AD/A-002 736

3-D DESIGN OF FREE-FORM B-SPLINE SURFACES

James Henry Clark

Utah University

Prepared for:

Defense Advanced Research Projects Agency

September 1974

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER UTEC-CSc-74-120	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD/A-002736
4. TITLE (and Subtitle) 3-D DESIGN OF FREE-FORM B-SPLINE SURFACES		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL REPORT
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) JAMES HENRY CLARK		8. CONTRACT OR GRANT NUMBER(s) DAHC15-73-C-0363
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department University of Utah Salt Lake City, Utah 84112		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order #2477
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, Virginia 22209		12. REPORT DATE September 1974
		13. NUMBER OF PAGES 82
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer-aided design, B-Splines, splines, surface design, 3-D graphics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes an experimental system for designing free-form B-spline surfaces using a head-mounted display. In this system, the interaction with the surfaces takes place in three dimensions as the designed object's shape is updated in real-time. The report also examines some of the problems that should be solved in building a practical three-dimensional computer-aided geometric design system for surfaces.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

3-D DESIGN OF FREE-FORM B-SPLINE SURFACES

by

James Henry Clark

September 1974 UTEC-CSc-74-120

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. DAHC15-73-C-0363 and Contract No. F30602-70-C-0300.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT[†]

This report describes an experimental system for designing free-form B-spline surfaces using a head-mounted display. In this system, the interaction with the surfaces takes place in three dimensions as the designed object's shape is updated in real-time. The report also examines some of the problems that should be solved in building a practical three-dimensional computer-aided geometric design system for surfaces.

[†]This report reproduces a dissertation of the same title submitted to the department of Computer Science, University of Utah, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

CHAPTER I

INTRODUCTION

Typically, three-dimensional computer-aided geometric design (CAD) systems have primarily been experiments on the feasibility of a particular mathematical representation like the Coons patch or Bézier patch [1,2,3,4,5,22]. This study focuses primarily on the problems of 3-D interaction in a three-dimensional environment using a head-mounted display and 3-D wand. The particular mathematical formulation on which much of this study is based is the B-spline (for Basis-spline), which was first proposed in a computer-aided geometric design context by R. F. Riesenfeld[6].

This work has been done with three global objectives in mind. The first objective was that the interaction with the 3-D surfaces actually take place in three dimensions rather than with various two dimensional orthographic and perspective projections. The devices used to accomplish this objective were a head-mounted display and a 3-D wand. The head-mounted display was built by Ivan E. Sutherland and coworkers at

Harvard Univereity[7]. This display was moved to the University of Utah when Ivan Sutherland joined the faculty here.

Thie 3-D interaction objective also required that a device be available that allows the user to communicate the geometric poeitional information to the system. A 3-D wand to allow this type of interaction wae constructed at the University of Utah and used in this system.

The second objective in building this system was that it be very close to real-time in response. The system should appear to the user to respond instantaneously, or at least delays in response should be no greater than one or two seconds in the woret case and far less in most cases. In my opinion, thle requirement is a very important one to impose on a computer-aided design system. When we sculpture an object in clay we always get Immediate response. To be forced to wait four or five seconds to see the results of deformations made to an object can be very frustrating, as anyone who has used a CAD system that responds in this way knows. The user can lose sight of creative ideas during these waiting periods. Therefore, In thls system I have attempted to shorten the computatlon algorithms for the B-spline surfaces and leave as much of the computation as possible to a special purpose graphics processor.

The third objective of this system was to provide a mathematical formulation for the user that requires little or no mathematical background on his part in order that he be able to use it effectively. This restriction seems easy to satisfy. However, when coupled with the requirement that the formulation also satisfy the physical needs of derivative continuity and the subjective attribute of "fairness", which is important from a designer's point of view, the requirement is more difficult to satisfy. All of these things should be present in the mathematical representation with no explicit intervention by the designer.

It is with this third objective that B-splines become important, for they intrinsically yield derivative continuity across patch boundaries. In fact, with these functions, one must explicitly introduce extra definition points to produce a breakdown in continuity. This is in contrast to Coons patch systems, e.g. Armit's Multipatch [4] and Multiobject [5] systems, or Bézier's Systéme UNISURF [2] , in which all derivative continuity across patch boundaries must be explicitly dealt with by the designer. Moreover, B-spline surfaces are locally defined. This means, for example, that changes in the design of the fender of a car do not change the hood shape, or modifying a nose on a bust being sculptured does not affect the shape of the mouth. These considerations are extremely important to a tractable three dimensional design environment.

As suggested in the three objectives mentioned above, the main goal of this research has been to design free-form basis spline surfaces, in particular B-spline surfaces, in a 3-D environment. The real-time requirement has made necessary that a wide variety of special purpose display and digitization hardware be available. Some of this equipment already existed when the work began, but some of it was acquired or built to make this research possible.

CHAPTER II
MATHEMATICAL FORMULATIONS

II.1 Requirements of a Mathematical Representation

One of the principal problems in computer aided geometric design is the representation of shape information in the computer. This means that we are more concerned with shapes than with functions of the form $y=f(x)$. Representations as functions of the form $y=f(x)$ have a number of properties that are undesireable in a CAD system, the worst of which is that for general kinds of shapes they are multivalued and often have infinite slopes. Large or infinite slopes are axis dependent. What is needed for shape descriptions is an axis independent representation. The representation should also be easy to input and to output to a display. Also, the internal details of the representation should not be of concern to the designer.

Because of the inherent difficulties in representing shapes with this type of functional representation, Coons[1], Forrest[8], Bézier[2], Gordon[9], and others have chosen a parametric vector-valued representation to represent the designed shapes. A curve in 3-space is of the form

$$f(t) = [\, x(t),\ y(t),\ z(t)\,],$$

where t is a parameter that varies between 0 and 1. This formulation is axis independent.

Just as with curves, surfaces (or volumes, etc.) can be represented in a number of different forms. The form $z=f(x,y)$ is, however, unsuited to the needs of geometric CAD. A form that is suitable for this class of problems is

$$f(s,t) = [\, x(s,t),\ y(s,t),\ z(s,t)\,],$$

where both s and t are parameters that vary between 0 and 1.

Prior to the period in which Coons and others did their initial work with surfaces for geometric CAD, doubly curved surfaces were avoided whenever possible in design systems because of the difficulties in representing these surfaces by plane projections and because of the cost of manufacturing them. When complex fillet surfaces were needed to blend portions of castings together, the job of interpreting them was left to the pattern maker. In the aircraft industry where the shape of the surface is critical for aerodynamic reasons, a technique known as "lofting" was used. This lofting procedure was carried out by specifying families of mathematical curves at a number of parallel plane sections and interpolating a eurface through these curve sections. This technique of course breaks down when the surface is complex enough to prevent definition by plans parallel curves.

II.2 Coons and Bézier Formulations.

Coone Patchee.

One of the earliest attempts to use the computer in geometric computer-aided design resulted from investigations into surface representations by Steven A. Coons at M.I.T. In his report[1], he describes a technique for blending the boundary curves of patches together in a way that ensures derivative and positional continuity under conditions that can easily be specified. His work was used as the basis for computer aided design systems by Armit[4], Ferguson[5], and Peters[22].

The method of surface description developed by Coons coneiets of building up a piecewise continuous surface by aesembling together surface patches. It is an interpolation approach becauee each patch is defined by a bivariate Hermite type interpolation to boundary conditions that consist of functions of a single variable. Each patch is specified by four boundary curves and possibly higher order conditions on these boundaries. The only restriction on these boundary curves is the "compatibility constraint" that they intersect at four corners. They are not restricted to be planar curves. A patch side may even be degenerate, thus allowing asymmetric triangular patches. Also, patches may be split so that complexity is introduced only where the shape is complex.

The following discussion of the Coons formulation closely follows Forrest[10]. In this discussion, Q's are used to denote the boundary curves that define the surface and P's are used to denote the defined surface.

The boundary curves of a Coons patch are denoted by Q(0,v), Q(1,v), Q(u,0) and Q(u,1). They intersect at the points Q(0,0), Q(0,1), Q(1,0) and Q(1,1) (see Figure 2.1). Using an abbreviated notation, we let i=0 or 1 and j=0 or 1. The boundary curves are then represented by Q(i,v) and Q(u,j) and the four corners by Q(i,j). The cross boundary(tangent) slopes are represented by $Q_u(i,v)$ and $Q_v(u,j)$.

We can now construct the canonical form of a Coons patch satisfying the boundary conditions Q(i,v), $Q_u(i,v)$, Q(u,j) and $Q_v(u,j)$:

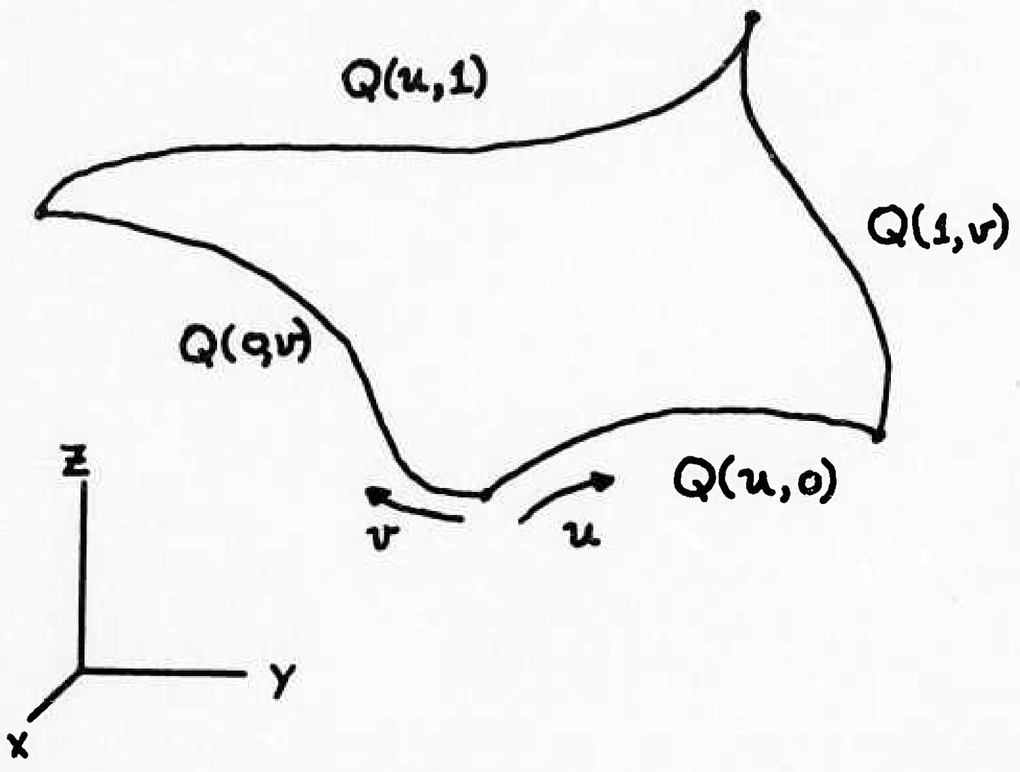


Figure 2.1 Boundary Curves for a Coons Patch

$$\begin{aligned} P(u,v) = {} & Q(i,v)\ f_i(u) + Q_u(i,v)\ g_i(u) \\ & + Q(u,j)\ f_j(u) + Q_v(u,j)\ g_j(v) \\ & - Q(i,j)\ f_i(u)\ f_j(v) - Q_u(i,j)\ g_i(u)\ f_j(v) \\ & - Q_v(i,j)\ f_i(u)\ g_j(v) - Q_{uv}(i,j)\ g_i(u)\ g_j(v). \end{aligned} \qquad (2.1)$$

The f_i's are functions introduced to blend the boundary curves together and the g_i's blend together the cross boundary slopes. The form for the f and g blending functions that gives slope continuity for the patches is

$$\begin{aligned} f_0(t) &= 1 - 3t^2 + 2t^3, \\ f_1(t) &= 3t^2 - 2t^3, \\ g_0(t) &= t - 2t^2 + t^3, \\ \text{and}\quad g_1(t) &= -t^2 + t^3. \end{aligned} \qquad (2.2)$$

The $Q_{uv}(i,j)$ terms in (2.1) are the cross partial derivatives at the four corners. Coons calls these the "twist vector" terms. They eliminate unwanted quasi-flat regions at the corners of a patch.

The main advantage of the Coons patch of (2.1) is that it is extremely general. The boundary curves may be of any form whatever. The Coons patch can therefore be joined to a previously defined surface very easily so long as the curve defining the boundary of the surface is parameterized.

The main disadvantage of the form of (2.1) is the inclusion of the twist vector term. These terms, representing the cross partial derivative of the surface with respect to the two parameters at the corners, are difficult for even the mathematician to use. Of course if the twist vector terms are not explicitly dealt with by the designer in a system, that is if the system keeps them hidden from the user, then they present no special problem aside from the computation.

Bézier Patches.

P. Bézier of Regie Renault in Paris has developed a system for curve and surface represtentation[11]. It is not quite as general as the Coons method, but it does not require that the user have as detailed a knowledge of the formulation as with the Coons formulation. Some people consider this feature an advantage. Arguments for this point of view will be discussed in section II.5. Bézier methods are discussed here because they form the basis of a system that is actually used to design automobiles at Renault and because of their relation to B-splines.

A Bézier space curve is a vector-valued polynomial approximation to a polygon, or sequence of points, v_0, v_1,..., v_m, of the form

$$B_m(s) = f_0(s)\ v_0 + f_1(s)\ v_1 + \ldots + f_m(s)\ v_m, \qquad (2.3)$$

where the $f_j(s)$ turn out to be the binomial probability density functions (Bernstein polynomials[9]).

Figure 2.2 shows the binomial basis functions for $m=5$. Note that since all of the functions except f_0 are zero when $s=0$, the curve interpolates, or passes through, the point v_0. Likewise, since all except f_m are zero when $s=1$, the curve interpolates at v_m. All other points control the global shape of the curve in the amount of the weights they are given by their respective basis functions.

An illustration of a Bézier curve is shown in Figure 2.3. An important feature of this type of curve is that it lies entirely within the convex hull of the points. The curve follows the global shape of the "control polygon", yet it is much smoother.

