The Use of Multinomial N-Dimensional Curve Fitting in Computer Graphics

by

Richard Huntley Cole

A thesis
presented to the University of Waterloo
in partial fulfillment of the
requirements for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, 1983

The Use of Multinomial N-Dimensional Curve Fitting in Computer Graphics

by

Richard Huntley Cole

A thesis
presented to the University of Waterloo
in partial fulfillment of the
requirements for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, 1983

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature 

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature 

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

The Use of Multinomial N-Dimensional Curve Fitting in Computer Graphics

Richard H. Cole

University of Waterloo
Department of Computer Science
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

Polynomial one dimensional curve fitting by least squares is a classical method for placing smooth curves along data points. Recurrence relations based on orthogonal polynomials construct these curves quickly and efficiently. The fitting method can be generalized to N dimensions using orthogonal multinomials.

Multinomial N-dimensional curve fitting can be applied to the area of graphical surface modeling. Parametric equations fit to spatial data points describing a surface are used to render graphical objects.

The parametric values assigned to data points have profound effects on the resultant curve fit. Using methods from statistical pattern recognition data points can be assigned parametric values that are statistically best.

This thesis presents original work in the automatic parameterization of data and the subsequent fitting of multinomial surfaces. A demonstration program has been developed using this work to produce and display three dimensional parametric surfaces.

The work reported here was funded in part by grants A3022 and A4076 of the Natural Sciences and Engineering Research Council of Canada.

Acknowledgements

I would like to thank, first of all, Richard Bartels who was my guide through all of this work and without whom this Thesis would have been impossible.

I would also like to thank Iris Wagner, my wife who inspired my interest in Computer Science and supported me throughout. All those in the graphics lab who found my dumb bugs and kept me company at all hours of the day and night are greatly appreciated.

Last and not least, I would like to thank Gaggia of Italy, makers of fine espresso coffee machines, by far the best fluidic peripheral I have ever seen.

Contents

Glossary

$\vec{s}$	Random data vector, also $\vec{x}$.
$s^{(n)}$	n^{th} member in a set of data.
$f^{(n)}$	Dependent variable associated with $s^{(n)}$.
$\epsilon^{(n)}$	Error between approximating function and $f^{(n)}$.
$f(s)$	Approximating function to data values.
Φ_i	i^{th} basis polynomial.
Ψ_i	i^{th} orthogonal basis polynomial.
j	Scalar index of orthogonal basis multinomial.
$< g, h >$	Inner product between two functions g and h.
c_i	Coefficient of terms in polynomial fit.
γ_i	Coefficient in fit using orthogonal multinomials.
a_i	Coefficient of i^{th} orthogonal basis polynomial.
α_i	Coefficient of i^{th} term in the recursion that produces the orthogonal basis multinomials.
q	Degree of the multinomial.
K	Number of independent dimensions.
k	Index of independent variable.
$Z_i(\vec{s}^{(n)})$	Residual difference between fit with i and $i-1$ basis multinomials.
$\mathbf{A}$	Linear Transformation mapping data vectors into a space where there features are uncorrelated.
$E[\vec{x}^{(n)}]$	Expected value of n data points.
$\vec{\mu}$	Mean vector of a set of data.
$\vec{m}$	Sample mean.
Σ	Covariance matrix.
S	Sample covariance matrix.
ϕ	Eigenvector of Σ.
Φ	Matrix of the eigenvectors.

1. Introduction

Discrete least squares fitting is a computational process used to model or smooth input data in which a function, sampled with error on a finite set of points, is approximated by a linear combination of a set of fitting functions. A variety of fitting functions, splines, trigonometric functions and exponentials as well as polynomials are used in varying applications. Multidimensional fitting uses fitting functions of more than one independent variable. Chapter 2 of this thesis presents a method of least squares fitting with polynomial and multinomial fitting functions.

1.1. Uses of Multinomial Equations in Computer Graphics

Least squares fitting can be used in computer graphics for a variety of applications. In all of them curves or surfaces are fit to input data points and used to model the data.

Input from empirical or other error prone sources can be passed through a "least squares filter" to eliminate much of the effect of noise before further processing.

