1982-02-01

A Hidden-Line Algorithm for Hyperspace

Robert P. Burton
rpburton@cs.byu.edu

David R. Smith

Original Publication Citation

Robert P. Burton and David R. Smith, "A Hidden-Line Algorithm for Hyperspace," SIAM Journal on Computing, XI, February 1, 1982, pp. 71-8.

SIAM J. COMPUT.
Vol. 11, No. 1, February 1982

0097-5397/82/1101-0006 $01.00/0

A HIDDEN-LINE ALGORITHM FOR HYPERSPACE*

ROBERT P. BURTON† AND DAVID R. SMITH‡

Abstract. An object-space hidden-line algorithm for higher-dimensional scenes has been designed and implemented. Scenes consist of convex hulls of any dimension, each of which is compared against the edges of all convex hulls not eliminated by a hyperdimensional clipper, a depth test after sorting and a minimax text.

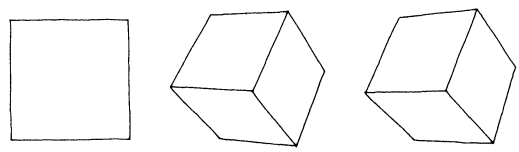
Hidden and visible elements are determined in accordance with the dimensionality of the selected viewing hyperspace. When shape alone is the attribute of interest, hidden-line elimination need be performed only in that hyperspace.

The algorithm is of value in the production of shadows of hyperdimensional models, including but not limited to four-dimensional space-time models, the hyperdimensional elementary catastrophe models and multivariate statistical models.

Key words. hyperspace, hidden-line elimination

Introduction. This paper describes an algorithm for solving the hidden-line problem in hyperspace. The lines are edges of convex hulls approximating the surfaces of hyperdimensional objects. The algorithm removes edges which would be invisible in a hyperdimensional scene. The scene may then be projected to lower dimensions. The development of a hidden-line eliminator for hyperspace is part of an ongoing effort to display and gain insight into the structures of higher-dimensional space. Of particular interest are four-dimensional space-time models, the seven elementary catastrophe models, of which five are hyperdimensional [1] and multivariate statistical models. While these models are numerically useful to some extent, they are of limited general utility in the absence of adequate hyperdimensional presentation techniques. Without an ability to present visible lines only, the four- (and higher-) dimensional analogues of front, rear and depth become hopelessly garbled in the generalized view.

Previous efforts to display hyperobjects [2], [3] have employed techniques which either discard one or more variables or hold them constant so as to restrict structures to three dimensions. Such techniques impose unacceptable constraints. To illustrate by analogy, consider a cube aligned with x-, y- and z-axes. The cube can be restricted to two dimensions by either eliminating or holding constant one of the coordinates. The cube then appears to be nothing more than a square (see Fig. 1a). A generalized technique, which permits the cube to be viewed from any position, with any orientation and in stereo, provides substantially more information, especially when hidden elements of the cube are removed (see Fig. 1b). Similarly, views of hyperobjects are significantly enriched when a generalized viewing capability is combined with a hyperdimensional hidden-line eliminator such as the one described in this paper.



(a) (b)

FIG. 1. (a) *A restricted view of a cube*; (b) *A generalized stereo-pair view of the cube with hidden lines removed.*

* Received by the editors November 6, 1979, and in final form March, 1981.

† Computer Science Department, Brigham Young University, Provo, Utah 84602.

‡ Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

The meanings of *visible* and *hidden*. The development of an algorithm for removing hidden lines is necessarily preceded by a determination of the meanings of *visible* and *hidden*. The definitions offered here accommodate the geometry of space and are hypothesized to accommodate the geometry of hyperspace.

