The drawing of 3 dimensional surfaces

Citation for published version (APA):
Eindhoven University of Technology
Computing Centre Note 4

THE DRAWING OF 3 DIMENSIONAL SURFACES

Lieuwe Sytse de Jong.
1. **Introduction.**

In this paper a unified approach is given for the graphical representation of a single-valued function $S$ of two variables $x$ and $y$, as seen from an arbitrary direction. It is assumed that the function is defined on a rectangle and that the viewing is non-perspective. The surface of $S$ is visualized by drawing the visible parts of a number of curves lying on the surface. The drawing plane is perpendicular to the viewing direction and is in fact the plane upon which the curves are projected.

On this hidden line problem a number of papers appeared. The earliest algorithms attacked the problem in 3-space (Kubert et al. 1968 [1]). The visibility of a particular point on the surface is there determined by intersecting the surface with a line parallel to the viewing direction, going through that point. If there are no further intersections, then the point is visible and its projection on the drawing plane may be drawn. In case the surface is represented by $N^2$ points, this visibility definition leads to an amount of work proportional to $N^3$ operations.

If, however, the problem is solved in the drawing plane, the computational effort may become proportional to $N^2$ operations. The surface is then divided into a number of elementary parts, which are projected and drawn in a certain order. This order is very essential. A particular point on the surface is visible, if its projection does not fall into or onto the projections of the elementary parts that, according to the order are to be projected before that point. So, at any moment in the algorithmic process the drawing plane divides into a free and a forbidden area. In this way, the visibility of points can be determined more efficiently.
The methods proposed by Williamson (1972, [2]) and Wright (1973, [3]) are based on this principle. Williamson considers lines of constant x on the surface (iso-x lines), but Wright considers the case of both iso-x and iso-y lines on the surface.

The forbidden area as it arises in the faster algorithms of Williamson and Wright may be memorized in two essentially different ways, leading to two different types of algorithms.

The first type takes advantage of the fact that there are only a finite number of addressable points on the screen of a graphical display. For instance, the addressable points of a T4010 of Tektronix are arranged in a raster of 1024 x 768 points. It is clear that the forbidden area may then be memorized by one or two arrays of 1024 elements. The visibility of any projected point is determined by comparing its coordinates with these arrays. The disadvantages of this type of algorithms is their low resolution and their device dependence. Both disadvantages can be overcome by introducing an artificial raster having much more points (10^6 x 10^6 say). However, doing so, would be very unpractical, since the amount of work obviously is proportional to the size of the raster. This amount of work is prohibitive even if rasters of a "normal" size are used. The algorithms known to the author reduce the work by investigating only a few points on the straight line pieces of which the visible part must be determined (e.g. the mid-points and the end-points). In case this procedure leads to visibility errors, one is advised to investigate more points on the line pieces.

The second type of algorithms remember the forbidden area by the points characterizing its boundary. If, for instance, the forbidden area is a polygon, then the only points to be memorized would be the corner points. Because the visibility of a point is now harder to determine, this type of algorithms tends to be more complicated and sophisticated. However, these algorithms do not have the disadvantages of the algorithms of the first type and can be made as fast.
The cross-hatching algorithm of Wright belongs to the first type, the line-hatching algorithm of Williamson belongs to the second type of algorithms. Here we shall consider only the second type.

The paper will be organized as follows. In section 2 we make some conventions, whereafter, in section 3, the general set-up of the algorithms is discussed. The last section is devoted to the complexity and efficiency of the algorithms and to the removal of the restrictions under which they work.

2. Conventions.

2.1. The surface.

We restrict ourselves to the graphical representation of continuous functions \( S \) of two variables \( x \) and \( y \) defined on a rectangle \( \{(x,y)\mid x_0 \leq x \leq x_n; y_0 \leq y \leq y_m\} \).

The coordinate systems considered here will always be rectangular and right-oriented.

Let

\[
x_b = x_0 < x_1 < x_2 \ldots < x_n = x_l
\]

and

\[
y_b = y_0 < y_1 < y_2 \ldots < y_m = y_l
\]

be sets of values on the \( x \)-axis and the \( y \)-axis.