Parametric curves or surfaces are fit to data points to create graphical objects. The surfaces follow the general trend of the input data but do not necessarily go through the data points themselves. Using a sufficient number of points and sufficiently high order multinomials, acceptable looking objects or portions of objects can often be obtained. Figures

9 and 13 present such objects.

Multinomial fitting is not easily applied to closed or periodic (multivalued) patches. However, using *a priori* knowledge of the desired shape, the underlying parametric coordinates of the data points defining a patch or curve can be selected to give acceptable approximations to simple closed surfaces and curves. End conditions are, however, a problem, since the borders of the surface fitted are unconstrained. Chapter 3 addresses these problems.

Least squares fitting can also be used as a preprocessor for data points. Surface representations using Cartesian Product B-splines require a regularly spaced mesh of control points. These control points could be generated from a multinomial least squares fit to randomly scattered data points.

Yet another possibility for the use of least squares multinomials is to smooth existing surface representations. With simple surface representations, e.g. polyhedra, problems arise in finding slopes and normals. A local fit to such piecewise linear surfaces can be computed for the sake of obtaining slopes and normals.

Illumination algorithms can also make use of surface fits. One possibility is to replace complex lighting calculations with a multinomial approximation, fit to sample illumination values. The number of independent variables describing the surface depends on the complexity of the algorithm.

2. The Least Squares Fit of Discrete Data

2.1. Introduction

This chapter summarizes the theory behind the fitting scheme used in this research. The scheme is to compute a least squares fit with orthogonal multinomials to data points. The orthogonal multinomials are derived from the surface data points. The algorithm presented fits an arbitrary order surface in multiple dimensions. The fit is limited only in that the number of independent multinomials comprising the fit must not be greater than the number of data points.

Section 2.2 presents the formulation of the least squares criterion and the use of orthogonal polynomials in a least squares fit. Section 2.3 generalizes the formulation to multiple dimensions and surface fitting using orthogonal multinomials.

2.2. Least Squares Curve Fit

A least squares curve fit is often used to model data. It is frequently assumed, for the sake of convenience, that the function represented by the data is some form of a polynomial. The data measurements represent this polygonal function plus some amount of noise. The least squares criteria fits the data with a linear combination of polygonal fitting functions. It is hoped that a least squares curve fit extracts the 'true' representation of the function out of the data, leaving the noise behind. The averaging effect of the least squares fitting process leads to smooth polynomial curves.

2.2.1. The Least Squares Criteria and Orthogonal Polynomials

If $f(s)$ is a polynomial to be fitted to a set of values $f^{(n)}$ defined at a number of points $s^{(1)}, \ldots, s^{(N)}$ then each $f(s^{(n)})$ differs from $f^{(n)}$ by some residual $\epsilon^{(n)}$, or

$$f(s^{(n)}) = f^{(n)} + \epsilon^{(n)}, \quad n = 1, \ldots, N \quad . \tag{1}$$

The function $f(s)$ is a linear sum of terms:

$$f(s) = c_1\Phi_1(s) + c_2\Phi_2(s) + \ldots. + c_M\Phi_M(s) \qquad (M \leq N) \, , \tag{2}$$

where the $\{\Phi_i\}$ are an *a priori* selected set of basis polynomials and the $\{c_i\}$ are the coefficients which are the variables in the fit. Conventionally the $\{\Phi_i\}$ are chosen to be of order $i-1$ or in the simplest case $\Phi_i = s^{i-1}$, s being the dependent variable. The coefficients are determined using

$$\underset{c_1, \ldots, c_m}{\text{minimize}} \sum_{n=1}^{N} \left(\epsilon^{(n)}\right)^2 \, .$$

This is the least squares criterion. The minimum is found by solving a set of linear equations known as the *normal equations*:

$$\begin{bmatrix} <\Phi_1,\Phi_1> & \cdots & <\Phi_1,\Phi_M> \\ \vdots & & \vdots \\ <\Phi_1,\Phi_M> & \cdots & <\Phi_M,\Phi_M> \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} <\Phi_1,f> \\ \vdots \\ <\Phi_M,f> \end{bmatrix} \tag{3}$$

where

$$<\Phi_i,\Phi_j> = \sum_{n=1}^{N} \Phi_i(s^{(n)})\, \Phi_j(s^{(n)}) \tag{4}$$

and

$$<\Phi_i,f> = \sum_{n=1}^{N} \Phi_i(s^{(n)})\, f^{(n)} \ . \tag{5}$$

Unfortunately, this set of equations is costly to solve and is often ill-conditioned with the conventional choice of basis polynomials. A development of the normal equations can be found in Conte and de Boor [Conte80].

