DEVELOPMENT OF SURFMAP, A THREE-DIMENSIONAL SURFACE MAPPING SYSTEM

By

Craig Alan Olbrich

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

1992

ABSTRACT

DEVELOPMENT OF SURFMAP, A THREE-DIMENSIONAL SURFACE MAPPING SYSTEM

By

Craig Alan Olbrich

SURFMAP is a system developed to digitize three dimensional surfaces and provide a mathematical representation of these surfaces. A coordinate measuring machine based upon laser triangulation concepts was assembled. Control software was developed to automate the digitization process and provide a user-friendly interface. Analysis software was developed to provide necessary information about the surfaces digitized.

SURFMAP has three levels of usage: A manual control coordinate measurement device allows for interactive digitization of surfaces and the mapping of areas. The automatic coordinate measurement device measures a grid of points across a user-defined area. The combination measurement device takes a surface as input and can either map a finer grid or use an adaptive search algorithm to provide only new information about the surface.

Examples of each aspect of the system are provided. The relative strengths and weaknesses of the system are discussed. Also included is a user's manual.

It would take a much less humble
person than I to claim these ideas as solely
mine own. I dedicate this work to all
those who have inspired me.

Inspiration, like loose change,
shows up in some very unlikely places.

Acknowledgments

I would like to express my appreciation for the financial assistance of NASA Lewis Research Center. I would also like to thank Dr. E. Goodman for his (quick!) assistance and support. Dr. C. Somerton deserves thanks for making his teaching a priority and not a chore. And of course, without Dr. H. Schock, none of this would have been possible.

I would also like to acknowledge Mike, whose boundless supply of energy kept me going when all sane people were home in bed. I would also like to thank all of those involved in Formula SAE, easily the most educational experience I have had.

For those whom I have neglected to mention, we do not live in a vacuum. I deeply indebted to those people who create the environment capable of sustaining those of us who thirst for knowledge.

Table of Contents

Table of Contents, Cont.

List of Figures

List of Tables

Nomenclature

A = Area

$B_{n,i}$ = Bernstein Basis Function

Ds(u)/du = total derivative of s with respect to u

H = height

H_i^3 = cubic Hermite functions

J_T = Jacobian Matrix

$N_{i,k}$ = B-Spline Blending Function

P(u) = B-Spline Curve

Q(u,v) = Surface Patch Eqn.

SA = Surface Area

$\overline{V}$ = Vector of control point vertices

V = Volume

V_i = Control Polygon vertex location

a_0, a_1, etc = general constants

h(u) = algebraic form of the cubic spline

$h_{i,j}$ = geometry array for hermite surface patches

k = order of curve

n = number of control points - 1

$\overline{r}$ = vector of data for interpolation

x,y,z = general three-space coordinates

Chapter 1

Introduction

Numerical control machining centers are becoming more common in the manufacturing sector. These systems are excellent for performing repetitive and precise machining processes that would normally be too tedious or complicated for an operator. Until the late 1950's when it became possible to control machine tools with computer-based control the same jobs were accomplished by what is called copymilling. Copymilling is the analog process of taking a master model and using that to prescribe the motion of the machine tool. Copymilling generates its program by measuring a master model and translating that directly into motion commands for the machine tool. With the advent of computer controls and computer generated machining instructions the question became how to represent the parts to be manufactured on the computer. For copymilling the parts were "prototyped" with wooden master models, but for numerical control machining these models needed to be represented in computer language, i.e. mathematically.

Two distinct methods exist for the generation of the mathematical models. The method that is primarily used today is that of design. The curve, or surface, is generated from a mathematical model. This is generally an iterative process with the designer comparing various curves or surfaces. The second method is that of fitting curves to given data. Fitted data are usually used to describe real objects for part checking, copying or other similar processes. The data for this second

process is usually generated by a coordinate measurement device. Coordinate measurement machines digitize surfaces. They measure a number of points on this surface, usually in a grid pattern, which represent the surface geometry. Many techniques exist for the generation of a mathematical model from the data provided by these two methods. A survey of these methods is presented in Chapter 2. The rest of this work is spent on the details of the design and implementation of an automated Coordinate Measuring Machine. This system, in addition to analysis programs included, digitizes surfaces and generates an appropriate mathematical representation.

These mathematical representations can then be analyzed using the simple analysis programs discussed in Chapter 5 or transferred to other applications. Possible uses range from using the surfaces as input for NC machining processes or part checking to measuring boundaries to be used as input for Computational Fluid Dynamics simulations.

Chapter 2.

Background

2.1 CMM information

A coordinate measurement machine is designed to measure the location of a specific point to within pre-determined error tolerances. These systems are used for a variety of purposes. The uses range from part checking in manufacturing environments such as automotive fit and finish processes to measuring the thickness of solder on circuit boards. The systems usually consist of a positioning arm of some type which holds a distance measurement device. The distance measurement device then measures the location of a point on the surface being inspected. Distance measurement devices can be divided into two groups, contact and non-contact devices. Contact measurement devices usually use a pointer or ball roller to indicate the position to be measured. The non-contact devices are typically optical systems and these can be divided into two subgroups, active (structured light) and passive (stereo disparity).[Jalkio et al. '85] When the distance to be measured is much greater than the wavelength of the light ($>10^7\lambda$) used, time-of-flight systems are employed. When the distances are very small($<<\lambda$) interferometry techniques comparing the incident wavefront with a reference wavefront are typically used.[Mercer and Beheim '91] and [Schmitt and Barsky '86]. Structured light devices are used for distances in between the previous two mentioned ($10^{-1}\lambda < Z < 10^6\lambda$). Typical commercial systems use structured light triangulation because these

systems have the best potential for fast, accurate, and inexpensive systems.

2.2 Curve and Surface representation

2.2.1 Introduction

A mathematical representation of a curve or a surface can be expressed in functional form like this:

$$y = f(x) \quad \text{(curve)} \tag{1}$$

$$z = f(x,y) \quad \text{(surface)} \tag{2}$$

These are called explicit definitions. For the uses presented in this work, and in general CAD implementations the parametric representations are more useful.

2.2.2 Curves

The parametric representation of a curve and a surface are as follows:

$$\left.\begin{aligned} y &= f(u) \qquad (3)\\ x &= g(u) \qquad (4) \end{aligned}\right\} \text{(curve)}$$

$$y = f(u,v) \tag{5}$$

$$\left.\begin{array}{l} x = g(u,v) \\ \\ z = h(u,v) \end{array}\right\} \quad \text{(surface)} \tag{6,7}$$

The most obvious advantage to parametric representation is their ability to express infinite derivatives in the coordinate space. Other advantages are that the implicit representations are independent of coordinate system and they are easily normalized between its boundaries. They are convenient for generation of machine tool paths since the parameter can be represented as a corresponding function of time, to get the machine tool's motion as a function of time. [Quilin '87, pg. 46-48]

There are many different methods for defining parametric curves. Three popular and useful methods are; The cubic spline, the Bezier spline, and the Basis spline representation.

Cubic splines are typically defined as follows:

$$y(x) = a_0 + a_1*x + a_2*x^2 + a_3*x^3 \tag{8}$$

for a plane curve and the parameters are defined as:

$$
\begin{aligned}
a_0 &= y(0) \\
a_1 &= y'(0) \\
a_2 &= (3y_1 - 3y_0 - 2y_0' - y_1') \\
a_3 &= (2y_0 - 2y_1 = y_0' + y_1')
\end{aligned}
\tag{9}
$$

where y_0, y_1 are the values of the endpoints and y_0', y_1' are the slopes of the curve at the endpoints.

The generalization of the cubic into three dimensions is:

$$
\begin{aligned}
z &= f(x) \\
x &= g(y)
\end{aligned}
\tag{10}
$$

and the parametric version is:

$$
\begin{aligned}
z &= f(x(y(u))) = f(u) \\
x(y(u)) &= g(y(u)) = g(u) \\
y(u) &= h(u)
\end{aligned}
\tag{11}
$$

This can be considered the two projections onto the z-x axis and the x-y axis. The algebraic form of the cubic in parametric form is:

$$h(u) = a_0 + a_1*u + a_2*u^2 + a_3*u^3 \tag{12}$$

which works for all three components. Normally the separate coordinates are represented by different equations, not functions of other

equations. Please note that this is the same form for the 2-d case but extended into 3-d.

Bezier spline curves are represented by the vertices of the Bezier Polygon. The parametric form looks like this:

$$x(u) = \sum_{i=0}^{n} B_{n,i}(u)\, V_i \tag{13}$$

[Quilin, '87]
where $B_{n,i}$ are the Bernstein Basis functions, or Bernstein polynomials.

$$B_{n,i} = \binom{n}{i} u^i (1-u)^{n-i}$$

and

$$\binom{n}{i} = \begin{cases} \dfrac{n!}{i!\,(n-i)!} & 0 \le i \le n \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

V_i are the locations of the vertices of the Bezier polygon.
[Rogers '87 pg. 291-292] u is the parameter 0 <= u <= 1

Bezier curves have several important properties. The maximum degree of the polynomial defining the curve is one less than the number of vertices in the polygon. So a cubic curve segment is defined by four control polygon points. The first and last points of the curve are coincident with the first and last points on the defining polygon. The

tangent vectors at the ends of the curve are parallel to the polygon spans at those points. The curve is invariant to affine transformations, which means that if you want to transform the curve, you can transform the control points. It is also important to note that the parameter values for the Bezier curve typically ranges from 0 to 1.

Basis splines, or B-splines, were first mentioned in the literature by Schoenberg in 1947. Gordon and Riesenfield [Gordon et al. '74], [Riesenfield, '73] showed in 1971 that B-splines were actually the generalization of Bezier curves. B-splines are piecewise polynomial curves represented similarly to Bezier curves.

$$P(u) = \sum_{i=1}^{n+1} V_i \, N_{i,k}(u) \qquad (15)$$

Where the B_i are the position vectors of the defining polygon. They are often called control points or deBoor points [deBoor]. They form the deBoor polygon. The $N_{i,k}(u)$ are piecewise polynomials of degree 'n' defined over a knot vector u0<u1<u2<u3<u4<... Mansfield, Cox and deBoor [Böhm et al. '84, pg 16] independently found a recursion relationship for calculating the $N_{i,k}$'s . The recursion formula is:

(Note: $N_{i,k} = N_i^k$)

$$N_i^r(u) = (u - u_i)\frac{N_i^{r-1}(u)}{u_{i+r} - u_i} + (u_{r+i+1} - u_{i+1})\frac{N_{i+1}^{r-1}(u)}{u_{i+r+1} - u_{i+1}} \tag{16}$$

where

$$N_i^0 = \begin{cases} 1 & u \in [u_i, u_{i+1}] \\ 0 & u \notin [u_i, u_{i+1}] \end{cases}$$

The properties of the B-splines are similar to those of the Bezier curves with a difference being the maximum order is equal to the number of polygon vertices. Another difference that is important is that B-splines have the property of local shape control, whereas Bezier curves do not. B-splines curves can be parameterized very flexibly, providing another method of control for the designer.

The knot vector is important to the definition of B-splines and there are two types; uniform and non-uniform with both types being either open or periodic. Uniform knot vectors have equally spaced parameter values while non-uniform do not. The knot vector is a vector containing the parameter values for the associated control polygon. The length of the knot vector is a function of the number of control points (n+1) and the order of the curve (k). [Hawkins, '87]

$$\text{Length} = n+1+k \tag{17}$$

The number of segments a curve will have is as follows:

$$\text{Segments} = n-k+2 \tag{18}$$

The parameter values for each segment is best illustrated by an example:

> Curve: 3rd order quadratic (k = 3), with 4 control points (n=3) gives 3-3+2 segments.
>
> A sample knot vector is:[0,0,0,1,2,2,2].
> The first segment runs from parameter values [0,1], the second from [1,2]

Note that by repeating the first and last knot values k times the curve is forced through the control points at the beginning and end. It is important to note here that when the number of polygon vertices is equal to the order of the spline curve and an open uniform knot vector is used the B-spline basis reduces to the Bernstein basis and the curve is a Bezier curve with one segment and a parameter range of $t=t_0$ to $t=t_f$. A sample knot vector might look like [0,0,0,0,1,1,1,1] for a cubic(k=4) curve with 4 control points.

Other types of splines appear in the literature. Beta-splines were developed by Barsky[Barsky, '81] and have the property of being visually C^2. Nu-splines were developed by Nielson and Gamma splines were developed by Böhm. Nu-splines and Gamma splines are G^2 continuous, which is a relaxation of the C^2 continuity constraints. These splines were developed because of the need to relax the geometric constraints. The C^r continuities were just not flexible enough for the designer and the idea of geometric continuity was developed. See [Farin, '90] for more details.

Cubic splines have the advantage of being easy to use and work very well for simple cases. They are however, not a local description and modification requires recomputation of the entire spline. Another problem is that the parameters used for its description are non-intuitive. The Bezier curve representation overcomes this problem by making the specified parameters a set of polygon vertices. This is much more intuitive, however Bezier splines are merely a subclass of Basis splines which have the added advantage of having local shape control. The Bezier curve is a one-segment B-spline curve that interpolates the endpoints of the control polygon. A B-spline can have multiple segments and the property of interpolation can by controlled by the multiplicity of the knot values. It is also worth noting that the Initial Graphics Exchange Specification(IGES) adopted Non-uniform Rational B-splines as their standard for curve and surface description in 1986. The computer storage requirements of the different types are also different. The cubic Hermite requires 7n+1 real numbers of storage, the Bezier 3n+3, and the B-spline 4n+7.[Farin, '90]

2.2.3 Surface Representation

Surface representation schemes generally expand directly from the curve schemes. The problem is how to represent a surface patch as a function of two parameters. A typical implementation will have a surface made of several patches. The mathematics of surface representation has been developed around three or four sided patches although a five sided patch has been discussed by Gregory ['86]. Each boundary of a patch is

a corresponding curve of one of the types mentioned above.

S.A Coons [Coons, '64] introduced two types of surface patches, the bilinearly blended and the bicubically blended. Bilinear blending is the linear interpolation of two identically parameterized curves. A four-boundary region is defined by interpolating in the two parametric directions. This has the drawback of two surfaces only having C^0 continuity at the boundaries. Coons improved this to C^1 by using cubic Hermite polynomials as blending functions.

The hermite form of the bicubic patch is:

$$Q(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} h_{ij} H_i^3(u)\, H_j^3(v) \quad 0 \le u,v \le 1 \qquad (19)$$

where $H_i^3(u)$ are the cubic hermite functions, see Farin [90] for the derivation, and

$$[h_{ij}] = \begin{bmatrix} x(0,0) & x_v(0,0) & x_v(0,1) & x(0,1) \\ x_u(0,0) & x_{uv}(0,0) & x_{uv}(0,1) & x_u(0,1) \\ x_u(1,0) & x_{uv}(1,0) & x_{uv}(1,1) & x_u(1,1) \\ x(1,0) & x_v(1,0) & x_v(1,1) & x(1,1) \end{bmatrix}$$

The x_u notation represents the partial derivative with respect to u. [Farin, '90 pg. 289]

These surfaces are C^2 continuous and they can be combined with other patches in C^1 continuity.

The corresponding Bezier surface description is as follows.

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} V_{i,j} B_{n,i}(u) B_{m,j}(v) \qquad (20)$$

where $J_{n,i}(u)$ and $K_{m,i}(v)$ are the Bernstein Basis functions un the u,v parametric directions, and the $V_{i,j}$'s are the vertices of the defining polygon net.

The surface will have continuity of two less that the number of polygon points in that direction, and surface patches can be combined with C^r continuity by analogy with the case for curves.[Böhm, '84]

The B-spline representation of a surface is similar to that of the Bezier form in that it is an extension of the curve form.

$$Q(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} V_{i,j} N_{i,k}(u) N_{j,l}(v) \qquad (21)$$

where the B_{ij} are the control polygon points and the N_{ik}, N_{jl} are the B-spline basis functions in the u,v directions respectively.

The continuity of the B-spline surface is either C^{k-2} or C^{l-2} depending upon the direction in question.

2.2.4 Surface Fitting

In general, surface information from digitized surfaces is available in two forms; gridded or scattered. There are two possible techniques that can be employed to generate a surface. The surface can be

interpolated, which produces a surface that passes through each defined point, or it can be approximated, which only approximates each data point to within some error. Interpolation schemes generally produce some oscillation, whereas approximation schemes are more fair. Farin['90. pg. 386] defines fair curves as those with continuous curvatures and consisting of only a few monotonic pieces. Rogers['87, pg 251] takes no attempt at a definition except to say that they are 'pleasing to the eye.' Farin et al.['87] present a process for fairing that essentially uses curve subdivision techniques for making the curvature smoother. Fairing of surfaces is a complex question in itself and the issue of an appropriate method is left for others to determine. Generally, the fewer control points defined, the more fair a surface is going to be.[Rogers, '87 pg 348] The literature seems to indicate that there is an optimal relationship between the number of control points and the number of data points. The optimal condition would be specified by the fairing criteria. It should also be noted that not all of the surfaces should be fair. Some surfaces have slope and/or curvature discontinuities. A method for determining these situations automatically has yet to be found by this author.