An object J is defined to be *visible* in a viewing space of dimension m to the extent that the points P constituting J or any section(s) of J intersected with the viewing space collectively extend at least into the $m-1$ dimensions of the viewing space orthogonal to the ray of vision and are not hidden. A point P on an object J is defined to be *hidden* from view at V by an object H if and only if a neighborhood of at least dimension $m-1$ exists about an intersection of the line segment VP and H, completely contained in H and extending into each of the $m-1$ dimensions orthogonal to VP (see Fig. 2). Opacity of objects is assumed.

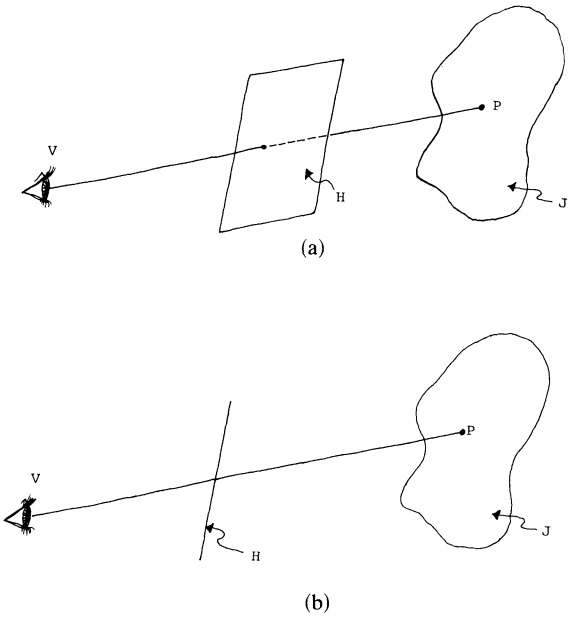


FIG. 2. *Hidden and visible.* (a) *An obscuring intervening object*; (b) *A nonobscuring intervening object.*

Each snapshot of the viewing space involves the viewer at V, the object(s) J which may be partially or completely hidden and the potential hider(s) H. Vision is limited to, but always includes, the $m-1$ dimensions which can be perceived plus depth which can be inferred, which together span the viewing space. Objects possessing additional dimensions become part of the scene to the extent that they intersect it or are projected into it.

Surface approximation. Being piecewise smooth, the surfaces of n-dimensional forms intended for graphic presentation are topologically equivalent to segments of $(n-1)$-space. Thus, the surface of a form in n-space can be approximated by polyhedra of dimension $n-1$. This is an extension of the practice of approximating surfaces of three-dimensional forms with polygonal patches.

Simplification. Both the initial absence of an intuitive feeling and a lack of experience in four and higher dimensions encouraged conceptual simplicity in the design of algorithms for producing, transforming and presenting hyperdimensional scenes. With this in mind, the initial hidden-element algorithms required all surfaces to

be approximated by simplices.[1] The final algorithm was extended to accommodate general convex hulls. The representation of objects as convex hulls provides both a disadvantage and an advantage. The advantage is robustness. Consider a patch in three-dimensional space, supposed to be a quadrilateral, but which actually consists of four nonplanar points. Its manifests itself as a tetrahedron, but nevertheless hides points in a predictable manner. The disadvantage is that concave objects must be built from multiple convex objects. Restriction of the effort to the development of a hidden-*line* algorithm was another obvious step. Attributes other than shape, such as color, for example, were ignored. These restrictions were easily accepted in part because the graphics equipment on which the algorithm was to be implemented was monochromatic and vector oriented.

Simplification was also achieved by collapsing successively from n dimensions to $n-1$ dimensions, and so on, resulting ultimately, for our inspection, in a stereo pair or a single two-dimensional image. The several levels of computation implied by these stages of collapse suggested an object-space algorithm rather than a screen-space algorithm [4]. Furthermore, the algorithm needed to provide output of the same form as its input. After some experience with the algorithm, however, we observed and were able to establish that hidden lines need be eliminated at only one dimensional level prior to projection into R^3 or R^2, making successive application of the algorithm unnecessary. Repeated viewing transformations are still required, nevertheless. Finally, the hidden-line algorithm was simplified by preprocessing the scene with a viewing transformation which can include perspective. The effect of this transformation is to place the viewer at infinity looking in the negative direction along the mth axis with the rays of vision parallel to the mth axis.