A surface can be generated by a two dimensional array of control points, using for the weighting functions the tensor product of the univariate weighting(or basis) functions. The tensor product basis functions are

$$f_{j,k}(s,t) = f_j(s)\ f_k(t), \qquad (2.4)$$

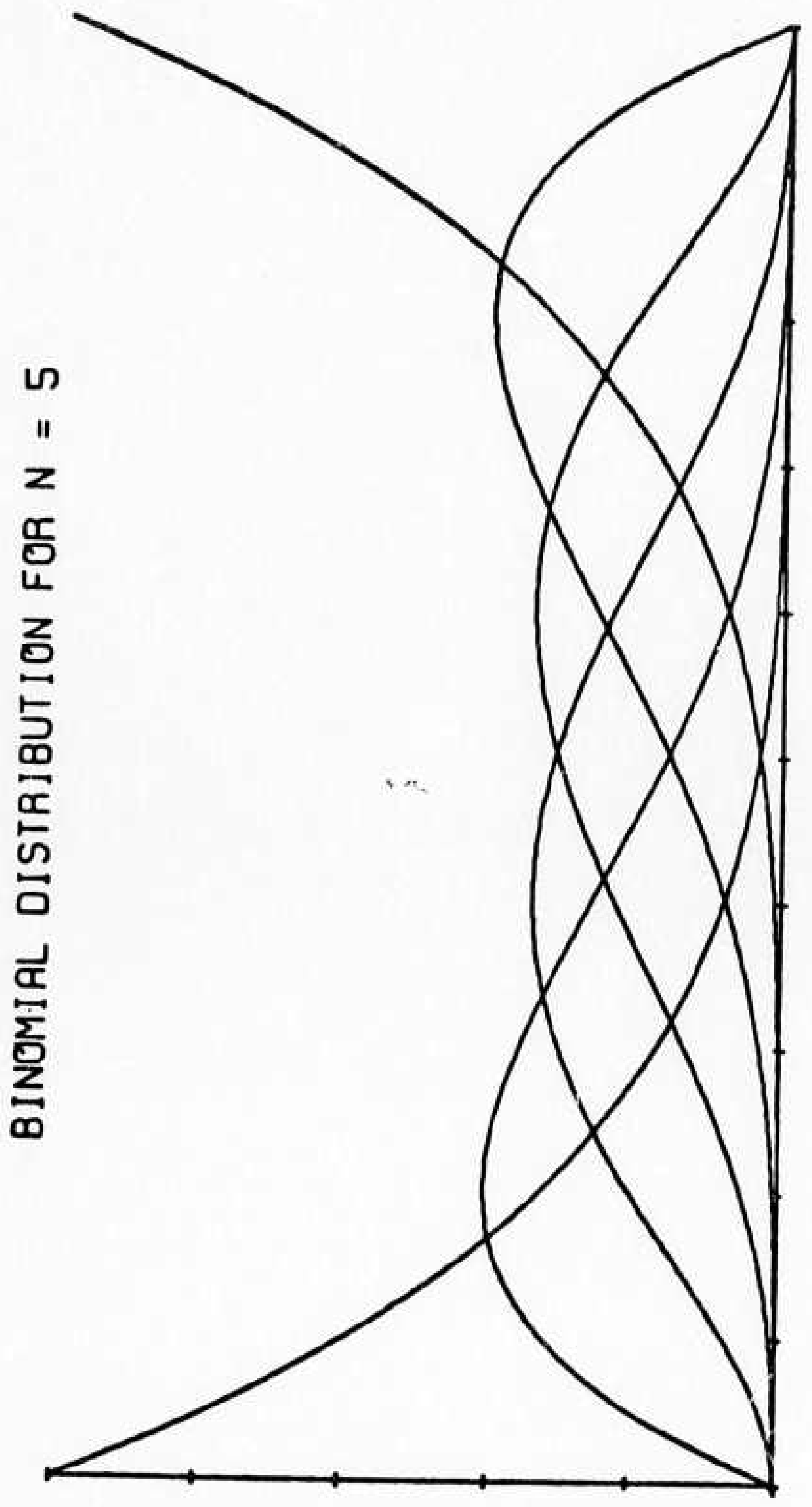


Figure 2.2 Bernstein Basis for Polynomials of degree 5.

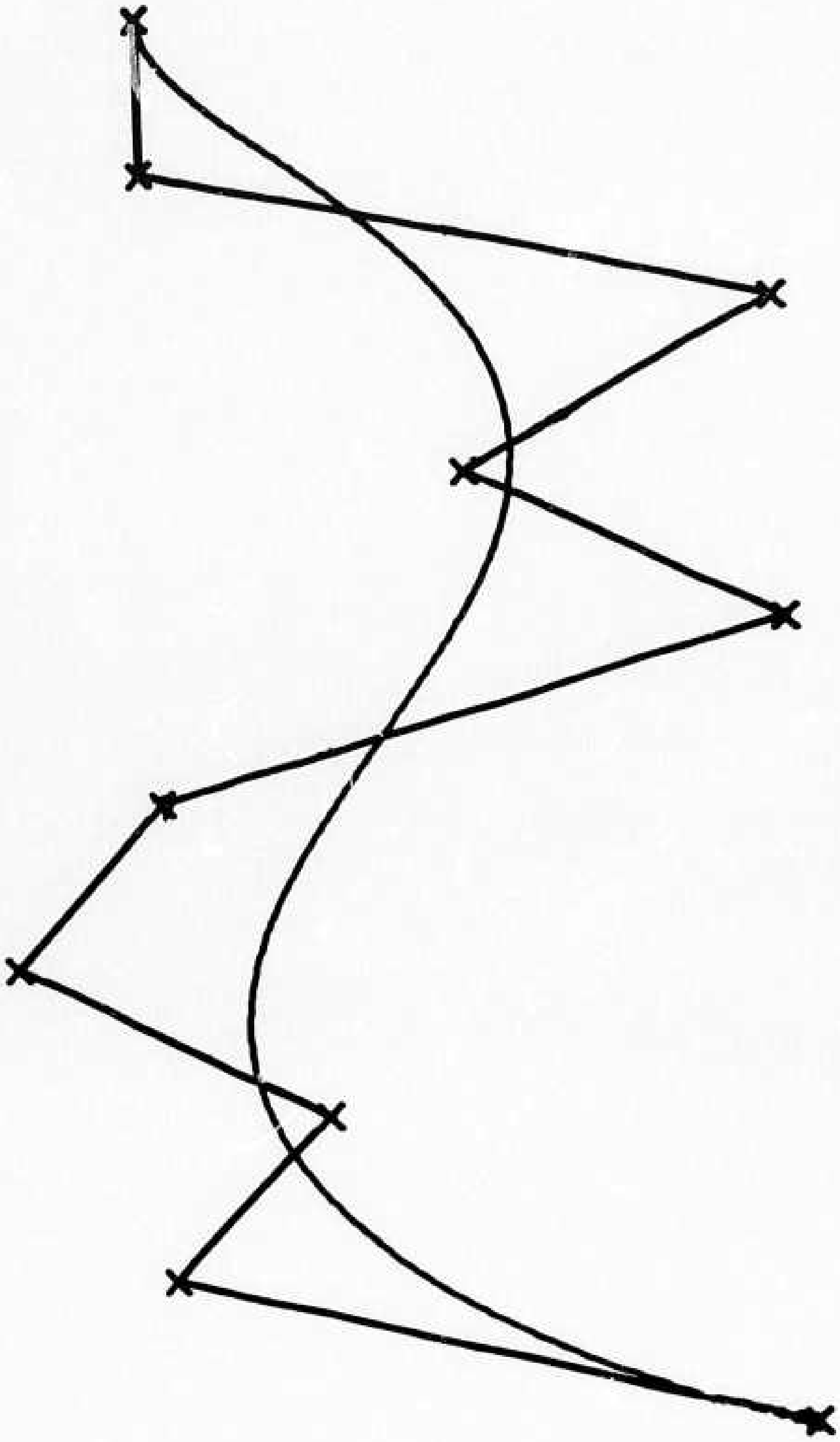


Figure 2.3 Bezier Curve for a 9-sided Polygon

giving for the surface equation

$$S_{m,n}(s,t) = f_{00}(s,t)\ v_{00} + f_{01}(s,t)\ v_{01} + \ldots + f_{mn}(s,t)\ v_{m,n}.$$

A two dimensional control point array and the generated Bézier surface are shown in Figure 2.4. Since this is a tensor product form, the surface interpolates the points at the corners of the control net. The boundary curves are the univariate curves associated with the bounding points in the control net.

II.3 Local Basis Formulations.

B-splines.

The first work with B-splines in a computer aided geometric design context was done by Richard F. Riesenfeld and is reported in his Ph.D. thesis at Syracuse University[6]. In the following discussion of B-splines we will use an approach that more closely follows a development due to Coons[12].

In defining the shape of an object we might like to be able to define the positions of a number of points on it and

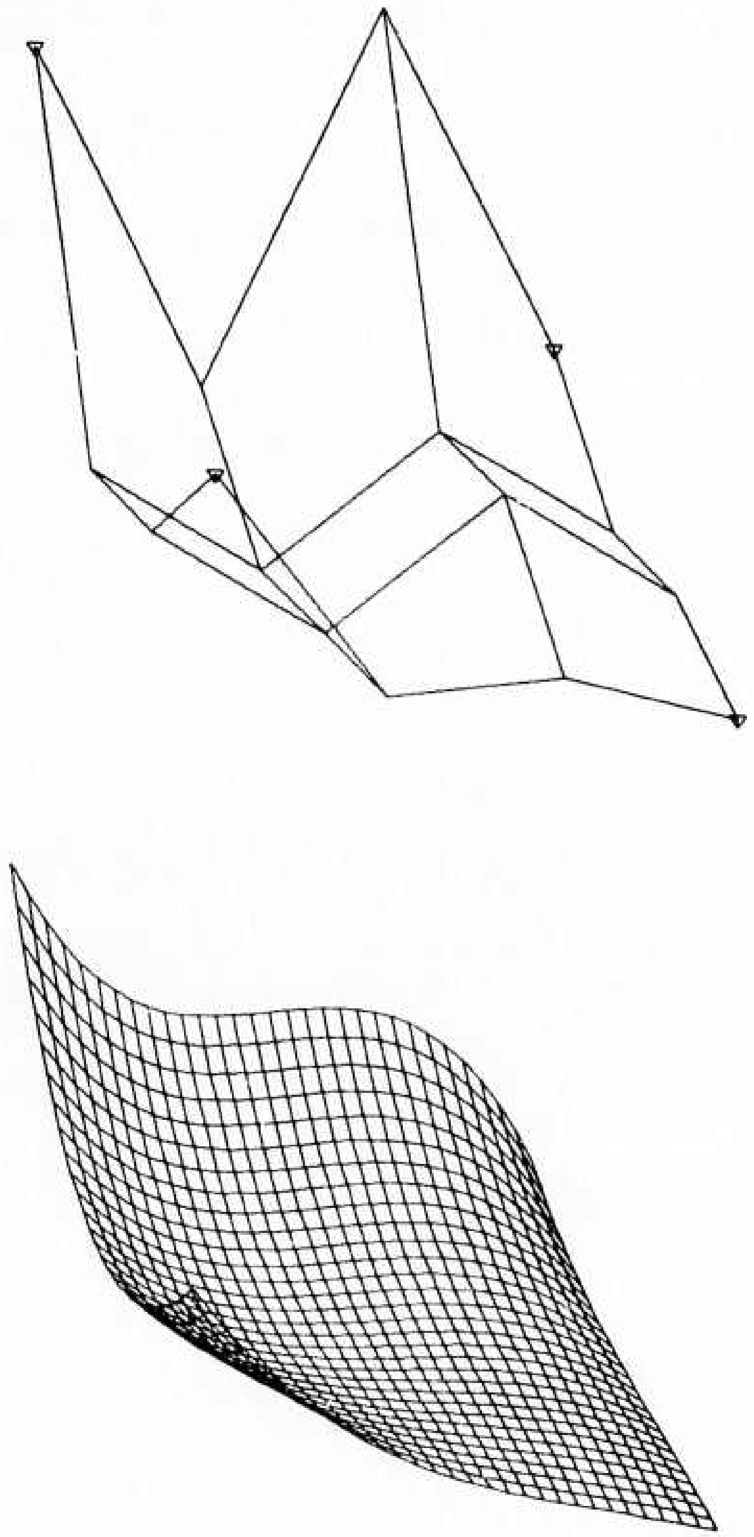


Figure 2.4 Bezier Surface and Control Point Array.

have the computer program use a suitable formulation to "fill in" the regions of the object not explicitly defined by the points. The number of defining points should not be too large if the surface is not complicated. Yet if the surface is complicated we cannot expect any mathematical formulation to be able to describe the surface adequately with just a few definition points.

Suppose we have an ordered sequence of points in 3-space, $(v_0, v_1, \ldots, v_q)$, that we wish to approximate with a curve. The curve is to be piecewise, i.e. it is to be made up of segments(splines). It is to be continuous to first derivative; each segment joins to the next with tangent continuity. We make no assumption about the degree of the various segments that together form the complete curve except that each segment is represented by a vector function of the form

$$P_n(s) = B_0(s)\ v_n + B_1(s)\ v_{1+n} + \ldots + B_m(s)\ v_{m+n}, \qquad (2.5)$$

where the functions $B_j(s)$ are polynomials of degree r in the parameter s.

The value of m is not yet known; it is for now less than or equal to q, the total number of control points. It represents the number of points affecting the shape of each segment. Likewise, r is not yet specified. It is simply a positive non-zero integer.

The $B_j(s)$ functions form a set of basis functions. The goal is to find what form these functions must take to satisfy the continuity requirements. This means that we must determine the coefficients in the basis functions and the values of m and r. Each basis function has r+1 unknown coefficients, and from (2.5) we see that there are m+1 basis functions. Therefore, there are (m+1)(r+1) unknown coefficients.

Imposing the continuity requirement, we obtain the following relations:

$$P_{j+1}(0) = P_j(1),$$
$$P'_{j+1}(0) = P'_j(1). \qquad (2.6)$$

These require that the basis functions be of the form

$$B_j(0) = B_{j+1}(1),$$
$$B'_j(0) = B'_{j+1}(1), \qquad (2.7)$$
$$B_0(1) = B'_0(1) = 0,$$
$$B_m(0) = B'_m(0) = 0,$$

for $j=0,1,\ldots,m-1$.

Equations 2.7 make up 2m+4 constraints to be applied to the basis functions. We obtain 1 additional constraint when we require that the basis functions be normalized, that is they must sum to 1 for all values of the parameter s. This last condition makes a total of 2m+5 constraints on the (m+1)(r+1) unknowns. Equating these two quantities:

$$(r+1)(m+1) = 2m+5,$$

or

$$r = 1 + 3/(m+1). \tag{2.8}$$

Obviously this implies r=m=2, since r and m must both be integers.

From the conditions of (2.7), the normalization condition and the restriction of m=2, the basis functions are determined to be:

$$\begin{aligned} B_0(s) &= (1-s)^2/2, \\ B_1(s) &= (-2s^2 + 2s + 1)/2, \\ B_2(s) &= s^2/2. \end{aligned} \tag{2.9}$$

A plot of these functions is shown in Figure 2.5.

The same argument holds for higher degrees of continuity. With the addition to (2.6) of the requirement that $P''_{j+1}(1) = P''_j(0)$, we see that for this case the functions are

$$\begin{aligned}
B_0(s) &= B_3(1-s),\\
B_1(s) &= B_2(1-s), \qquad (2.10)\\
B_2(s) &= (-s^3 + 3s^2 + 3s + 1)/6,\\
B_3(s) &= s^3/6.
\end{aligned}$$

The usual tensor product form for surfaces yields for the bivariate B-spline:

$$B_{j,k}(s,t) = B_j(s)\, B_k(t). \qquad (2.11)$$

A halftone picture of the bivariate basis corresponding to (2.9) is shown in Figure 2.6. This picture was generated using the Watkins process[13] by breaking each patch into 16 polygons. This is a plot of the functions in parameter space. Each function has been plotted separately in its own coordinate system, and the coordinate systems have been displaced in such a way as to illustrate each function separately and at the same time show the tangent continuity conditions between functions.