The problem with scattered data is actually the same problem that the gridded data methods solve with approximations, that of determining the parameter value for each point in question. Given the parameter values the surface interpolation is easily possible, without it they must be approximated. This is usually an interactive process with the parameter

values adjusted depending upon surface fit. In gridded data format the parameter estimations work well because the parameters can be as evenly spaced as the points they represent. For scattered data this is a more challenging question. The methods for curves will not extend to surfaces because of the inability to determine constant parameter values in the first place. See Böhm ['84 pg. 49-50] for a discussion and description of methods for scattered data. Non-uniform grids are considered gridded in this sense. The estimation of parameters is generally possible, assuming the boundaries of the region are known.[Rogers, '87 pg 456] The adaptive grid is meant to show the most information with the least amount of data.

To fit a curve or a surface to some known data the computer must follow the general algorithm: (Tensor-Product method)

1) specify points on the curve or surface patch.
2) calculate/estimate parameter values.
3) calculate corresponding weighting functions.
4) Solve,

$$[X] = [A][C][B] \tag{22}$$

where A, B are the matrices of Basis functions for the two directions and C is the matrix of polynomial locations. X is the known data.

Thus:

$$[C] = [A]^{-1}[X][B]^{-1} \tag{23}$$

For the case of a curve [B] = [I]. If the number of data points is

the same as the number of control points desired, the curve or surface, may oscillate. A solution to this problem is to provide more data. This gives you an overdetermined system because the method of solution is to multiply by the transpose of [A] and get

$$[A]^T[X] = [A]^T[A][C]$$

which can be solved by using a Cholesky factorization.

$$[C] = [[A]^T[A]]^{-1}\,[A]^T[X] \tag{24}$$

The resulting curve minimizes the sum of all squared distances, $|s(u_i) - x_i|$.

Methods exist in which each control point is given a weighting factor which controls the variation of the surface on the patches.[Foley, '87] This method appears useful for relatively small amounts of data but no development was found for B-splines so the concepts are left for others to pursue. As noted in Foley, the splines are not local and solving the linear system is non-trivial.

2.3 Using the Surfaces

Clearly the technology exists to examine the splines for their usefulness and correctness before any expensive machining or manufacturing procedures are undertaken. Viewing the resulting surfaces and curves in a graphics display proves to be a valuable tool. The curve or surface can be examined immediately after creation to determine

whether it meets the required criteria. Offset spline curves (in 3-d) are often used for machine tool paths in NC machining. Processes also exist for the graphical verification of the cut, after tool shape has been taken into account. [Wysocki, '87] and [Oliver, '86].

The spline surface representations can also be used to perform required numerical analysis of the surfaces. It would be possible to integrate the description of the surface and produce the Surface area and Volume of the surface in question. The formula for Surface Area is:

$$SA = \iint_R \left| \partial f/\partial u \times \partial f/\partial v \right| du\, dv \tag{24}$$

and the formula for Volume is

$$V = \iint_R f(x,y)\, dx\, dy \tag{25}$$

where f(x,y) is the functional representation of the surface. To calculate this integral for the parametric surface a change of variables must be used and the formula is as follows:

$$\int_x \int_y f(x,y)\, dx\, dy = \int_u \int_v z(u,v) \left| J_T \right| du\, dv \tag{26}$$

where z(u,v) is the parametric representation of the height and J_T is the Jacobian,

$$J_T = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix} \tag{25}$$

The same method is applicable for the case of calculating areas under curves.

$$A = \int f(x)\,dx = \int z(u) \left| \frac{\partial x}{\partial u} \right| du \tag{26}$$

Note that all of these methods require the calculation of the partial derivatives of the parametric equation with respect to u,v. These derivatives exist for the Basis and cubic spline equations. The form of the first derivative for B-splines is as follows:

$$\frac{Ds(u)}{du} = \sum_{i=0}^{n} V_i^{(1)} N_i^{n-1}(u) \tag{27}$$

where

$$V_i^{(1)} = n \frac{V_i - V_{i-1}}{u_{i+n} - u_i} \tag{28}$$

V_i = control polygon coordinates
n = order of curve
i = 0,n

See [Böhm, '84] for the other types of curves.

The next process involving spline surfaces is that of surface intersections. This is a very complicated process because two surface patches intersect in a curve whose degree is much higher that of the two original ones. For example a pair of bicubic patches intersect in a curve of degree $4*3^4 = 324$. Sederburg and Meyers['86] show they cannot be expressed exactly by polynomial parametric curves of the proper order. So clearly numerical approximation to these curves is needed.

Two methods to accomplish surface intersections are popular in the literature. Subdivision methods divide two intersecting surface patches into piecewise linear approximations and use the intersection of the faces. Marching methods find a single point on the curves and follow the curve solving three polynomial equations for each point. The Marching techniques assume a method for finding all of the intersection curves for a given patch. Sederburg['88] presents a method of finding patches with no closed loop intersection curves. These methods require significant computation for simple use in part checking where simpler methods might serve adequately.

Chapter 3.

The SURFMAP system.

The spline representation chosen for use in this project is the Basis spline. It was chosen because of its flexibility in interpolating many types of surfaces. Local shape control is important to the fitting process to speed implementation and computation. Another major factor in the choice was the universality of the representation. As mentioned in Chapter 2 the Basis splines have the powerful Bezier splines as a special case, and the non-uniform, rational B-splines are the IGES standard for surface description. A further goal of the project was portability of data, as a full fledged analysis-design system was not planned for this work.

The tasks planned for this system were digitization of surfaces and mathematical representation of these surfaces. Simple analysis routines were developed to take advantage of the information. The theory suggests some methods for improving spline fits in that gridded data is more readily analyzed than the corresponding scattered data. What is not obvious is the advantage of being able to change the grid spacing for regions of high curvature or other important surface features, in essence subdividing the surface before any spline fit algorithm is attempted.

The system for digitizing surfaces presented in this work is very powerful but has its limitations nonetheless. It is relatively slow (~ 2 seconds/data point) in comparison to other systems available. [Schmitt '86] The algorithms presented in this paper are an attempt to automate the digitization of surfaces and reduce operator involvement. The system

is presented as developed. The preliminary steps of the process are presented when shown to be useful.

SURFMAP is a system developed at the MSU Engine Research Facility that digitizes surfaces and provides the mathematical representations of these surfaces, allowing for further analysis using the simple analysis programs contained within or analysis provided by other software packages. SURFMAP contains the routines that perform the control, data acquisition, storage and analysis of these surfaces. These programs, in conjunction with the necessary hardware described below, make up a system which creates computer representations of surfaces and uses these representations to perform simple analysis of these surfaces. SURFMAP digitizes the surface using a non-contact laser measurement device as the measuring pointer. This digital image of the surface is then interpolated using B-splines and the result is a surface spline approximation to the real surface. This surface data can then be used as input to a CAD/CAM system for further design or numerical control machining, for example. Other uses include using the surface as input to a computational fluid dynamics simulation, or computational calculations such as surface area or volume.

The goals for SURFMAP were to develop an automatic system that could be used to generate surface information with a minimum amount of operator involvement. This system starts with a real surface, and with the use of subprograms included generate the information in the necessary forms, (i.e. CAD formats, image formats, etc.) An additional

goal was to develop a system flexible enough to allow for the testing and development of algorithms and procedures for the automatic measurement of surface geometry.

Chapter 4.

The Equipment

The equipment that makes the digitization of surfaces possible has five main parts. They are: The Point Range Sensor, The traverse table, The Modulynx stepper motor control system, The Sony encoder system, and The Zenith TurboSport 386 laptop computer. The Point Range Sensor is mounted on the traverse table in a vertical direction. The traverse table is moved by stepper motors connected to each of three axes. These Stepper motors are controlled by the Modulynx control computer. The current position of the table is measured by linear position encoders connected to a Sony LM22S position display computer. Both the Sony and the Modulynx computers are capable of being controlled through serial communication ports and both are connected to the Zenith TurboSport in this manner. The Zenith controls the other computers by sending commands through these serial connections.

An important aspect of this system is that many of the components used were already in place for other uses, or are non-specific to use as a digitizer. The Modulynx-Table-Sony system is used to hold laser doppler velocimetry equipment. The only component which is specific to the system is the Point Range Sensor. In some respects, this system is valuable as an addition to existing equipment.

The Point Range Sensor is a measurement device that works on the principle of triangulation to determine the distance from a surface to the laser head, see figure 1 for a schematic. The Point Range Sensor

works by projecting a beam of laser light through a lens, in a direction perpendicular to the major axis of the sensor. The beam of light is focused to provide a decent spot within the projected measurement region. A photo-detector array is placed with a lens focusing the region of interest onto the array. The location of the laser spot image upon this array gives a relative location of the surface. The Sensor is calibrated between the short and long ranges and the resulting accuracy is +/- 0.01 mm., or 0.787E-3 inches. This offset is a function of the type of head chosen and is larger for the less accurate sensor heads and much smaller for the more accurate ones. The Point Range Sensor chosen was the PRS-800 model, which has an offset of 58 mm. and a working range of 8 mm. The Point Range Sensor itself is controlled by a full length AT style bus card. This card contains most of the electronics that control the laser head. An interface box between the laser head and computer card contains the power supply for the laser and acts as a multiplexor device, allowing for multiple sensor heads to be controlled by the one card. The programming interface is actually accomplished through device driver software supplied with the system. This software allows communication between the application programs and the Point Range Sensor device to proceed through normal I/O channels. The laser device is classified as a class 2 device with a maximum power output of 2 mW.

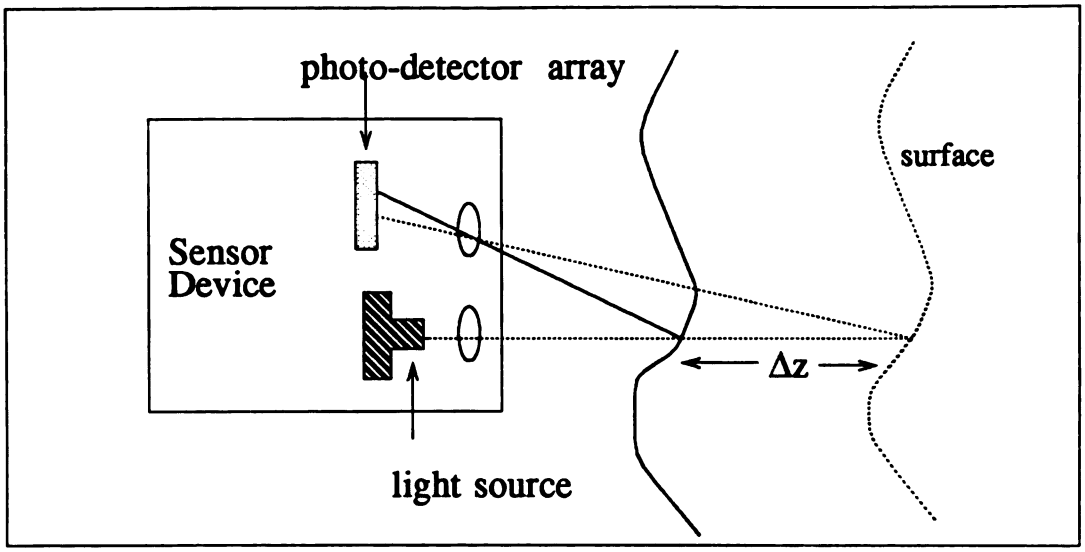


Figure 1.
Point Range Sensor Schematic

It is important to note some of the limitations of the Point Range Sensor as it has a large effect upon how the device is used. The surface of the object to be digitized must diffuse enough of the light to provide a large enough spot size. Thus, very shiny metal surfaces will reflect the beam spectrally and make measurement impossible. This can be avoided if shiny surfaces are dusted with talcum or sprayed with diffuser. The fifteen degree angle that is required to perform the triangulation actually makes some measurements impossible. For example, when trying to digitize a gear, see figure 18. The device worked fine for some of the surface. When the beam was located in between two of the teeth, the sensor was unable to register a surface. At this point it should be noted that the choice of laser device causes the limitation and for different devices, the limitation will be quantifiably different although qualitatively similar. The angle required for triangulation is different for each type of sensor head and this angle is what limits the types of surfaces that can be

measured and the amount of discontinuity that can be tolerated. The PRS-30 device has much greater accuracy at the expense of relative ease of measurement, for example. The PRS-1600 has a smaller angle but its accuracy is limited. So there is a trade-off to either have extreme accuracy and inflexible measurement conditions or choose less accuracy and more flexible measurement conditions. For devices such as engine parts and other parts which might only be accurate to within 0.001 inches, the PRS-800 was determined to be accurate enough.

The traverse table is capable of independent movement in three directions. In the longitudinal direction it has a movement range of 980 mm. In the latitudinal and vertical directions it has a movement range of 480mm. It is the wide range of movement capabilities provided by the table which makes this system unique. The use of the traverse table as a mounting device for the Point Range Sensor is not ideal. The traverse table is relatively large and moving the large mass of the bed is not an instantaneous process, although the high mass and large stiffness of the system make the natural frequencies of the traverse table very high and the system is thus somewhat noise resistant. These are good properties for a laser diagnostic system, which is the table's normal usage, but not for a movement arm of a CMM. The large mass makes acceleration and deceleration of the table a more time consuming process and thus inherently slows down the measurement process. It should be noted that it is precisely this large size that makes this system unique. The traverse table position can be measured within the ranges mentioned above to

accuracies that are noted below in the discussion on the Sony encoder system. The size of the table makes it slow to move around, and also makes it extremely stable, which is good for measurement.

The stepper motors that turn the power screws which move the table are controlled by the Modulynx controller. The controller manages the signals sent to the stepper motors and provides computer interface to the system, allowing the entire Modulynx subsystem to be controlled by an outside computer interface. This was originally intended to be a computer terminal, but was adapted to suit the needs of the SURFMAP programs. The external computer interface is accomplished through a serial connection. Commands in the form of ASCII data is transmitted over these lines and interpreted by the controller. See the appropriate manual for a description of the commands.

The Sony Encoder subsystem consists of two parts, a small computerized display/interface device and the linear position sensors. The linear position sensors are not actually position sensors. They are displacement sensors. They measure the amount of movement from one position to another. The Sony keeps track of the cumulative displacements and sums them to produce a position. The inability of the Sony to keep track of an absolute reference is a hindrance to the SURFMAP system, as it would be to any CMM. See the section on surface comparisons for more details, but the general idea is that a specific reference point is needed and one is not available. Thus some other method of fixing the reference point must be determined. See the

Recommendations in section 7.3.

The Zenith TurboSport 386 controls the other four subsystems. It contains the program structure which decides what to do and it stores the data in a form that is readily available for analysis. It also performs the some of the analysis. The Zenith has two serial ports which are connected to the Sony and to the Modulynx respectively. These two subsystems are controlled by commands sent by the Zenith. The Zenith operates at 12 Mhz and is capable of serial transmission speeds up to 56k baud, though the other hardware limits this to 1200 and 2400 for the Modulynx and the Sony respectively.

The five components fit together as follows: The Zenith is connected to the Sony by a 2400 baud serial connection. A command that tells the Sony to return the current values of the display is sent over the serial connection when the information is needed. The Sony then sends the values on the display back to the Zenith. This is then decoded and the position is known. The Zenith decides what movement is required next and sends the appropriate command over the 1200 baud serial connection to the Modulynx. The Modulynx then sends commands to the appropriate stepper motor and it moves the required number of pulses. The Zenith also requests information from the Point Range Sensor. The Point Range Sensor receives a request for a measurement and it returns either a measurement, or some sort of diagnostic suggesting either a retry attempt be made or that there is no surface within range of the sensor. The Zenith manages all of this information

and makes it possible to digitize a surface.