The viewing transformation. A generalized viewing transformation in $R^m (m \geqq 2)$ includes:

(1) translation of the scene so that the point to be *looked from* is at the origin 0_0 of object coordinates; and

(2) rotation through the appropriate angles in the planes determined by the axis pairs $(1, m), (2, m), \cdots, (m-1, m)$. Rotation in the (i, m)-plane directs the gaze so as to ultimately place the point to be *looked at* on the viewing axis. Rotation in planes where the axis pair does not include m is implicitly restricted. Simply specifying *look at* and *look from* positions without an order for rotations leaves the scene free to tumble with $\binom{m-1}{2}$ unrestricted degrees of freedom. While this would not affect spatial relationships or the visibleness of elements of the scene, it would significantly affect the orientation of the scene relative to the viewer as well as subsequent projections to lower dimensions.

The rotation scheme based on lexicographic ordering of axis pairs accommodates the human habit of keeping the eyes parallel to the horizon in three-dimensional space; if this third degree of rotational freedom were unrestricted the gaze would remain fixed, but the scene would be free to rotate about the viewing axis. Lexicographic ordering also facilitates easy calculation of the Euler angles. Corresponding advantages are experienced in higher dimensions.

Scenes are projected orthogonally to lower dimensions except possibly during the final projection.

Hidden-line elimination. Hidden-line elimination is carried out by comparing each convex hull H against each edge E to determine which portion of the edge, if any,

[1] An *n-simplex* is a convex hull of $n+1$ affinely independent points. A set of points $\{x_i\}$ is *affinely independent* if and only if for some fixed j the set of points $\{(x_i - x_j) | i \neq j\}$ is linearly independent.

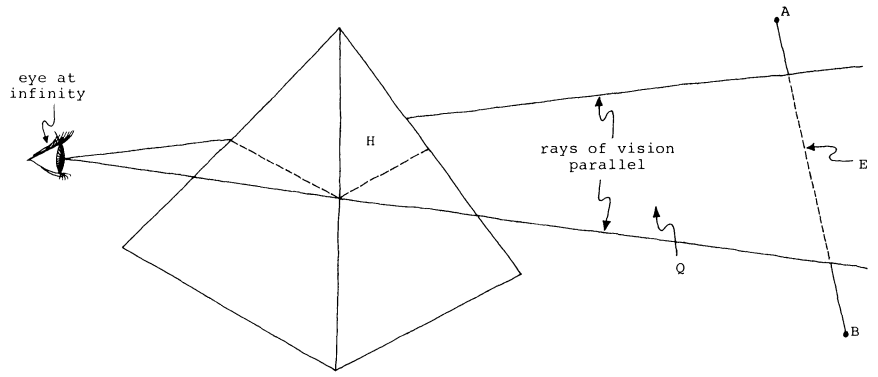


FIG. 3. *A partially obscured edge.*

is hidden by the convex hull. Requiring the hull H to be convex assures that all points of E which H hides are contained in a connected interval.

An edge E with endpoints $A = (a_1, a_2, \cdots, a_m)$ and $B = (b_1, b_2, \cdots, b_m)$ may be hidden at least partially by a convex hull H if H intersects the partial plane of view Q (see Fig. 3). From the criterion of Carathéodory,

$$X \in \textit{convex hull} \text{ of a finite set } A = \{X_i | i = 1, 2, \cdots, m\}$$

if and only if

$$X = \sum_{i=1}^{m} \alpha_i x_i, \quad \text{where } \alpha_i \geqq 0, \quad i = 1, \cdots, m, \quad \sum_{i=1}^{m} \alpha_i = 1.$$

