Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE
DELLA INFORMAZIONE

PISA

VECTOR AND RASTER HIDDEN SURFACE REMOVAL USING PARALLEL CONNECTED STRIPES

C. Montani, M. Re

Nota interna B85-08

Agosto 1985

VECTOR AND RASTER HIDDEN SURFACE REMOVAL USING PARALLEL CONNECTED STRIPES

Claudio MONTANI° and Michele RE°°

° Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Via Santa Maria, 46, 56100 Pisa (Italy).

°° Dipartimento di Scienze dell'Informazione, Università degli Studi, Corso Italia, 40, 56100 Pisa (Italy).

ABSTRACT

A list priority algorithm for the removal of the hidden surfaces from a three-dimensional scene is presented. The general lines of the solution proposed are similar to those of the classical list priority algorithms (determination of the depth relations between the faces of the solids of the scene, construction of a priority list, removal of the hidden parts), but a new sorting procedure and the use of an efficient data structure to represent the faces of the solids (the Parallel Connected Stripes data structure) make the algorithm interesting. The PCS data structure allows the algorithm to work on several faces at a time, to solve cyclic overlaps, to remove hidden parts efficiently and to make low memory requirements. The algorithm produces both vectorial output (the boundaries of the faces and the holes), and raster output (similarly to a scan line algorithm) without any rasterization process.

1. INTRODUCTION

This paper proposes a new solution to the problem of the removal of hidden surfaces from three-dimensional scenes. Despite the fact that the first hidden-line or hidden-surface algorithms date back to the sixties, interest in new solutions offering low memory requirements, reasonable computational complexity and easy firmware implementation is still high. The extensive literature on the problem is evidence of this fact [1].

The algorithm we present is not so much new in its general lines as in the solutions it adopts and its application possibilities.

Sutherland et al.[2] classify hidden-line or hidden-surface removal algorithms into three categories: algorithms operating in object space (with unlimited resolution), algorithms operating in image space (with a prefixed resolution, the pixel size of the output device) and so-called "list priority" algorithms (generally characterized by a sorting step in object space and a display step in image space).

The algorithm which we propose is a list priority algorithm. The innovative elements of our solution are represented by the new sorting procedure and by the data structure used to represent the faces of the solids of the three-dimensional scene. Unlike Newell's sorting procedure [3], our method can determine sets of faces of the same

priority rather than priorities between single faces.

The computational complexity of the algorithm is almost linear with the number of faces of the solids. This is because we have chosen techniques similar to those adopted by Wittram [4] and Tamminen [5].

Each face of the solids of the scene being examined is represented by the "Parallel Connected Stripes" data structure [6]. The choice of this coding scheme has offered us several advantages:

a) each face can be of unlimited topological complexity and, consequently, the constraints which have to be imposed on the input data are very low. Faces with nesting holes and cyclic overlaps involving, at least, three faces are permitted;

b) the use of the PCS data structure has meant that a display procedure based on the "silhouette" concept [7] can be implemented. This concept makes it possible to retain only a small amount of information in main memory, for each step of the algorithm. The logical difference between one or more of the faces and those faces partially hiding them (this step is very often neglected in the bibliography) is very efficient and, under some conditions, can be realized for a whole set of faces of the same priority;

c) the algorithm can be forced to work in object space or in image space indifferently by simply modifying a single parameter of the data structure (the thickness of the PCS stripes) and using different procedures to convert from PCS to vectorial or raster information.

2. THE PCS DATA STRUCTURE

In 1984 the Parallel Connected Stripes data structure was defined as a coding scheme for the efficient representation of polygonal surfaces [6]. PCS naturally belongs to the raster representation class [8], but the classical concept of scanning line is replaced here by that of "scanning stripe". Let us recall the main characteristics of the PCS data structure by means of an example. For further details, see [6], mentioned above.

Figure 1 shows the boundaries of two polygonal surfaces. Each boundary is coded by a Freeman chain, i.e. by the absolute coordinates of the starting point of the chain and by a sequence of links describing the basic displacement along the nodes of a regular square grid of step size h, in 8 possible directions. Let us suppose that our polygons are cut along the horizontal lines of the grid into stripes (of thickness h) and the substripes belonging to each polygon are obtained from these stripes. The set of substripes obtained by this process is illustrated in Figure 2. A satisfactory description of every stripe of Figure 2 can be obtained by simply indicating its coordinate Y and the number of substripes which belong to it. The coordinate Y of a stripe is indicated by the ordinate of its lower side. Each substripe has a trapezoidal form (which may be triangular at the maximum or minimum local points) and can

therefore be described by the types (i.e. one of the three possible inclinations) and by the value of the abscissas of its west and east sides (Figure 3).

The information represented graphically in Figure 2 is described digitally in Table 1. This table shows the abscissas of the west and east sides and, within brackets, the side types for every substripe.

The PCS data structure is particularly suitable for the representation of topologically complex polyedrons (convex or concave polyhedrons with or without nesting holes). Since PCS is a raster representation, logical operations (union, intersection, difference) on two PCS's consider the stripes of identical ordinates (parallel scanning) and their corresponding substripes, iteratively. The replacement of a scanning line by a scanning stripe implies that the conversion process from vectorial representation to PCS data structure and viceversa does not present ambiguous situations.

3. THE ALGORITHM

Our algorithm is based on the ability of the PCS data structure to handle complex polygonal surfaces and to facilitate logical operations on distinct polyhedrons.

If a priority list of the faces of the solids of our three-dimensional scene can be constructed, then the following solution to the hidden-surface problem can be proposed. At the i_th step of the algorithm, we have two

PCS's in main storage: the first contains the i_th face to be drawn, whereas the second PCS data structure contains the silhouette of the i-1_th faces already drawn (i.e. the logical union of the i-1_th faces preceding the face under examination in the priority list). The parts of the face to be drawn are obtained by differentiating the first PCS from the second. The new silhouette is given by the logical union of the current silhouette with the face parts just drawn.

This is the basic idea of the algorithm. In addition, we will show in Section 5 how the PCS data structure can also help to solve ambiguous situations of the scene to be displayed.

In this section we will use the example given in Figure 4 in order to clarify the different steps of the algorithm. Obviously in a true three-dimensional scene (our example shows a two and half-dimensional scene) viewing and projection transformations must be applied to the faces of the solids to de displayed.