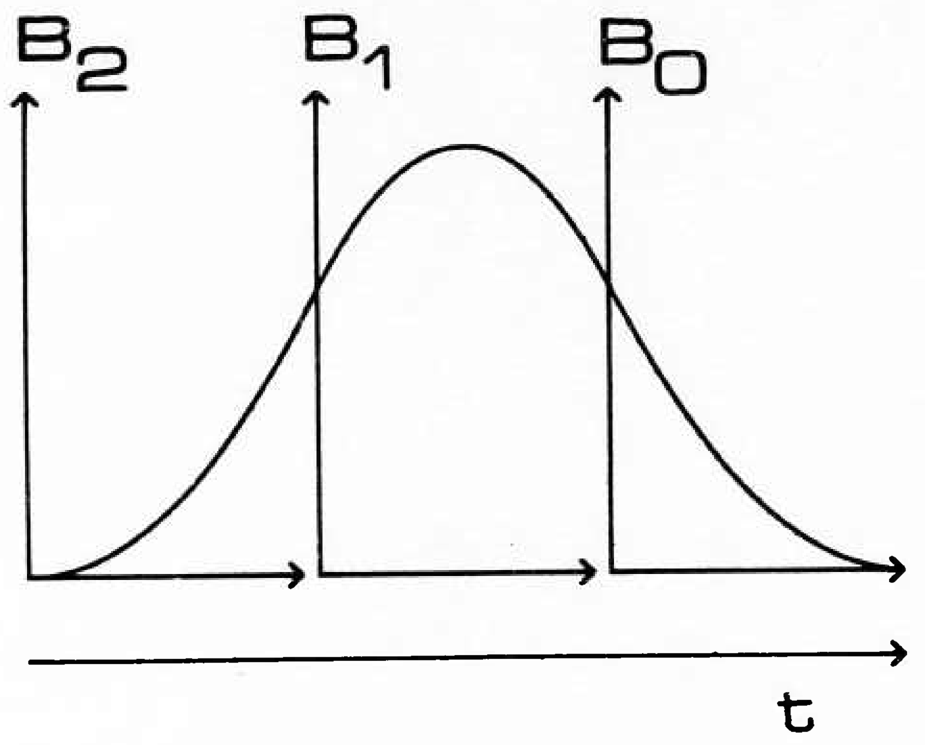


Figure 2.5 Quadratic B-spline Basis Functions

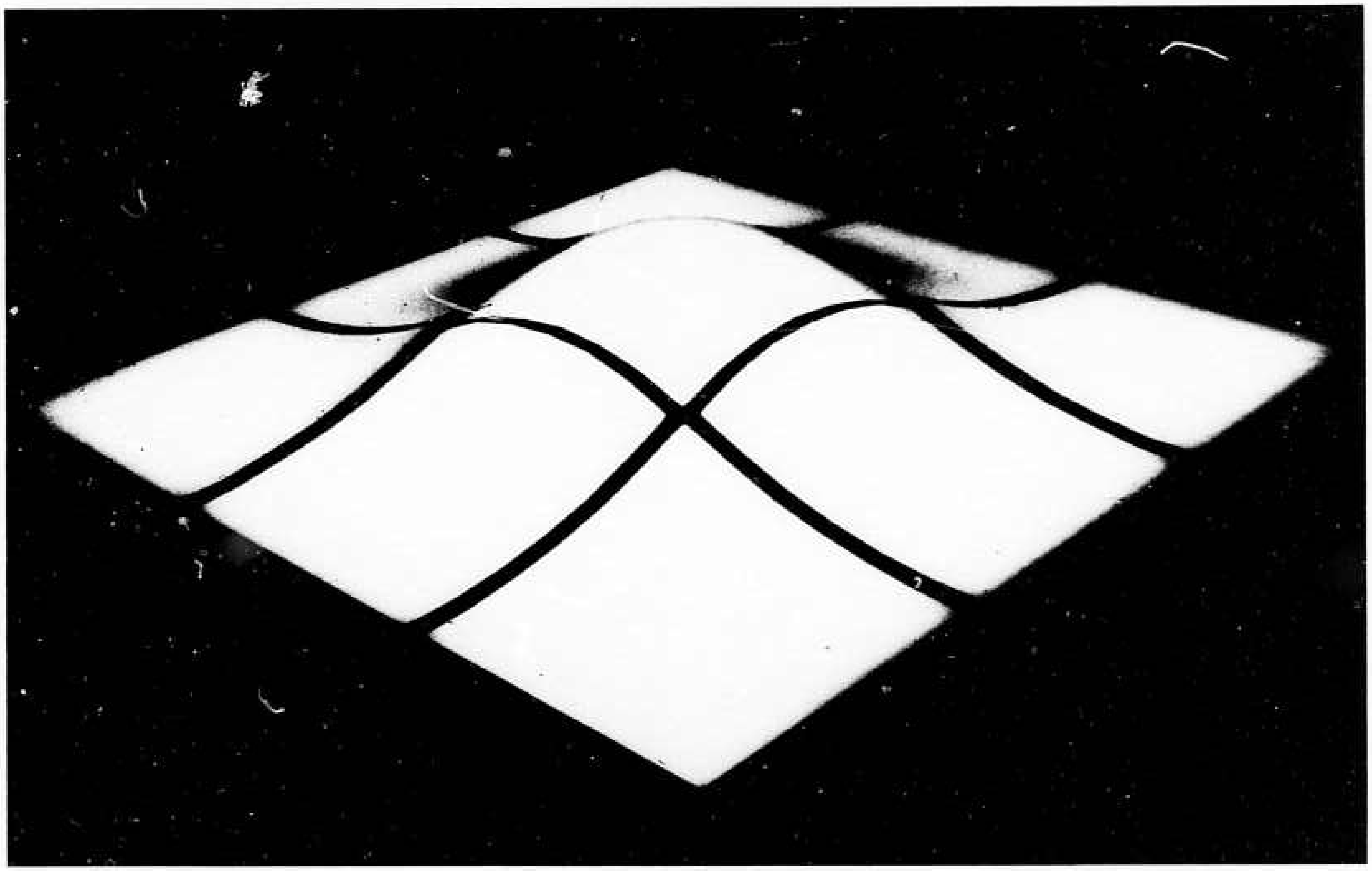


Figure 2.6 Tensor Product of Quadratic B-spline Basis

Figure 2.7 shows a set of points that has been approximated by a curve using the basis functions of (2.9). Each vertex is indicated by an x. Notice that the curve interpolates at the endpoints. This is because multiple vertices occur there, and hence the curve is degenerate there. Figure 2.8 shows a surface that was generated using their bivariate form. The controlling points for the surface are shown by the control net in the same figure.

Interpolating Splines.

E. Catmull and R. Rom have described a method for getting interpolating splines with a local basis formulation[14]. The form of the basis functions for interpolating splines can be obtained in a way similar to the B-spline development above.

Suppose we have a sequence of points, v_0, v_1,..., v_m, that we wish to fit with a piecewise cubic curve that is of the form

$$P_n(s) = f_0(s)\ v_n + f_1(s)\ v_{1+n} + f_2(s)\ v_{2+n} + f_3(s)\ v_{3+n}. \qquad (2.12)$$

If we impose the constraints

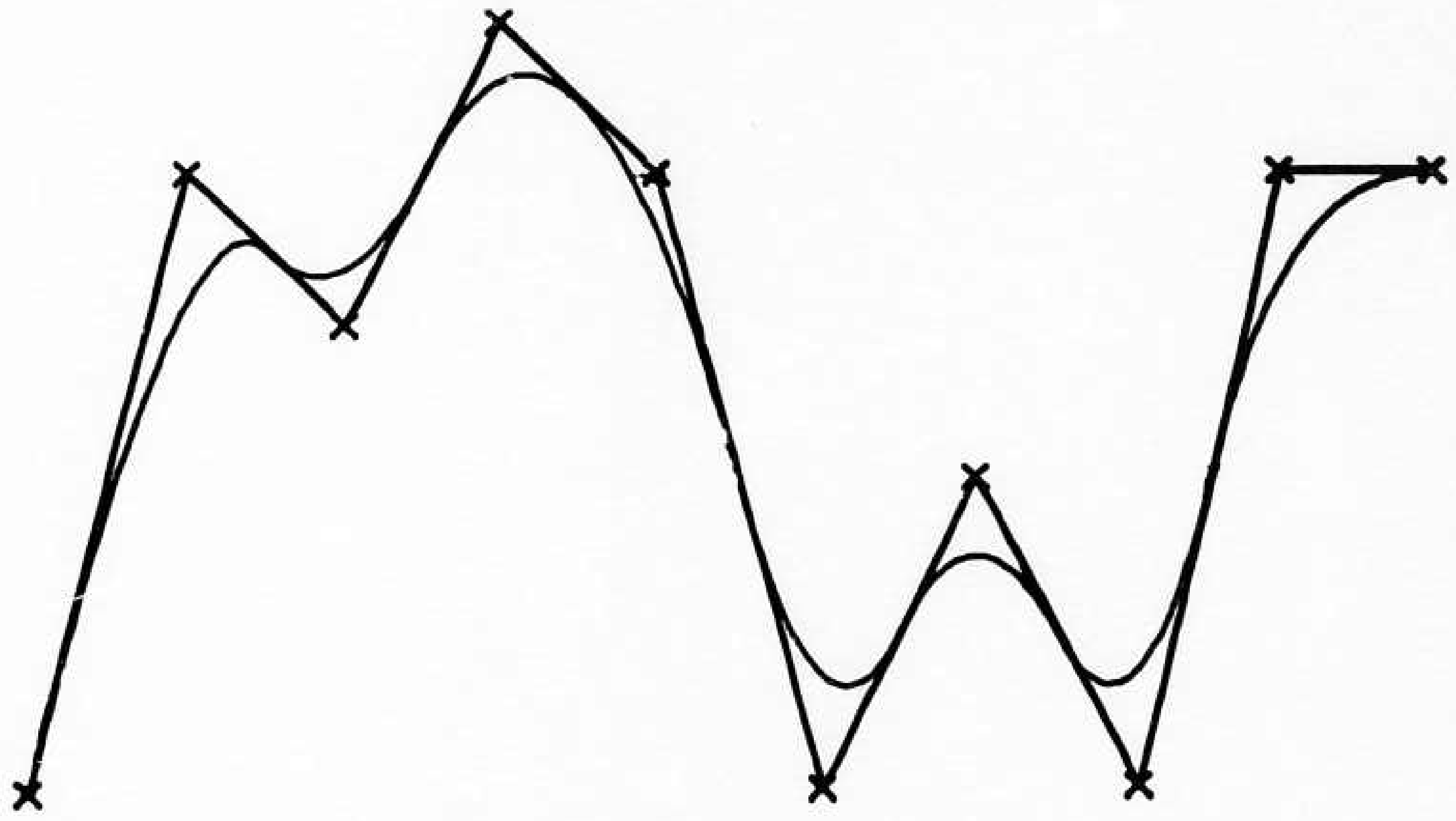


Figure 2.7 Quadratic B-spline Curve; Tangent continuity

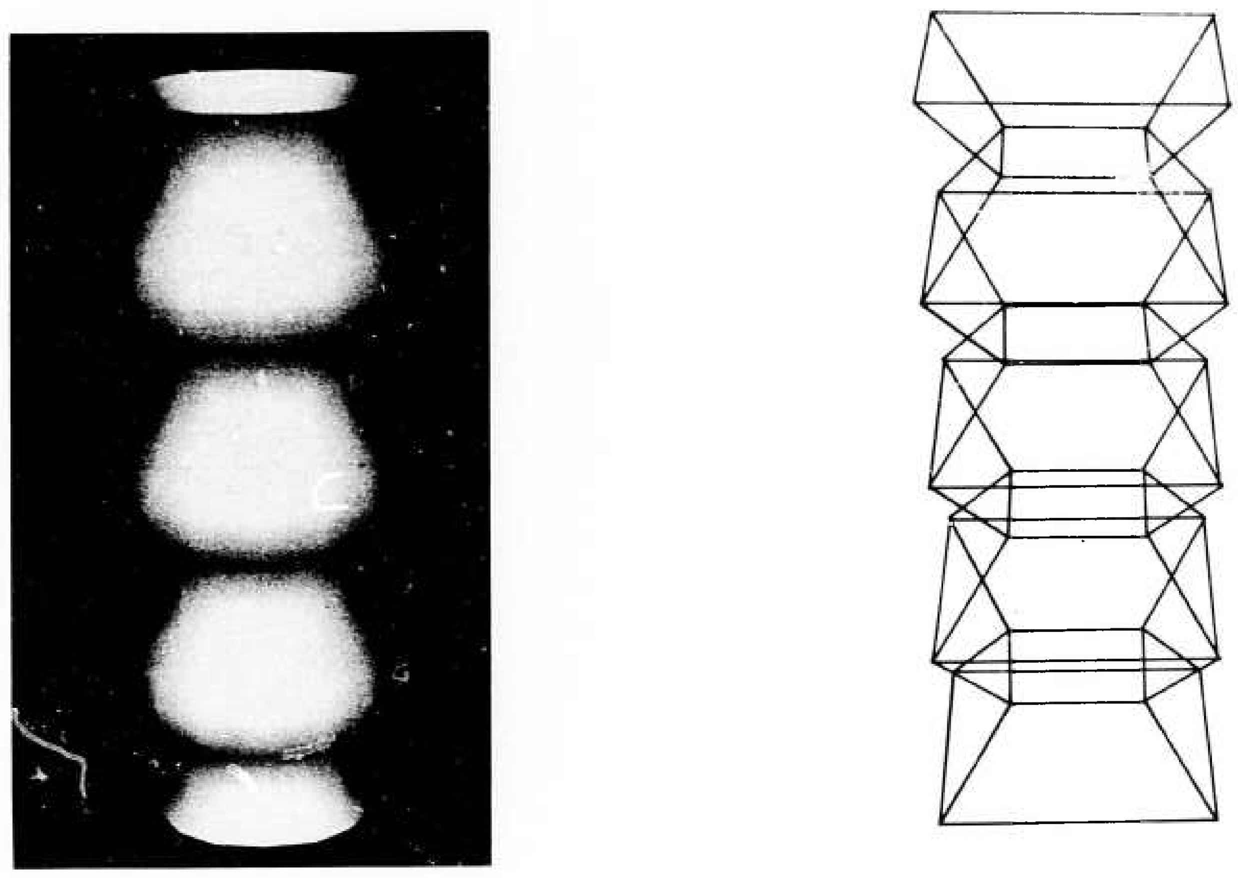


Figure 2.8 Control Point Array and Generated Biquadratic B-spline Surface.

$$P_n(0) = v_{1+n},$$
$$P_n(1) = v_{2+n}, \qquad (2.13)$$
$$P'_n(0) = c(v_{2+n} - v_n),$$
$$P'_n(1) = c(v_{3+n} - v_{1+n}),$$

where c is a positive constant, we find that the basis functions are determined to be

$$f_0(s) = c(-s^3 + 2s^2 - s),$$
$$f_1(s) = (2-c)s^3 + (c-3)\ s^2 + 1, \qquad (2.14)$$
$$f_2(s) = f_1(1-s),$$
$$f_3(s) = f_0(1-s).$$

The remaining parameter, c, can be adjusted to control the magnitude of the tangent of the curve at its ends. The value c=1/2 was arbitrarily chosen for all of the work described in this thesis. However, this remaining parameter might be adjusted to improve the subjective "goodness of fit".