The definition of a convex hull is less restrictive than the definition of an n-simplex. The edge E can be expressed parametrically as

$$E = \{X \in R^m | X = A + (B - A)t, 0 \leqq t \leqq 1\}. \tag{1}$$

The partial plane of view Q can be expressed as the directed sum

$$\begin{aligned} Q &= E + e_m s, \qquad 0 \leqq s \\ &= \{X \in R^m | X = A + (B - A)t + e_m s, 0 \leqq t \leqq 1, 0 \leqq s\}, \end{aligned} \tag{2}$$

where e_m is the mth vector in the natural basis of R^m. Letting $X_i = (x_{1i}, x_{2i}, \cdots, x_{mi})^T$ be the ith vertex of the convex hull, we have

$$H = \left\{ X \in R^m \middle| X = \sum_{i=1}^{m} \begin{bmatrix} x_{1i} \\ \vdots \\ x_{mi} \end{bmatrix} \alpha_i, 0 \leqq \alpha_i, \sum_{i=1}^{n} \alpha_i = 1 \right\}. \tag{3}$$

Intersecting Q and H yields

$$\begin{bmatrix} x_{11} \\ \vdots \\ x_{m1} \end{bmatrix} \alpha_1 + \cdots + \begin{bmatrix} x_{1n} \\ \vdots \\ x_{mn} \end{bmatrix} \alpha_n = A + (B - A)t + e_m s = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} + \begin{bmatrix} b_1 - a_1 \\ \vdots \\ b_m - a_m \end{bmatrix} t + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} s. \tag{4a}$$

Rearranging, we get

$$\text{(4b)}\qquad \begin{bmatrix} a_1-b_1 \\ \vdots \\ a_m-b_m \end{bmatrix} t+\begin{bmatrix} x_{11} \\ \vdots \\ x_{m1} \end{bmatrix} \alpha_1+\cdots+\begin{bmatrix} x_{1n} \\ \vdots \\ x_{mm} \end{bmatrix} \alpha_n+\begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix} s=\begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix}.$$

Incorporating the restriction $\sum_{i=1}^{n} \alpha_i = 1$ yields

$$\text{(4c)}\qquad \begin{bmatrix} a_1-b_1 \\ \vdots \\ a_m-b_m \\ 0 \end{bmatrix} t+\begin{bmatrix} x_{11} \\ \vdots \\ x_{m1} \\ 1 \end{bmatrix} \alpha_1+\cdots+\begin{bmatrix} x_{1n} \\ \vdots \\ x_{mn} \\ 1 \end{bmatrix} \alpha_n+\begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 0 \end{bmatrix} s=\begin{bmatrix} a_1 \\ \vdots \\ a_m \\ 1 \end{bmatrix},$$

which corresponds to

$$\text{(4d)}\qquad \begin{bmatrix} a_1-b_1 & x_{11} & \cdots & x_{1n} & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ & & & & 0 \\ a_m-b_m & x_{m1} & \cdots & & -1 \\ 0 & 1 & \cdots & & 0 \end{bmatrix} \begin{bmatrix} t \\ \alpha_1 \\ \vdots \\ \alpha_n \\ s \end{bmatrix} \begin{bmatrix} a_1 \\ \\ \vdots \\ a_m \\ 1 \end{bmatrix}$$

or, as the adjoint matrix

$$\text{(4e)}\qquad \begin{bmatrix} a_1-b_1 & x_{11} & \cdots & x_{1n} & 0 & a_1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ & & & & 0 & \\ a_m-b_m & x_{m1} & \cdots & x_{mn} & -1 & a_m \\ 0 & 1 & \cdots & 1 & 0 & 1 \end{bmatrix}$$

still restricted by $0 \leqq t \leqq 1$, $0 \leqq s$ and $0 \leqq \alpha_i$.

The portion of E hidden by H can be determined by finding the values of t for which the adjoint matrix has a solution. Since H is convex the solution will be a connected interval.