The steps of the algorithm can be summarized as follows:

a) determination of the depth relations between the faces of the scene ;

b) construction of the priority list of the faces;

c) output of faces with the elimination of hidden parts.

In order to limit the main memory requirements, the polygonal lines coding the boundaries of the faces and the coefficients of the equation of the plane determined by each face are stored in a direct access file on secondary

storage. However, on the countrary, the extents of each face (i.e. the minimum and maximum values of the circumscribing parallelepiped) are stored in main memory.

The three-dimensional scene we envisage, adopts a left-handed coordinate system with an X positive axis in a rightwards direction, a Y positive axis going upwards and a Z positive axis moving away from the observer.

The algorithm is preceded by a step in which the faces on minimum X are sorted. This expedient enables us to obtain a complexity linear with the product of N*Y, where N is the total number of faces and Y denotes the Y_complexity (i.e. the average number of faces overlapped in X by any generic face) [4].

In the first step of the algorithm the depth relations of the faces must be determined. This is achieved by applying 6 tests of increasing complexity to each pair of faces (see Table 2) until one of these tests is satisfied.

It is worth while noting that the information present in main memory is sufficient for tests 1-3. Test 6 is performed by locating any existing point in the X-Y plane, which belongs to both the PCS data structures representing the faces under examination.

Table 3 shows the depth relations between the faces in our example. In the table, the relation R<F represents the priority of face R versus face F, whereas A>C implies that face A is successive to face C. For simplicity, Table 3 presents the specular relations towards the main diagonal ($F_1 > F_2$ implies $F_2 < F_1$). Columns NMIN and NMAX represent the

counters of the "<" and ">" relations for each face, respectively.

Starting with the depth relations the priority list can be constructed. The basic idea of this step is very simple. By observing the depth relations, we can isolate a set of faces which are not hidden from the others (those for which NMAX=0) and a set of faces which do not hide others (NMIN=0). These two sets are the first and the last priority class, respectively; we insert them into two lists named FRONT and BACK.

The faces without depth relations (NMIN=0 and NMAX=0) can be inserted into the FRONT or BACK list, indifferently. The successive computations are reduced if they are inserted into the BACK list. The rows and columns of faces just considered are deleted from the table and the NMIN and NMAX counters for the remaining faces are updated. The procedure starts again from the beginning, searching for two new sets of faces at front and at the back of the scene, and considering only the remaining faces. These two new sets will be inserted at the end of the FRONT list and at the beginning of the list BACK list, respectively.

The steps of this procedure are shown in Table 4. At the end of the process the FRONT and BACK lists will be linked to form the priority list. It should be noted that the algorithm just described can not solve cyclic overlaps. A simple solution to this problem is presented in Section 5.

Our priority list is suitable for the output step of the

algorithm. Figure 5 shows the output step for the example given in Figure 4. Two PCS data structures are present in main memory: The first structure contains the silhouette of the parts of the scene already displayed, i.e. the union of all faces which, at the i_th step of the algorithm, have been differentiated from the silhouette and displayed. The second PCS data structure contains the faces to be displayed, i.e. those belonging to the i+1_th priority class.

For example, at Step 2 of the algorithm (the output of faces U and C) PCS-2 is differentiated from PCS-1 in order to give the information desired. The same information is added to PCS-1 to obtain the silhouette of next step (Step 3, in the example).

It should be noted that all faces of the same priority are considered at each step of the algorithm rather than single faces. This is possible when faces of the same class do not have common edges or different output parameters, such as colour or pattern texture. In the first case two polyhedrons having common edges will become (on PCS-2) a unique polyhedron; in the second case the output attributes of each face would be lost.

4. VECTORIAL AND RASTER PRESENTATION

So far we have offered no hypothesis for the step size of the square grid overlapped to the scene and, consequently,

for the thickness of the stripes of the PCS data structure. We can thus assert that our algorithm works in object space, i.e. in a coordinate space independent of output peripheral. The resulting precision (i.e. the quality of the output) is functional of the thickness chosen for the stripes.

If the output device is vectorial or if a vector type output is required, the display is achieved by reconstructing, at each step, the boundaries of polyhedrons (and their holes) obtained by the difference operations. The procedure which converts from PCS data structure to vectorial information can be found in [6].

If the output device is of raster type and we want to display the faces of the scene as polyhedrons of uniform colour, it is not necessary to differentiate the faces on the basis of their own priority. It is sufficient to use the overwriting technique, examining the priority list, going from back to front. If the device used does not permit this (e.g. a photoplotter), or if it is desired to reduce the number of I/O device operations, the differentiation step must be performed. In this case, the algorithm can be forced to work on image space by assuming the thickness of the stripes to be equal to (or a multiple of) the pixel size of the output device. In this way the reconstruction of the boundaries of the polyhedrons to be represented is no longer necessary. In fact, it is sufficient to scan the stripes of the data structure and to color the pixels covered by all substripes. For the stripe of ordinate Y of Figure 6, for

example, we will color the pixels from X_{60} to X_{84} and from X_{100} to X_{125} at Y_{120} of the screen.

However, in this way, we reduce each substripe of the PCS data structure to a rectangular rather than a trapezoidal form. We feel that the extra information given by the PCS (the sides of type 1 and 3 of a substripe) could be efficiently exploited by implementing an appropriate anti-aliasing technique. We hope this consideration will be the subject of a future paper.

Figures 7 and 8 show two examples of vectorial applications of our algorithm. Figure 9 is a raster application example.

5. CYCLIC OVERLAPS

A simple extension of the list-priority algorithm enables us to solve cases of cyclic overlaps for, at least, three faces. However, our algorithm is unable to solve the ambiguous situation shown in Figure 10. Let us suppose that the set of faces $(F_1,\ldots,F_n)$ of a scene presents a cyclic overlap among the faces F_1, F_2 and F_3. The depth relations are (Figure 11):

$$F_1 < F_2 \,,\; F_2 < F_3 \,,\; F_3 < F_1 \,.$$

In order to solve this stalemate situation, it is sufficient to differentiate the PCS's representing faces F_1 and F_2, respectively. The new face to be considered in the construction of the priority list will be $F_2 = F_2 - F_1$. The

depth relation

$$F_1 < F_2 \quad (F_2 > F_1)$$

can be deleted by the depth relations table and the cyclic overlap will be solved.

It should be noted that the faces of the set presenting depth relations with F_1, F_2 or F_3 (but not belonging to the cyclic overlap themselves) are assigned to the priority list by the ordinary procedure described in Section 3.