As with B-splines, the bivariate form for these functions is obtained by using the tensor product of the univariate basis functions.

II.4 Computation Algorithms for Basis type splines.

The surface formulation for both the B-splines and the interpolating splines discussed in section II.3 is, for cubics, of the form

$$S_{m,n}(s,t) = \sum_{i,j=0}^{3} f_{jk}(s,t)\ v_{j+m,k+n}. \qquad (2.15)$$

It is interesting to look at the number of computations required to find a point on the surface. We choose a particular value for s and t, evaluate the sixteen bicubic functions $f_{jk}(s,t)$ for this choice of s and t, and perform the sum over j and k from 0 to 3. Each part of the sum involves 3 multiplies, one for each coordinate. In addition, each point influences the shape of 16 different surface patches $S_{m,n}$. Therefore, each time a point's coordinates are changed, the above computations must be performed 16 times for each value of s and t at which the surface is to be evaluated.

One of the goals of the system described in this thesis was to be able to update the surface in 1/20 second or less when a control point is moved. Obviously this goal cannot be met if the number of computations cannot be reduced from the number mentioned above.

Since the graphical equipment used in this system is capable of drawing only straight line segments, a parametric line on a surface patch is drawn as a sequence of straight line segments. The number of straight line segments per parametric line and the number of parametric lines per patch determine the number of points at which the surface equation must be evaluated.

The first step in reducing the arithmetic in the computations is to choose appropriate intervals for values of the parameters s and t and form a table of precomputed values for each of the 16 basis functions. If for example we choose intervals of 1/5 for s and t, then a table of 36 values for each function is generated, corresponding to these values of s and t. We can then do a table lookup instead of evaluating the function.

The next step in reducing the arithmetic is to make use of the local character of the approximations that the basis splines give. A given point influences only a limited part of the surface's shape. Thus, for a particular choice of parameter intervals for the basis functions, the surface can be completely evaluated at initialization time and stored in tablular form. When a point is moved, only those patches it influences need to be altered. For the parameter spacing mentioned above, this means that 30*3 storage locations are needed for each patch.

The final reduction in arithmetic is obvious when we recognize that a movement of one point causes only one of the terms in (2.15) to change. That is, if point (q,r) is changed by Δv_{qr}, then the surface changes by

$$\Delta S_{q-j,r-k}(s,t) = f_{j,k}(s,t)\ \Delta v_{q,r}, \tag{2.16}$$

for j,k=0,1,2,3.

Thus for the parameter spacing mentioned above, if a point is moved, 16 patches must be updated. A minimum of 30 points on each must be re-evaluated. (We can omit one boundary parametric line per patch since it would be evaluated twice.) Each of these evaluations involves 3 multiplies and 3 adds, one for each coordinate. Thus 1440 multiplies and 1440 adds must be performed. The PDP-10 instruction time required for this number of operations is approximately 25 milliseconds. This is well within the 50 millisecond requirement mentioned above.

II.5 Why Use B-splines?

In the previous sections of this chapter, the essential points of most of the major mathematical representations that are in use in CAD systems today were discussed. B-splines were chosen for the major mathematical formulation in this work.

From Section II.2 and from the form of (2.1), we see that the Coons patch is the most general formulation of those discussed. The boundary functions $Q(i,v)$, $Q(u,j)$, $Q_u(i,v)$ and $Q_v(u,j)$ may be any parametric functions. For example, the space curves might be semi-circles or parts of a complex molding. This flexibility makes the Coons formulation preferable in cases in which this type of flexibility is needed. With Bézier and B-spline approaches, this level of generality is not readily available.

The principal disadvantage of the Coons approach is that the twist vector terms in (2.1) present problems to people who are not familiar with cross partial derivatives. Even for those who are familiar with this concept, it is not usually an intuitive one as, for example, tangent or curvature continuity is. In the Bézier and B-spline formulations, this twist vector concept is imbedded implicitly in the formulation. Consequently, they are easier formulations to use in many applications.

For the purposes of this system, another discouraging feature of the Coons approach is that the interior region of a Coons patch is determined solely by the shapes of the boundaries. Of course, the form of the functions in (2.2) influences the interior shape, but these functions are fixed for all boundaries. With the B-spline formulation, the user has control points in the interior of the patch as well. These control points may also influence the shape of the boundary curves.

B-splines and Bézier surfaces are both formulated in terms of basis functions. However, the Bézier basis is global, and the degree of the basis functions varies with the number of points being approximated. With B-splines, once the degree of continuity in the surface is specified, the basis functions are fixed in degree. It is a local approximation. The amount a given point influences the surface is limited in extent by the weight of its basis function. For the purposes of the free-form goals in this work, this feature was considered desirable.

Another reason that B-splines are preferred over the others is that B-spline surface patches are automatically continuous with adjacent patches. This is true if they share boundary points, and it is an intrinsic property of the local basis approach. With both Coons and Bézier patches, the continuity between patches is explicitly dealt with by matching tangent vectors, or tangent boundary functions, at the boundaries of the patches. Of course this is implicitly what we do with B-splines when we cause adjacent patches to share boundary points, but it is less distracting to deal with points than with relative vectors.

When a control point of a B-spline surface is moved, its influence on the surface is local, and moving it does not alter the continuity. This is not true of the other approaches.

All of the advantages of B-splines discussed to this point are also shared by the Catmull-Rom(CR) interpolating splines discussed before. Actually, these functions might be considered preferable to B-splines because they interpolate. However, there is a very important property that B-splines have that is not shared with the CR splines. A B-spline curve always lies within the convex hull of the points it is approximating. The convex hull for a cubic curve is shown in Figure 2.9. The reason for this property is that the B-spline curve is a weighted average of the vertex coordinates with the basis functions as weights. Since the basis functions are always less than unity and are positive for all values of the parameter, the curve lies inside their convex hull.

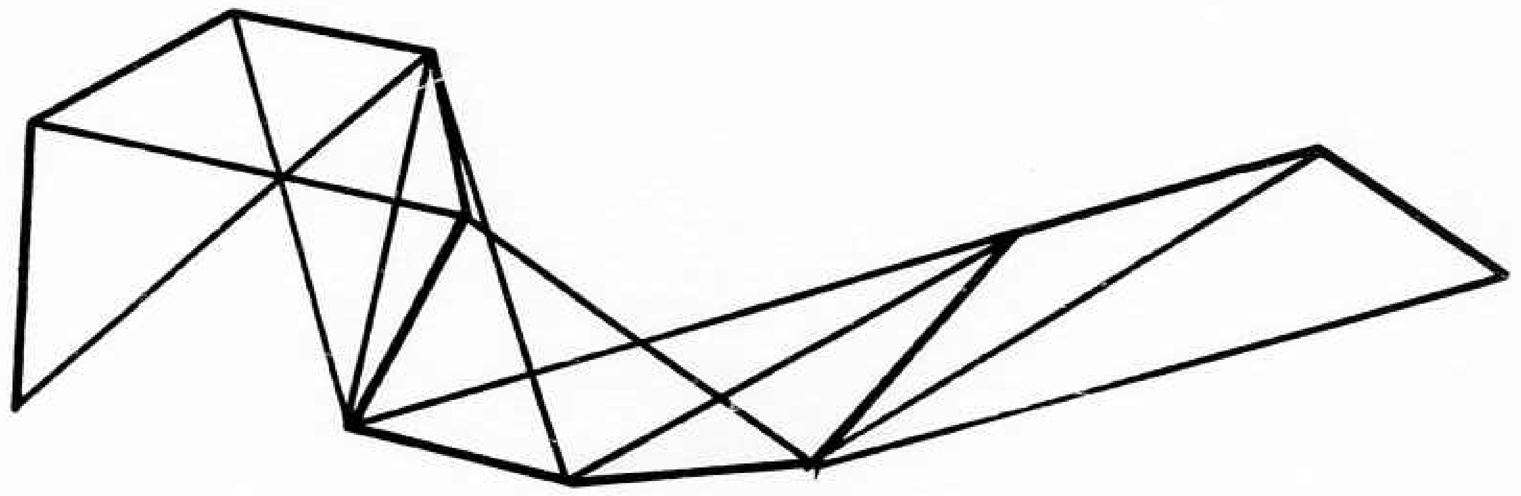


Figure 2.9 Convex Hull for a Cubic B-spline Curve

The final criterion in choosing B-splines was the ease with which they can be calculated incrementally. This is true of both B-splines and the interpolating type. In fact, the computatione for both are identical; only the basis tables must be changed to go from one to the other.

CHAPTER III

THE 3-D B-SPLINE DESIGN SYSTEM

III.1 System Hardware Configuration

Figure 3.1 shows the hardware configuration for the design system. The main computing engine for the system is the PDP-10 computer. This machine controls the operation of the rest of the system.During normal operation, the PDP-10 program stays in a loop in which it reads the counter values for the head position and wand position, the function switches and the wand buttons. This is done through the PDP-10 I/O Buss interface. From these counter values, the main program computes the separate matrices that make up the head position matrix and finds the wand position. Then if any changes must be made to the surface, the incremental computation is done, and the display file for the display processor is updated.

The secondary computer in the system is the LDS-1 display processor. This machine's only task is to generate the composite viewing transformation and execute the drawing

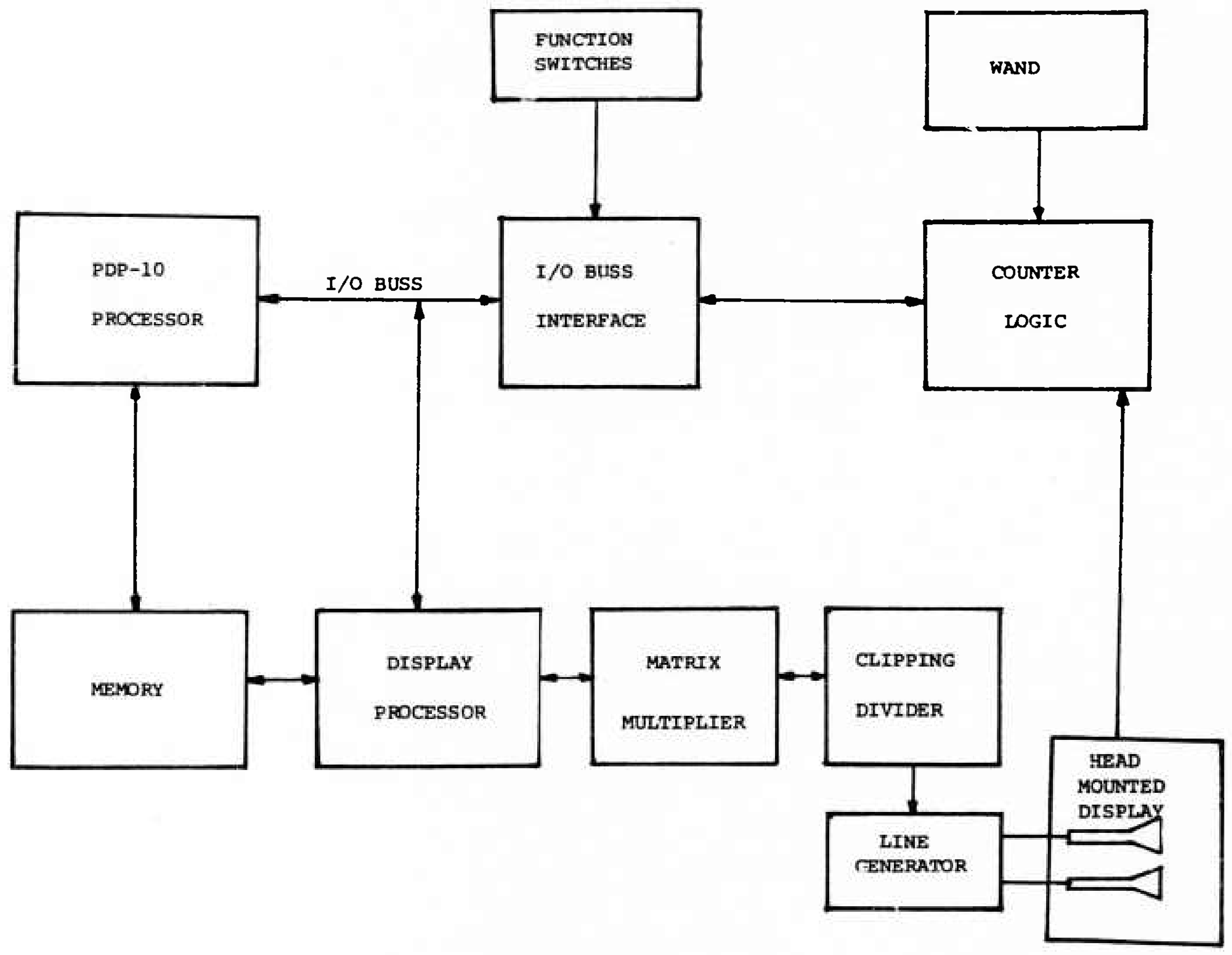


Figure 3.1 Hardware Configuration for the Line-drawing System.

instructions in the display file. It has access to the main memory via the PDP-10 Memory Buss. Figure 3.2 is a picture of the display processor and I/O Buss interface equipment.

When the work described in this thesis first began, the system as shown in Figure 3.1 did not exist. There were two principal components missing: the Matrix Multiplier and the Clipping Divider. In addition, the display processor, the counter logic and the line generator did not work properly. The Matrix Multiplier was purchased from Systems Concepts, and the Clipping Divider was designed and built at the University of Utah[15]. When both of these devices were operational, the rest of the system was repaired.

III.2 The 3-D Environment

Head-Mounted Display.

The principal hardware component of the 3-D environment is the head-mounted display. This display was the first of its kind ever to be built. As mentioned before, it was brought to the University of Utah when Ivan Sutherland joined the faculty here in 1968.