We assume that \( S \) is known on the rectangular raster formed by the points \((x_i,y_j)\), either by a formula or by a table. The points \((x_i,y_j, S(x_i,y_j))\) shall be denoted by \( S_{ij} \).

Instead of dealing with \( S \), we shall be concerned with a piecewise bilinear function \( N \) that approximates \( S \) as follows:

\( N \) is linear in \( x \) and \( y \) on each mesh \((x_{i-1} \leq x \leq x_i, y_{j-1} \leq y \leq y_j)\), and in the raster points it holds that \( N(x_i,y_j) = S(x_i,y_j) \).
Obviously, it depends on the number of elements in the sets \( \{x_i\}_{i=1}^n \) and \( \{y_j\}_{j=1}^m \) and on the spread of the elements of these sets, how well the surface of \( S \) is approximated by the surface of \( N \).

The iso-\( x \) lines of interest are

\[
X_i := \{(x,y,z)|x = x_i; y_0 < y < y_m; z = N(x,y)\}, \ (0 < i < n)
\]

and the iso-\( y \) lines of interest are

\[
Y_j := \{(x,y,z)|x_0 < x < x_n; y = y_j; z = N(x,y)\}, \ (0 < j < m).
\]

We note that \( X_i \) is equivalent with the broken line \( S_{i1}S_{i2} \ldots S_{im} \) and that \( Y_j \) is equivalent with the broken line \( S_{1j}S_{2j} \ldots S_{nj} \).

(The broken line ABCDE is the set of the straight line pieces AB, BC, CD and DE; A, B, C, D and E are called the corner points of the broken line).

2.2. The viewing.

Next, let us introduce some terminology concerning the viewing. We shall say that a spectator is in the neutral position with respect to a coordinate system, if the \( x \)-axis points to his right (parallel to the horizon), the \( y \)-axis points away from him and the \( z \)-axis points upwards. The viewing position is defined by two angles \( \psi \) and \( \theta \). From the neutral position the spectator moves into the viewing position by going \( \psi \) radians to the right (in the \( X-O-Y \) plane) and, next, \( \theta \) radians upwards (perpendicular to the \( X-O-Y \) plane).

\[\text{figure 1}\]
Let us suppose that the spectator is in the viewing position with respect to OXYZ. If we define a new coordinate system O'X'Y'Z' such that the spectator is in the neutral position with respect to O'X'Y'Z' then we must have the relations

\[
\begin{align*}
x' &= x \cos \psi + y \sin \psi \\
y' &= -z \sin \theta + (y \cos \psi - x \sin \psi) \cos \theta \\
z' &= z \cos \theta + (y \cos \psi - x \sin \psi) \sin \theta
\end{align*}
\]

between the two sets of coordinates of a point in space. The drawing plane is a plane \(y' = d\), parallel to the X'O'Z'-plane, where \(d\) is taken to be smaller than all \(y'\)-coordinates of the surface. The projection of a point of the surface on the drawing plane is simply obtained by setting its \(y'\)-coordinate to \(d\) and leaving the other coordinates. So, the projections of the points \(S_{ij}\) on the drawing plane are given by

\[
P_{ij} = \begin{bmatrix} x_1 \cos \psi + y_j \sin \psi \\ d \\ S(x_i, y_j) \cos \theta + (y_j \cos \psi - x_i \sin \psi) \sin \theta \end{bmatrix}
\]

The projections of the iso-x and iso-y lines (\(X_i\) and \(Y_j\)) are respectively the broken lines

\[
I_i := P_{i1} P_{i2} \ldots P_{im}, \quad (0 < i < n)
\]
\[
Y_j := P_{1j} P_{2j} \ldots P_{nj}, \quad (0 < j < m)
\]

If \(\psi \neq 0 \mod(\pi/2)\), then we see that dependent on the sign of \(\sin \psi\) or \(\cos \psi\), the \(x'\)-coordinates of the corner points of the broken lines \(I_i\) or \(J_j\) are either strictly increasing or strictly decreasing. From now on we assume that \(\psi \neq 0 \mod(\pi/2)\). We shall say that \(I_i\) and \(J_j\) are proper broken lines because of the monotonicity of the \(x'\)-coordinates of their corner points.
2.3. A notion of distance.