A problem arises in the elimination process when all convex hulls are compared against all edges. If s is allowed to be zero, a convex hull will eliminate its own edges. One solution might be to avoid testing a convex hull against the edges which bound it. However, convex hulls may hide some of their own edges. Furthermore, an edge may be shared by two or more convex hulls; the task of remembering all the convex hulls which a given edge bounds is cumbersome. A simple solution to the problem requires only slight modification of the adjoint matrix. By adding a small number $\varepsilon > 0$ to the mth coordinates of the endpoints A and B, the edge is moved a distance ε closer to the viewer. The visible or hidden status of the edge is easily determined now because it no longer lies on the surface of the convex hull from which it arose. The modification

appears in the $(m, n+3)$ element of the adjoint matrix:

$$\begin{bmatrix} a_1-b_1 & x_{11} & \cdots & x_{1n} & 0 & a_1 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ a_m-b_m & x_{m1} & \cdots & x_{mn} & 0 & a_m+\varepsilon \\ 0 & 1 & \cdots & 1 & -1 & 1 \\ & & & 0 & & \end{bmatrix}. \tag{4f}$$

The value of ε is suggested by the depth sort information. Experience has indicated that a value of 10^{-5} times the maximum depth works well.

The adjoint matrix is transformed to permit solution for $t_{\min}$ and $t_{\max}$ by the simplex method of linear programming [5]. The adjoint matrix is modified to yield t as a function of s and α_i. One step of Gauss elimination suffices, using an element of column 1 as the pivot for partial pivoting. Row independence is assured by completing the forward Gauss elimination process on the adjoint matrix and ignoring the zero rows remaining at the bottom. If after elimination the last nonzero row has a nonzero element in the adjoint $(n+3)$ column only, the system is inconsistent and there is no solution for t, in which case H does not hide any portion of E.

The adjoint matrix is now in tableau form for the simplex method, except that column 1 is superfluous and can be ignored. The value of t is the element $(1, n+3)$ of the adjoint matrix. The simplex method transforms the matrix to yield the values of $t_{\min}$ and $t_{\max}$. Although details of the procedure are not presented here, it should be noted that the initial basic feasible solution is found using artificial variables. A special loop in effect accomplishes Gauss elimination by pivoting on the artificial variables without expanding the matrix to contain them explicitly. A value for *Big M* [6, p. 63], the arbitrary large number needed for the initial basic feasible solution, can be determined during the depth sort which is discussed below. Experience indicates that good results can be obtained with *Big M* equal to 10^2 times the maximum depth. Although the simplex method enforces the restrictions $0 \leqq s$ and $0 \leqq a_i$, it does not restrict the range of t. Therefore, the interval $[t_{\min}, t_{\max}]$ must be intersected with the interval $[0, 1]$. If t is unbounded, then E is parallel to e_m; the projected image of E is a single point and can be ignored.

Depth and minimax tests. By performing a depth test and a minimax test, the situation can often be resolved without employing the adjoint matrix described above. Edges determined from coordinates of vertices and lists of vertices comprising convex hulls are entered into a hash table which is heap sorted into a linear table implicitly eliminating shared or otherwise redundant edges. Edges are sorted according to depth so that the edge with the greatest depth appears first in the list. The depth of each convex hull is compared against the depth of each edge. If both endpoints of the edge are closer than the convex hull to the viewer, the convex hull cannot possibly obscure any part of the edge. Furthermore, it cannot obscure any of the subsequent edges since edges are listed in order of decreasing depth. When the depth test fails, a minimax test is applied to the other $m-1$ dimensions to identify cases where the convex hull cannot hide the edge, i.e., cases where the ith coordinate values of the edge entirely exceed or entirely fall short of the ith coordinate values of the convex hull.