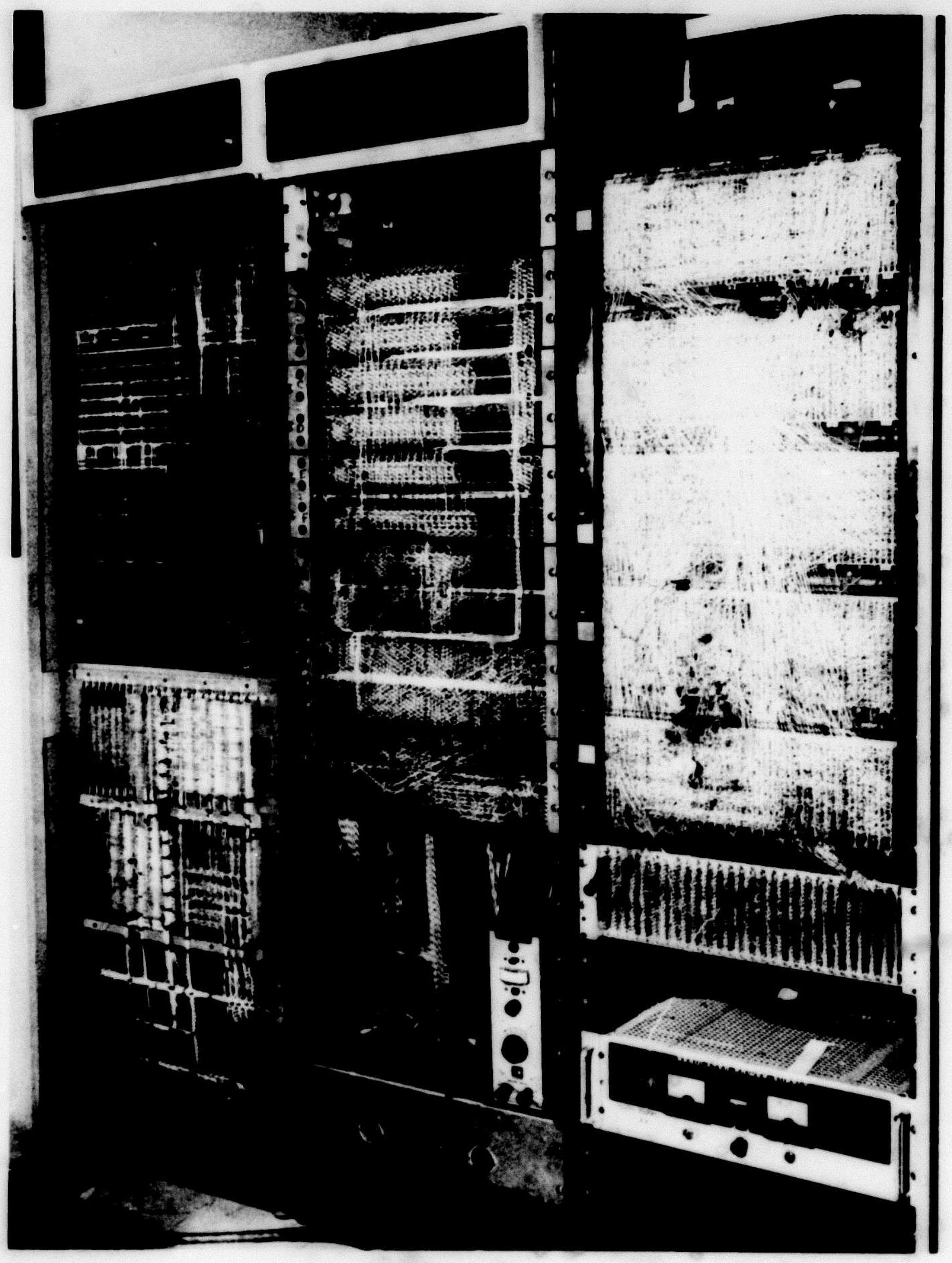


Figure 3.2 Picture of Display Hardware, including counter logic and I/O and Memory Buss Interfaces.

Figure 3.3 is a picture of the head-mounted display, which provides the three-dimensional display environment. The mechanism of the head-mounted display permits the computer to sense the position of the observer in the room and the angular attitude of his head. Consequently, the computer "knows" what the observer ought to be able to see and presents the appropriate scene on two miniature CRT's, one for each of the observer's eyes. If there is a three-dimensional object defined "in" the computer, the observer can see it, walk around it, move closer or farther away, or even walk through it.

The display uses six shaft encoders to measure the independent quantities needed to determine the six degrees of freedom that one rigid body has relative to another. The six quantities determine the orientation of the viewing part of the head-mounted display relative to the room. There are five angular measurements and one displacement measurement. The first of these determines the angle of rotation of the entire display mechanism relative to the room(see Figure 3.4). The next two are the angles of rotation about the axes of a universal joint, which is located at the top of the room. Then the displacement measurement is made from the universal joint at the top of the room to one located just above the viewing mechanism. The last two measurements are the angles in the

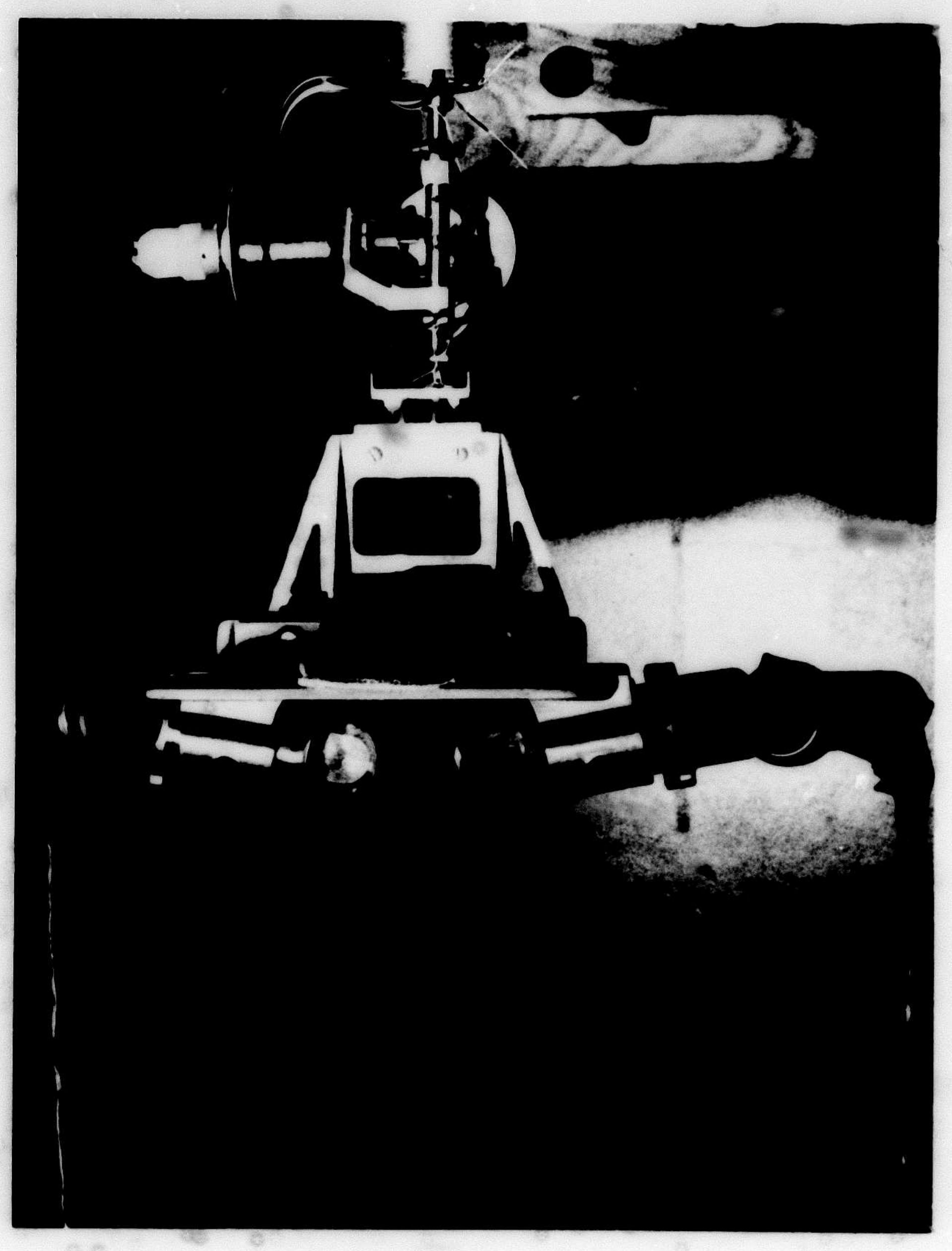


Figure 3.3 Head-Mounted Display Viewing Mechanism.

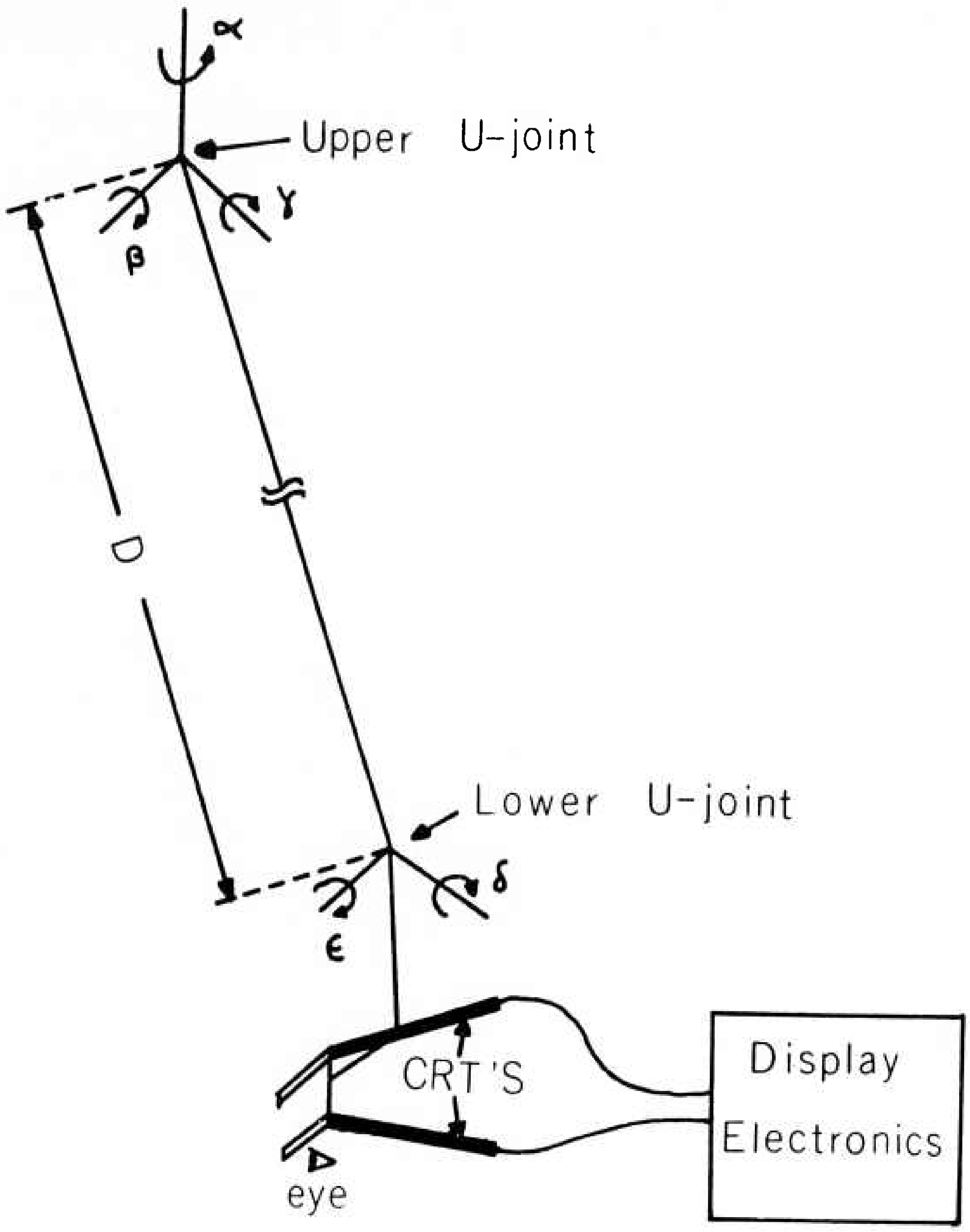


Figure 3.4 Function Diagram for Head-Mounted Display Position Sensing Mechanism

lower universal joint. A more detailed description with the appropriate transformation matrices shown is given in Appendix 1.

When the system is first initialized, the head-mounted display is placed in a calibration stand. The counters for the six shaft encoders are then preset to the calibration values that fix the location of the origin of the room-fixed coordinate system relative to the CRT headset. Movements of the display mechanism cause the shaft encoders to send pulses to their respective counters causing them to count up or down. The counters are triggered whenever a pulse comes from the shaft encoders. The counter values are read by the main PDP-10 program each time the head position matrix is to be calculated. The values are then used to compute the matrices, which are combined by the matrix multiplier to form the head position matrix. Details of these computations are given in Appendix 1.

3-D Wand.

A substantial effort has been made at the University of Utah to build a good, real-time, 3-D digitizer that has no mechanical constraints. One such effort has been an attempt by several students to build an acoustical digitizer, or 3-D tablet, using a method like that used in the SAC Tablet[16]. Three very long cylindrical microphones were mounted along mutually perpendicular axes, and the appropriate electronics were used to find the 3-D coordinates of the SAC spark pen. Because of the difficulty and expense in maintaining the microphones, this project was abandoned.

The most recent 3-D digitizer attempt made at the University of Utah was the Burton Box by Robert Burton[17]. This system consisted of 4 rotating disks with small radial slits cut in them. Behind each disk were two photomultiplier tubes and an optical system. As the disks rotated, a light emitting diode(LED) was very briefly turned on. The LED and the slit that passed in front of the photomultiplier define a plane containing the LED. If the LED is "visible" by all of the photomultipliers, then the system simultaneously solves 8 planar equations for an intersection point. This point represents the location of the LED.

The problems with this system were numerous. The worst problem was that the standard deviation of the 3-D coordinates of a stationary LED was 7 millimeters. Stated in other terms, one out of every 100 samplings of the coordinates of the point would be outside of the radius of a ping-pong ball. Another annoying problem with the Burton Box was that it could not be in operation for more than 30 minutes at a time due to excess heat generated by the motors that were rotating the disks. This heat caused the photomultiplier outputs to be excessively noisy.

Much has been learned about the 3-D digitization problem through the efforts described above. I believe that an approach that utilizes some of the ideas in Burton's thesis will lead to a system that can accurately determine 3-D coordinates in real-time. A possible system that accomplishes this is proposed in Appendix 2.

The only reliable 3-D wand to be built at the University of Utah is based on a mechanical position sensing mechanism. In this device, the wand position is found by measuring the lengths of 3 wires attached to a point on the handle of the wand and extending to three housings mounted on the ceiling. The housings are located at the corners of an isoceles triangle. Inside each housing is a drum around which the wire is wrapped, a negator-spring motor, and a shaft encoder. The shaft encoder records the angular displacement of the drum, and from this measurement the amount of wire extending from the housing is determined. The purpose of the spring motor is to maintain a constant tension in the wire for varying lengths. The wand position is computed from the intersection of three spheres with centers at the three housings and radii equal to the lengths of wire. The details of the computation will be found in Appendix 1.

Display Equipment.

The line drawing display equipment used in this system is built around the Evans and Sutherland LDS-1 display processor. This processor is a special purpose computer that has access to the PDP-10 storage by way of the Memory Buss(see Figure 3.1). The processor-to-processor communication takes place over the PDP-10 I/O Buss.