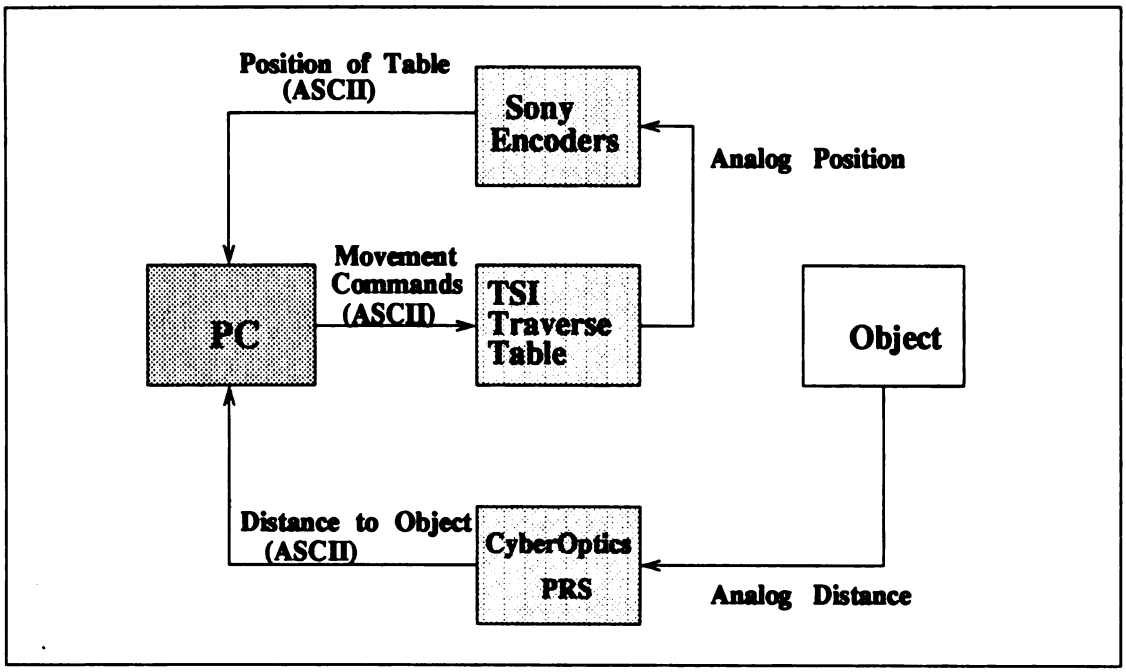


Figure 2.
SURFMAP Information Flow Diagram

Chapter 5.

The Program Organization

5.1 General Description

The SURFMAP software package is a combination of subprograms designed to perform different tasks in the process of digitizing and analyzing surfaces. The two main sections are the subprograms that take care of the data acquisition and the subprograms that do the analysis of these surfaces. The data acquisition routines are the major part of the software and they consist of two levels of code; the hardware dependent code and the management routines. The hardware dependent code consists of routines that move the table, read data from the Sony, read from the sensor and report the condition of the hardware. The management routines do the decision making and information control. These routines do the mundane tasks such as storing the data, keeping track of the current position, and also the non-mundane tasks such as deciding where to move to next, whether the new location is "safe", etc. The analysis routines perform some simple analysis such as the calculation of volumes, surface area, areas and performing spline fits to the data.

The purpose behind the structure of the code was to make the system easily modifiable and hardware independent. The idea is that if the hardware changes, the only routines that need to change are the hardware control routines, and the management routines need not change. Complete isolation is not possible because the data needs to be

written in a form that all of the routines understand, for example, the analysis routines need to know the form of the data to perform the surface fits. The hardware routines thus all return simple "C" data structures and are independent of the method of data storage used by the calling procedure. See figure 3. This flexibility was designed into the system because the system was seen as a constantly developing system, a test structure for new ideas and methods. Thus flexibility was a requirement, not an option.

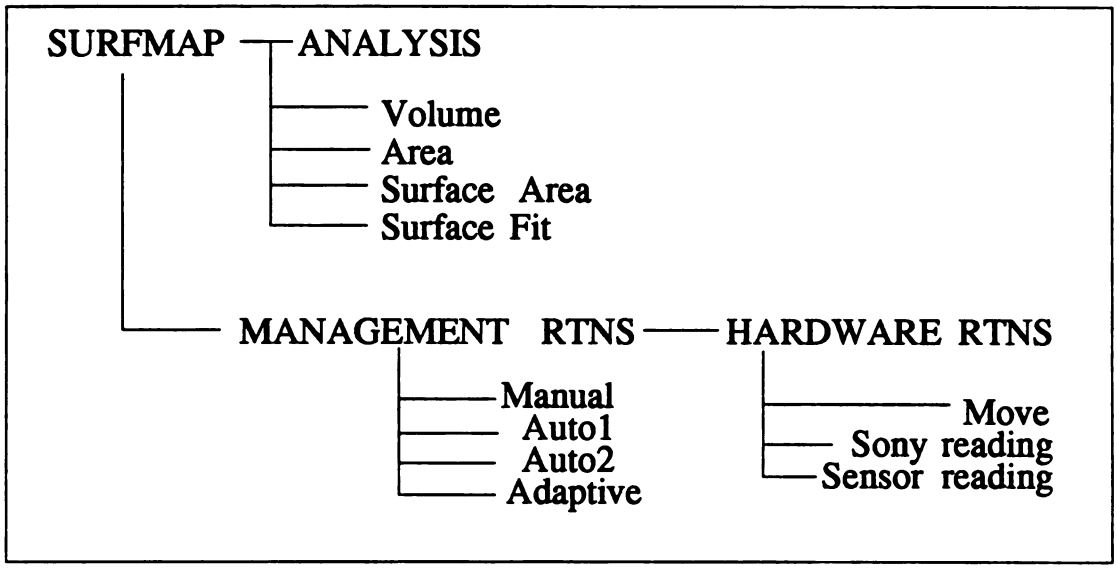


Figure 3. Program hierarchy

The code has three main goals at the center of its algorithms. It has to measure the data point, search for a surface when none is available, and determine an accurate representation of the surface for use in analysis of these surfaces.

The measurement of the data point proceeds as follows; the sensor is first checked to see if there is a surface within range. Assuming that

there is a surface, the next step is to measure the position of the traverse table from the Sony encoders. These two measurements are combined to make up the relative position of the current point. If there is no surface detected at the current position, the program initiates a series of movements designed to locate the surface at the current x,y position. See below for a description of the algorithm pertaining to locating a surface.

The issue of how to search for a surface has many aspects. The question of speed, which has been a consideration throughout this design process, would have us find the surface as quickly as possible. However, because the system is automated it is not necessarily as critical how fast the surface is found but that one is found when possible. A surface is not detected for one of three main reasons. The surface may have some feature which makes surface measurement impossible such as a reflective surface. The surface may be out of range of the sensor, implying the height of the table needs to be adjusted appropriately. The surface may have discontinuities such as holes or edges which make measurement of the surfaces near or inside those areas impossible.

The searching generally proceeds as follows:

1) The need is realized. 2) An action is decided upon (See the appropriate Section below), and carried out. 3) If no surface has been found the program decides that this point is an anomaly, (i.e part of a hole, or some other surface discontinuity) and moves on to the next point.4) 1, 2, and 3 are repeated until all of the actions have been completed or the surface is found.

Table 1. Searching Algorithm

A method for dealing with these surface discontinuities is explained further in Section 4.

5.2 Description of capabilities.

The following is a description of the capabilities of the Manual program. The Manual program was designed to use the devices under its control as a user-controlled coordinate measurement device. To this end the table movement controls are managed by the PC and respond directly to user inputs. The program allows for output in either x,y format or x,y,z format, to make the measurement of areas more straightforward. See Chapter 6 on calculating the compression ratio for an example of how it is used. The basic procedure involves the user moving the pointing device to the point to be measured. A key is depressed and the point is recorded. This continues until all of the points needed are marked, and then the user quits the program. The data is written to the file in the

order that it was taken. The data taken can then be used as input to the Adaptive routines, or a surface can be fitted to this data.

The Automatic sections were designed to require less operator attention. There are two different subprograms that make up this section, Auto-1 and Auto-2. The main difference between them is the complexity of the search algorithm. The development of the search procedures necessitated different levels of searching ability and each level has proven to be useful to some extent. The automatic routines begin by defining a measurement volume. It is inside this volume that it searches for a surface. The procedures start at one corner of the rectangularly shaped volume and search for surface points in a grid pattern across the measurement volume. When the Auto-1 program cannot find a surface it prompts the user for input. When the Auto-2 program needs to look for a surface it performs some actions based upon a list given at start-up. The currently supported actions are:

1. Move the Z axis in the direction of the slope of the previous two points.
2. Move to the top of the measurement volume.
3. Step downwards until reaching the bottom.
4. Reached bottom of volume.
5. Skip to the next measurement position.

Table 2.
Searching Commands

The Adaptive routines take an estimated surface in the form of coordinates and digitize the surface in a finer pattern. This is equivalent to saying that in parametric space the grid structure is finer. This is to

produce a representation of the surface that contains more information with as little data added as possible. There actually are two possibilities for this procedure. The user can choose to do either a finer grid structure with no adaptation or an adaptive-searching algorithm with the intention of seeking out surface details. Because of this ability to locate surface details the routine inherently produces unequally spaced grids. This introduces difficulties when fitting surfaces to the data. See Chapter 2, Section 4.

The searching routines work by one of two methods. The gradients of the surface are considered and the table position adjusted appropriately. If that does not work, the program begins a blind search mechanism which continues until it either finds a point, or exhausts the search actions. If the Auto-1 routine does not find a point, it prompts the user for adjustment. Auto-2 merely skips the point, assuming that it is near a discontinuity, e.g. a hole, and cannot measure because of this discontinuity. Assuming this generates some inherent error at sharp edges or other discontinuities.

Another routine, called the 'impact' procedure, examines each new move before it is accomplished and tries to predict the situations where the sensor head would contact the surface. This is an attempt to keep surfaces with large height differences from being contacted by the sensor head and moved so as to make all previous measurements invalid. The surface can theoretically vary as much as the vertical table movement range. The relatively large size of the sensor head necessitates a check

on the expected destination to insure proper clearance. This routine works by taking the current position and adding the planned displacement to it. Next the edges of the sensor head are calculated. These edges are compared with the current known surface. If the height of any of these edges is less than that of the surface, the movement is aborted. In the automatic routines the surface is defined as it is measured. The Adaptive routine uses the previously loaded surface information is its calculations.

A post-processor was written to look at the relative slopes of the data and generate another field based upon whether the point was considered an "interior" point or an "exterior" point. This is designed to aid in the fitting of the data by indicating points which should be the edges of discontinuous surface patches.

The variable grid spacing algorithm works in a similar fashion to the chordal deviation method presented in Wysocki, ['87]. This is implemented in the adaptive subprograms. Three constant parameter points are considered from the previous surface information. The chordal deviation of the third point with respect to the other two is calculated. If it is less than the maximum, the point is OK, and the process is stepped along. It if is greater that the minimum then the second and third points are bisected. If new information is present the points are re-examined as above. If not, the process moves onward until the surface is completed. This has the effect of spreading out the points on flat areas and bunching them up in areas of high curvatures.

With just a minor adjustment to the variable grid spacing argument

the same algorithm can detect jump discontinuities. When a minimum step size between data points B&C are achieved and the height is still greater that the maximum chordal deviation, the two points represent a discontinuity. The variable grid space algorithm coupled with the Point descriptor post-processor enables the interpolation of complex surfaces with discontinuities.

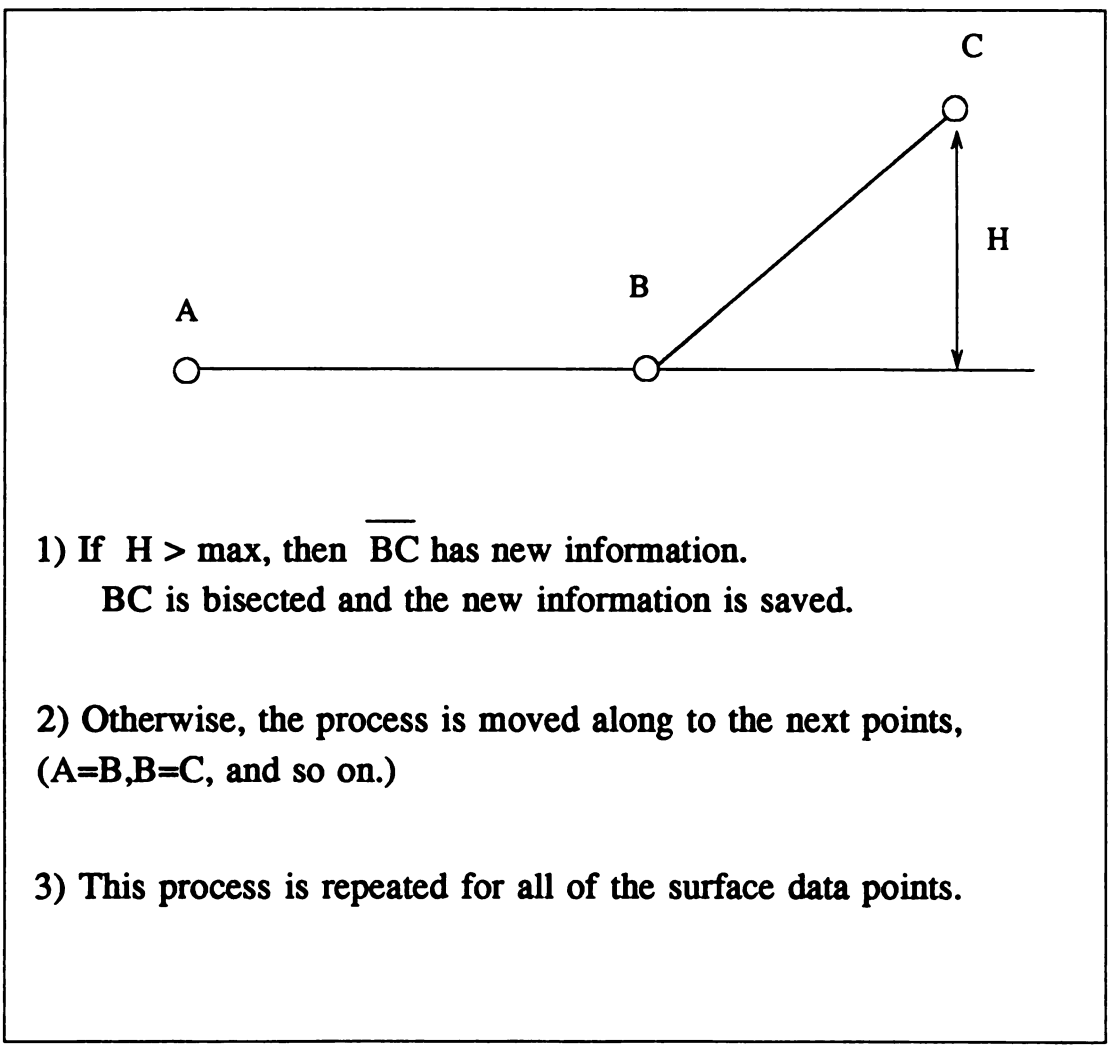


Figure 4 .
Adaptive Searching Algorithm
(based on Chordal Deviation Method)

5.3 Algorithms

5.3.1. The Manual Program.

The Manual program begins by allowing the user to move to the first point. It then measures the height if necessary and prompts for any required position corrections. It then checks to see if the point is good and saves it if appropriate. The user next has the opportunity to exit the program or continue, in which case the process is repeated until the user exits.

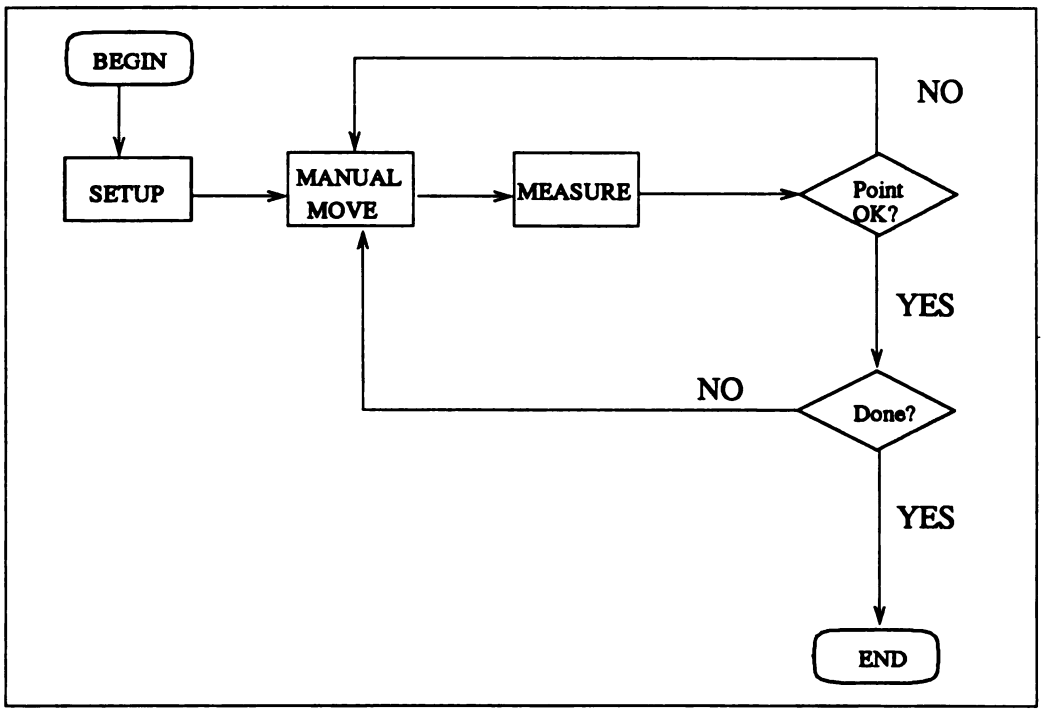


Figure 5.
The Manual Algorithm

5.3.2 The Automatic Programs

The Automatic routine inputs from the user the boundaries of a two-dimensional region which contains the surface to be digitized. The step sizes in the two directions are input next and the system starts at the origin of the region and proceeds in the 'x' direction first and then steps the 'y' direction when the end of the region is encountered. See figure 5 below for a schematic of how the movement proceeds.