REFERENCES

[1] J. G. Griffiths, "A bibliography of hidden-line and hidden-surface algorithms", Computer Aided Design, 10(3), 1978, 203-206.

[2] I. E. Shuterland, R. F. Sproull and R. A. Schumacker, "A characterization of ten hidden-surface algorithms", Computing Surveys, 6(1), 1974, 1-55.

[3] M. E. Newell, R. G. Newell and T. L. Sancha, "A new approach to the shaded picture problem", Proc. ACM National Conference, 1972, 443-450.

[4] M. Wittram, "Hidden line algorithm for scene of high complexity", Computer Aided Design, 13(4), 1981, 187-192.

[5] M. Tamminen, "Hidden lines using the EXCELL method", Computer Graphics Forum, 1(3), 1982, 96-105.

[6] C. Montani, "Region representation: parallel connected stripes", Computer Vision Graphics and Image Processing, 28, 1984, 139-165.

[7] M. P. Wein, P. Tanner, G. Bechtold and N. Burtnyk, "Hidden line removal for vector graphics", SIGGRAPH '78 Proc., published as Computer Graphics, 12(3), 1978, 173-180.

[8] R. D. Merrill, "Representation of contours and regions for efficient computer search", Communications of the ACM, 16(2), 1973, 69-82.

Captions of the Figures

Figure 1: The boundaries of two plygons represented in the picture are coded by the chains A and B. C is the boundary of the hole of B. $A^{\circ}=(6,11)$, $B^{\circ}=(12,2)$ and $C^{\circ}=(11,8)$ are starting points of the chains. The direction of the chains is not significant.

Figure 2: The original picture is cut into stripes of height h. Each stripe is made up of one or more substripes (the "inside" of the polygons).

Figure 3: Possible types of sides (west and east) of a substripe and respective abscissas.

Figure 4: Two and half-dimensional scene.

Figure 5: Output phase for the example of Fig. 4. At each step the faces to be drawn (PCS-2) are differentiated from the silhouette (PCS-1) and displayed. This information is added to the silhouette for the next step.

Figure 6: Scanning of a stripe for raster output.

Figure 7: Vectorial application of the algorithm.

Figure 8: Another vectorial application of the algorithm.

Figure 9: Raster application of the algorithm.

Figure 10: Cyclic overlap between two faces.

Figure 11: Cyclally overlapping polygons.

Captions of the tables

Table 1: Data structure of Fig. 2 represented in digital form.

Table 2: Tests of increasing complexity for the determination of the depth relations.

Table 3: Depth relations between the faces of the example in Fig. 4.

Table 4: Construction of the priority list for our example. At each step the faces for which NMAX=0 or NMIN=0 are inserted in the FRONT or BACK lists, respectively.