Both problems can be overcome by the selection of a more appropriate basis, i.e. by selecting better Φ_i. Selecting orthogonal polynomials as the Φ_i, that is, choosing the Φ_i to satisfy:

$$<\Phi_i,\Phi_j> \begin{cases} = 0 & \text{if } i \neq j \\ > 0 & \text{if } i = j \end{cases} \tag{6}$$

simplifies the normal equations to diagonal form,

$$\begin{bmatrix} <\Phi_1,\Phi_1> & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & <\Phi_M,\Phi_M> \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} <\Phi_1,f> \\ \vdots \\ <\Phi_M,f> \end{bmatrix} \tag{7}$$

from which the solution for the coefficients is given merely by

$$c_i = \frac{<\Phi_i,f>}{<\Phi_i,\Phi_i>} \ , \tag{8}$$

an easy calculation.

The use of orthogonal polynomials has an additional advantage. If the Φ_i are chosen to be of increasing degree, an increase in the order of the fit simply extends the previous fit. Only $<\Phi_{M+1},\Phi_{M+1}>$ and $<\Phi_{M+1},f>$ are needed to produce c_{M+1} from (8), and the fit $c_1\Phi_1(s) + c_2\Phi_2(s) + \ldots + c_{M+1}\Phi_{M+1}(s)$ is simply (2) with a single term added.

However, there remains the selection of the orthogonal polynomials. An orthogonal basis of polynomials can be obtained from the conventional basis by the Gram-Schmidt process. In the case of a single independent variable, i.e. for curves, Forsythe [Forsythe57] pointed out that the Gram-Schmidt process can be reduced to a three-term recurrence relation. A discussion of this can be found in Conte and de Boor [Conte80]. In higher dimensions, i.e. for surfaces, a more complicated recurrence relation pertains for the selection of orthogonal multinomials. The next section presents the Gram-Schmidt process and Forsythe's construction in one dimension. The recurrence relation with respect to higher dimensions will be examined in a later section.

2.2.2. Construction of Orthogonal Polynomials

The Gram-Schmidt process can be expressed as follows.

Given one basis for a space $\Phi_1, \ldots, \Phi_M$, a second basis $\Psi_1, \ldots, \Psi_M$ is formed iteratively:

$$\begin{aligned} \Psi_1 &= \Phi_1 \\ \Psi_j &= \Phi_j - \sum_{i=1}^{j-1} a_{ji}\Psi_i \end{aligned}, \qquad (9)$$

where the coefficients a_{ji} are chosen to remove from Φ_j all contributions which it might contain from any of the Ψ_i already constructed. What is left will be orthogonal to all of the Ψ_i already constructed. Recall a few simple properties of the inner product $<,>$ which are used to solve for the coefficients a_{ji}:

$$
\begin{aligned}
\langle g, h \rangle &= \sum_{n=1}^{N} g(s^{(n)}) h(s^{(n)}) \ ; \\
\langle g, g \rangle &\geq 0 \quad \text{and is positive if } g \neq 0 \ ; \\
\langle g, h \rangle &= \langle h, g \rangle \ ; \\
\langle g_1 + g_2, h \rangle &= \langle g_1, h \rangle + \langle g_2, h \rangle \ ;
\end{aligned}
$$

and

$$
\langle \beta g, h \rangle = \beta \langle g, h \rangle \ ,
$$

where the g's and h are any two functions $g(s)$, $h(s)$ defined on the data set $\{s^{(n)}\}$ and β is any real number.

The above properties are used to sketch out the Gram-Schmidt process.