The LDS-1 processor is initialized by commands issued over the I/O buss by the controlling PDP-10 program.These commands initialize various configuration and status registers in the processor, Matrix Multiplier, and Clipping Divider. Then a command is issued to start the processor at a certain storage location, and the LDS-1 begins executing the display program.

All drawing commands and pipeline device commands for the Matrix Multiplier and Clipping Divider are passed on to the command pipeline. If the command is to draw a line, the Matrix Multiplier transforms the endpoints of the line into the viewing coordinate system and passes it on to the Clipping

Divider. The Clipping Divider then clips the line to the pyramid of vision, thereby eliminating part or all of the line, and performs the perspective division on the endpoints of the clipped line. The resulting line is scaled to scope coordinatee and given to the line generator, which generates the analog ramps for beam deflection and sends them to the deflection electronics for the scope.

This LDS-1 Display Processor is a very powerful device. A number of its features have made the implementation of this system much easier. Probably its most inportant feature is that it is a processor and it executes its own program. This means that it can interrogate memory locations and has its own instruction set. Since the LDS-1 instruction word has the same boundaries as the PDP-10 instruction word, it can be programmed by making OPDEFS in the MACRO10 Assembler. It also has a stack and subroutine capability.

Another useful feature of this display system is that the Matrix Multiplier can multiply matrices together. For example, in finding the viewing transformation matrix each frame, the PDP-10 program generates 8 matrices and the Matrix Multiplier combines these separate transformations to form the composite viewing transformation.

III.3 Communication with the System.

All communication with the basis spline design system is through a set of five buttons on the handle of the wand (see Figure 3.5), a panel of 18 switches, and the teletype. When the system is initialized, it allows the user to select a file to start the design. These files are stored on DECTAPE. The file selected is read into storage and the corresponding surface is created and entered in the display file. The user can then interact with the surface by moving the control points. Interaction with the surface in this system consists of free, i.e. unrestricted and unconstrained, movements of the control points. After completing some stage of the design, intermediate results of the design can be saved, and the user may continue designing.

System Features.

A number of features were added to the first version to make it easier to design objects with the system. Since the objects designed with this system might be larger than the space defined by the limits of the mechanical head position sensors, the ability to rotate and translate the object was added. This is done by selecting the appropriate switch on the switch console. The first two buttons on the wand are then

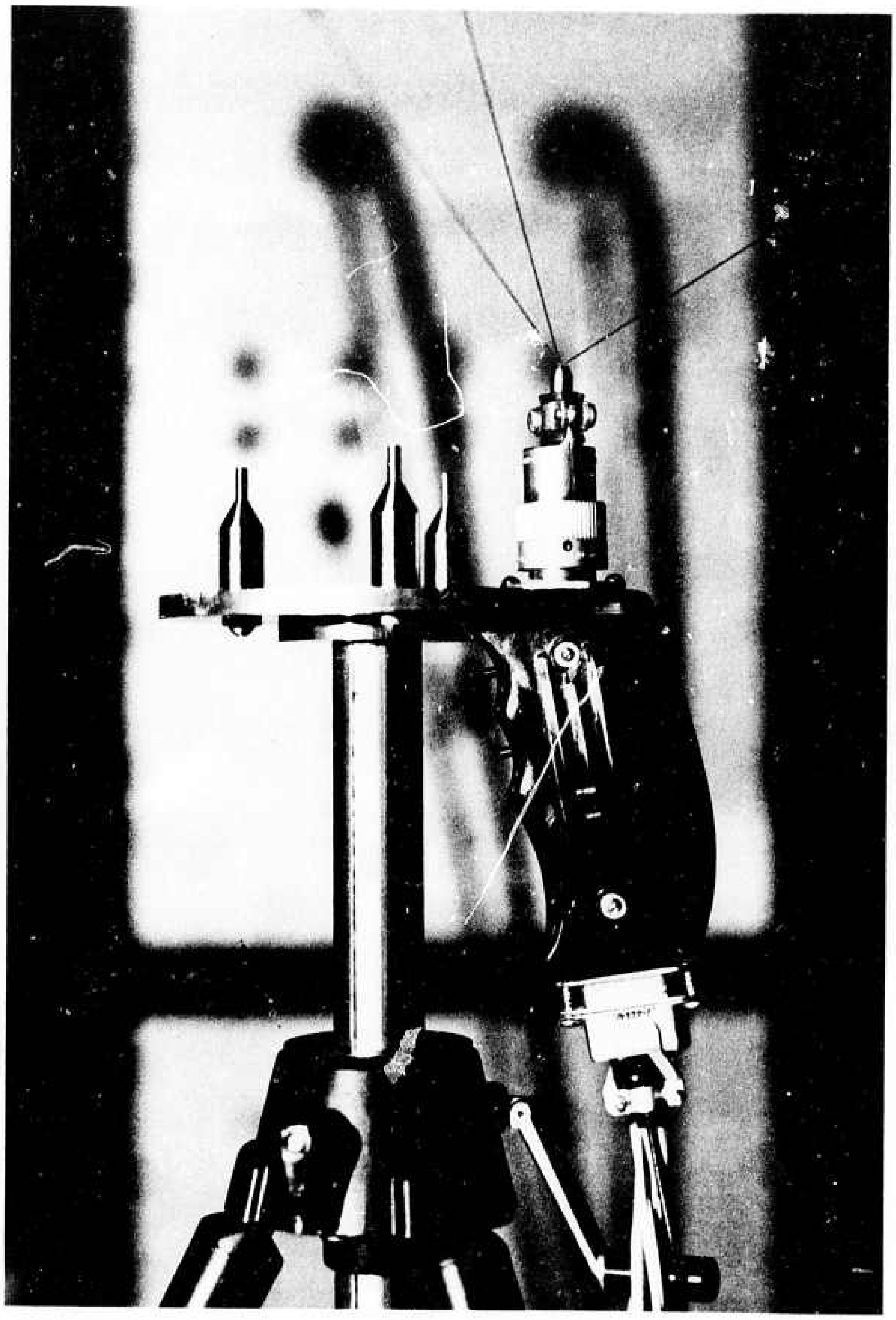


Figure 3.5 Handle for 3-D Wand.

treated as a two bit speed control. The switch on the back of the handle is used to reverse direction. The object is rotated or translated by making the appropriate changes to its transformation matrix rather than by changing the data.

In normal operation, the system displays only isoparametric curves(rendered as collections of straight line segments) in the surface and emphasized dots to represent the control points. Figure 3.6 shows some typical views as seen on the display. As control points are moved, the appropriate surface patches are updated in real-time. This gives the user immediate feedback on the shape of the object. The control net is not displayed because there would then be too much detail being displayed, and the object would be difficult to see. However, since the user is allowed to move the control points freely, it occasionally happens that the connectivity of the control points becomes confusing. In these cases, it is almost essential to have the control net displayed. Therefore another switch was added that allows the user to select either the control net or the surface to be displayed.

In order to be able to update the surface in real-time, a special computation algorithm had to be used. The details of this algorithm were discussed in Section II.4. Since the LDS-1 uses fixed point arithmetic, this algorithm suffers from

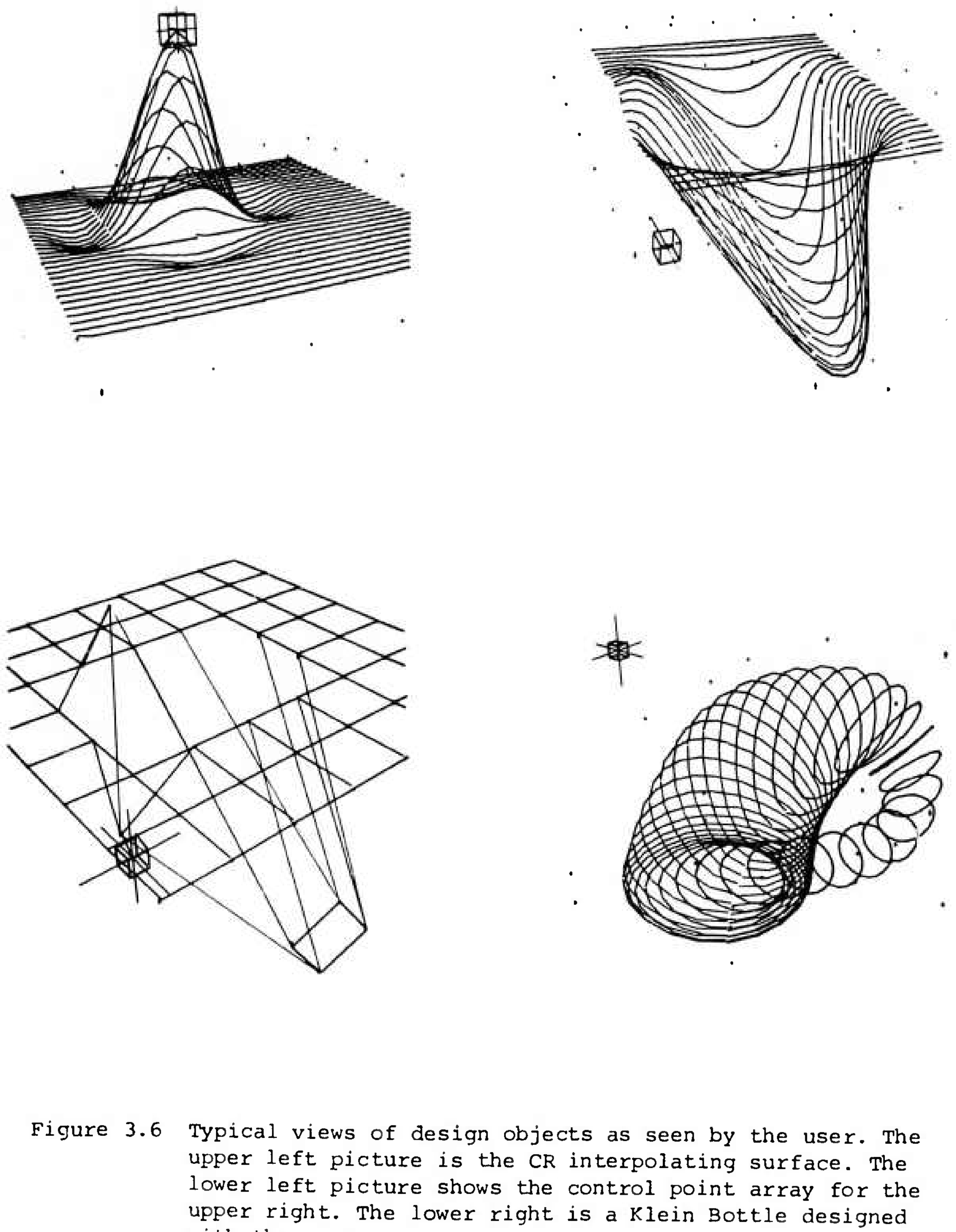


Figure 3.6 Typical views of design objects as seen by the user. The upper left picture is the CR interpolating surface. The lower left picture shows the control point array for the upper right. The lower right is a Klein Bottle designed with the system.

round-off problems. As a result of accumulated round-off errors, the part of the surface influenced by a given point begins to look ragged after the point has been moved around a lot. When the surface begins to look too ragged, the user can cause the surface to be completely computed over again by pressing one of the wand buttons. This computation usually takes from 2 to 5 seconds of computer time.

To keep the PDP-10 loop under 50 milliseconds all of the time, a special searching method is employed to find when the wand is close enough to a point to grasp it. A search through the complete set of control points is performed only in two cases: 1) if both buttons 1 and 2 on the wand are pressed or 2) if the wand is not already grasping a point. In this way, the main program does not usually have to both find which point(s) the wand is near and also update the surface for the point(s) being moved. The only case for which the update must be done while searching for points is when both buttons 1 and 2 are pressed. This exception allows the user get additional points while also holding onto those he has.

Another special case arises when the user decides he wants to separate two(or more) points that he has merged together. To allow this kind of control, button 1 is used to grasp only 1 point at a time, and button 2 is used to get more round-off problems. As a result of accumulated round-off errors, the part of the surface influenced by a given point begins to look ragged after the point has been moved around a lot. When the surface begins to look too ragged, the user can cause the surface to be completely computed over again by pressing one of the wand buttons. This computation usually takes from 2 to 5 seconds of computer time.

To keep the PDP-10 loop under 50 milliseconds all of the time, a special searching method is employed to find when the wand is close enough to a point to grasp it. A search through the complete set of control points is performed only in two cases: 1) if both buttons 1 and 2 on the wand are pressed or 2) if the wand is not already grasping a point. In this way, the main program does not usually have to both find which point(s) the wand is near and also update the surface for the point(s) being moved. The only case for which the update must be done while searching for points is when both buttons 1 and 2 are pressed. This exception allows the user get additional points while also holding onto those he has.

Another special case arises when the user decides he wants to separate two(or more) points that he has merged together. To allow this kind of control, button 1 is used to grasp only 1 point at a time, and button 2 is used to get more than one point.

Human Factors.

Several problems have been encountered by the people who have used this system. The most annoying of these problems is that the wires for the wand get tangled in the position sensing mechanism for the head-mounted display. This makes it difficult to maneuver around in the room as freely as one would like. Another problem is that it is not easy to attach the viewing portion of the head-mounted display to the head. This is because of the tension in the mechanical position sensing mechanism. The user must hold onto the head position sensing mechanism with one hand and onto the wand with the other. These difficulties in using the system were accepted since to improve the situation we would have to improve the sensing mechanism and perhaps make it non-mechanical, which is a complete problem in itself.

An interesting problem arises when we try to use the wand to grasp a point in the room. When the first version of the system was written, the wand was displayed as a small cube about 3/4 inch per side, and the control points for the surface were displayed as dots. To grasp a point, the user had to manipulate the wand until one of the control points was inside the cube and then press a button on the wand handle. Alternately, he could first press the button and then search for the point. This was done by holding the wand so that, from the user point of view, the point was inside the boundaries of the cube on the diplay, then moving the wand forward and backward until the depth of the point was found. This procedure proved to be very tiresome.

The first change to this procedure was to draw a cross hair inside the cursor cube that is about four times the size of the cube. As a que that the wand is close to a point, when the wand gets less than 4 inches from any point, the cross hair decreases in size as the Manhattan distance (that is the rectangular distance) to the closest point decreases. This addition makes it much easier to find a point with the wand.

One final improvement that was made was to draw a line from the center of the cursor cube to the nearest point when the cursor is less than five inches from any point. This provides additional aid in finding the depth of the point.

A number of improvements in this aspect of the system have been suggested by people who have used the system. One of the best suggestions was to use the wand as an aiming device. Used in this way, the wand can make relative movements to points by alming at the point and grasping-at-a-distance when a button is preesed. Another suggestion was to display the control points as small cubes rather than dots, thereby giving a good depth cue with the display of the point. Another was to display the environment in stereo. All of these suggestions seem to be reaeonable attempts at solving the problems of finding a control point with the wand. However, improvements of this eort come from an endless source, and none of them attack the real problem. The real problem is that one needs a multiple polnt wand and more freedom to move about in the 3-D environment than this system will allow because of the mechanical poeition sensing mechanisms.