Next, let us introduce a notion of distance for the planes \( \{x=x_i\} \) (\( 0 < i < n \)) and \( \{y=y_j\} \) (\( 0 < j < m \)) with respect to the drawing plane. With this notion we bring about a relative ordering within the set of iso-x planes, within the set of iso-y planes and also within the sets of objects that will be introduced in the next section.

If, going along into the direction of the positive \( y' \)-axis, the plane \( \{x=x_i\} \) is met before the plane \( \{x=x_k\} \) then the plane \( \{x=x_i\} \) is said to be closer than the plane \( \{x=x_k\} \).

Notice that, if the plane \( \{x=x_i\} \) would be opaque, nothing could be seen of the plane \( \{x=x_k\} \). This implies that if we would make a drawing of these planes as seen from the viewing direction, first the plane \( \{x=x_i\} \) and, thereafter, the plane \( \{x=x_k\} \) should be drawn.

For the planes \( \{y=y_j\} \) we introduce the notion of distance in a similar way. Note that a necessary condition that the planes \( \{x=x_i\} \) and \( \{y=y_j\} \) are met indeed, is that \( \psi \neq 0 \text{ mod}(\pi/2) \).

2.4. The visibility of a point on the surface.

Finally, let us be more specific about the visibility of a point on the surface. A point \( P = (x'_p, y'_p, z'_p) \) lying on the surface is a visible point on the lower side of the surface, if for all points \( Q \) on the surface with \( x'_q = x'_p \) and \( y'_q < y'_p \), it holds that \( z'_q > z'_p \). It is said to be a visible point on the upper side of the surface, if for all points \( Q \) with \( x'_q = x'_p \) and \( y'_q < y'_p \) it holds that \( z'_q < z'_p \). It is easy to verify that, if \( P \) is visible, a line from \( P \) to the drawing plane, parallel to the \( y' \)-axis, intersects the surface only in \( P \).
3. The general set-up of the algorithms.

3.1. Introduction.

Having defined the viewing direction and a drawing plane, the algorithm breaks up in the following parts.

A. {the definition of a number of opaque objects of which the contours give a good (?) impression of the surface}

B. {the ordering of the objects in such a way that the visible part of the image of any object can be drawn without taking account of its successors in the order}

C. {the drawing of the contour of the first object, projected on the drawing plane, and the initialization of an area in which no further drawing is allowed}

D. while {any objects} left do
   {pick the next object, draw the part of its contour, which is visible (i.e. the part situated outside the forbidden area) and update the forbidden area}

Note that all computations are relative to the drawing plane, which is one reason for the efficiency of this algorithmic set-up. A second reason is found in the introduction of ordered objects and a forbidden area. When drawing an object, it is only necessary to take account of a few other objects. We shall see that within this set-up still much can be gained by looking into the possible data-structures with which the forbidden area may be memorized and updated. In the remainder of this section we shall discuss the various parts of the algorithm.

3.2. The various objects.

In this paper we distinguish the following opaque objects.

A. Partitions contained in an iso-x (or iso-y) plane of which the upper (c.q. lower) side is an iso-x or iso-y line. To be more precise, the iso-x partitions are given by:
   \[(x,y,z) \mid x=x_i, y_0 \leq y \leq y_m, z \leq N(x,y)\}, (0 \leq i \leq n)\].
B. Strips lying on the surface, each strip being the area between two adjacent iso-x (or iso-y) lines.
So, the iso-x strips are given by:
\[ \{(x,y,z)|x_{i-1} < x < x_i, y_0 < y < y_m, z = N(x,y)\}, (1 < i < n). \]
C. Meshes lying on the surface; each mesh being the area enclosed by two adjacent iso-x and two adjacent iso-y lines.
In formula form:
\[ \{(x,y,z)|x_{i-1} < x < x_i, y_{j-1} < y < y_j, z = N(x,y)\}, \]
\[ (1 < i < n, 1 < j < m). \]

Figures 2, 3 and 4 give an impression how a surface might look, if these three different objects are used.
Using partitions, one can give a graphical representation of only one side of the surface. It is of course not necessary to draw a skirt as in figure 2. In the drawing plane a partition is characterized by a broken line $I_i$ (or $J_j$). If $\psi \neq 0 \mod(\pi/2)$, we can memorize a partition by a proper broken line $C = C_1C_2 \ldots C_k$ with $k = n$ (or $k = m$): the $x'$-coordinates of the $C_i$ are strictly increasing.