2.2.2.1. The Gram-Schmidt Process

The desired scheme is

$$
\begin{aligned}
\Psi_1 &= \Phi_1 \\
\Psi_j &= \Phi_j - \sum_{i=1}^{j-1} a_{ji} \Psi_i \ , \ \text{for } j > 1 \ .
\end{aligned}
\tag{10}
$$

We wish to chose the a's so that

$$
\langle \Psi_m, \Psi_j \rangle = 0 \ , \ \text{for } m < j \ , \tag{11}
$$

i.e. such that Ψ_j is orthogonal to all previous polynomials. Using this

$$
\begin{aligned}
\langle \Psi_m \ , \ \Phi_j - \sum_{i=1}^{j-1} a_{ji} \Psi_i \rangle &= \langle \Psi_m \ , \ \Phi_j \rangle - \sum_{i=1}^{j-1} a_{ji} \langle \Psi_m \ , \ \Psi_i \rangle \\
&= \langle \Psi_m \ , \ \Phi_j \rangle - a_{jm} \langle \Psi_m \ , \ \Psi_m \rangle \\
&= 0
\end{aligned}
\tag{12}
$$

(by the orthogonality of the Ψ_i s)

which implies that

$$a_{jm} = \frac{<\Psi_m, \Phi_j>}{<\Psi_m, \Psi_m>} \quad . \tag{13}$$

2.2.2.2. Forsythe's Method

Forsythe [Forsythe57] showed that, if the Φ's are the powers of s, $\Phi_j = s^{j-1}$, then Ψ_j can be written as

$$s\Psi_{j-1} + \text{(lower-order terms)} \quad ,$$

and (10) can be written in the slightly different form

$$\Psi_j = s\Psi_{j-1} - \sum_{i=1}^{j-1} \alpha_{ji}\Psi_i \quad . \tag{14}$$

In this case all α's, save two, turn out to be zero and the Gram-Schmidt process reduces to a three-term recurrence:

$$\Psi_1 = 1 \tag{15}$$

$$\Psi_2 = s\Psi_1 - \alpha_{21}\Psi_1$$

and

$$\Psi_j = s\Psi_{j-1} - \alpha_{j,j-1}\Psi_{j-1} - \alpha_{j,j-2}\Psi_{j-2} \quad , \; j>2 \quad .$$

This ensures that the highest order term of Ψ_j is s^{j-1}. The coefficients, α, are given by

$$\alpha_{ji} = \frac{<s\Psi_{j-1}, \Psi_i>}{<\Psi_i, \Psi_i>} \quad . \tag{16}$$

2.3. Least Squares Extension to Orthogonal Multinomials

The extension of the least squares curve fit to multiple dimensions (multiple independent variables) follows the formulation of least squares presented in the last section. In multiple dimensions the terms of the fit are multinomials rather than polynomials. The construction of the orthogonal multinomials however is more complicated than the

construction for the polynomials. The next section presents the generalization of the least squares curve fit to K dimensions. The construction used closely follows that presented by Weisfeld [Weisfeld59], but also contains improvements found in Bartels and Jezioranski [Bartels83].

2.3.1. Production of Orthogonal Multinomials

Consider a set of data points representing a function with more than one independent variable $s_1, \ldots, s_K$, where K is the number of dimensions. If each vector represents the values of the independent variables at a point, there exists a related dependent value f. Let

$$(\vec{s}^{(n)}, f^{(n)}) = (s_1^{(n)}, \ldots, s_K^{(n)}, f^{(n)}), \quad n = 1, \ldots, N \tag{17}$$

where $\vec{s}^{(n)}$ is the vector representing the n^{th} data point and $f^{(n)}$ is the value of the dependent variable at that point.