Off-line Shaded Pictures.

All of the surfaces in this system were rendered by drawing some of the parametric lines in the surface patches. No attempt was made to remove any hidden parts of the surfaces or provide any type of shading. The reason that the surfaces were rendered in this fashlon is that smooth shaded hidden eurface algorithms take too long to compute to be useful in a real-tlme system. (Actually, with enough hardware, this could be done in real-time. Shaded picture systems exist that provlde this type of response[18].) Nonetheless, a mechanism has been provided that allows the user to record a design sequence in real-time and reconstruct the sequence in a smooth-shaded, halftone picture movie.

An interrupt service routine was incorporated in the design system as a selectable feature. Each 1/12 second, both the head position transformation matrix and all incremental changes to the coordinates of control points tha' have taken place since the last frame are written on disk. Another set of programs that uses the Watkins hardware then uses this information to create a halftone movie. Figure 3.7 is a halftone picture of a Klein bottle generated by this process. Several sequences have been recorded in this fashion, including a walk through a Klein bottle. These sequences and several live action sequences, which were taken from the face of the CRT during a design sequence, have been submitted as part of this thesis[19].

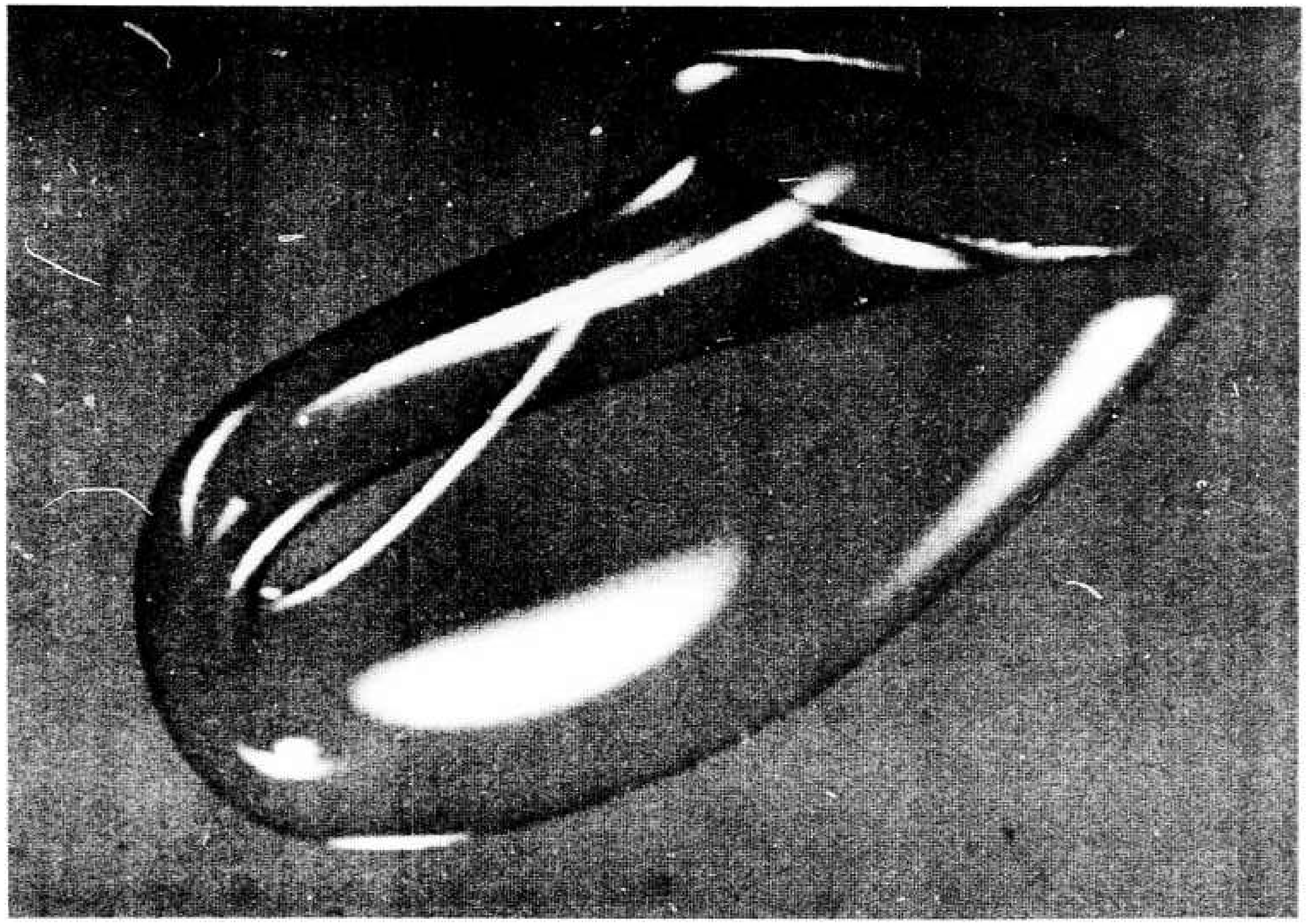


Figure 3.7 Halftone picture of the Klein bottle produced with the system.

CHAPTER IV
CONCLUSION AND SUMMARY

IV.1 System Evaluation.

The system described in this thesis is an experiment in the feasibility of 3-D surface design in a three-dimensional environment. The B-spline surface formulation was chosen to represent the surfaces for the reasons given in Section III.5.

One of the premises of this work is that 3-D design should be done in a 3-D environment. To design 3-D objects using 2-D perspective projections and 2-D input devices unnecessarily constrains the user. Generally, those who have used this system agree that it is much more convenient to have a 3-D input device and to manipulate the objects in a 3-D environment than with other kinds of systems. However, because the manipulation is totally unrestricted and unconstrained, it would be difficult to use this system as it currently is implemented to design something that must satisfy physical constraints.

There are several very important features that are required before a 3-D surface design system becomes useful in this sense. The first and most important is the ability to constrain the geometry of the surface to certain shapes. As the system now exists, the user may grasp any control point and move it to any place in the operating environment. Movement of this point does not influence the position of any other point. Experience has shown that this is often too much freedom.

The type of restriction we might like to impose on the design is that a group of the points making up an object always lie in a plane. Then if several of the points in the group are moved, the others might be moved by the program to maintain the planar constraint.

Another obvious feature that should be added is the ability to specify planes of reflection for objects being designed. Objects are often symmetric about at least one plane, e.g. cars, aircraft, etc. If the system is capable of generating the mirror image, then by manipulating one side, the program can modify the other symmetrically.

A viable system should also be able to handle complex geometries. Because of the rectangular mesh in a tensor product surface, it is not clear that the tensor product form of B-eplinee alone is sufficient to handle these complex requirements. The simple tensor product, e.g. bicubic, basis is certainly not sufficient to handle cases in which the surface must join to a boundary of predefined shape. Some combination of B-splines with a blending function method like the Coons method is probably the correct way to handle this special case[20].

A drawback of this system that many people seem to be able to accept is the limitations of the mechanical position sensors for the wand and head mounted display. From Figure 3.3 we can see the mechanical linkage for the head mounted display extending up to the ceiling. The three housings for the wand can also be seen in the picture. Each of the wand housings has a wire extending to the wand handle. These wires and the head mounted display linkage intefere with each other considerably.

A really good real-time 3D digitizer that can find the 3-D coordinates of 10 or more points in real-time would alleviate this difficulty. If we could mount 3 or more LED's on a helmet and determine their 3-D coordinates in real-time, we could use this information to find the orientation of the helmet in the room. In other words, we could eliminate the mechanical linkage shown in Figure 3.3.

Also, with such a device we could attach LED's to the ends of the thumb and index and middle fingers of each hand. Then we could grasp a point by bringing the thumb and index finger to close on it [17].

IV.2 Other Uses for the Head Mounted Display.

The head mounted display was used in this system to view the surfaces being designed in three dimensions. However, there are several other uses for such a combination of position sensors as that used to sense the head position. One such use is to describe a "path" of a rigid object through space. For example, suppose we want to record a sequence of positions (orientation and displacement) for an airplane to follow in an animated film. Describing such paths is a very tedious task in computer animation. But this position sensor would allow us to "fly" the display mechanism around the room, recording its position each frame time. The results can then be used to position the aircraft in the animated film.

A very difficult problem in 3-D computer graphics is creating a 3-D data base for objects to be displayed. Clearly, the head mounted display sensing mechanism can be used to make measurements of this type. The position sensing mechanism used in this display has a precision of approximately 1/16 inch at the lower, or viewing, end. To use it as a digitizer, we can attach a sharp point, or scribe, to the viewing mechanism and calibrate the position of the point. Then the head mounted display can be used to find the 3-D coordinates of a set of points on the object, so long as the object is large enough that the 1/16 inch precision is not an important factor in the accuracy of the measurements.

IV.3 Lessons in Hardware.

The original goal of this work was to build a system for designing B-spline surfaces in a three dimensional environment. A secondary goal was to utilize the head mounted display. Both of these goals required that a considerable effort go into building and repairing special purpose display equipment.

The hardware in this system comes from a variety of sources. The Clipping Divider was designed and built here at the University of Utah, and the Matrix Multiplier was built by Systems Concepts, a special purpose hardware company in San Francisco. The head mounted display was built at Harvard University in 1967 when Ivan Sutherland was on the faculty there, and the remainder of the system was built by various etudents and etaff at the University of Utah.

A very significant problem with a hardware syetem of the sort used to implement this design system ie the difficulty in maintaining it. Certainly it is possible to have maintainence contracte with each of the separate vendors for the equipment, but how does one isolate which component is bad? Either one maintainence engineer should be responsible for maintaining the whole system, or at least the line drawing part of it should be purchased from one manufacturer. Then that one company can be responsible for keeping it operational.

REFERENCES

[1] Coons, S. A., Surfaces for Computer-Aided Design of Space Forms, M.I.T. Project MAC TR-41, June 1967.

[2] Bézier, P. E.," Numerical Control in Automobile Design and Manufacture of Curved Surfaces", Curves and Surfaces in Engineering, I.P.C. Science and Technology Press, Guildford, England, 1972.

[3] Sabin, M. A., The British Aircraft Corporation Numerical Master Geometry System. Proc. Roy. Soc. Lond. A321, 197-205, (1971).

[4] Armit, A. P., Computer Systems for Interactive Design of Three-Dimensional Shapes, Ph.D. Thesis, University of Cambridge Computer Laboratory, November, 1970.

[5] Ferguson, J., "Multivariable Curve Interpolation", Journal of the ACM, Vol. 11, No. 2, 221-228, April, 1974.

[6] Riesenfeld, R. F., Applications of B-spline Approximation to Geometric Problems of Computer Aided Design, Ph.D. Thesis, Syracuse University, 1972. Published as University of Utah UTEC-CSc-73-126.

[7] Sutherland, I. E., "A Head-Mounted Three-Dimensional Display," AFIPS Conference Proceedings, Vol. 33 (Fall Joint Computer Conference, 1968) pp. 757-764.

[8] Forrest, A. R., Computational Geometry, Proc. Roy. Soc. Lond. A321, pp. 189-195, 1971.

[9] Gordon, W. J., and Riesenfeld, R. F., Bernstein-Bézier Methods for the Computer-Aided Design of Free-Form Curves and Surfaces, Journal of the ACM, Vol. 21, No. 2. 293-310, April 1974.

[10] Forrest, A. R., On Coons and Other Methods for the Representation of Curved Surfaces. Computer Graphics and Image Processing 1, pp. 341-359 (1972).

[11] Bézier, P.E., "Procede de Definition Numerique des Courbes

et Surfaces Non Mathematique; Systeme UNISURF.", Automatisme 13, May 1968.

[12] S. A. Coons gave a seminar at the University of Utah in January 1972 in which he discussed the B-spline work being carried out by R. F. Riesenfeld of Syracuse University.

[13] Watkins, G. S., A Real-Time Visible Surface Algorithm, Ph.D. Thesis, University of Utah, 1970.

[14] Catmull, E. and Rom, R., "A Class of Local Interpolating Splines", Computer Aided Geometric Design, ed. by R.E. Barnhill and R.F. Riesenfeld, Academic Press, 1975.

[15] Blanchard, M., Clipping Divider documentation, University of Utah, 1974.

[16] SAC Tablet documentation, Universtiy of Utah.

[17] Burton, R. P., Real-time Measurement of Multiple Three-Dimensional Positions, Ph.D. Thesis, University of Utah, 1973.

[18] Case Western University has a halftone picture system, built by E and S. Computer Company, that provides real-time smooth-shaded pictures of environments with up to 1500 polygons.

[19] Copies of this movie may be obtained from the University of Utah Computer Science Library.

[20] Coons, S. A., "Surface Patches and B-spline Curves", Computer Aided Geometric Design, ed. by R.E. Barnhill and R.F. Riesenfeld, Academic Press, 1975.

[21] Peters, G.J., "Parametric Bi-Cubic Surfaces", Computer Aided Geometric Design, ed. by R. E. Barnhill and R. F. Riesenfeld, Academic Press, 1975.

[22] Catmull, E., Ph.D. Thesis, University of Utah, 1974.