Strips on the surface may serve for simultaneous drawings of the lower and the upper side of the surface. Such a strip is characterized by one pair of broken lines $I_{i-1}I_i$ (or $J_{j-1}J_j$). If again $\psi \neq 0 \mod(\pi/2)$, then the strip may be memorized by two proper broken lines $C = C_1C_2 \ldots C_k$ and $F = F_1F_2 \ldots F_k$ (where $k = n+1$, or $k = m+1$) with $C_1 = F_1$, $C_k = F_k$. The broken line $C_2 \ldots C_{k-1}$ shall correspond with the part of the strip that is nearer to the drawing plane. Note that if $F_2$, $\ldots$, $F_{k-1}$ are lying above the $C$-line, then the upper side of the strip is entirely visible.

Meshes on the surface may also be used for simultaneous drawings of the lower and the upper side of the surface. They too can be memorized by two proper broken lines $C$ and $F$. However, now the $C$- and $F$-line contain only three points.

3.3. The ordering of the objects

Having defined the objects, they must be arranged such that the correct visible part of any of them can be drawn without accounting for the successors in the order. Generally, such an arrangement might be impossible, but also several arrangements might exist. The order does depend on the viewing direction.
Partitions are ordered according to their distance from the drawing plane as introduced in section 2 (each partition is situated in one iso-plane).

For strips an order is found by arranging them according to the distance of their C-lines.

Meshes may be put in order in several ways. We propose the following order. First the \( m \) meshes within each iso-\( x \) strip are arranged according to the order of the iso-\( y \) strips to which they belong as well. Thereafter the iso-\( x \) strips are ordered in the way described above. Figure 5 gives an impression of the order thus obtained. Figure 6 gives an alternate way for ordering the meshes.

For the three kinds of objects thus ordered, we shall prove the claim that each object may be drawn without making allowance for its successors in the order.

First let us consider partitions. Since any partition is nearer to the drawing plane than all its successors (i.e. leaving the drawing plane in the viewing direction, any partition is met before its successors), the assertion holds for partitions.
Next, we consider strips on the surface. Let $S_1$ and $S_2$ be any pair of strips (iso-x say), such that $S_1$ precedes $S_2$. Let $S_1$ be determined by the iso-x lines $I_i$ and $I_{i+1}$ and let $S_2$ be determined by $I_j$ and $I_{j+1}$. We assume that the viewing position is such that, if $I_k$ is nearer than $I_l$, it holds that $k < l$.

So must have that $i < i+1 < j < j+1$, and, hence it follows that $I_{i+1}$ is not farther than $I_j$. Consequently, when going in the viewing direction points of $S_1$ as well as points of $S_2$ are met, then the points of $S_1$ are always met before the points of $S_2$. This argument shows that $S_1$ may be drawn before $S_2$.

Finally let us consider the meshes. We suppose that $M_1$ and $M_2$ are two meshes and that $M_1$ precedes $M_2$. If $M_1$ and $M_2$ belong to different iso-x strips, then the claim holds, because it holds for strips. If $M_1$ and $M_2$ belong to the same iso-x strip, then they belong to two different iso-y strips ($I_k$ and $I_{l}$ say). Because $M_1$ precedes $M_2$, $I_k$ must be nearer than $I_l$, which shows that $M_1$ may be drawn before $M_2$.