Partially obscured edges. If a convex hull hides a middle portion of an edge, the edge is divided into two segments. To avoid entering the resultant new endpoints into the vertex array and to avoid additional edge definitions, the hidden intervals of an edge

are placed in a singly-linked list. Each node in this list gives the minimum and maximum values of t ($0 \leqq t \leqq 1$) in (1) for which part of the edge is obscured. The use of linked lists saves computer time since all separate, visible segments of an edge can be checked simultaneously against possible obscuring convex hulls. This would be impossible if new endpoints were calculated and additional edges generated. Within the list, obscured portions of an edge which are found to overlap or nearly abut are combined to minimize the number of linked nodes. If the combined interval is [0, 1], the edge is completely hidden; the associated node is subsequently returned to the pool of available storage.

Clipping. Clipping in hyperspace is conceptually simple, being an extension of three-dimensional clipping [7]. Similar to its analogue in three dimensions, the hyper-pyramid of vision in R^4, for example, imposes the following constraints:

$$-w \leqq c_x x \leqq w,$$

$$-w \leqq c_y y \leqq w,$$

$$-w \leqq c_z z \leqq w,$$

where each of the c_x, c_y and c_z represents the cotangent of half the angle of vision in the given direction.

Even though the introduction of new points into the point array should be avoided, a new endpoint must be generated for an edge which lies partly within the field of vision, but crosses the zero-depth plane. In other cases it is sufficient to treat a clipped edge as a partially obscured edge, using the linked list. While this approach saves storage, it necessitates modification by the perspective routine of the t-parameters in the hidden-segment linked list [7].

The convex hulls should be processed by an object clipper. However, a convex hull totally within or outside the field of vision poses no problems. Even if a convex hull lies partly within the field of vision it need not be clipped in most cases; only the edges, not the convex hull itself, are ever visible. However, when a partially visible convex hull crosses the zero-depth hyperplane, division by depth in the perspective transformation becomes troublesome [7]. In such cases the convex hull must be clipped to a hyperplane slightly in front of the viewer. The convex hull is clipped by regenerating the edges and clipping them against this limiting hyperplane. Intersection points are entered into the hash table. The description of the convex hull now includes the hyperplane intersection points, but excludes vertices on the viewer's side of the clipping hyperplane. The hash table is placed in vacated storage locations in the vertex array, permitting all points to be treated uniformly.

The key to the hash table can be the sum of the vertex's first $m-1$ (for R^m) coordinates, multiplied by a constant [8]. Even though most transformations will cause a vertex to generate a key which differs from the one by which it was entered into the table, the objective of avoiding redundant points is realized at each dimensional level of clipping.

Classification and performance. The hidden-line algorithm presented in this paper is an object-space algorithm. The algorithm is closely related to Roberts' algorithm [9], particularly because it sweeps the area from an edge out to infinity. Basic relationships are formulated as matrix systems which are solved by standard methods.

As is true for Roberts' algorithm, computation required is roughly proportional to the square of the complexity of the scene. Minimization of computation is heavily dependent upon the ability of the preliminary tests to resolve the situation, thereby avoiding the need for the matrix solution. Experience with three-dimensional scenes

having from 400 to 1400 triangles with 700 to 2150 edges yields execution times (transformation, clipping, sorting, hidden-line elimination and plot file generation all divided by the product of the number of edges and triangles) of 83 μsec to 130 μsec to test one triangle against one edge, running on a DECsystem-1070. Failures of the depth test and minimax test result in higher execution times.

Discussion. The algorithm presented in this paper performs hidden-line elimination in R^n ($n \geqq 3$) and differs from conventional hidden-line elimination algorithms in its hyperdimensional capabilities. When an image is projected into R^3 or R^2 after hyperdimensional hidden-line elimination, information is preserved which would be lost if hidden-line elimination were performed in R^3 using conventional algorithms.