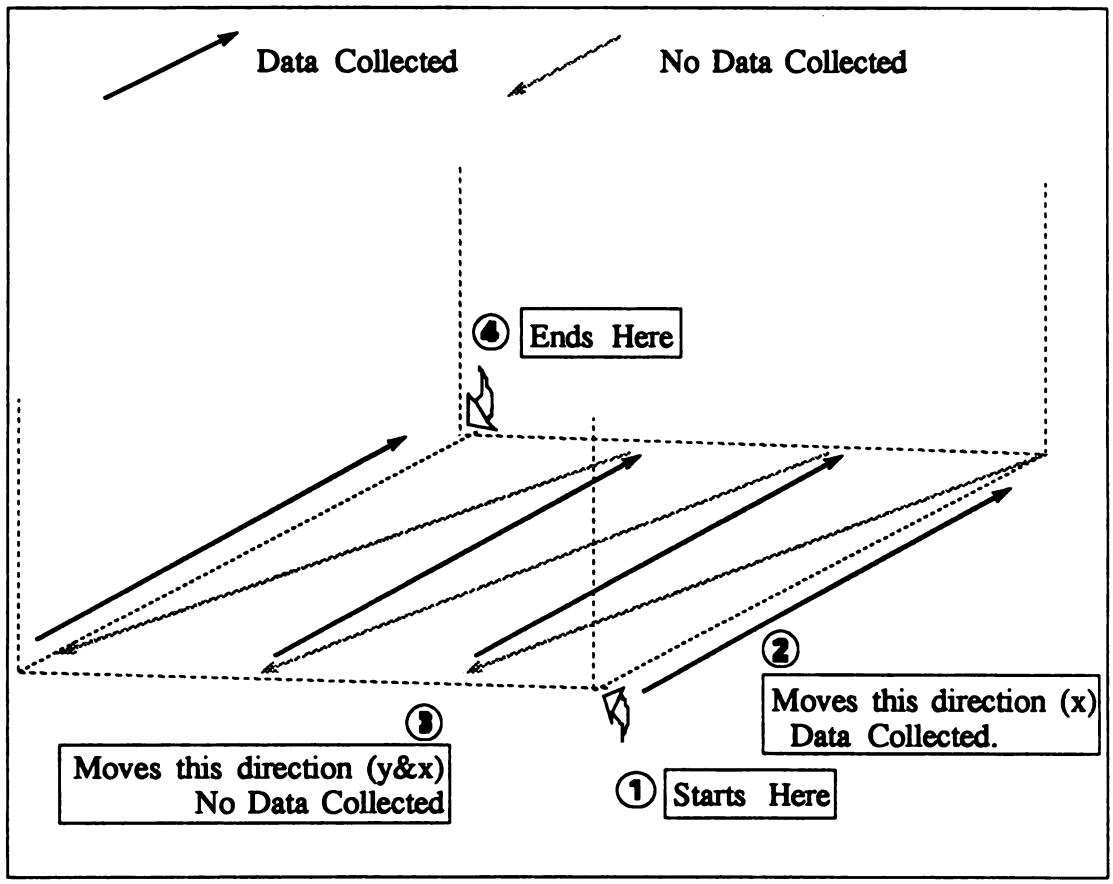


Figure 6.
Automatic Movement Pattern

The data is saved differently in Auto-1 and Auto-2. In Auto-1 the current point is written to a data file immediately and in Auto-2 the data

point is kept in a linked list of points until the end when it is written to a file. In Auto-1 the information for the searching routines is updated with every new point; in contrast, Auto-2 uses the saved surface data to calculate the required parameters when they are needed. See Figure 7 below for the general automatic algorithm.

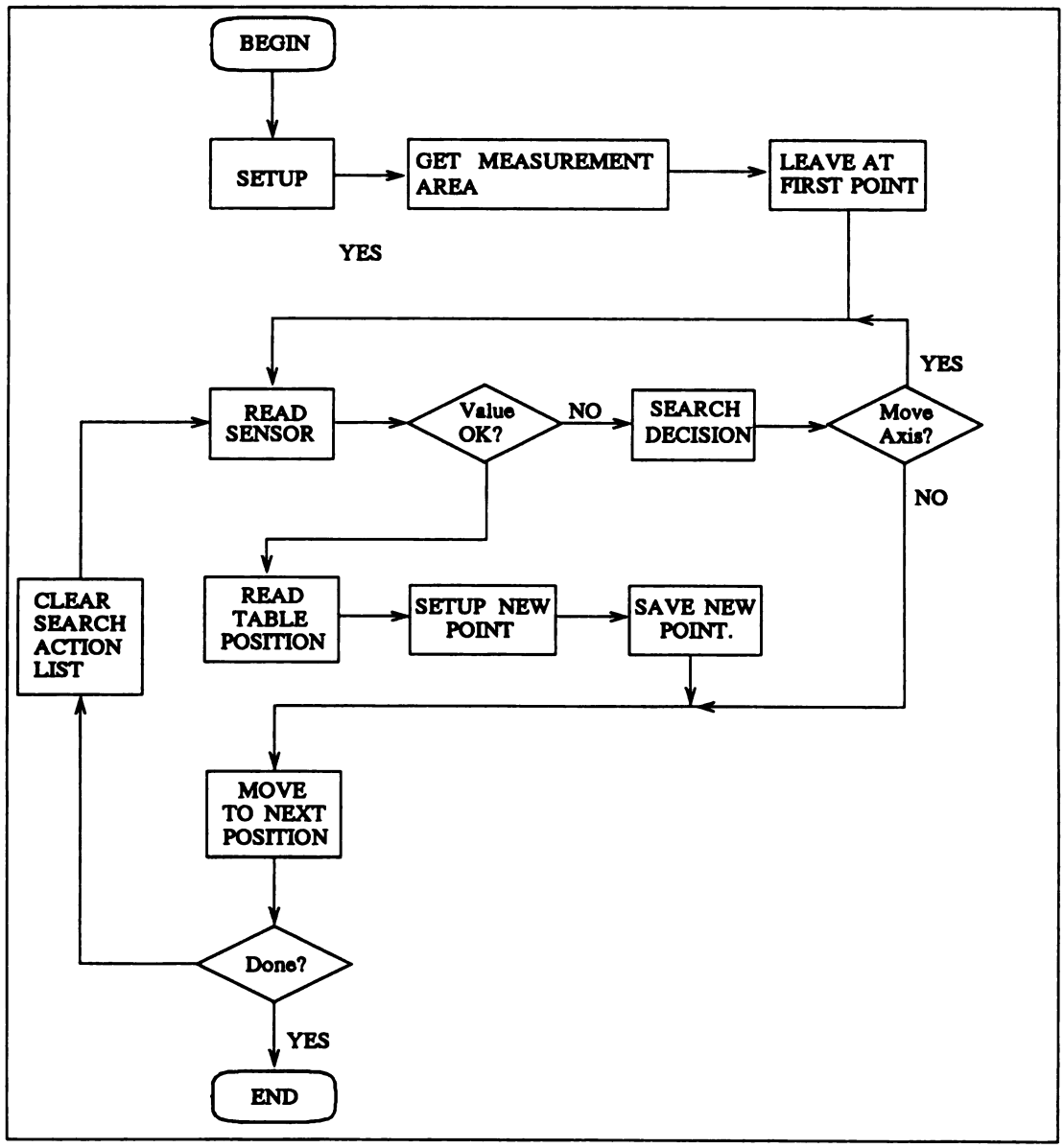


Figure 7.
The Automatic Algorithm
Auto-1 and Auto-2

5.3.3 The Adaptive Routines.

The adaptive routine reads in a previously digitized Surface in the form of coordinates. This surface is then either subdivided or searched adaptively. The complete set of points is output to the data file. This routine uses the previous surface information as an estimate of the surface slopes. See Figure 7 for a detailed description of the algorithm.

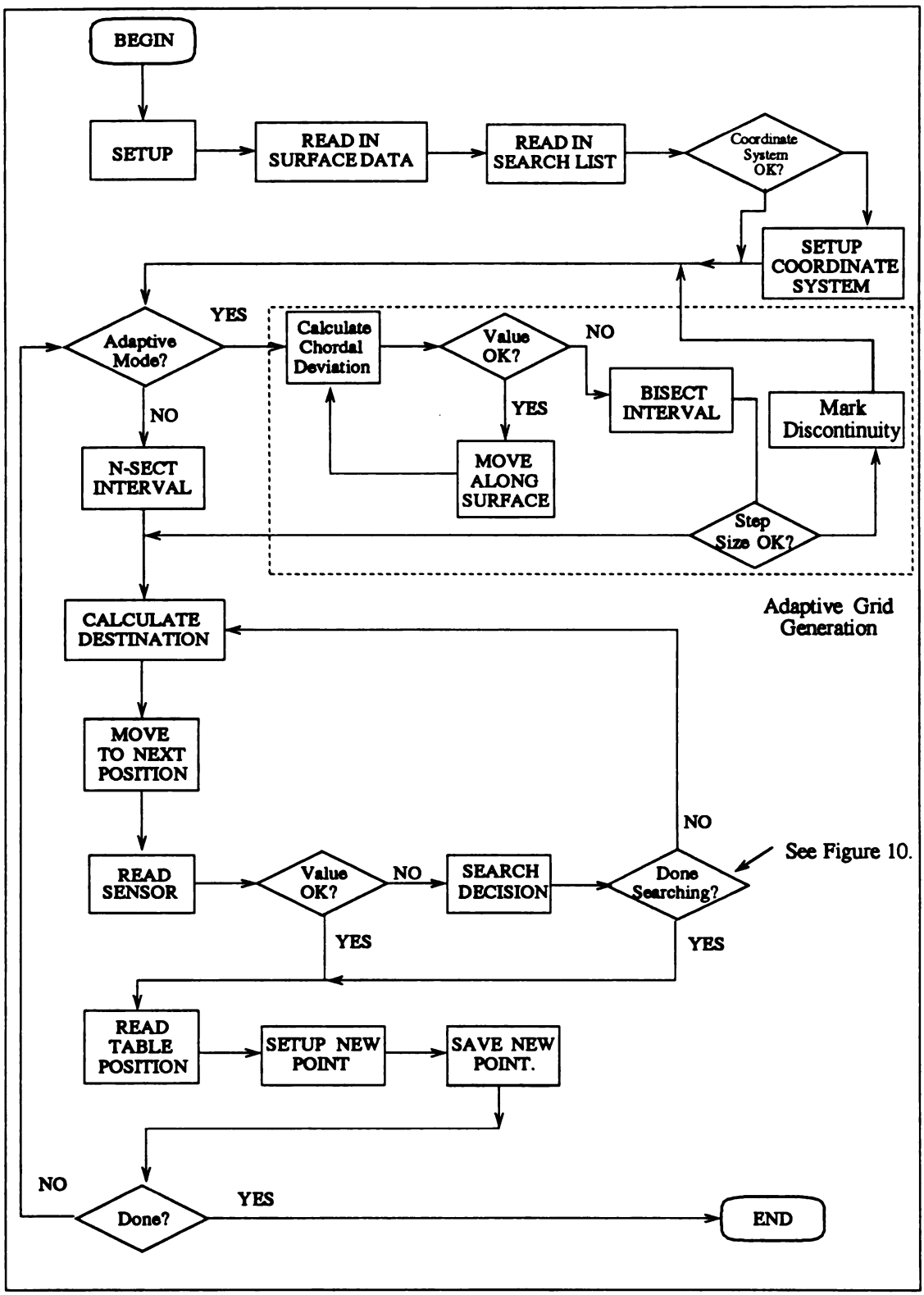


Figure 8.
The Adaptive Algorithm

5.3.4 Miscellaneous Support Algorithms

5.3.4.1 Searching Algorithms

The searching routines are list-processed decision trees. That is to say the appropriate actions are entered into a queue in the order of execution. The system then executes the items on the list appropriately until the list is completed. The first searching routine has a fixed order in which is searches for surfaces. The second searching routine reads the searching procedure from a file. This file is a list of integers indicating what actions to preform in what order. See Table 2 in section 5.2. See Figures 9 and 10 below for the algorithms. The case numbers in the figures represent actions from the appropriate action table, see Table 1.