The foregoing is rephrased by the following lemma:

**Lemma 3.3.**

The objects are so ordered that, if $P'$ and $Q'$ are the projections of points $P$ and $Q$ belonging respectively to object $A$ and the farther object $B$, it holds that

$$x'_P = x'_Q + y'_P < y'_Q.'$$

**Remark:** The lemma does not hold in case the viewing is perspective.

### 3.4. The characterization of the forbidden area.

The projected contour of the first object can be drawn completely, because it has no predecessors. The forbidden area initializes of course with the broken line(s) characterizing the contour of the first object. However, memorizing the forbidden area in this way is algorithmically unwise if one is dealing with strips or meshes, because it hinders an easy update. We shall illustrate this in the case of strips (see figure 7).
The contour of the first strip consists of the two proper broken lines C and F, where C corresponds with the nearest part of the strip and F corresponds with the farthest part. After projection it is seen that the forbidden area divides into parts descended from the upper-surface (the shaded parts) and into parts descended from the lower-surface (the blank parts in figure 7). A point Q at the upper surface but not on the first strip, is visible if $z'_P < z'_Q$ for all P on the first strip with $x'_P = x'_Q$ and $y'_P < y'_Q$ (lemma 3.3 states that $y'_P < y'_Q$ is implied by $x'_P = x'_Q$). Consequently, the area in the drawing plane that is forbidden for future projections of points of the upper-surface is:

$$\{(x',z') | x < x'_C; z' < C(x') \land z' < F(x')\}.$$ 

In the figure this would be the partition with upper boundary $UB := F_1 F_2 S_1 C_3 S_2 F_5 F_7; UB$ is a proper broken line. Similarly, we find as the area that is forbidden for future projections of points of the lower-surface:

$$\{(x',z') | x < x'_C; z' > C(x') \land z' > F(x')\}.$$ 

In the figure, this would be the partition with lower boundary $LB := C_1 S_3 F_2 S_4 C_6 C_7; LB$ is a proper broken line.

So, initially, the forbidden area is characterized by two proper broken lines LB and UB describing the boundary of the area; LB lying below UB.
Also, in case meshes are considered, such a characterization is possible. If partitions are used, it is obvious that only one line LB is needed.

From a computational point of view these characterizations are very pleasant, especially if they pertain during the computational process.

**Theorem 3.4.**

If the surface is connected and the viewing is non-perspective, then at any time in the algorithmic process an area forbidden for future projections exists. In case strips or meshes are used by the algorithm, this area may be characterized by two proper broken lines $\text{LB} = \text{LB}_1 \text{LB}_2 \cdots \text{LB}_t$ and $\text{UB} = \text{UB}_1 \text{UB}_2 \cdots \text{UB}_k$ such that

(i) $\text{LB}_1 = \text{UB}_1$ and $\text{LB}_t = \text{UB}_k$

(ii) LB lies below UB

(iii) The area above LB is forbidden for future projections from the lower-surface.

(iv) The area below UB is forbidden for future projections from the upper-surface.

In case partitions are used, the forbidden area is characterized by only one broken line UB, below which future drawing is forbidden (in case the partitions are used for the drawing of the lower surface, then we are dealing with a broken line LB, above which ....). The line UB is almost proper, viz.: its x'-coordinates are non-decreasing.

**Remark:** for shortness sake we shall call an area in which future drawing is forbidden and which satisfies the other demands of the theorem, a proper region. (Note that the meaning of a "proper region" depends on the objects used.)

**Proof of the theorem.**

It is straightforward to verify that the area initialized by the first partition, strip or mesh is a proper region.

Let us suppose that the forbidden area as it is found after drawing k objects is a proper region. We intend to show that then the area as it is obtained after drawing (k+1) objects is a proper region.
It is obvious that the area determined by the \((k+1)\)-th object is a proper region itself. The union of this area and the area corresponding with the first \(k\) objects is the new forbidden area. In case of partitions it is trivial that the new forbidden area is a proper region provided that we postulate that it is connected (this is a natural demand; otherwise we would be looking across the partitions).

In the case of meshes or strips there are a few possibilities for the union of the forbidden area corresponding with the first \(k\) objects and the forbidden area of the \((k+1)\)-th object. These possibilities are indicated in the following figure. The lower contour corresponds with the proper region of the first \(k\) objects, the upper contour corresponds with the \((k+1)\)-th object.