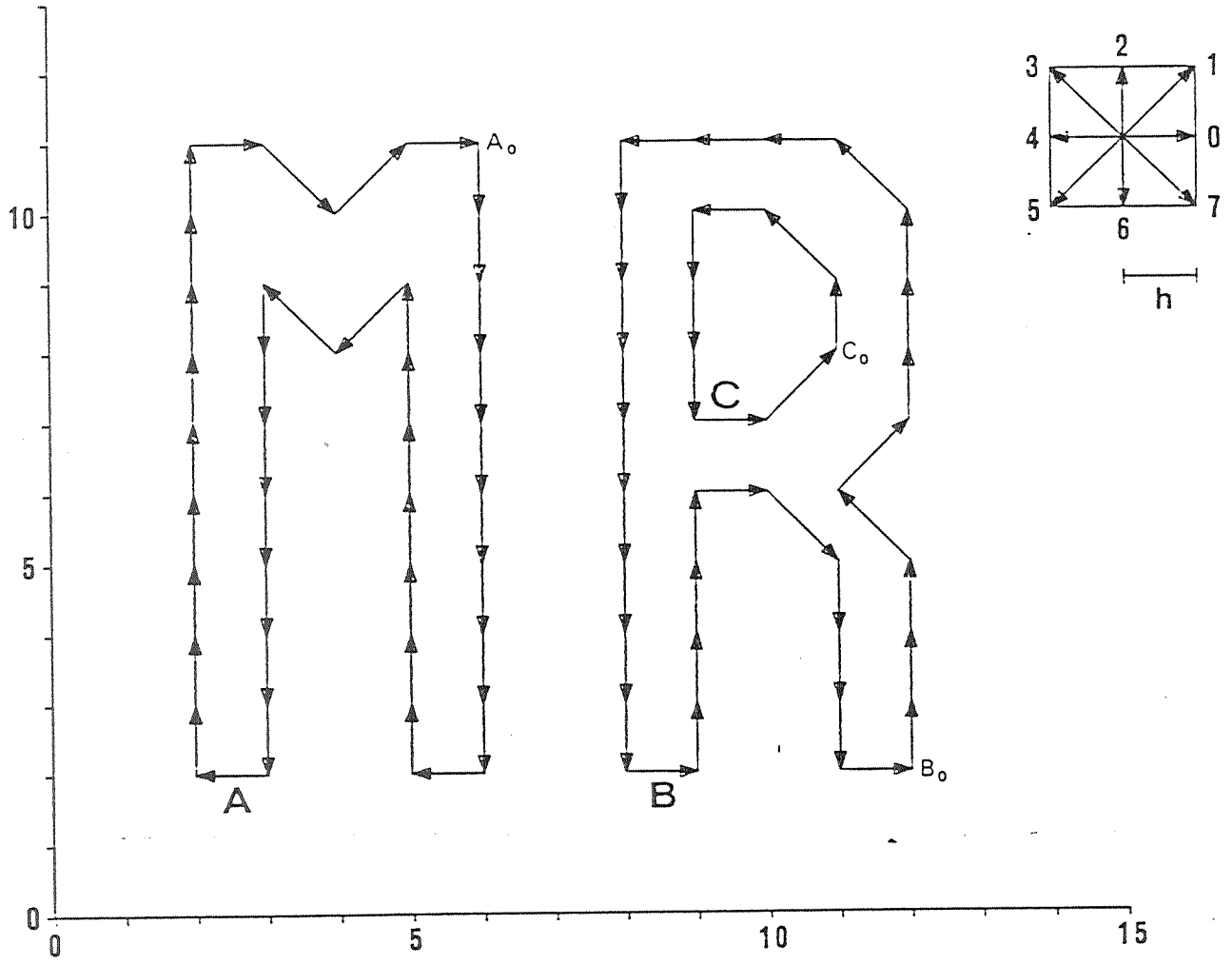

Fig. 1

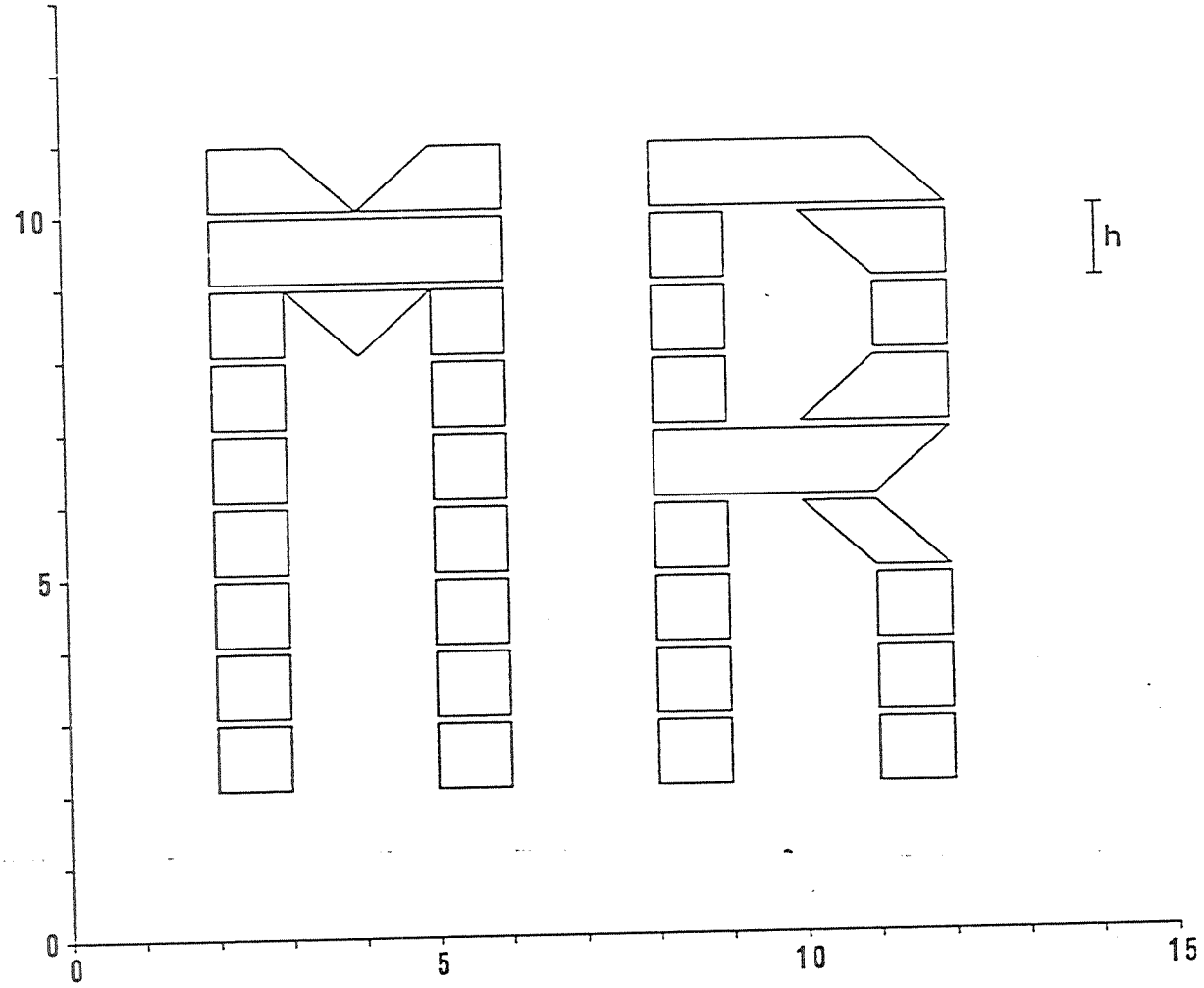

Fig. 2

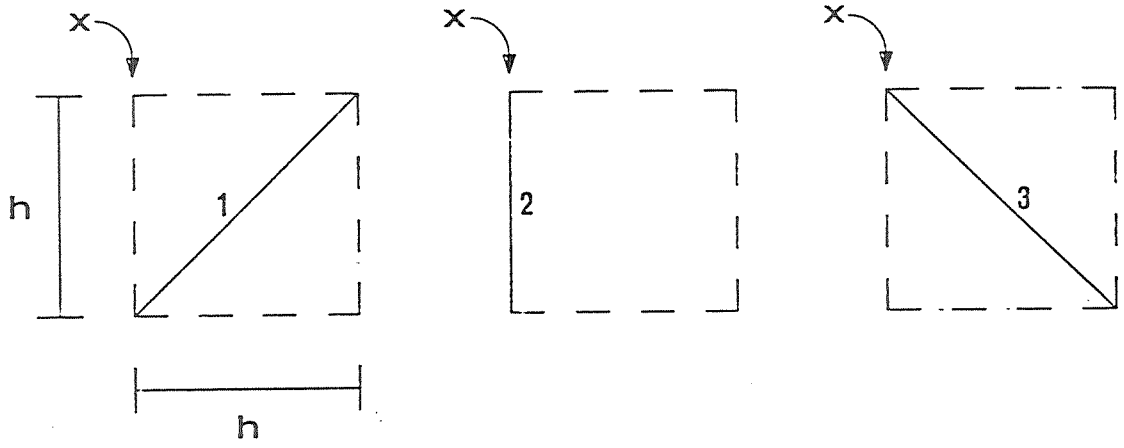

Fig. 3

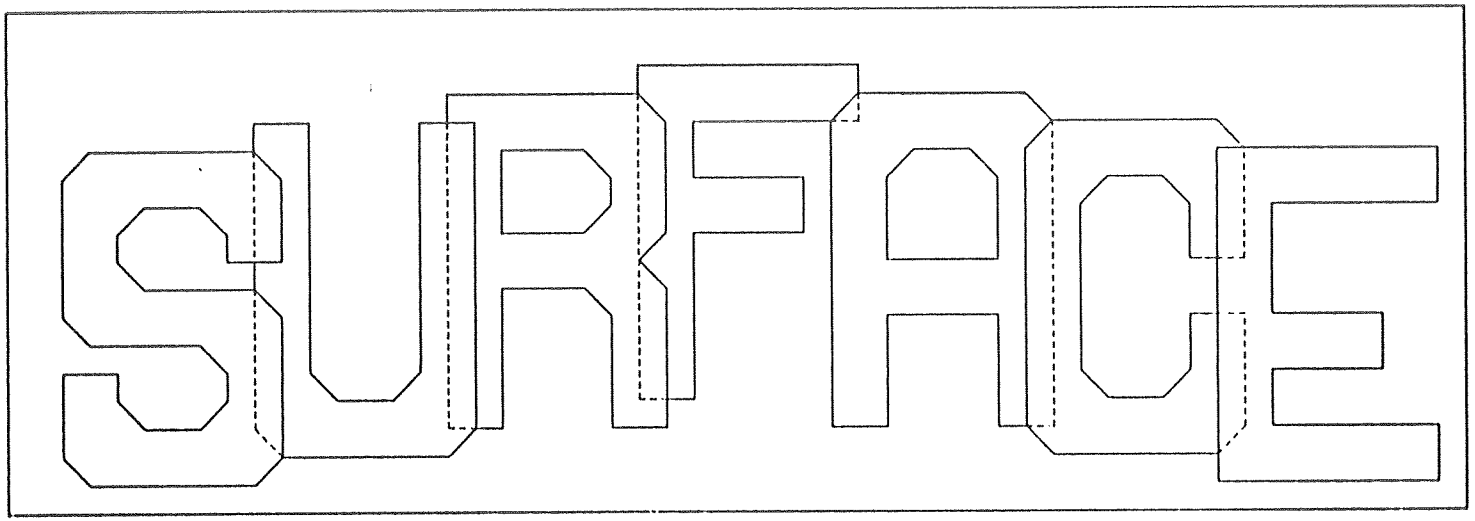

Fig. 4

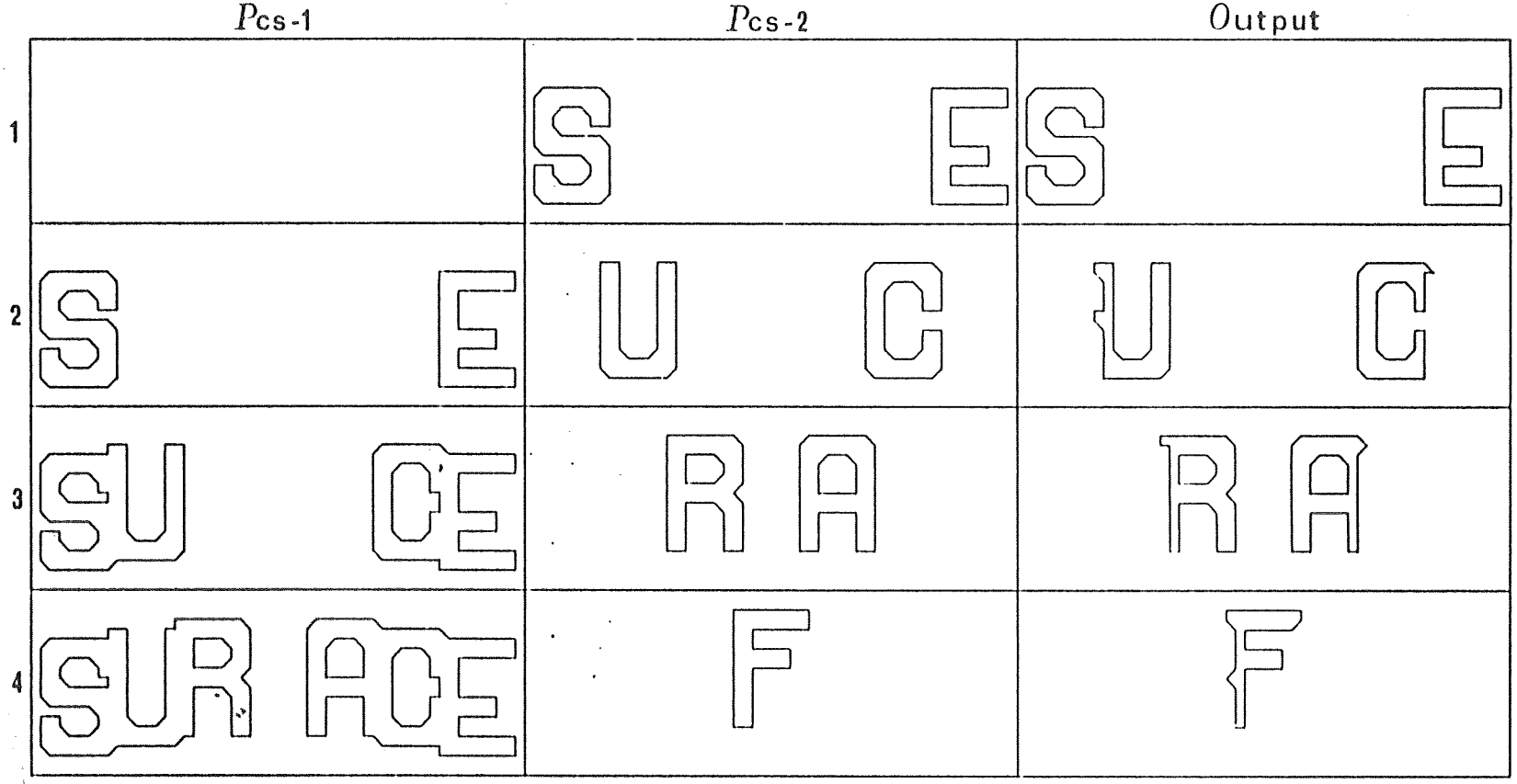

Fig. 5

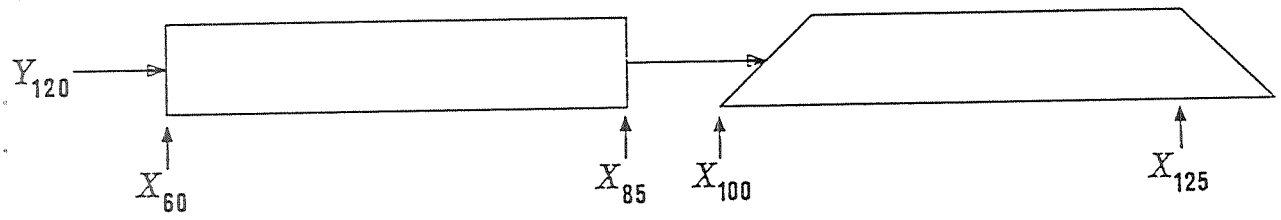

Fig. 6

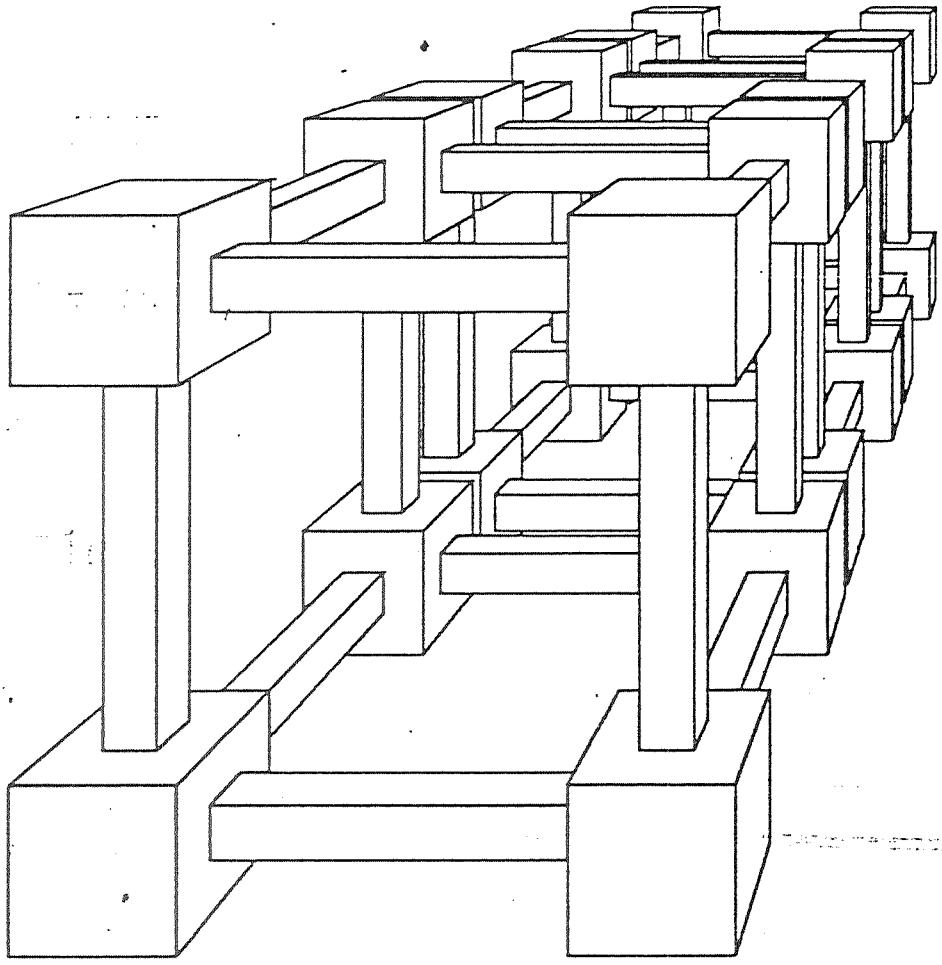

Fig 7

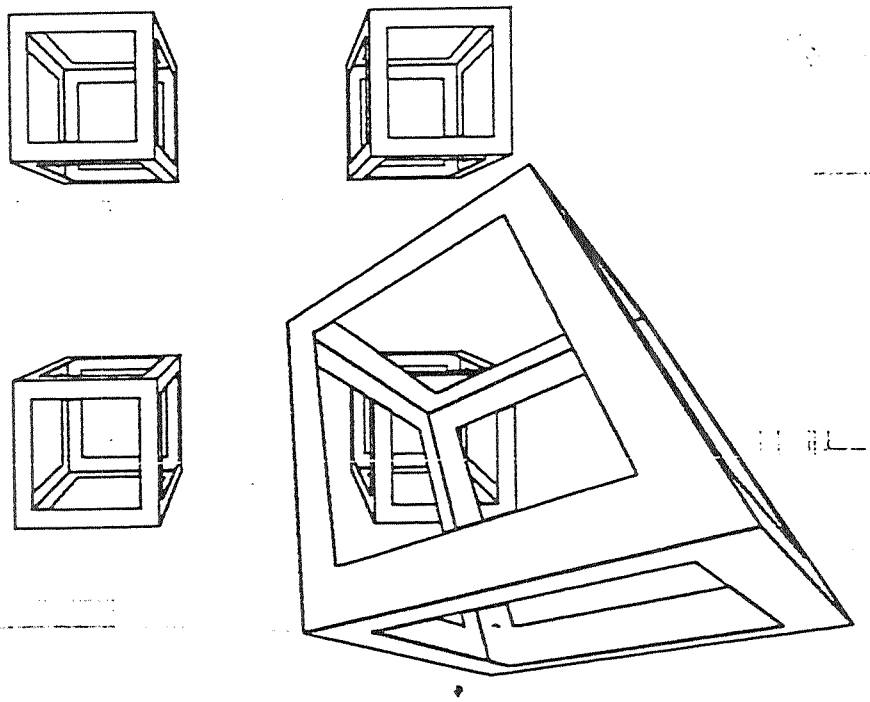

Fig 8

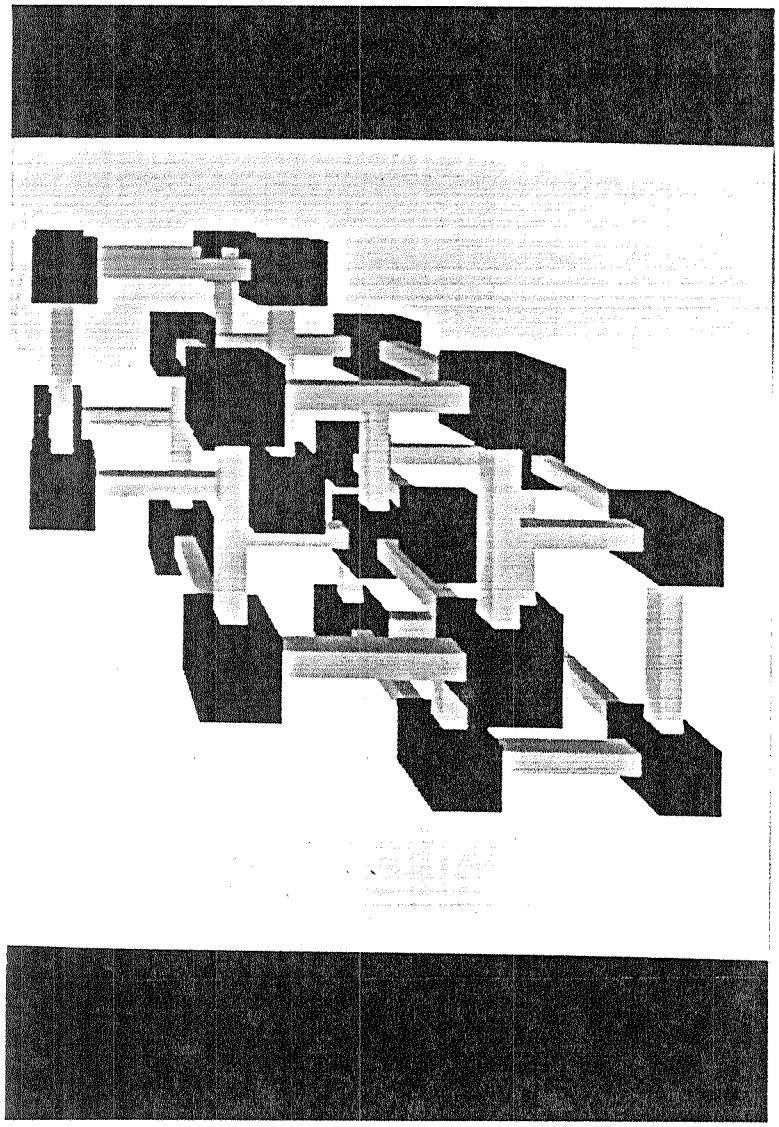

Fig 9

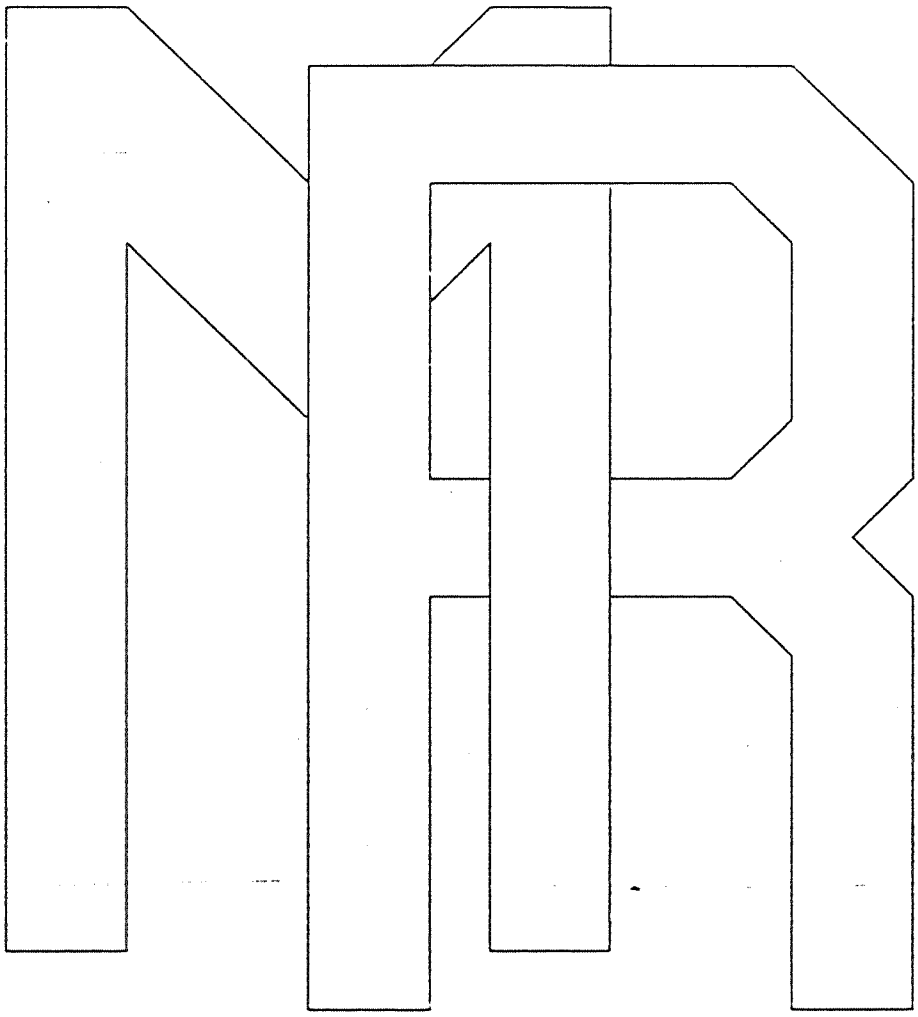

Fig. 10

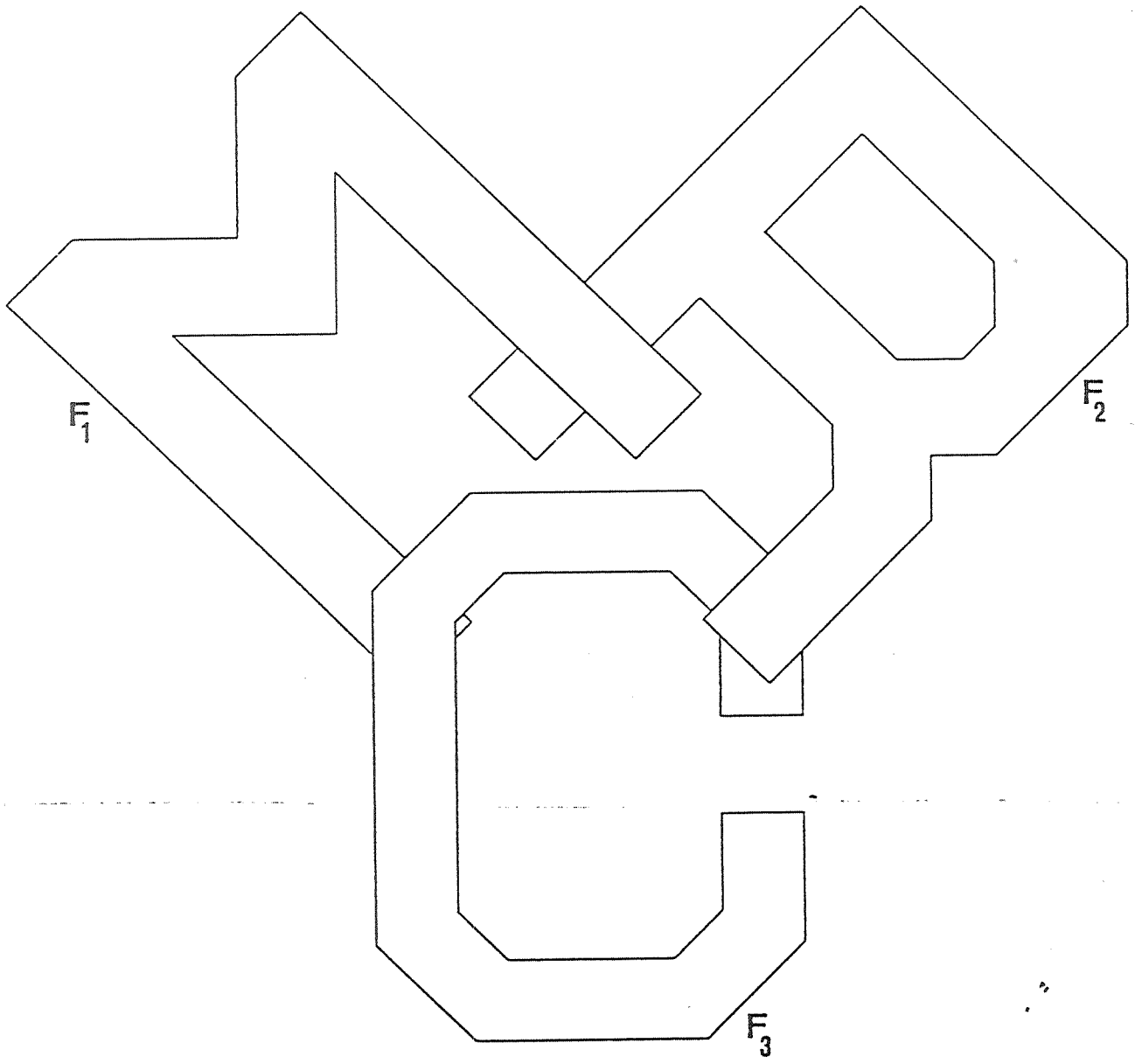

Fig.. 11