APPENDIX 1

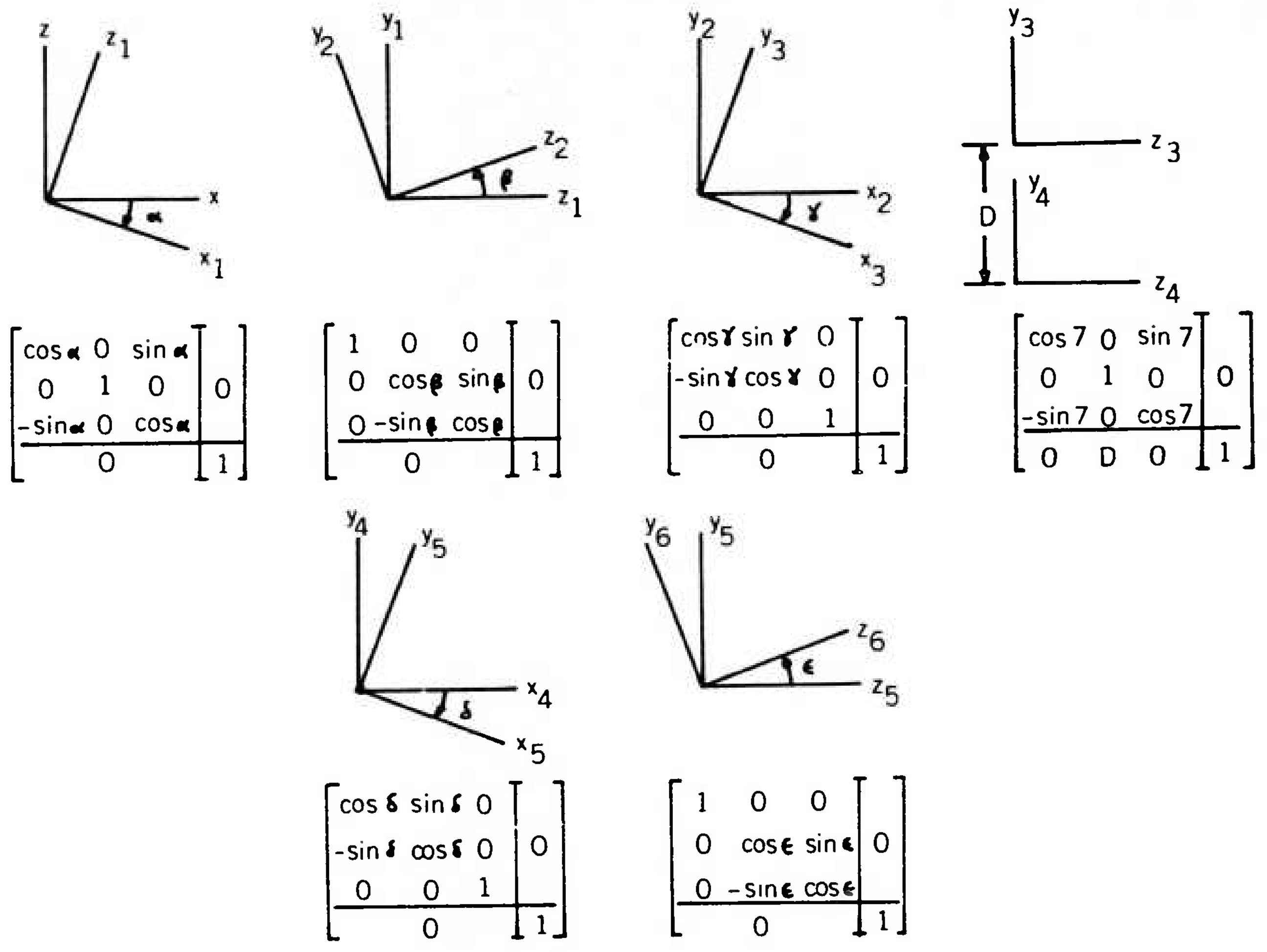
POSITION DETECTION COMPUTATIONS

A1.1 Head-Mounted Display Computations.

The mechanical position sensor for the head-mounted display measures the 6 independent quantities necessary to determine the position of one rigid body, the display, relative to another, the room. There are 5 angular measurements and 1 length measurement (see Figure 3.4).

The first angular measurement measures the rotation of the head-mounted display relative to the room. The coordinate system attached to the ceiling is labeled (x,y,z). The (x_1,y_1,z_1) system is attached to the head-mounted display. If the head-mounted display is rotated in the sense shown in Figure A1.1, then the matrix shown is the appropriate transformation to take points described in the (y,x,z) system to points described in the (x_1,y_1,z_1) system.

Also shown in Figure A1.1 are the transformations for the rotations at the upper universal joint. The first of these



$$\left[\begin{array}{ccc|c} \cos\alpha & 0 & \sin\alpha & \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & \\ \hline & 0 & & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & \\ \hline & 0 & & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} \cos\gamma & \sin\gamma & 0 & \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & \\ \hline & 0 & & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} \cos 7 & 0 & \sin 7 & \\ 0 & 1 & 0 & 0 \\ -\sin 7 & 0 & \cos 7 & \\ \hline 0 & D & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|c} \cos\delta & \sin\delta & 0 & \\ -\sin\delta & \cos\delta & 0 & 0 \\ 0 & 0 & 1 & \\ \hline & 0 & & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} 1 & 0 & 0 & \\ 0 & \cos\epsilon & \sin\epsilon & 0 \\ 0 & -\sin\epsilon & \cos\epsilon & \\ \hline & 0 & & 1 \end{array}\right]$$

Figure A1.1 Illustrations of separate matrices used in the head position matrix computation.

rotations is a rotation about the x_1 axis, and the second is about the z_2 axis.

The next transformation uses the length measurement D. This represents a translation along the y_3 axis Also shown is a correction rotation about the y_3 axis. This corrects for a 7 degree misalignment between the axes of the upper and lower universal joints.

The two transformations for the lower universal joint are shown next. There is also one final transformation to account for the translation from the position of the universal joint to the position of the eye.

When all of these matrices are multiplied together, the room to eye transformation is formed. This matrix transforms points described in the left hand coordinate system at the top of the room to points in the left hand coordinate system at the eyeball.

A1.2 Wand Computations.

The wand position is computed from the lengths of wire extending from the three housings that enclose the shaft encoders. Figure A1.2 shows the coordinate system relative to

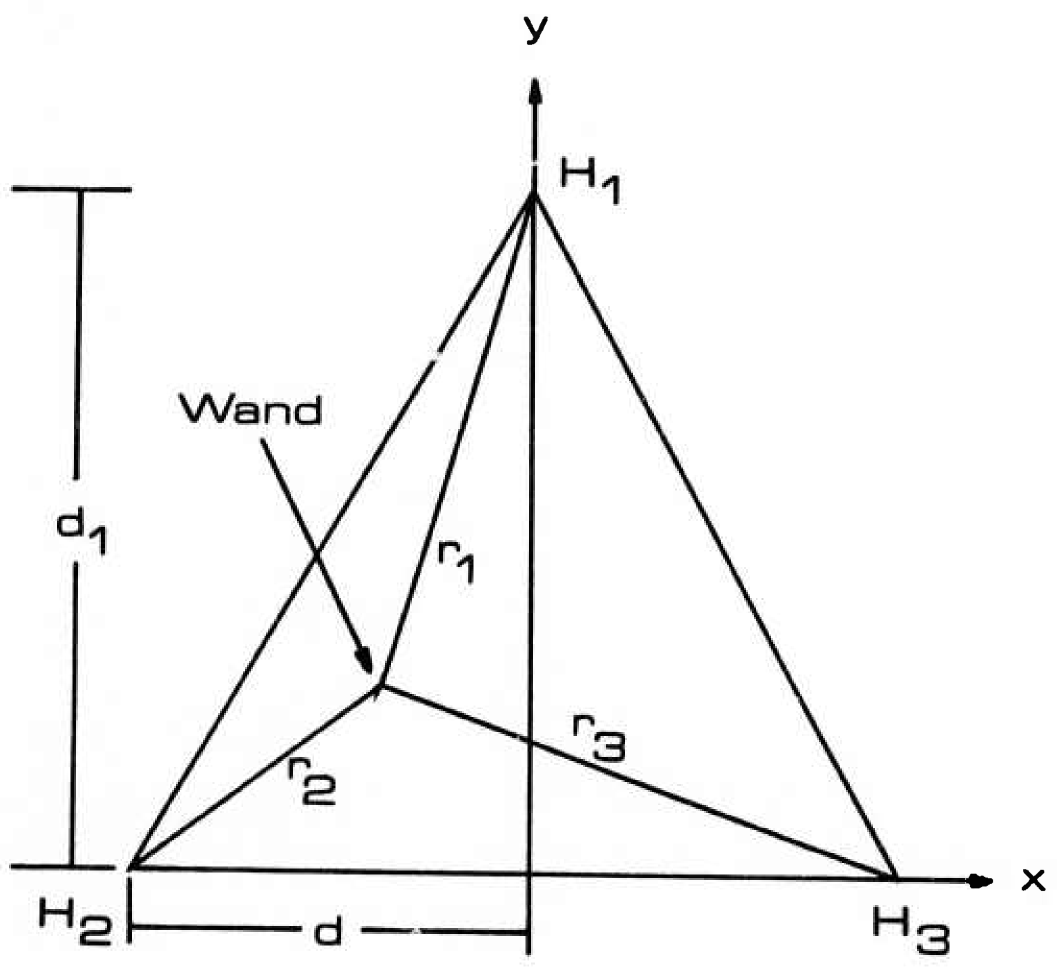


Figure A1.2 Coordinate System for Wand Computations

which the computations are done. The sensor housings are labeled H_1, H_2, H_3, and the distances to the wand are r_1, r_2, and r_3. The wand position is found from the intersection of three spheres. From the equation of a sphere we find:

$$r_1^2 = x^2 + (y-d_1)^2 + z^2,$$

$$r_2^2 = (x-d)^2 + y^2 + z^2,$$

and $$r_3^2 = (x-d)^2 + y^2 + z^2.$$

Subtracting the r_3 equation from the r_2 equation we obtain

$$x = (r_2^2 - r_2^2) / 4d.$$

Then eliminating z from the first two equations we get for y:

$$y = (r_2^2 - r_2^2 - 2dx + 2d^2) / 2d_1.$$

Then z is obtained from any of the three equations using the previously computed values of x and y. Since a square root must be taken in any case, the sign is negative because the wand is always assumed to be below the plane containing the housings.

Bliss routines to find the head position matrix and the wand position are available in the <GRAPHICS> directory at UTAH-10 under file name HMDW.BLI.

APPENDIX 2

3-D DIGITIZING IN REAL-TIME

It was mentioned in Chapter 4 that a real-time multi-point 3-D digitizer would make it possible to eliminate the mechanical position sensors for the head mounted display. Here we look at a possible method for finding 3-D coordinates in real-time.

Reticon Inc. now produces a linear array of photodiodes that has 512 light sensitive elements. Each element is 1 mil square, so the entire sensitive portion of the device is 0.001 inches wide and 0.512 inches long.

If we mount the device with a cylindrical lens in front of it as shown in Figure A2.1, then a portion of the room is divided into 512 planes that extend out from the focal line of the lens. If an LED is turned on somewhere in this field, some of the light from this LED will fall onto the array, and one of the photodiodes will sense it. Once we have determined which photodiode in the array was discharged, we know in which plane the LED lies. The 3-D coordinates of the LED are determined by

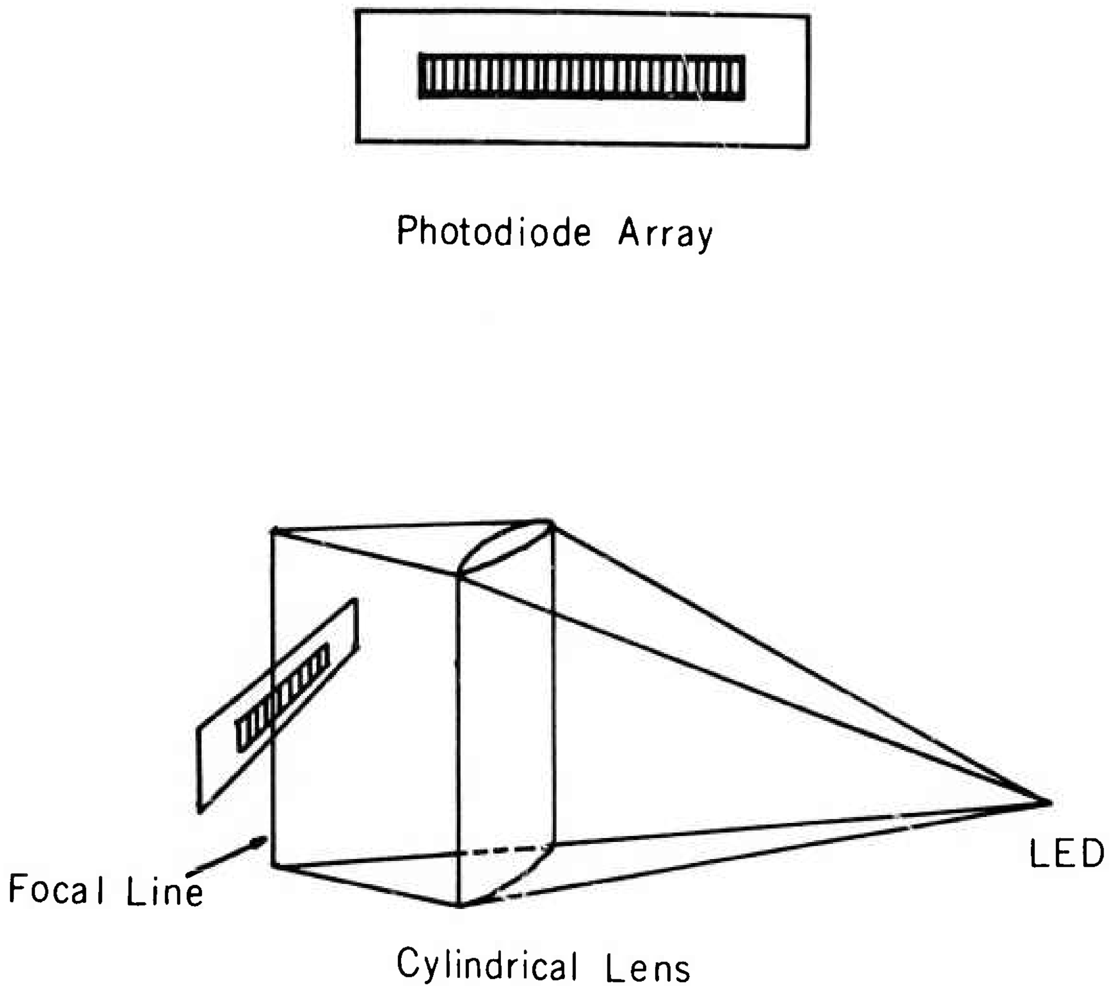


Figure A2.1 Configuration for finding the one dimensional coordinates of an LED.

using 3 or more of these systems. fhe point of intersection of 3 or more planes gives the position of the LED.

The problem with this approach is in the time required for the light from the LED to discharye the photodiode. If the diode cannot be discharged in a time somewhat less than 1/30 second, then this device is not of much value for real-time poeition detection.

The energy required to saturate one of the photodiodes is 58.4*10-$6 ergs. A typical LED radiates 0.0763 microwatts per equare centimeter at 10 feet. If we do not have the lens to focus the light from the LED, then the incident area is the size of one of the photodiode elsements, or 10-$6 square inches. In this case the energy incident on it is 4.88*10-$6 ergs/sec. This gives 11.95 seconds for the time required to saturate the photodiode. If we can use a 1 inch wide aperture in front of a cylindrical lens, then this time is reduced to approximately 1/50 second, which is closer to the required time.

The obviouc problem with this approach is in the optics. If there were some optical system that would focus the entire field of view in the room onto a line, then we would gain several orders of magnitude in energy incident on an element in the array, thereby reducing the integration time. A simple cylindrical lens does not do this. It will focus one point in the room onto a line in the image space of the lens, but if the point is moved, the line moves. The result is that a cylindrical lens focuses the room onto a plane. Nonetheless, this approach appears to be a promising method for real-time position detection.

ACKNOWLEDGMENTS

This study came about as the result of a seminar given by Steven A. Coons at the University of Utah in which he gave a stimulating presentation of the B-spline work being carried out at Syracuse University by R. F. Riesenfeld. I thank Steve. I also thank Ivan Sutherland for encouraging me to get aquainted with Steve, for encouraging me to pursue this work, and for being my committee chairman. It has been stimulating to work with him.

I also thank David Evans and Rich Riesenfeld for sitting on my thesis committee. I could not have made a better choice.

This work required a considerable effort in hardware design. I thank Malcolm Blanchard for his willingness to implement the design of the Clipping Divider. I especially thank Martin Newell, without whom this project might not have been completed. Martin was a source of constant encouragement, help, and ideas. In many ways, he was my resident advisor.