The first and second possibility admit the existence of points \(A\) and \(B\) with \(x'(A) = x'(B)\) such that all points \(C\) on the surface with \(x'(C) = x'(A)\) and \(z'(B) < z'(C) < z'(A)\) have not yet been projected. From the continuity of the surface along the line \(x' = x'(A)\) it follows that such points \(C\) do exist. However, one can show that the way in which the objects are ordered assures that the points \(C\) should already have been projected. So the first two possibilities may be excluded.
Also the third possibility: a vertical line piece as part of the boundary of the new forbidden area, may be excluded: a line piece \( x' = \text{constant} \) cannot correspond with a boundary line of the surface \( (\psi \neq 0 \mod(\pi/2)) \). Hence, due to the continuity of the surface, one should have that the image of the first \((k+1)\) objects contains points at the left as well as at the right of the line \( AB \).

Consequently, it is possible to characterize the new forbidden area by two proper broken lines \( UB \) and \( LB \). It is not difficult to verify that also the other demands are satisfied. Hence we conclude the validity of the theorem.

**Remark:** The non-perspective viewing is not explicitly used in this proof, however it is concealed present, because without it meshes could not be ordered in the way they are.

### 3.5. The actual realization.

In this subsection we shall be concerned with the actual realization of the algorithm. In the case of partitions, the forbidden area initializes with the iso-\( x \) (or \( y \)) line that characterizes the first partition. Later on in the algorithmic process — after the drawing of \( k \) partitions say —, the new forbidden area that corresponds with \((k+1)\) partitions has as its boundary

\[
UB_{\text{new}}(x') = \max \{UB_{\text{old}}(x'), I_{k+1}(x') \mid \ldots < x' < \ldots \}.
\]

Algorithmically we can find \( UB_{\text{new}} \) as follows. First of all we determine the leftmost point of \( I_{k+1} \) that belongs to \( UB_{\text{new}} \) (this is either the leftmost point of \( I_{k+1} \) or the left most intersection point of \( UB_{\text{old}} \) and \( I_{k+1} \)). If such a point does not exist, then \( UB_{\text{old}} \equiv UB_{\text{new}} \), otherwise we know a first part of \( UB_{\text{new}} \). In the latter case we walk from the point just found to the right along the edge of the new forbidden area, keeping left at the intersection points of \( UB_{\text{old}} \) and \( I_{k+1} \), and simultaneously drawing if we walk on a part of \( I_{k+1} \). Since \( UB_{\text{new}} \) can be expected to be a mixture of joint pieces of \( UB_{\text{old}} \) and \( I_{k+1} \), a linked list is the natural datastructure for \( UB_{\text{new}} \). The complexity of this procedure depends critically on the number of intersections.
In the case of strips or meshes, the initial UB and LB are determined as the maximum respectively the minimum of the C- and the F-line of the first strip or mesh. The procedure to do this is quite similar to the one described for partitions. Having drawn the image of the first k strips or meshes, the new forbidden area is determined by the boundaries

\[
\begin{align*}
UB_{\text{new}}(x') &= \max \left\{ UB_{\text{old}}(x'), C_{k+1}(x'), F_{k+1}(x') \right\} \quad \ldots < x' < \ldots \\
LB_{\text{old}}(x') &= \min \left\{ LB_{\text{old}}(x'), C_{k+1}(x'), F_{k+1}(x') \right\} \quad \ldots < x' < \ldots 
\end{align*}
\]

In the case of strips we know that they are so arranged that \( C_{k+1} = F_k \) for all eligible k. This implies that the maximization and minimization process above can be restricted to \((UB_{\text{old}} \text{ and } F_{k+1})\) or \((LB_{\text{old}} \text{ and } F_{k+1})\) respectively. The actual determination of \( UB_{\text{new}} \) and \( LB_{\text{new}} \) is then quite similar to the procedure in the case of partitions. Again the linked list is the natural data structure for LB and UB.

In the case of meshes the situation is somewhat different. Although each mesh has a part in common with its predecessors in the order, this part is not always the same. Hence the reduction in the maximization and minimization process varies. We distinguish three possibilities.