TABLE 1

STRIPE		SUBSTRIPE1		SUBSTRIPE2		SUBSTRIPE3		SUBSTRIPE4		SUBSTRIPE5	
Y	N	West Side	East Side	West Side	East Side	West Side	East Side	West Side	East Side	West Side	East Side
10	3	2(2)	3(3)	4(1)	6(2)	8(2)	11(3)				
9	3	2(2)	6(2)	8(2)	9(2)	10(3)	12(2)				
8	5	2(2)	3(2)	3(3)	4(1)	5(2)	6(2)	8(2)	9(2)	11(2)	12(2)
7	4	2(2)	3(2)	5(2)	6(2)	8(2)	9(2)	10(1)	12(2)		
6	3	2(2)	3(2)	5(2)	6(2)	8(2)	11(1)				
5	4	2(2)	3(2)	5(2)	6(2)	8(2)	9(2)	10(3)	11(3)		
4	4	2(2)	3(2)	5(2)	6(2)	8(2)	9(2)	11(2)	12(2)		
3	4	2(2)	3(2)	5(2)	6(2)	8(2)	9(2)	11(2)	12(2)		
2	4	2(2)	3(2)	5(2)	6(2)	8(2)	9(2)	11(2)	12(2)		

TABLE 2

	Test	Depth Relation	Next Compare
1	$X_{max}(F_i) < X_{min}(F_{i+1})$		F_{i+1}, F_{i+2}
2	$Y_{min}(F_i) > Y_{max}(F_{i+1})$ $Y_{max}(F_i) < Y_{min}(F_{i+1})$		F_i, F_{i+2}
3	$Z_{min}(F_i) > Z_{max}(F_{i+1})$	$F_i > F_{i+1}$	
	$Z_{max}(F_i) < Z_{min}(F_{i+1})$	$F_i < F_{i+1}$	F_i, F_{i+2}
4	Face F_i is wholly on that side of the plane of F_{i+1} which is nearer the viewpoint or Face F_{i+1} is wholly on that side of the plane of F_i which is further the viewpoint	$F_i < F_{i+1}$	F_i, F_{i+2}