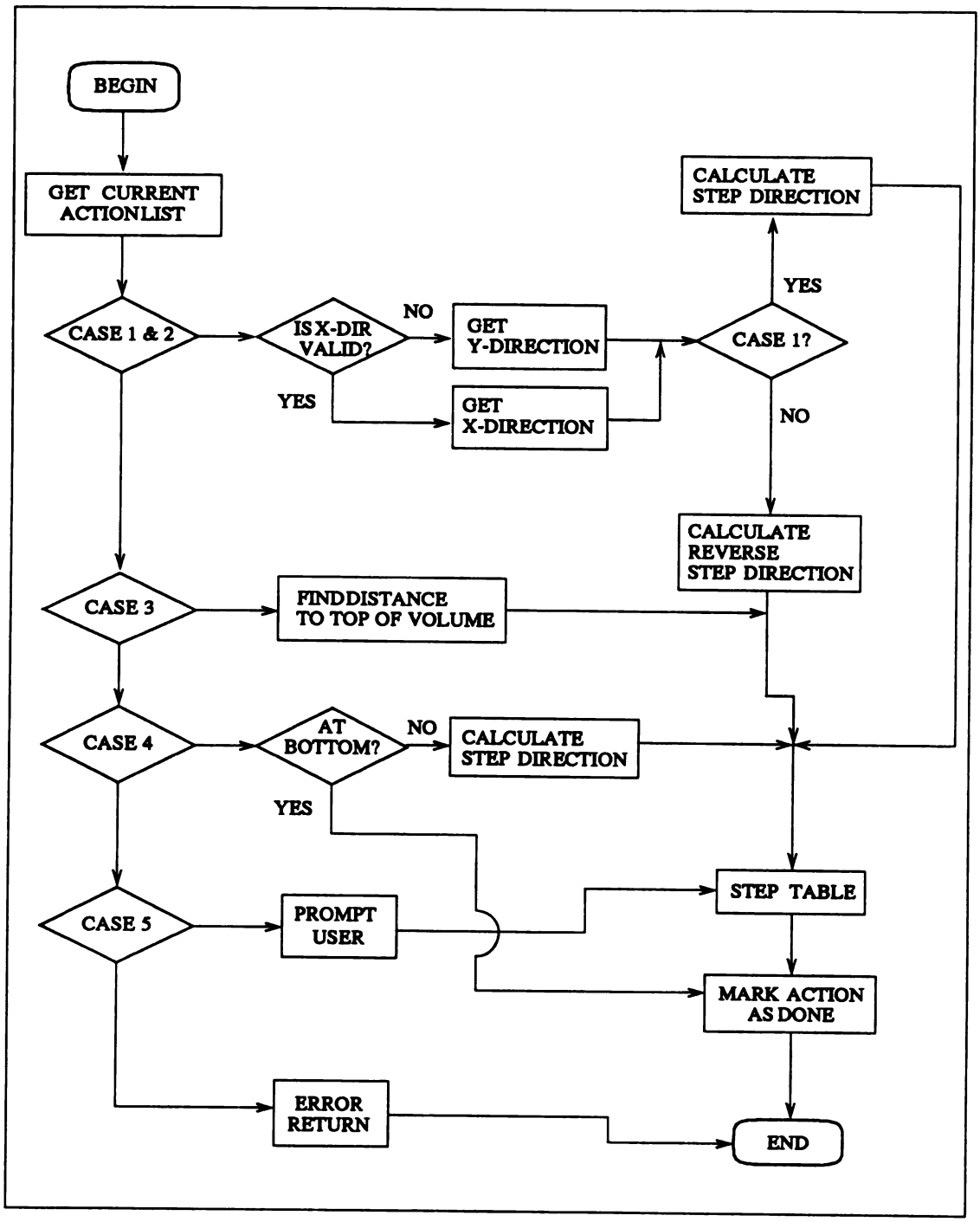


Figure 9.
The First Searching Algorithm

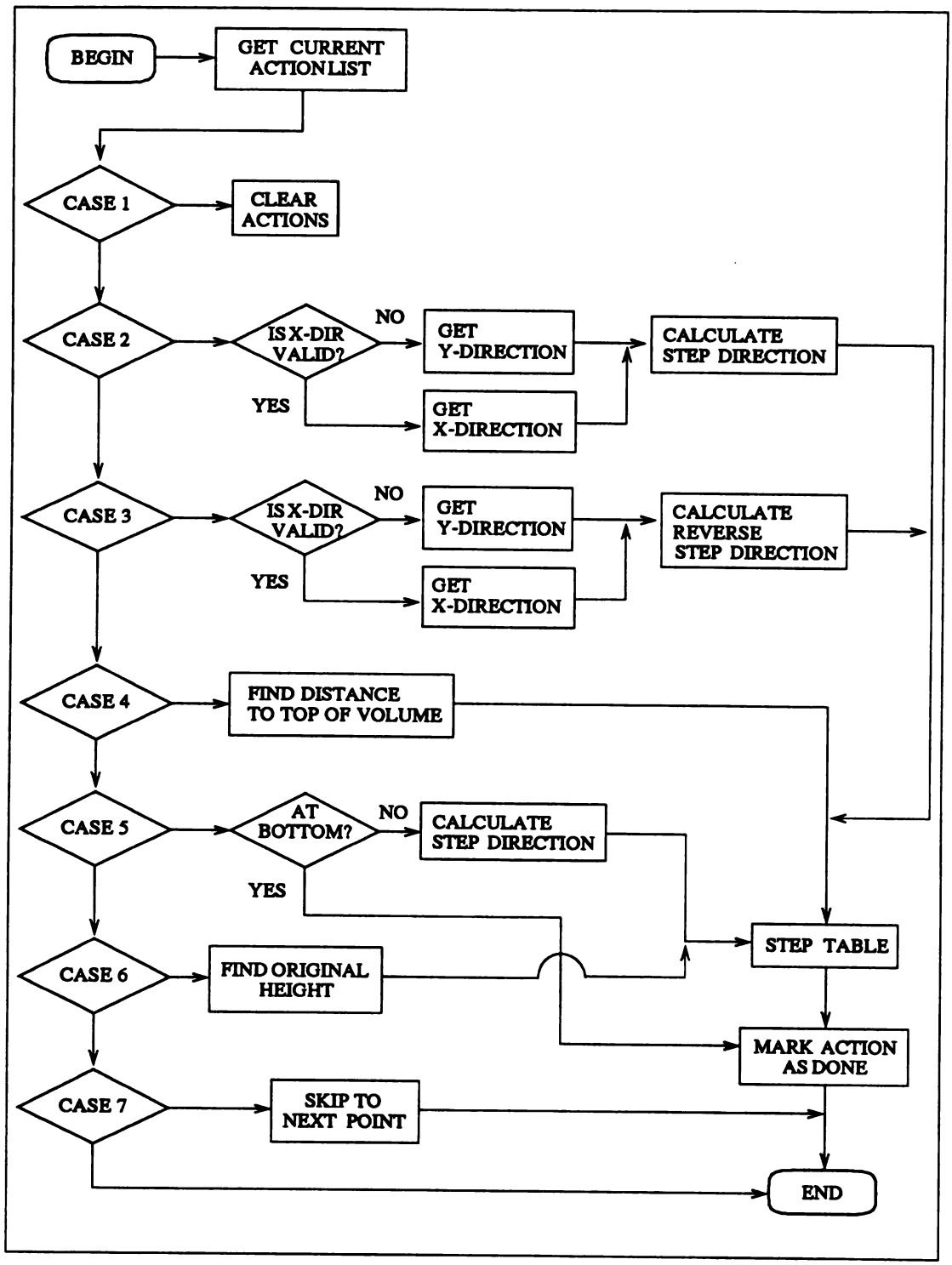


Figure 10. Second Searching Algorithm.

5.3.4.2 Point Descriptor Field

The point descriptor fields are decided upon by the relative angles between the vectors from the first to the second and second to third. If this angle is greater than the user-prescribed minimum the second point is marked discontinuous. The process is them moved along to the next three points,(i.e first=second, second=third, third=new) See Figure 11 below.

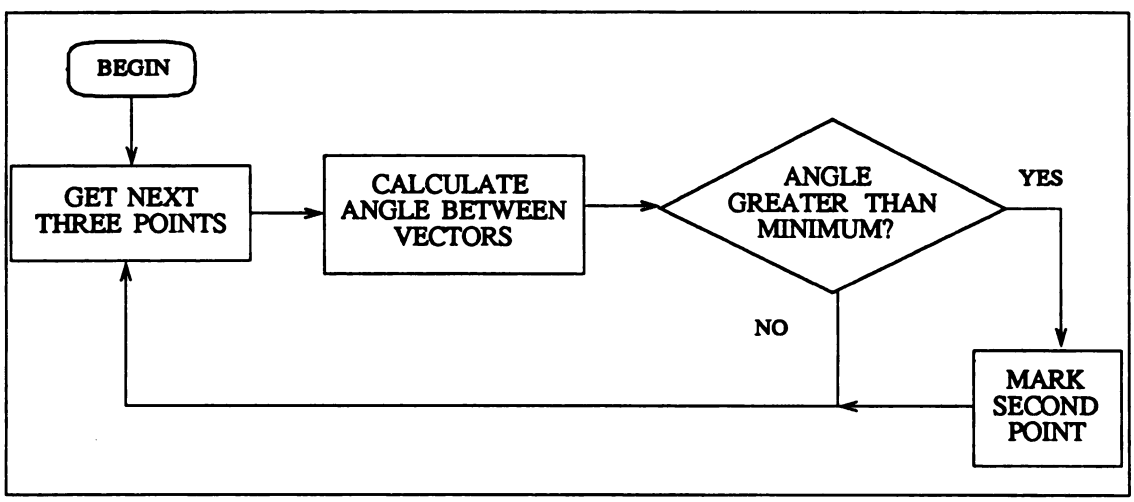


Figure 11
Point Descriptor Field Generation

5.3.4.3 Variable Grid Spacing

The variable grid spacing algorithm is the most important part of the adaptive routine. As stated previously the routine looks at three consecutive data points and decides if there is enough information to describe the surface between them. Consecutive points are points that are closest together in a constant parameter direction. The chordal deviation is calculated and compared to the maximum allowable. If new

information is needed the routine returns a request for the new information. The surface is processed until all of the intervals contain no new information or the interval spacing is too small for further investigation. When the interval is small and the height is still too large, the points around it are marked discontinuous. See Figure 4.

5.4 Analysis Routines

As described in section 2.4 the two methods for describing surfaces are an interpolation scheme and an approximation scheme. The interpolation scheme used here comes directly from Farin['90].

The surface is written as a tensor product surface, see equation 21, and the problem is to solve for the control polygon points V_i.

$$Q(u,v) = \left[N \bullet M \right] \left[V \right] \tag{29}$$

This method interpolates one control point for each data point, and provides no data compression. The matrix of blending function coefficients can be written as a tridiagonal matrix, and the whole thing is a tridiagonal system. Instead of trying to calculate the inverse of the blending functions matrix and multiplying ,

$$\left[N \bullet M \right]^{-1} \left[Q \right] = \left[V \right] \tag{30}$$

The system is solved by doing a lower-upper decomposition and backsubstitution. [Press et al.]

The only detail left is to calculate the blending functions. These are calculated as in Farin['90] by examining the relationship between the data points x_i and the control point V_i. The V_i are converted to Bezier points by converting them to piecewise Bezier curves. In short the system is written in the following manner. This system is solved for each

row of data points, and produces a row of control points.

$$\begin{bmatrix} 1 & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{L-1} & \beta_{L-1} & \gamma_{L-1} \\ & & & & 1 \end{bmatrix} \begin{bmatrix} \overline{V} \end{bmatrix} = \begin{bmatrix} \overline{r} \end{bmatrix} \qquad (31)$$

Where $r_0 = B_1$

$$r_i = (\Delta_{i-1} + \Delta_i)\, x_i \qquad (32)$$

$$r_L = B_{3L-1}$$

and

$$\alpha_i = \Delta_i^2 / (\Delta_{i-2} + \Delta_{i-1} + \Delta_i)$$

$$\beta_i = \Delta_i(\Delta_{i-2} + \Delta_{i-1}) / (\Delta_{i-2} + \Delta_{i-1} + \Delta_i) + \qquad (33)$$

$$\Delta_{i-1}(\Delta_i + \Delta_{i+1}) / (\Delta_{i-1} + \Delta_i + \Delta_{i+1})$$

$$\gamma_i = \Delta_{i-1}^2 / (\Delta_{i-1} + \Delta_i + \Delta_{i+1})$$

$$\Delta_i = u_{i+1} - u_i$$

The actual interpolation done in Figure 20. was done using SURFPAK, a software package published by the Case Center for CAE/CAM at Michigan State University. This software package has the ability to fit B-spline, Bezier, and Coon's patches to data presented in topologically rectangular grids. The approximation scheme is set up as in equation 24. This can be solved by the LU decomposition method mentioned above or by the singular value decomposition(SVD). The LU method produces very large delicately balanced coefficients which make the results very unstable. The SVD method does not have this

characteristic and allows for the specification of an error tolerance. The surface approximations done here, (See Figures 21,23), were performed by SURFPAK. SURFPAK uses a slightly different method of approximation, see the SURFPAK User's Guide['84].

The discontinuous nature of the surfaces that are used and the noted inability of the fitting methods used to produce discontinuities[Rogers, '87] suggests using a method for surface comparison that does not rely on the fit parameters. The approach used here is to compare the discrete points on the surface. The surfaces that are to be compared must be adjusted so that the surface points on one surface coincide with the other surface. This can be accomplished by translating and rotating the points into the correct position using standard geometric transforms. Since this is best accomplished visually the Wave software licensed by Precision Visuals Inc. is recommended. When the surface grids are different, points between grid points must be approximated for the comparison to be valid. This can also be accomplished by using the Wave software. See the Appendix (Section 6) on the SURFER routine. The SURFER routine fits cubic splines between the data points and generates new grids of data. Note that the splines used here are not used for surface description for the reasons mentioned in Chapter 2. Here they are only used to approximate between data points.

The surface comparison is achieved by aligning the surfaces and comparing the relative heights. This is good for the comparison of

different spline fits to surfaces, by comparing the data with the fit results. The accuracy is a function of the accuracy of the two data points.

$$H = z1 - z2$$

and the uncertainty is:

$$dH = \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| dx_i \quad = \left| dz \right| + \left| dz \right| \quad = 2dz \tag{32}$$

where dz is the uncertainty in the height measurement.

The area and volume calculations are handled on a discrete basis for two reasons. The accuracy of the spline fit is in question and generally the digitized data has small enough increments to keep the error small. See below.

The area under a curve is calculated using the general trapezoidal integration scheme. The area underneath the line between the two data points is calculated. The volume is calculated by first calculating the area under each row of data. This is multiplied by the width and summed over the whole surface to get the volume. The area is calculated as follows, see figure 12:

$$A = z_1(x_2 - x_1) + 1/2 * (x_2 - x_1)(z_2 - z_1) \tag{33}$$

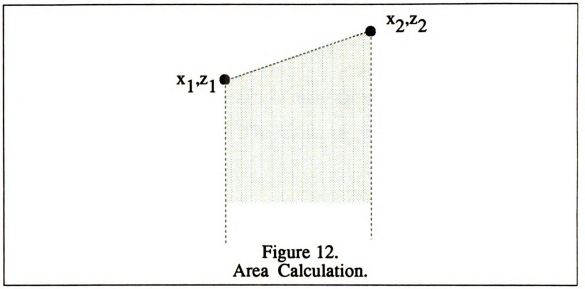


Figure 12.
Area Calculation.

The error in the area is:

$$dA = \sum_{j=1}^{n} \sum_{i=1}^{m} \left| \frac{\partial A_j}{\partial x_i} \right| dx_i \qquad (34)$$

where n = number of points, m = number of independent variables.

$$\sum_{i=1}^{m} \left| \frac{\partial A_j}{\partial x_i} \right| dx_i = \left| x_2 - x_1 \right| dz + \left| z_2 + z_1 \right| dx \qquad (35)$$

The error in the volume is calculated similarly.

$$dV = \left| \frac{\partial V}{\partial A} \right| dA + \left| \frac{\partial V}{\partial w} \right| dw \qquad (w = \text{width}) \qquad (36)$$

The surfaces are read in constant y-coordinate values, such as is output by the Auto-1 and Auto-2 procedures. The area under each line is calculated and then multiplied by the width.

Chapter 6.

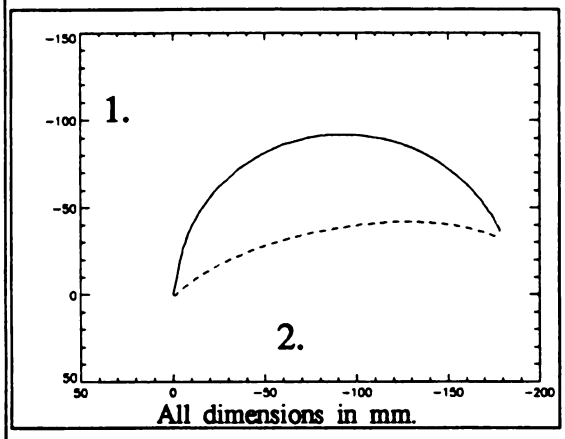
Examples

This chapter contains results of using the programs in the SURFMAP software package. The results are presented in increasing order of complexity.

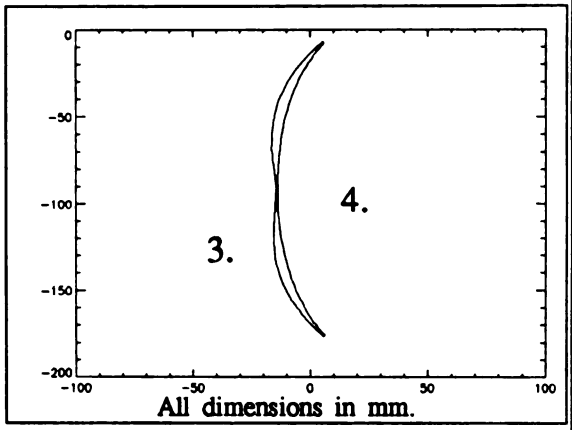
The measurement of areas is accomplished by using the system in its manual mode of operation. An example of the results of such a calculation are provided in figure 13. The positions of locations on the inside edge of the rotary engine housing were digitized. In the same coordinate system the edge of the rotor was also digitized. These two curves were digitized at the intake closed position and the full compression position. These two sets of intersecting curves were analyzed with the procedures in section 2.4.4 to generate the area between the curves. Error analysis is also provided and is calculated in accordance with the method described in section 5.4.

Figure 13 also shows the results of a volume calculation. The volumes of the different pocket geometries were calculated. The rotor pocket geometries were digitized using the automatic subprogram. See figures 14a,15, and 16 for the respective geometries. The volume of the Mazda 12A rotor pocket is 23.37 cc. and the others are 51.1 cc. and 21.47 cc. Figure 14b shows a problem inherent in the digitization process. The surface shown has a drastic change in slope in the region within the ellipse. Digitization inaccurately represents the edge of the pocket. On the real surface the edge is continuous along its length. The

Intake Closed



Full compression



Curve	Area Underneath	Error
1.	13,145.4 mm^2	9.6 mm^2
2.	5,631.7 mm^2	2.2 mm^2
Total Area =	7,513.7 mm^2	11.8 mm^2
3.	2335.4 mm^2	1.4 mm^2
4.	1829.0 mm^2	1.2 mm^2
Total Area =	506.4 mm^2	2.6 mm^2

CHAMBER VOLUME : Depth = 70 mm

(maximum)	= 525.96 cc	1.35 cc
(minimum)	= 35.45 cc	0.182 cc

TOP spark plug = 0.6 cc
BOTTOM spark plug = 1.35 cc

Total Volume (max) = 525.96 + 0.6 + 51.1 = 577.66 ± 3.45 cc
Total Volume (min) = 35.45 + 0.6 + 51.1 = 88.5 ± 2.3 cc

Compression Ratio (Deere Rotor) = 577.66 / 88.5 = 6.53 ± 0.11
Compression Ratio (Mazda Leading Deep Recess) = 9.04 ± 0.10

Figure 13. Compression Ratio Calculation

best representation here will be an inappropriate spline curve with large changes in its curvature.