1. \[ \begin{array}{c}
M_1 \quad M_2 \quad \ldots \quad M_k \quad M_{k+1} \\
C_1 \quad C_2 \quad C_3 \quad \ldots \quad \ldots \quad \ldots \\
\end{array} \]

2. \[ \begin{array}{c}
M_1 \quad M_2 \quad \ldots \quad M_k \\
C_1 \quad C_2 \quad C_3 \\
\end{array} \]

3. \[ \begin{array}{c}
M_{k+1} \quad M_k \quad \ldots \quad \ldots \quad \ldots \quad M_{2k} \\
C_1 \quad C_2 \quad \ldots \quad \ldots \quad \ldots \\
\end{array} \]

figure 9
In the first case we have that $C_1 C_2$ is already a part of UB or LB. In the second case we have that $C_2 C_3$ is already a part of UB or LB. In the third case we have that $C_1 C_2 C_3$ is already part of UB or LB. The algorithm should distinguish these three different cases. The determination of UB\textsubscript{new} and LB\textsubscript{new} is then again quite similar to the case of partitions. Linked lists are the natural datastructures.

4. Assessment of the algorithms.

4.1. The computational complexity.

Assuming that the array of altitudes by which the surface is given is $n \times n$, we shall show that the complexity of the algorithmic set-up introduced in this paper is equal to $n^3$. However, it is very unlikely that in practice so many operations are encountered. Normally, one may expect an operation count of $n^2$. The computational effort is mainly determined by the optimization procedure, which is necessary in the updating process of the forbidden area. In this process the intersection of two proper broken lines plays a leading part as far as the efficiency is concerned. One can show that two proper broken lines with $k$ respectively $t$ cornerpoints can have at most

$$T(k,t) := k + t + \min(k,t) - 2 \text{ intersection points.}$$

Algorithmically, it is not difficult to arrange things in such a way that, indeed, all intersection points are found in at most $T(k,t)$ attempts. If the broken lines would not be proper, then as much as $k \times t$ intersection points and attempts to find these points would be necessary!

In the case of partitions or strips, the forbidden area needs $n$-times updating (including the initialization of the first area).
If \( k \) and \( t \) are always of the order of \( n \), then the computational effort will be of the order of \( n^2 \). However, if the number of corner points of the forbidden area has the maximal theoretical increase of \( T(k,n) + n - 2 \) (assuming that the strip-boundaries have \( t = n \) corner points), then the total operation count could be of the order of \( n^3 \). Consequently, the complexity is \( n^3 \). This is also true for meshes. Figure 10 suggests a situation with a large number of intersections.

\[
\text{figure 10}
\]

4.2. The efficiency of the algorithms.

From subsection 4.1., it follows that it is important to program the intersection of line pieces in the drawing plane as efficient as possible. Let us suppose that we want to intersect the line pieces \( AB \) and \( CD \), having coordinates relating to the drawing plane given by \( a, b, c \) and \( d \).

One obvious way to investigate whether \( AB \) and \( CD \) do intersect is to compute the intersection point \( S \) of the lines

\[
x = a + \lambda(b - a)
\]

\[
x = d + \mu(c - d) \quad (\lambda \in \mathbb{R}, \mu \in \mathbb{R})
\]

and to verify whether \( S \) - if it exists - belongs to both \( AB \) and \( CD \) (via \( \lambda_S \in [0,1], \mu_S \in [0,1] \)).

However, doing so, one is always confronted with the full complexity of the intersection process of two lines. Simple tests are possible to decide - in case \( AB \) and \( CD \) are not too close to another - that an intersection does not exist. Applying them may save a lot of computational effort.
The sets of $x'$-coordinates and $y'$-coordinates of AB and CD are respectively

$$X_{AB} := \{\min(a_1, b_1), \max(a_1, b_1)\},$$

$$X_{CD} := \{\min(c_1, d_1), \max(c_1, d_1)\},$$

$$Y_{AB} := \{\min(a_2, b_2), \max(a_2, b_2)\},$$

$$Y_{CD} := \{\min(c_2, d_2), \max(c_2, d_2)\}.$$ 

It is quite clear, that if $(X_{AB} \cap X_{CD})$ is empty or $(Y_{AB} \cap Y_{CD})$ is empty that then no intersection point can exist.