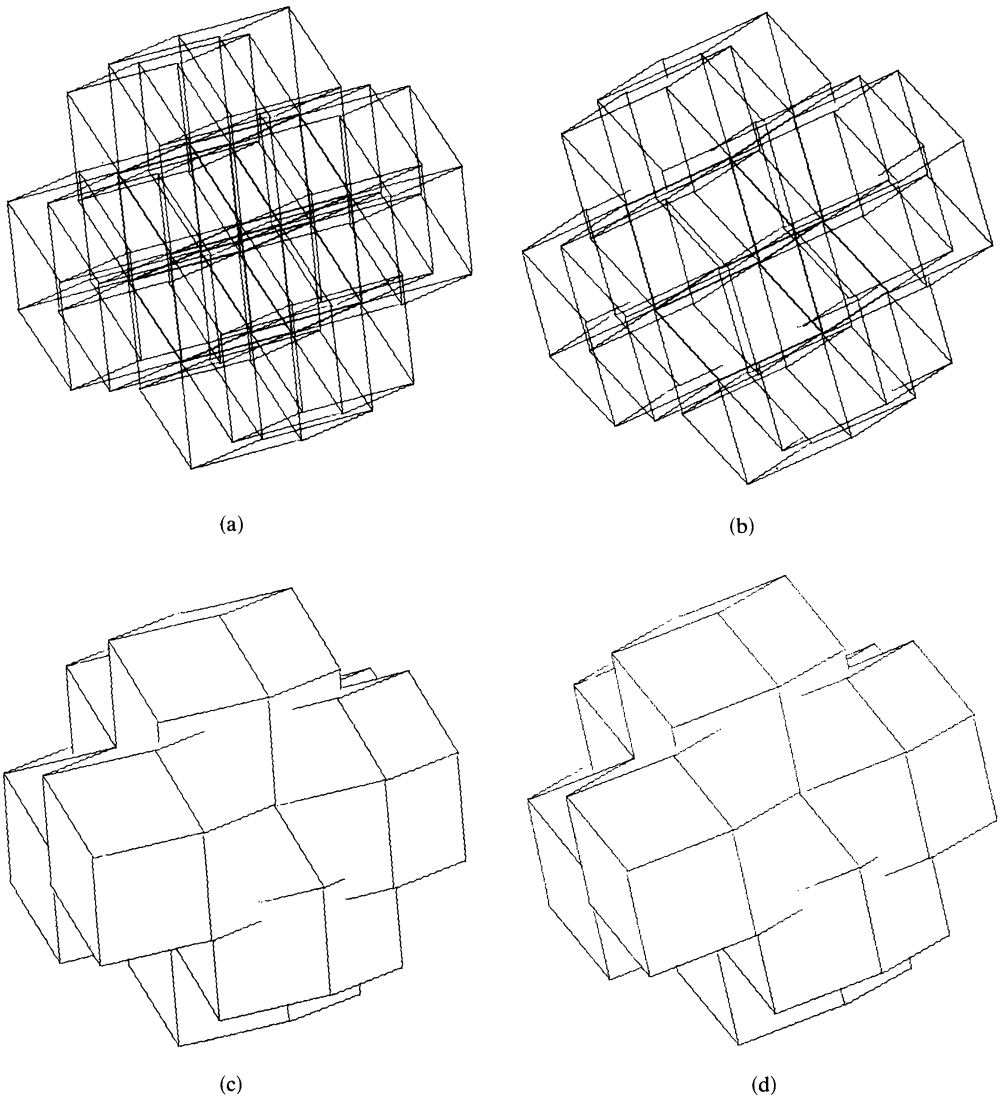


FIG. 4. *A generalized view of tesseracts (hypercubes) on each of the eight hyperfaces of a central tesseract.* (a) *No hidden-line elimination.* (b) *Four-dimensional hidden-line elimination.* (c) *Four-dimensional hidden-line elimination followed by three-dimensional hidden-line elimination.* (d) *Three-dimensional hidden-line elimination only.*

The information which is preserved is that which would be visible from hyperspace in a three- or two-dimensional projection of hyperspace.[2] Figure 4a shows a generalized view of a four-dimensional object with no hidden lines removed, projected into R^2; Fig. 4b shows the same object after four-dimensional hidden-line elimination and subsequent projection into R^2. Information is preserved in Fig. 4b which would have been lost had the object first been projected into R^3, followed by hidden-line elimination, followed by projection into R^2, as in Fig. 4d.