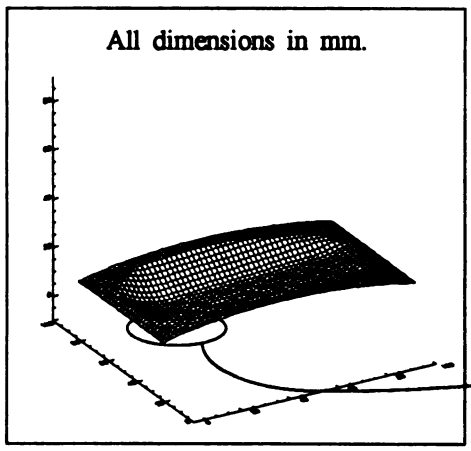


Figure 14a
Mazda 12A Rotor Pocket Leading Deep Recess.

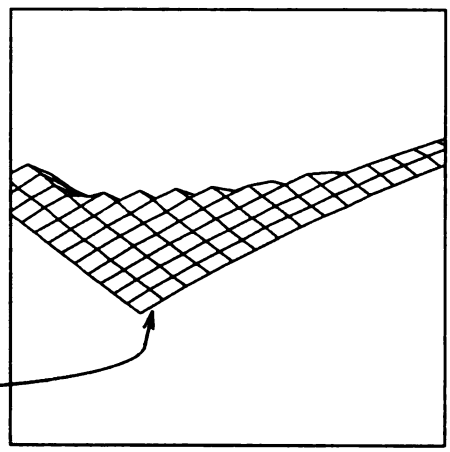


Figure 14b
Detail of Pocket Section Showing Discontinuous Edges.

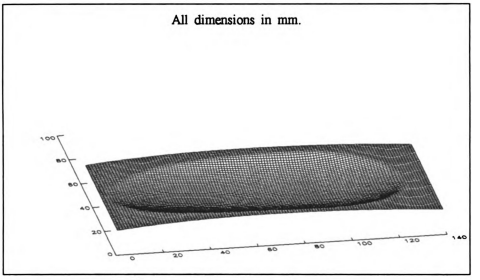


Figure 15. John Deere Trailing Deep Rotor Pocket design

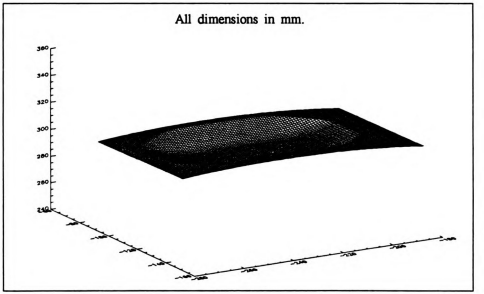


Figure 16. Symmetric Pocket Design

The next example is an automatic surface digitization. This example was provided to show one of the limitations of the system. The gear was digitized to show the sensor limitations, see Figure 17. When the laser spot is located between the teeth and the photo-detector array is blocked, no spot can be detected even though a valid surface is available. This could be simply solved by the addition of a rotary stage for holding the objects. See Section 7.3

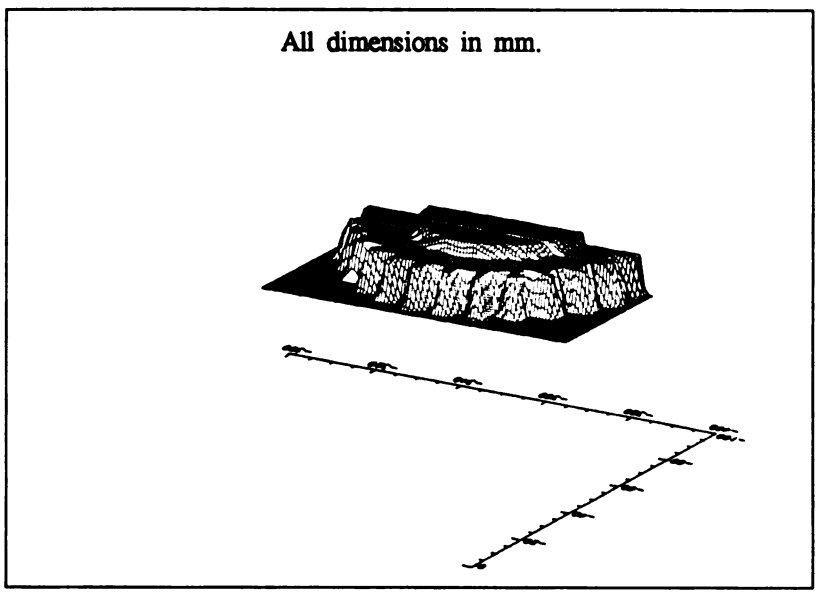


Figure 17.
Digitized Transmission Gear
From A 600cc
Motorcycle Engine

Figure 18 is a cross section of a mass airflow sensor housing for a Ford Aerostar. This is a good example of an object with severe discontinuities. This is also a sample of a surface that came from two separate patches that were combined as noted in section 5.4.4 The

complicated mouth section where the hot wire is installed is one section and the rest of the body is the other.

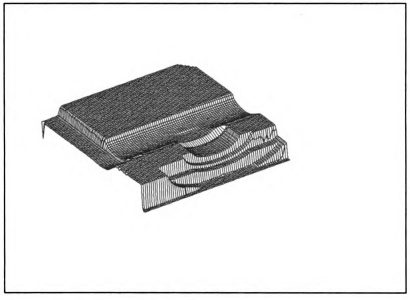
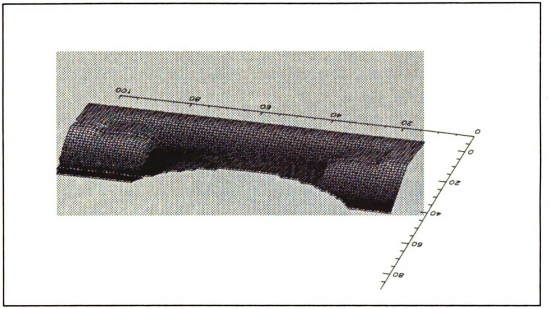
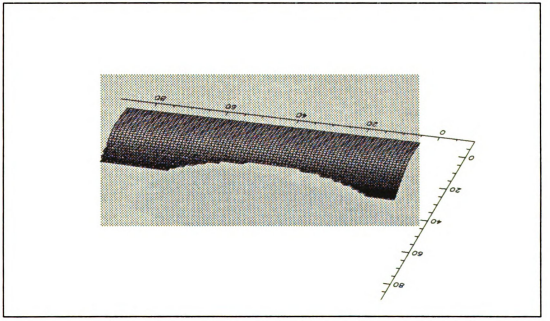


Figure 18.
Mass Airflow Sensor Cross Section

The figures on the following pages are sections of the air filter cover from a Ford Aerostar. These two figures are interesting in that they provide an example of the range of analysis provided by SURFMAP. The detail of the entry region in figure 19a is done with a grid spacing of 1 mm. and the side section in figure 19b has a spacing of 2 mm. The physical size of the first section is approximately 1/3 that of the second.



Section of Production Air Cleaner Cover at the Entrance to the MAFS



Section of Prototype Air Cleaner Cover at the Entrance to the MAFS

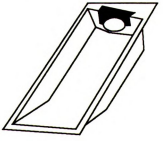
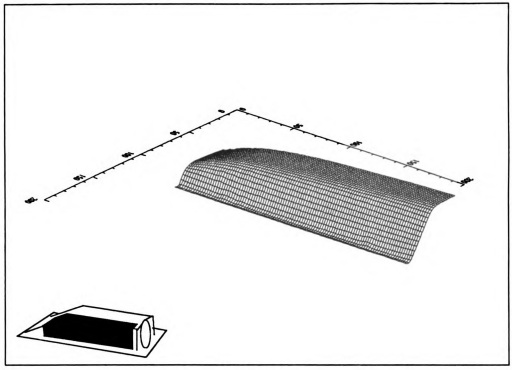
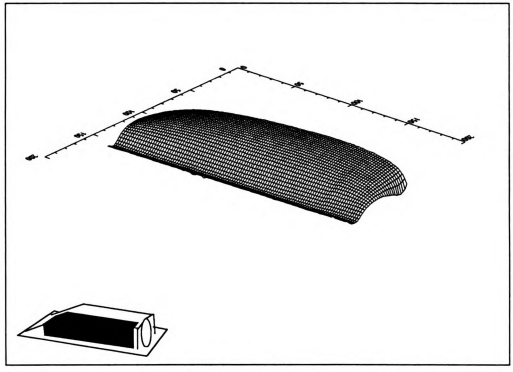


Figure 19a. Air Cleaner Sections
(All dimensions in mm.)



Original Cover



Refined Cover

Figure 19b. Sectional Views of Air Cleaner Covers
(All dimensions in mm.)

The next subject of discussion is that of the spline fits to the data. Figure 20 is a spline interpolation of the rotor pocket. This generates *n* control points from *n* data points. Note from the figure that there is some oscillation from the method used for calculation. See section 5.4.1. Figure 21 is a surface approximation based on the procedures mentioned previously. This is a coarse fit, which allows for greater smoothing and the specification of fewer control polygon points.[Rogers '87] Figure 22 is a surface approximation with a smaller error tolerance, which generally requires more polygon points. With coarser fits the amount of data can be reduced more than in a fine fit surface. The actual character of the surface must be considered, smooth surfaces will obviously be easier to approximate, whereas surfaces with discontinuities and/or large changes in curvature will require more information.