In case we want to compute the possible points of intersection, we have to solve the system of equations:

$$\begin{bmatrix}
(b_1 - a_1) & (d_1 - c_1) \\
(b_2 - a_2) & (d_2 - c_2)
\end{bmatrix}
\begin{bmatrix}
\lambda \\
\mu
\end{bmatrix} = 
\begin{bmatrix}
d_1 - a_1 \\
d_2 - a_2
\end{bmatrix}$$

A possible solution should have $\lambda, \mu \in [0,1]$. This is only possible, if for $i = 1, 2$:

$$\min(0, (b_i - a_i)) + \min(0, (d_i - c_i)) \leq (d_i - a_i) \leq \max(0, (b_i - a_i)) + \max(0, (d_i - c_i)).$$

This test is equivalent with the one we gave above.

A satisfactory way to solve the system above is by Gaussian elimination with row pivoting. That process can be stopped as soon as it is found that $\mu$ or $\lambda$ is not contained in $[0,1]$. One may show that the complete intersection process implemented in the above described manner needs the following number of multiplications/divisions:

0, (if the preliminary tests indicate that an intersection does not exist)

2, (if the matrix of coefficients is singular)

4, (if it appears that $\mu \in [0,1]$)

6, (if it appears that $\lambda \in [0,1]$)

8, (if $AB$ and $CD$ do have an intersection point).
Let us give an impression of the execution times of the new algorithms. The cross hatching version (which uses meshes as objects) has an execution time of about $1.3 \times 10^{-3} \times N^{1.88}$ seconds on the B7700 ($N \times N$ points on the surface). The algorithm of Wright needs the same amount of processor time but requires twice as much storage on that machine. Moreover, visibility errors can occur with that algorithm, which can only be cured at the expense of more computer time.

4.3. The restrictions.
A number of restrictions were introduced under which the new algorithms should work:

a. The azimuthal angle $\psi$ should not be equal to 0, $\pi/2$, $\pi$ or $3\pi/2$.

b. The drawing is non-perspective.

c. The surface is defined on a rectangle.

The first restriction is needed to ensure that in the cross-hatching version the meshes can be characterized by proper broken lines. For the other versions this restriction is not necessary. However, it is not a severe restriction: if the azimuthal angle is very little perturbed from a given value of 0, $\pi/2$, $\pi$ or $3\pi/2$, then the algorithm can be applied.

The second restriction is required to enable the ordering of the meshes as it was done in subsection 3. It is always possible to divide the surface as seen in perspective from some point in space, such that both sub-surfaces can be drawn with the algorithms discussed here. So that restriction is also not essential.

The third restriction is more stringent. The line-hedging versions can work without it. It is for instance possible to define the surface on a circle-shaped region. The cross-hedging version cannot work in this fashion, since on one hand the ordering of the meshes may still be possible, but on the other hand the characterization of the boundaries of the meshes is no longer possible in the same uniform manner. However, also this can be cured for.
5. Conclusions.
In this paper an algorithmic set-up is presented for the drawing of two-dimensional surfaces which divides the surface in a number of elements of which the contours give a good impression of the surface. These elements are ordered such that the correct visible part of any of them as seen from a certain direction can be drawn without taking account of the successors. The determination of the visible part of such an element is done in the drawing plane by comparing its boundaries with the boundaries of a forbidden area which is the region in the drawing plane already used for previous projections. All boundaries are characterized by "proper" broken lines which facilitates the above process very much.
The resulting algorithms are very efficient and deliver fault free projections of the surface to be drawn.
6. References.

[1] Kubert, D. J. Szabo, S. Giulieri
   The perspective representation of functions of two variables
   J.A.C.M., 2 (1968), pp 193-204

   Hidden line plotting program
   C.A.C.M., 15 (1972), pp 100-103

   A two-space solution of the hidden-line problem for plotting functions of two variables
   IEEE, Transac. on Computers, C-22 (1973), pp 28-33