp;
      IF yp <= mb[xp] OR yp >= mt[xp] THEN
         IF flag THEN fill(xp,yp,mp[xp],i,ip[xp])
           { from the point (xp,yp) with an intensity i to }
           { the point (xp,mp[xp]) with an intensity ip[xp]}
                ELSE switch on(xp,yp,i);
      mc[xp]:=yp;
      ic[xp]:=i
END;
PROCEDURE set up masks;
BEGIN mt:=mlt;mb:=mlb;
      mp:=mc;ip:=ic
END;
```

Algorithm 4.2.

The algorithm shown above is very simple and clear to understand. We have not dealt with the problem how MP and IP arrays were originally set up. The FILL procedure is actually the Bresenham's algorithm that is modified so that x coordinate is constant, y coordinate is changed with the step 1 and the intensity level is appropriately changed in order to get the whole intensity scale from the current intensity represented by IC[x] value to the previous intensity represented by IP[x] value or vice versa. The procedure itself must draw only outside the given strip of invisibility that is given by MT and MB arrays.

5. Hidden-Contouring Problem Specification

The hidden-contouring problem is described in literature very rarely because it covers several non-trivial tasks. The first is the contouring problem itself, the second is the problem of hidden-line elimination. The known algorithms are very complicated [3]. The main effort seems to be spent on the part that deals just with the hidden line removal. The known algorithms just use the algorithm for contouring and hidden line removal in the "pipe-line" way. The problem can be easily solved again in a very similar manner to the hidden-line elimination described above. If fig.5.1. is analyzed we can see that the problem can be easily described as follows. Firstly two margins must be drawn and then for each grid all contours must be drawn with respect to the visibility. The contour drawing order is substantial because the contours must be drawn so that the higher contours are drawn later on, e.g.:

contour(p) < contour(p+1)   for all p

When all contours for the given grid have been drawn it is necessary to draw "back margins" of the given grid. If any user needs smaller grids or smoothly interpolate contours he can subdivide grid mesh or he can employ the smooth interpolation (some kind of smooth interpolation can be done direct in the raster environment). The whole problem is now simple but we have to find a simple method which determines silhouetting in order to get the proper outlook. The silhouetting problem solution is described in literature very rarely because of high complexity of computation. Actually it is a problem of determining whether the partial derivation of the given function according to the observer's eye direction is zero. The hidden-contouring problem solution (without the solution of silhouetting) can be schematically described by the algorithm 5.1.

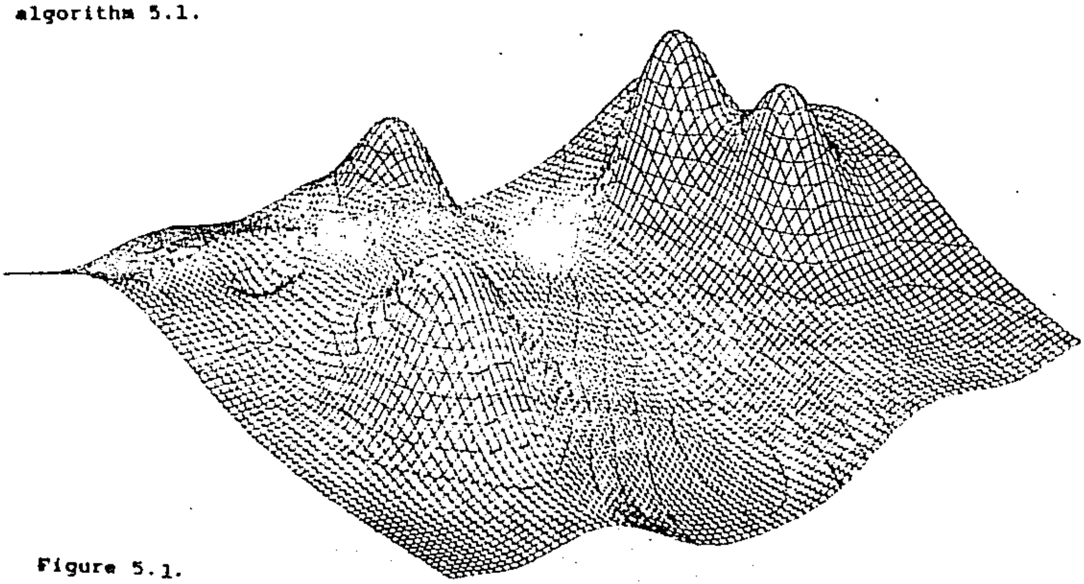


Figure 5.1.

6. Conclusion

The above presented algorithms are based on the Bresenham's algorithm for drawing straight lines in the raster environment. The algorithms for drawing functions of two variables are intended for the use in the raster environment, e.g. for raster displays or digital plotters. The presented algorithms for hidden-line, hidden-surface and hidden-contour removal are very fast, easy to implement and they do not need operations in the floating point representation for the determining whether the line segment or its part is visible or not.

All mentioned algorithms were implemented on the 8-bit microcomputer with 8080 microprocessor running at 3MHz. Presented results were drawn in 2 sec. approximately including the visibility solution with the resolution 256x256 points with 8 colours. The time of function computation, scaling and rotation is not included, because a host computer was used. The drawing was about 15% slower than drawing without the solution of the visibility.

7. Literature

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

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

[3] Clapworthy, G.J.: The Representation of a Geographical Terrain from Cartographic Data, MSc.Thesis, The City University of England, London, 1985

[4] Earnshaw, R.(Ed.): Fundamental Algorithms for Computer Graphics, NATO ASI, Series F, Vol.17, Springer Verlag 1985.

[5] Piteway, M.L.V., Watkinson, D.: Bresenham's Algorithm with Gray Scale, Communication of ACM 23(11), November 1980, pp.625-626.

[6] Skala, V.: Drawing of Functions of Two Variables with Respect to Visibility, TR 209-05-78, Computer Science Dept., Technical University, Plzen 1978 (in Czech).

[7] Skala, V.: Hidden-Line Processor, TR 209-03-84, Computer Science Dept., Technical University, Plzen 1984.

[8] Skala, V.: Hidden-Line Processor (a special case), CSTR/29, Brunel Univ., Middlesex, England, 1984.

[9] Skala, V.: An Interesting Modification to the Bresenham Algorithm for Hidden-Line Solution, Fundamental Algorithms for Computer Graphics, NATO ASI, Series F, Vol.17, Springer Verlag, (Edited by Earnshaw R.A.), 1985, pp.593-601.

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

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

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

[13] Wright, W.A.: A Two-Space Solution for the Explicit Functions of Two Variables, IEEE Trans. on Computers 22(1), January 1973, pp.28-33.

[14] Boller, D.J.: Efficient Hidden Line Removal for Surface Plots Utilizing Raster Graphics, NATO ASI, Vol.17, SerF, Springer Verlag, pp.603-615, 1985.