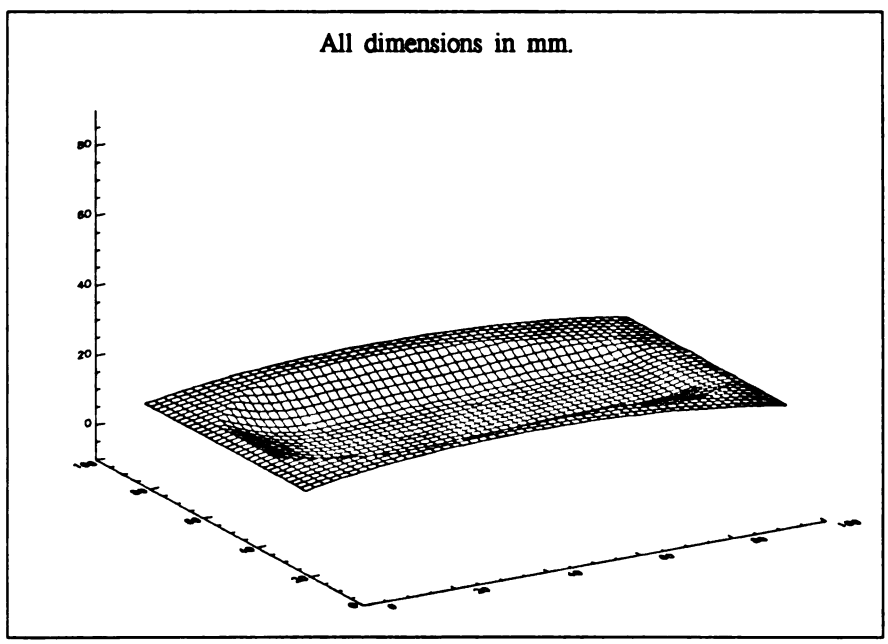


Figure 20.
Mazda 12a Spline Interpolation

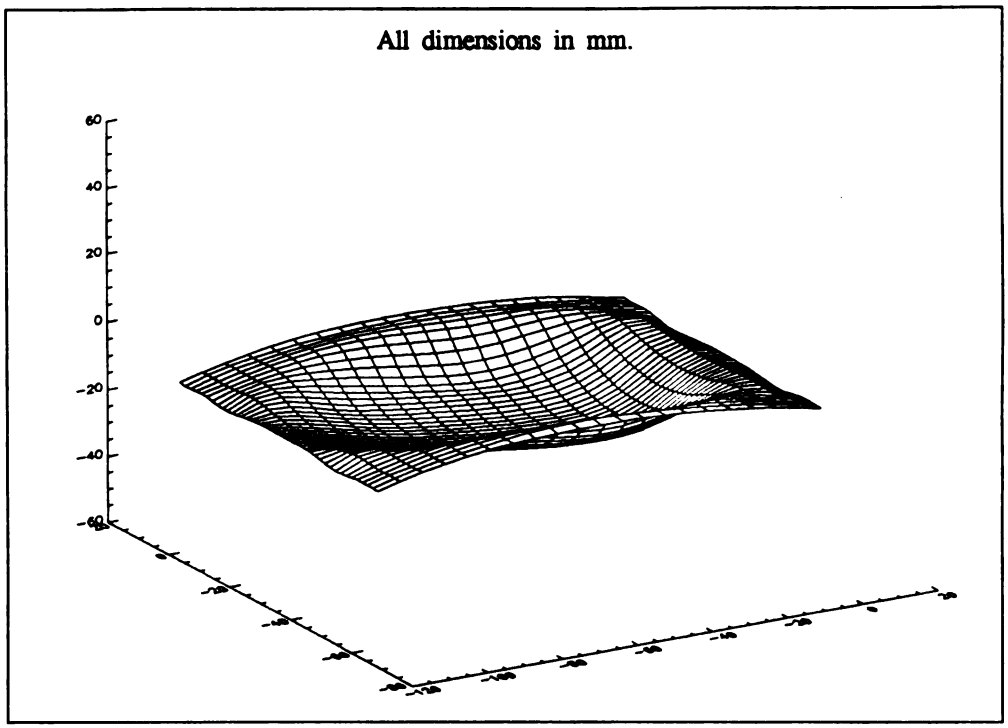


Figure 21.
Coarse B-Spline Approximation

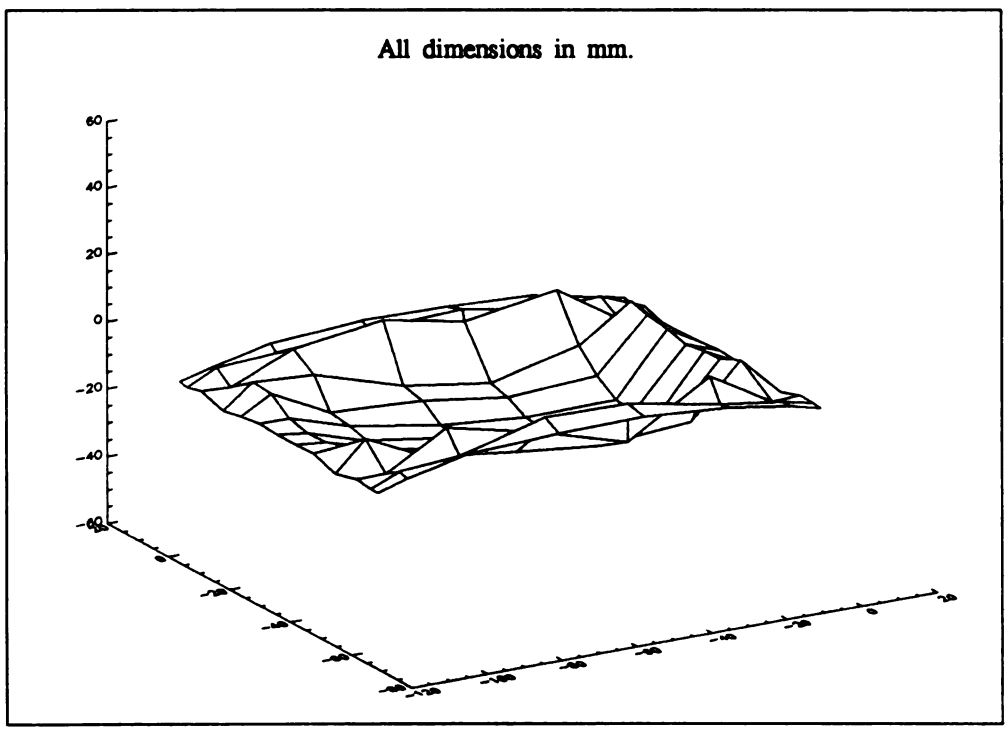


Figure 22.
Control Point Net for Coarse Approximation

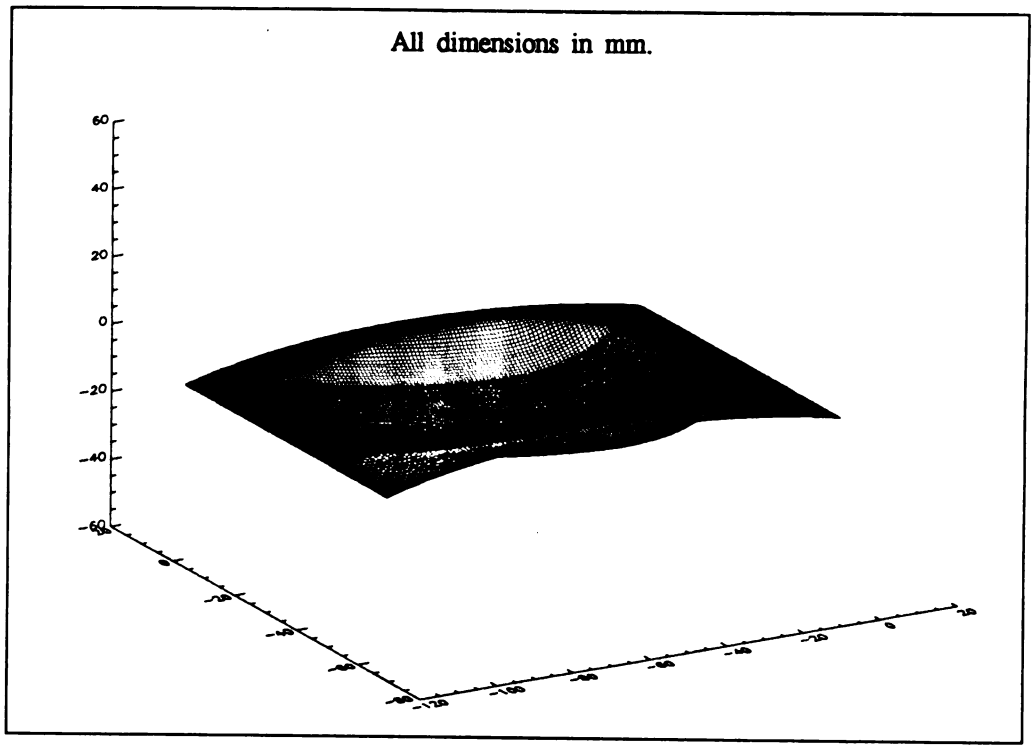


Figure 23.
Fine B-Spline Approximation

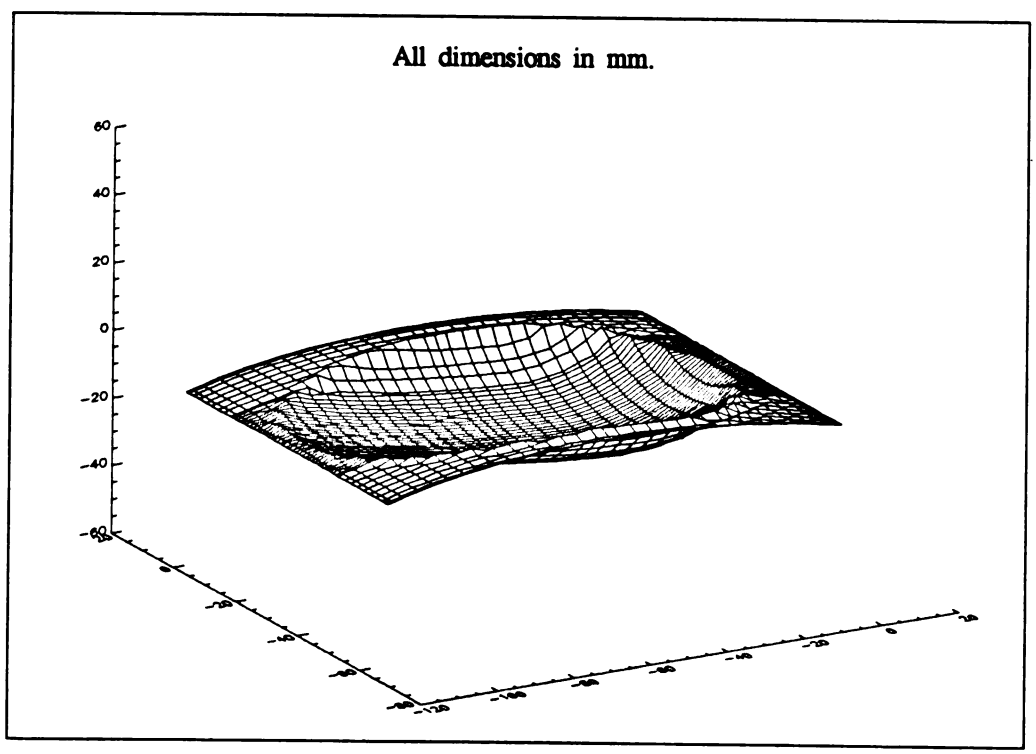
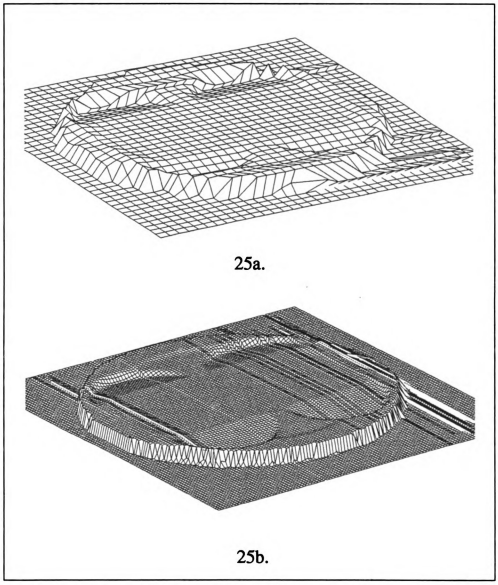


Figure 24.
Control Point Net for Fine Approximation

Figure 25 shows the results of the adaptive subprogram. Figure 25a. shows the coarse net of data points from using the automatic procedure. Figure 25b. shows the fine grid of data points measured by dividing the previous intervals by 4. The grid spacing of Figure 25a. was 2mm. x 2mm. Figure 25b. has a spacing if 0.5mm. x 0.5mm.

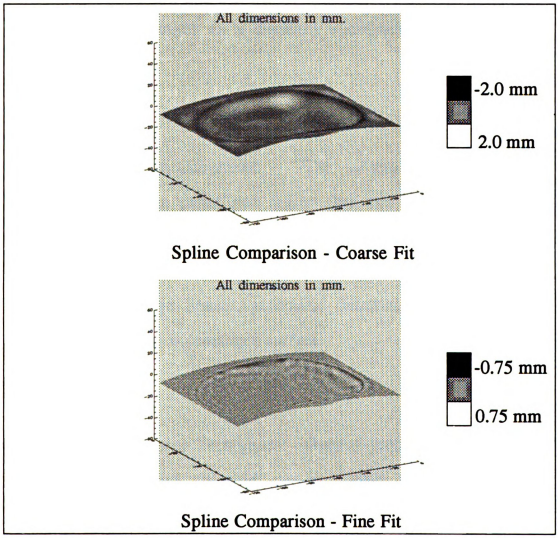


25a.

25b.

Figure 25.
Piston Head from a 600 cc. Kawasaki Engine

The surface comparison was performed using two different spline fits, the coarse spline approximation from Figure 21. and the fine approximation from Figure 23. The results of the comparison are presented in Figure 26. The dark regions represent areas where the approximated surface was below the actual surface, the light regions represent areas above the actual surface. See the Figure 26 for the scale.



Spline Comparisons
Figure 26

Chapter 7.

Results and Recommendations

7.1 Capabilities

The software package described in this work has the following capabilities.

- Interactive Manual Control: The system can be used interactively similar to a standard coordinate measurement machine.

- Automatic measurement: The system measures the surface inside a predefined region that can be as large as 960 mm. by 480 mm. with a maximum height change of 480 mm. Because the height is larger than the effective range of the Point Range Sensor, automatic searching routine were developed to aid in locating a surface.

- Grid Refinement Techniques: Once a preliminary surface grid is measured, the grid can be further subdivided into smaller increments or an adaptive searching algorithm can be used to measure only those intervals which could provide new information.

- Surface Fitting: Two schemes are used, a surface interpolation scheme which produces tensor-product B-spline surfaces, and a surface approximation scheme that is useful for data compression.

7.2 Limitations

SURFMAP has the following limitations.

- Due to Point Range Sensor Construction, only non-reentrant surfaces can be digitized. This could me modified be allowing another axis of movement, see the Recommendations, below.

- This package provides one surface at a time. This does not allow for a complete model that contains all surfaces that make up the object. To extend this process to the next stage of complexity, solid modeling techniques need to be used which incorporate the information developed by this package.

- The 15 degree angle needed by the Point Range Sensor for measurement limits the kinds of discontinuities that can be measured. The ability to rotate the sensor head with respect to the object would eliminate this limitation.

7.3 Recommendations

For immediate improvements the following issues need to be addressed.

- The sensor movement device: Using the traverse table limits the flexibility of the movement. The sensor would be more useful if given another degree of freedom. A suggestion for the accomplishment of the capability is the use of a four or five axis control arm, or perhaps a rotary stage.

- Surface description with discontinuities: The methods available in the literature for defining a surface with discontinuities need to be examined and implemented.

- Surface input and output should be done using spline description methods instead of surface points. This can save memory space in the computer and reduce the amount of data storage on disk by orders of magnitude.

Appendix

Appendix
SurfMap User's Guide

Section 1. Introduction

This section is the user's guide for the SURFMAP digitizing system. The first part contains a few hints for setting up objects for digitization. Following that is a step-by-step procedure describing how each of the main subprograms is used. A brief description of the very important input data file is next. This file contains most of the settings for the system that can be changed in the software. Next is a description of the pertinent settings, including information on plug-in cards, etc. Rounding out the practical user information are some notes on methods for displaying the data, including of description of how some of the analysis programs work.

The last part is a description of the SURFMAP system for someone interested in modifying the programs. It gives a description of the communications software used, shows the important data structures developed , and includes a directory tree showing the location of important subdirectories. A brief description of the simple menu system developed for this system is next.

Section 2. Setting up objects for digitization.

The digitization process works more easily if you keep the following notes in mind while setting up objects. These are just some practical suggestions for getting the most out of the system.

Object Surface: Surface diffusivity is a very important issue. The surface to be digitized must diffuse the laser spot rather than reflect it. Thus it is clear that polished metal surfaces are too reflective, also clear plexiglass is too transparent. A non-glossy coat of paint works well for metal surfaces while dusting surfaces such as plexiglass was effective. The issue is diffusivity, and any diffuse surface will work well. Care should be taken on surfaces that have areas of irregular diffusivity containing "spots" of highly reflective material. The surface should be treated to make the diffusivity as uniform as possible.

Object Orientation: Remember the limitations of the digitizer. The digitizer cannot do re-entrant regions, and only provides a projection of the surface. Set up the object to take best advantage of the digitizer's capabilities. If the plan is to digitize the surface in parts, be prepared with reference geometry to make matching the parts easier. Make the support for the object stable enough so that it isn't likely to move. Use clay or tape to fasten objects down in case the support is bumped.

Section 3. Procedures for Use

First set up the object appropriately as discussed in Section 2. The rest of this section will by step-by-step discussion of the procedures.

3.1 Manual Subprogram

1) The Program prints out the setup information. Examine the information to see if it is correct. Press a key to scroll through it.

2) If error messages are generated, exit and fix the problem. The error messages that would appear here typically are problems with the sensor. Password errors usually require a password to be set. This is accomplished through the use of the "Startup" program included in the PRS directory. See the CyberOptics manual for more information and Section 10.1 for the location of the PRS directory.

3) Enter the name of the file you with to have the data stored in.

4) If you with the data to be stored in x,y,z format, enter 1, otherwise enter 2.

5) Move to the point you wish to measure. See Section 3.6 for information on the movement commands.

6) When you complete the movement, the point will be measured. If no surface is located within the range of the

sensor, the routine will ask for an action to perform. See Section 3.7 for more information.

7) After a point is located, the routine prompts to see if you are done, proceeding back to step 5 if the answer is negative.

Section 3.2 Auto-1 Subprogram

1) The Program prints out the setup information. Examine the information to see if it is correct. Press a key to scroll through it.

2) If error messages are generated, exit and fix the problem. The error messages that would appear here typically are problems with the sensor. Password errors usually require a password to be set. This is accomplished through the use of the "Startup" program included in the PRS directory. See the CyberOptics manual for more information and Section 10.1 for the location of the PRS directory.

3) Enter the name of the file you with to have the data stored in.

4) Enter the distance between data points in the X and Y directions. This is the spacing of the grid.

5) Move to the last point you wish to measure. See Section 3.6 for more information on movement routines. If no point is found the user is asked for direction. See Section 3.7 for more information concerning the actions.

6) Move to the starting point. The program asks for these two points in reverse order so that it does not have to move back to the first point to begin measurement. The user is prompted for any necessary changes.

7) The program automatically steps along the grid determined by the previous input. If this is unsatisfactory, the user can change it by pressing a key and answering the appropriate questions. The table can also be moved in the same manner, if a large section were to be skipped. This is NOT recommended.

8) If the point cannot be measured by the sensor, the user is prompted for instruction. See Section 3.6 or 3.7 as appropriate.

9) The procedure in steps 7 and 8 repeats until the surface is complete, or an error condition requires exiting.

Section 3.3 Auto-2 Subprogram

1) The Program prints out the setup information. Examine the information to see if it is correct. Press a key to scroll through it.

2) If error messages are generated, exit and fix the problem. The error messages that would appear here typically are problems with the sensor. Password errors usually require a

password to be set. This is accomplished through the use of the "Startup" program included in the PRS directory. See the CyberOptics manual for more information and Section 10.1 for the location of the PRS directory.

3) Enter the name of the file you with to have the data stored in.

4) Enter the distance between data points in the X and Y directions. This is the spacing of the grid.

5) The user is prompted for a range of heights. They should be above the highest point you wish to measure and below the lowest.

6) Move to the last point you wish to measure. See Section 3.6 for more information on movement routines. If no point is found the user is asked for direction. See Section 3.7 for more information concerning the actions.

7) Move to the starting point. The program asks for these two points in reverse order so that it does not have to move back to the first point to begin measurement. The user is prompted for any necessary changes.

8) The program automatically steps along the grid determined by the previous input. If this is unsatisfactory, the user can change it by pressing a key and answering the appropriate questions. The table can also be moved in the same manner,

if a large section were to be skipped. This is NOT recommended.

9) The search routine then proceeds until done or interrupted.

Section 3.4 The Adaptive-1 Subprogram

1) The Program prints out the setup information. Examine the information to see if it is correct. Press a key to scroll through it.

2) If error messages are generated, exit and fix the problem. The error messages that would appear here typically are problems with the sensor. Password errors usually require a password to be set. This is accomplished through the use of the "Startup" program included in the PRS directory. See the CyberOptics manual for more information and Section 10.1 for the location of the PRS directory.

3) Enter the name of the file you with to have the data stored in.

4) Enter the name of the file that contains the surface information for input.

5) Enter the number of points per interval that you desire.

6) The program automatically steps along the grid determined by the previous input. If this is unsatisfactory, the user can change it by pressing a key and answering the appropriate

questions. The table can also be moved in the same manner, if a large section were to be skipped. This is NOT recommended.

8) The program continues until completed.

Section 3.5 The Adaptive-2 Subprogram.

1) The Program prints out the setup information. Examine the information to see if it is correct. Press a key to scroll through it.

2) If error messages are generated, exit and fix the problem. The error messages that would appear here typically are problems with the sensor. Password errors usually require a password to be set. This is accomplished through the use of the "Startup" program included in the PRS directory. See the CyberOptics manual for more information and Section 10.1 for the location of the PRS directory.

3) Enter the name of the file you with to have the data stored in.

4) Enter the name of the file that contains the surface information for input.

5) The surface is inspected using the chordal deviation method mentioned previously. The user is prompted only when absolutely necessary.

6) The procedure in step 5 repeats until the surface is complete, or an error condition requires exiting.

Section 3.6 The Movement Routines

1) *Move to a specific point.* Enter in the new point, each coordinate separated by **spaces.**

2) *Run an axis.* Choose the direction you wish to move the axis and press the arrow key associated with it. Press "Insert" to stop the table. Press "h" for fast movement or "l" for slow movement.

3) *Step an axis.* Enter the amount that you want the axis stepped. Next, enter in the direction by pressing an arrow key.

Section 3.7 The Searching Routine.