5	Face F_i is wholly on that side of the plane of F_{i+1} which is further the viewpoint or Face F_{i+1} is wholly on that side of the plane of F_i which is nearer the viewpoint	$F_i > F_{i+1}$	F_i, F_{i+2}
6	The projections of faces F_i and F_{i+1} onto the X-Y plane do not overlap		F_i, F_{i+2}
	The projections of faces F_i and F_{i+1} onto the X-Y plane overlap. Let P be a point belonging to both the projections and $P_i(Z)$, $P_{i+1}(Z)$ be the Z components of P on faces F_i and F_{i+1}, respectively.		
	$P_i(Z) < P_{i+1}(Z)$	$F_i < F_{i+1}$	F_i, F_{i+2}
	$P_i(Z) > P_{i+1}(Z)$	$F_i > F_{i+1}$	F_i, F_{i+2}

TABLE 3

	NMIN	NMAX	S	U	R	F	A	C	E
S	1	0		<					
U	1	1	>		<				
R	1	1		>		<			
F	0	2			>		>		
A	1	1				<		>	
C	1	1					<		>
E	1	0						<	

TABLE 4

	NMIN	NMAX	S	U	R	F	A	C	E
S	1	0		<					
U	1	1	>		<				
R	1	1		>		<			
F	0	2			>		>		
A	1	1				<		>	
C	1	1					<		>
E	1	0						<	

Step 1: FRONT ---> S,E /

BACK ---> F /

	NMIN	NMAX	U	R	A	C
U	1	0		<		
R	0	1	>			
A	0	1			>	
C	1	0		<		

Step 2: FRONT ---> S,E / U,C /

BACK ---> R,A / F /

PRIORITY LIST ---> S,E / U,C / R,A / F /