Example 1 : Production of Multinomials

Consider the ordering of monomials Φ_i suggested by the following example with three independent variables. The row numbers correspond to the degree of the multinomials in that row.

$$\begin{aligned} &\text{Row 0 :} \quad \Phi_1 \\ &\text{Row 1 :} \quad \Phi_2 \ \ \Phi_3 \ \ \Phi_4 \\ &\text{Row 2 :} \quad \Phi_5 \ \ \Phi_6 \ \ \Phi_7 \ \ \Phi_8 \ \ \Phi_9 \ \ \Phi_{10} \\ &\text{Row 3 :} \quad \Phi_{11} \ \cdots \end{aligned} \tag{18}$$

The simplest production for the Φ's gives:

$$\begin{aligned} &\text{Row 0 :} \quad 1 \\ &\text{Row 1 :} \quad s_1 \ \ s_2 \ \ s_3 \\ &\text{Row 2 :} \quad s_1^2 \ \ s_1 s_2 \ \ s_1 s_3 \ \ s_2^2 \ \ s_2 s_3 \ \ s_3^2 \\ &\text{Row 3 :} \quad s_1^3 \cdots \end{aligned} \tag{19}$$

Each row contains all the possible multinomials of the appropriate degree. These

multinomials are, however, not orthogonal.

Following Forsythe's approach for obtaining orthogonal multinomials, we want to find k_j and j' such that

$$\Phi_j = s_{k_j}\Psi_{j'} + \text{(lower-order terms)} ,$$

giving us

$$\begin{aligned} \Psi_1 &= 1 \\ \Psi_j &= s_{k_j}\Psi_{j'} - \sum_{i=1}^{j-1} \alpha_{ji}\Psi_i \ . \end{aligned} \tag{20}$$

where

$$\alpha_{ji} = \frac{\langle s_{k_j}\Psi_{j'}, \Psi_i \rangle}{\langle \Psi_i, \Psi_i \rangle} \ . \tag{21}$$

As in the one dimensional case, many of the α's turn out to be zero: there exists an index j'' for which α_{ji} is zero for $i<j''$. This constructs multinomials such that each new Ψ_j has a leading term consistent with (19). If we think of the set of monomials arranged in rows, multinomials are produced as follows:

Example 2 : Production of Orthogonal Multinomials

$$\begin{aligned} &\text{Row 0: } \Psi_{1_{-1}} \\ &\text{Row 1: } \Psi_2 = s_1\Psi_1 - T_2 \quad \Psi_3 = s_2\Psi_1 - T_3 \quad \Psi_4 = s_3\Psi_1 - T_4 \\ &\text{Row 2: } \Psi_5 = s_1\Psi_2 - T_5 \quad \Psi_6 = s_1\Psi_3 - T_6 \quad \Psi_7 = s_1\Psi_4 - T_7 \quad \Psi_8 = s_2\Psi_3 - T_8 \quad \Psi_9 = s_2\Psi_4 - T_9 \quad \Psi_{10} = s_3\Psi_4 - T_{10} \\ &\text{Row 3: } \Psi_{11} = s_1\Psi_5 - T_{11} \quad \Psi_{12} = s_1\Psi_6 - T_{12} \quad \Psi_{13} = s_1\Psi_7 - T_{13} \quad \Psi_{14} = s_1\Psi_8 - T_{14} \ \cdots \end{aligned} \tag{22}$$

where

$$T_j = \sum_{i=j'}^{j-1} \alpha_{ji}\Psi_i \ .$$

The example demonstrates the ordering and the production for multinomials that will be used

throughout the following discussion.

2.3.1.1. Scalar Indexing

It is conceptually advantageous to refer to the multinomials using scalar indices. Let j be the scalar index. Figure 1 shows the relationship between the j^{th}, j'^{th}, j''^{th} multinomials and k_j, independent variable associated with the j^{th} multinomial. The illustration presents the relationship for three independent variables.