1) *Modify Sensor Word.* Enter in the string to send to the sensor. See the CyberOptics manual for more details. This is not generally useful.

2) *Move.* See Section 3.6

Section 4. The Input Configuration File

The input configuration is kept in the "prs_init.dat" file in each directory. The file is a list of number and letters, kept in a strict order. Do not change the order of the file without changing the input routine. Note that this file is usually kept as a read-only file for protection from accidental deletion.

The first 4 lines are the port configuration information. See Table 3. for the section of code that performs this input. The rest of the information is read by the code in Table 4. below.

```
if((configuration_file=fopen("prs_init.dat","r"))!=NULL) {
   fscanf(configuration_file," %i",&(dl->sonyport));
   fscanf(configuration_file,"
       %i,%i,%i,%i",&baud1,&par1,&stop1,&dat1);
    screen_control(6,"Port 1 configuration is:");
   sprintf(buffer,"baud=%i,parity=%i,stop bits=%i,data bits=%i\n",
     baud1,par1,stop1,dat1);
   screen_control(6,buffer);
   fscanf(configuration_file," %i",&(dl->modport));
   fscanf(configuration_file,"
       %i,%i,%i,%i",&baud2,&par2,&stop2,&dat2);
    screen_control(6,"Port 2 configuration is:");
    sprintf(buffer,"baud=%i,parity=%i,stop bits=%i,data bits=%i\n",
      baud2,par2,stop2,dat2);
        screen_control(6,buffer);
}
```

Table 3.
Port Configuration Code

```
void read_config(FILE *config) {
/* 28 items */
        fscanf(config,"%f",&prs->origin[_W_]);
        fscanf(config,"%f",&prs->origin[_X_]);
        fscanf(config,"%f",&prs->origin[_Y_]);
        fscanf(config,"%f",&prs->origin[_Z_]);
        fscanf(config,"%f",&prs->end[_W_]);
        fscanf(config,"%f",&prs->end[_X_]);
        fscanf(config,"%f",&prs->end[_Y_]);
        fscanf(config,"%f",&prs->end[_Z_]);
        fscanf(config,"%f",&prs->datadensity[_X_]);
        fscanf(config,"%f",&prs->datadensity[_Y_]);
        fscanf(config,"%f",&prs->datadensity[_Z_]);
        fscanf(config,"%i",&prs->autoexposure);
        fscanf(config,"%i",&prs->spotsize);
        fscanf(config,"%i",&prs->range);
        fscanf(config,"%i",&prs->minexposure);
        fscanf(config,"%i",&prs->maxexposure);
        fscanf(config,"%i",&prs->lightlevel);
        fscanf(config,"%i",&prs->topframe );
        fscanf(config,"%i",&prs->leftframe);
        fscanf(config,"%i",&prs->height);
        fscanf(config,"%i",&prs->width);
        fscanf(config,"%i",&prs->quietmode);
        fscanf(config,"%i",&prs->camera );
        fscanf(config,"%i",&prs->seelaser );
        fscanf(config,"%i",&prs->startscan);
        fscanf(config,"%i",&prs->endscan );
        fscanf(config,"%i",&prs->sensorrow);
        fscanf(config,"%i",&prs->avgrows );
        fscanf(config,"%i",&prs->avg );
        fscanf(config,"%i",&prs->units );
        fscanf(config,"%i",&prs->range);
        fscanf(config,"%lf",&prs->resolution);
        return;
}
```

Table 4.
Sensor Configuration Code.

Section 5. Equipment Configuration Information

This section contains the required settings for all the equipment. Also included is the location of all the plug-in cards.

Section 5.1 The Zenith TurboSport

Note that the PRS device driver must be initialized in the config.sys file for the device driver to work. The line should be as follows:

device=c:\thesis\prs\driver.sys

The computer must have the two-port serial card installed in the expansion box. See the Zenith manuals for information on how to set up the expansion box. The port settings must be as follows. These communication parameters correspond to the settings on the Sony and Modulynx computers.

Port 1: Connected to the Sony LM22S.
2400 baud, even parity, 7 data bits, and 2 stop bits.

Port 2: Connected to the Modulynx.
1200 baud, space parity, 7 data bits and 2 stop bits.

Note that the programs will set these automatically.

The serial card must also be set to work as port 1 and port 2. See the documentation on the serial card for more information.

Section 5.2 The Modulynx Computer.

The Modulynx computer must be set up to be controlled externally. The red switch must be set to "computer", NOT "local." Internal to the computer is the communication card. This card has dip switches corresponding to the communication configuration. The card must be set up the same as Port 2 in section 5.1

Section 5.3 The Sony LM22S

The Sony must be set up in accordance with the communication parameters described in section 5.1. The only other important setting is the switch for metric or English units. Use ONLY the metric units (mm.).

Section 6. Manipulating the data

Making a 3-d plot of the data is a good and fast way to examine the data for inconsistencies. The graphics package WAVE published by Precision Visuals works well. The data must be arranged in a rectangular grid. This works well with the Auto-1 and Auto-2 programs, although some adjustment could be needed. Typically duplicating a point or two works well.

Once you have the data in the correct form, read it into WAVE using the "array_read" procedure. This reads the data into a (3 x n) array. The x-coordinate is row 0, the y-coordinate is row 1, and the z-coordinate is row 2. These should be split into separate 2-d arrays using the "reform" command. Note that the user should know the dimensions

of the 2-d array. The reform command is usually:

x = reform(input(0,*),r,s) where r,s are the dimensions of the 2-d array.

Perform this calculation for all three components and the data is ready for display. To display a wireframe of the surface the command is:

surface, z, x, y

See the WAVE documentation for more details. See also the "shade_surf" command.

The "SURFER" routines fit cubic splines to the data in a given density of control points. See the procedure for details. There are 4 SURFER routines, and a number of assorted procedures to do other stuff. Note that the built-in "surface" routines is enough to display the data, however the other stuff is added for completeness. Table 5. is a list of all of the WAVE procedures developed for use.

2dconnect.pro	maf.pro
2dpoints.pro	maf2.pro
ambrot.pro	notchdisplay.pro
array_bound.pro	notdis2.pro
array_bounds.pro	pc1.pro
axis_set.pro	pdf.pro
cc1_air5.pro	points.pro
cc_1.pro	ptsdis.pro
cc_2.pro	rdpts.pro
column_write.pro	rot.pro
connect.pro	rot2.pro
connect1.pro	rp1.pro
connect2.pro	shad_splin.pro
connect3.pro	splin_dis.pro
connectall.pro	surfer.pro
curv_cover.pro	surfer2.pro
curvature.pro	surfer3.pro
draw1.pro	surfer4.pro
flatdis.pro	surfread.pro
flatdis2.pro	threevbls.pro
house.pro	twodconnect.pro
invertarray.pro	twodpoints.pro
invertmatrix.pro	wire_splin.pro

Table 5.
WAVE procedures

The calculation of the surface difference proceeds as follows.

1) Calculate and approximation using either the cubic spline the WAVE provides or Bezier/B-splines provided by SURFPAK. This will be your "standard" so a very tight spline fit is recommended, or use the interpolation scheme that SURFPAK provides.

2) Generate and approximation of the second surface with the same x,y coordinates of the of the first approximation. Use the SURFER4 procedure.

3) The two arrays should be the same dimension, subtract the second from the first.

4) Use the result of step 3) as the shading parameter for the shade_surf procedure. Byte-scale the array first.

5) Display, use a command like:

WAVE> shade_surf, z, x, y, shad=Bytscl(difference)

Section 7. The Communication Software

The communication software is the commercially available package, "C Asynchronous Manager." The programs included need the object file ASYNCH.OBJ to be linked with the other files to enable the communication. This file is usually kept as a read-only file for protection against accidental deletion.

Section 8. Important Data Structures

This section contains descriptions of the important data structures contained in the SurfMap programs.

Section 8.1 Location List Structure

The location list structure contains the information for each data point. The x and y coordinates are self explanatory. The *w* coordinate contains the height of the table. The *z* coordinate contains the *w* coordinate combined with the sensor reading. Thus the *z* coordinate minus the *w* coordinate contains the distance measured by the sensor. See Table 6. for the structure definition.

Section 8.2 Row Header Structure

The index structure contains pointers to each "row" of data points, including pointers to other rows. See the appropriate source code for more details. See Table 6. for the definition.

Section 8.3 Device List Structure

The device list structure contains the port definitions of the various devices. See Table 6.

Section 8.4 Action List Structure

This structure implements the linked-list that contains the actions to be performed in the search procedure. It contains the action to be performed, and a switch to indicate completeness.

Section 8.5 PRS structure

This structure contains most of the global definitions for use throughout the programs. See Table 7. for the definition of the structure.

Section 8.6 Movement List Structure

The movement list is the list of parameters for the table movement. It contains the base speeds, accelerations, and high speeds of each axis. See the appropriate documentation. Also see Table 6.

Section 8.7 Field Info Structure

The field information is the row and column locations of the beginning and end of each defined window. See the appropriate source code for more information. See Table 6.

Section 8.8 Search Information Structure

This structure is used in the Auto-1 program to save the information needed to perform searching actions. It contains the slope in the two orthogonal directions, and the last point definition. See Table 6.

```
/* data structure declarations */
struct movement_list {
   long x_base,y_base,z_base,x_high,y_high,z_high,x_acc,y_acc,z_acc;
   };

struct location_list {
     int status;
     float w,x,y,z; /* w is the height of the table, z is the height
combined with the sensor reading */
     struct location_list *next,*previous;
     };

struct device_list {
     int sensor;
     int sonyport,modport;
     };

struct index {
     struct location_list *row;
     struct index *next_column,*previous_column;
     };

struct action_list {
   int field,done;
   struct action_list *next;
   };

struct search_information {
   float x,y,z;
   int points;
   float slopey,slopex;
   };

struct field_info {
  short int row_begin,row_end,column_begin,column_end,
  current_col, current_row, label;
}
```

Table 6
Structure Definitions

```
/* data structure declarations */

struct A {
        float origin[4];          /*origin of window */
        float end[4];             /* opposite corner of window. */
        float datadensity[3];     /* step size in the x,y,z directions */
        int autoexposure;         /* sets autoexposure control algorithm. */
        int spotsize;             /* optimum spot size in pixels */
        int range;                /* range of spotsize in pixels */
        int minexposure;          /* shortest possible exposure time*/
        int maxexposure;          /* longest exposure time */
        int lightlevel;           /* light level of laser */
        int topframe;             /* top of frame buffer for detector window
                                     processing, in pixels.*/
        int leftframe;            /* left side of frame buffer */
        int height;               /* height of frame buffer */
        int width;                /* width of frame buffer */
        int quietmode;            /* controls error messages  */
        int camera;               /* controls camera mode */
        int seelaser;             /* controls visibility of laser */
        int startscan;            /* start of scan for row averaging*/
        int endscan;              /* end of scan */
        int sensorrow;            /* center row for averaging */
        int avgrows;              /* number of rows to average */
        int avg;                  /* controls averaging */
        int units;                /* controls units */
        int rnge;                 /* controls data collection  */
        double resolution;        /* resolution for movement accuracy */
        };

/* end of definitions */
```

Table 7.
PRS Structure Definition

Section 9. The Menu System

The menu system works for the programmer through the "screen_control" subroutine. A typical line of code is:

```
screen_control(MESSAGE_WIN,"print this message");
```

The MESSAGE_WIN macro is defined in the screen.h include file. It points to the message window, which is defined in the init_scr.c procedure. The programmer must initialize the window first, then just include the procedure definitions and macros.

Section 10. Description of the File System

Section 10.1 Schematic

The file system schematic is presented in Figure 27. below. See the following subsection for locations of important files.

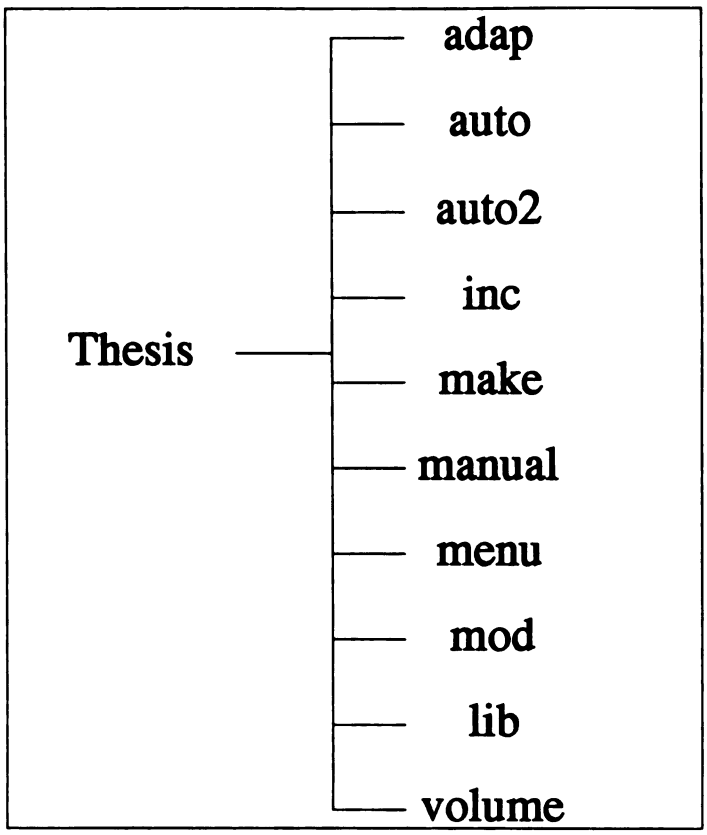


Figure 27. File System Tree.

Section 10.2 Location of Important Files

The most important files are contained within the **inc**, **make**, and **lib** subdirectories. The include files and the library files need to be set up in the search path for the compiler. See the compiler documentation. The make files are kept in each pertinent subdirectory and in the **make** subdirectory.

References

References

Barsky, B. (1981) The Beta Spline: A Local Representation based on Shape Parameters and Fundamental Geometric Measures., PhD. Thesis, University of Utah.

Böhm, W., Farin, G., and Kahnman, J. (1984) A Survey of Curve and Surface Methods in Computer Aided Geometric Design., Computer Aided Geometric Design 1(1) pg. 1-60

Coons, S., (1964) Surfaces for Computer aided Design., Technical Report, MIT.

Farin, G., Rein, G., Sapidis, N., Worsey, A.J. (1987) Fairing cubic B-spline curves., Computer Aided Geometric Design 4, pg. 91-103

Farin, G. (1990) Curves and Surfaces for Computer Aided Geometric Design., Academic Press, San Diego, CA.

Foley, T.A. (1987) Weighted Bicubic Spline Interpolation to Rapidly Varying Data, ACM Transactions on Graphics 6(1)

Gordon,W. and Riesenfield, R.E. (1974) B-spline curves and surfaces, in Computer Aided Graphic Design, Academic Press, NY, pg 95-126

Gregory, J., (1986) N-sided surface patches., in J. Gregory, editor, The Mathematics of Surfaces., Clarendon Press, pg. 217-232

Hawkins, J.L. (1987), The Wonder of B-Splines, Case Center for CAE Technical Report, Michigan State University.

Jalkio, J.A., Kim, R.C., and Case, S.K. (1985), Three Dimensional Inspection using Mulitstripe Structured Light., Optical Engineering 24(6), pg. 966-974

Mercer, C.R., and Behiem, G., (1991) Fiber-Optic Phase-Stepping system for interferometry. Applied Optics 30(7) pg. 729-734

Oliver, J.H. (1986) Graphical Verification of Numerically controlled Milling Programs for Sculptured Surface Parts, PhD. Diss, Michigan State University.

Precision Visuals, Inc. Boulder Co. 80301

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.,(1988) Numerical Recipes in C, The Art of Scientific Computing.,Cambridge University Press,Cambridge

Quilin, D., and Davies, B.J. (1987) Surface Engineering Geometry for Computer-Aided Design and Manufacturing., Ellis Horwood LTD. England.

Riesenfield, R.E. (1973), Applications of B-spline Approximation to Geometric problems of Computer Aided Engineering, Diss. Syracuse University.

Rogers, D.F. and Adams, J.A. (1990) Mathematical Elements for Computer Graphics, 2nd Ed. McGraw Hill, NY.

Schmitt, F.J.M., Barsky, B., Du, W., (1986) An Adaptive Subdivision Method for Surface-fitting from Sampled Data. SIGGRAPH Proceedings 20(4) pg. 179-188

Sederburg, T.W. and Meyer, R.J. (1988) Loop Detection in Surface Patch Intersections., Computer Aided Geometric Design 5, pg. 161-171

Wysocki, D.A., (1987) Generation, Verification, and Correction of Numerical Control Tool Paths, Master's Thesis, Michigan State University.