Hidden-line elimination need be performed only once when shape alone is the attribute of interest. Any subsequent hidden-line elimination would obliterate the results of all previous hidden-line elimination and alone would yield a shape equivalent to the aggregate shape that would result from successive applications of the algorithm from higher dimensions. By way of illustration, consider Fig. 4c, which shows the results of four-dimensional hidden-line elimination followed by three-dimensional hidden-line elimination. Figure 4d shows the results of three-dimensional hidden-line elimination alone.[3] The apparent equivalence of Figs. 4c and 4d can be explained as follows.

Assume that a point P is hidden from a viewer at V by a hypervolume H in four-dimensional space (see Fig. 5). The line segment VP intersects H in one point R provided only that VP and H are not contained in the same three-dimensional

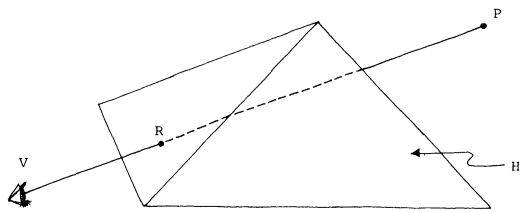


FIG. 5. *A point hidden in R^4 is hidden in R^3.*

hyperplane. Since the collapse from four to three dimensions takes place along the axis which contains V, R and P, the points R and P coincide after projection into R^3. P is simply absorbed into H. The shape of H is unchanged. In the case under consideration here, three-dimensional hidden-element elimination leaves no clue that four-dimensional hidden-element elimination took place previously. Attributes such as color would persist, however, unless hidden-element elimination took place in each successive dimension. When shape is the only attribute of interest, a significant savings is realized because the costly process of removing hidden lines need be performed only once. Computation is also minimized because the space in which hidden-line elimination is performed is dimensionally the least of all the spaces in which hidden-line elimination would be performed if successive application of the algorithm were necessary.

Acknowledgments. The authors gratefully acknowledge the consultation and contributions of other members of the Hyperspace Research Group. Appreciation for their support is extended to Brigham Young University and Eyring Research Institute.

[2] In the same sense, a photograph represents hidden-element elimination performed in R^3 and subsequent projection to R^2 to be viewed from R^3.

[3] The algorithm would never be used in R^3; a variety of superior algorithms could be summoned for that purpose. An illustration involving R^3 is chosen to simplify the presentation of the concept.

REFERENCES

[1] E. ZEEMAN, *Catastrophe theory*, Scientific American, 234 (April, 1976), pp. 65–83.
[2] C. PANATI, *Catastrophe theory*, Newsweek, 83 (January 19, 1976), p. 55.
[3] J. GORMAN, *The shape of change*, The Sciences, (September/October 1976), p. 21.
[4] I. SUTHERLAND, R. SPROULL AND R. SCHUMACKER, *A characterization of ten hidden-surface algorithms*, ACM Computing Surveys, 6 (1974), pp. 1–55.
[5] D. STEINBERG, *Computational Matrix Algebra*, McGraw-Hill, New York, 1974.
[6] F. HILLIER AND G. LIEBERMAN, *Operations Research*, Holden-Day, San Francisco, 1974.
[7] W. NEWMAN AND R. SPROULL, *Principles of Interactive Computer Graphics*, 2nd ed., McGraw-Hill, New York, 1979.
[8] M. STEPHENSON AND H. CHRISTIANSEN, *A polyhedron clipping and capping algorithm and a display system for three-dimensional finite element models*, Computer Graphics, 9 (Fall 1975), pp. 1–16.
[9] L. ROBERTS, *Machine perception of three dimensional solids*, MIT Lincoln Laboratory TR 315, 1963.