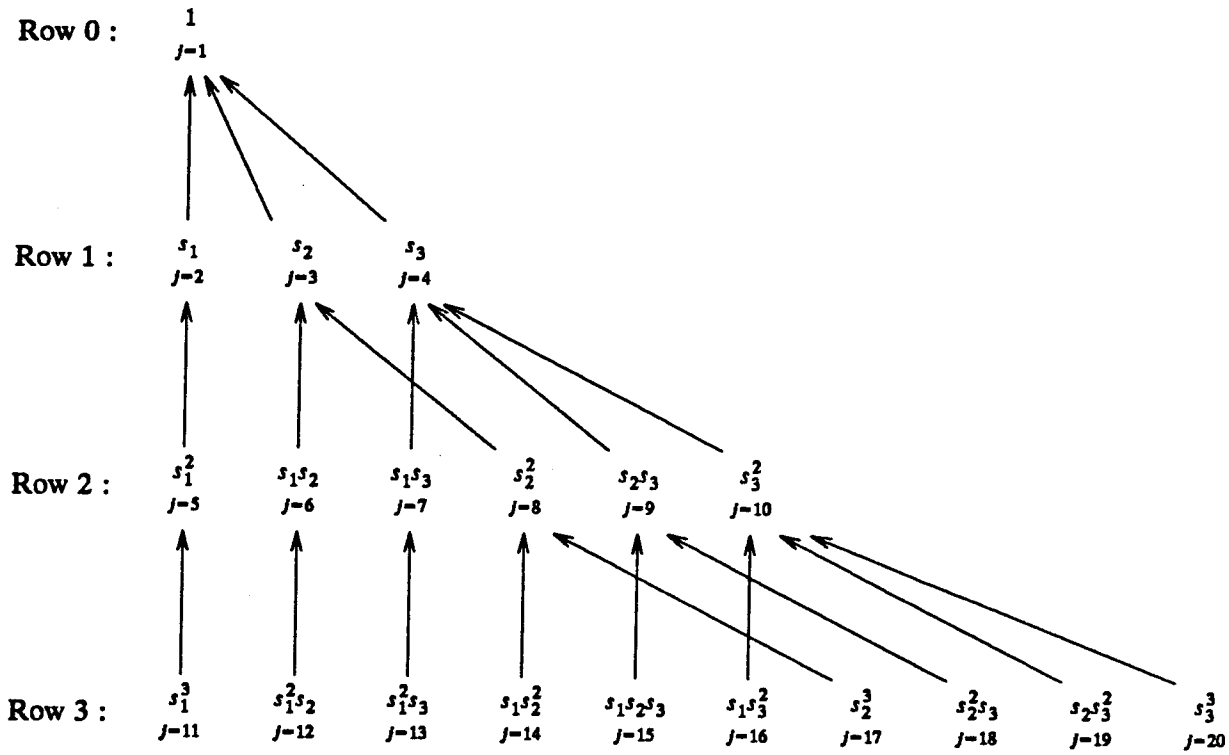


Figure 1 : Relationship between the Multinomials.

Tracing the arrows back one row in the diagram gives the j' multinomial (at the head of the arrow) corresponding to the j^{th} multinomial (at the tail of the arrow). The corresponding j'' for each j can be found by following the arrows back one more row. Bartels and Jezioranski show that coefficients of the terms with indices $< j''$ will be zero. Each j'' corresponds to j' as j' does to j. It can be seen that in this case the relation that gives j' for

each j also gives j'' when applied to j' (This is also true for the general case). The algorithm for finding j' from j is explained below.

We will use q to index the rows. There is only one term in the zeroth row and it is identically 1. The number of terms in a row will be referred to as the run length The q^{th} row can be divided into K ranges. Each range represents the j's which are produced by multiplying the j'^{th} monomial by s_{k_j} for a particular $k_j \leq K$. The run length of each row can be obtained from

$$\text{run length} = \frac{(K+q)!}{K!\ q!} - j \quad , or \tag{23}$$

$$\text{run length} = \binom{K-q-1}{q} \quad , \tag{24}$$

where K is the number of dimensions, q is the degree of the monomials in the present row and j is the index of the last monomial in the previous row. The length of the k^{th} range in the q^{th} row can be obtained from

$$\text{range length} = \begin{cases} 1, & \text{if } k=K \\ \binom{K-k-1-q}{q-1}, & \text{if } k<K \end{cases} . \tag{25}$$

As we proceed from one range to the next in a particular row, j being incremented by one, j' is also incremented by one. As j is incremented past the end of the k^{th} range in the current row, j' is reset to the beginning of the $k-1^{th}$ range in the previous row. This process continues to the end of the row, at which point the present row becomes the previous row, the length and range lengths of the new row are calculated, and the algorithm continues.

This is justified in Bartels and Jezioranski [Bartels83] with j, k_j, j', and j'' defined such that

$$\Psi_j = s_{k_j}\Psi_{j'} - \sum_{i=j'}^{j-1} \alpha_{ji}\Psi_i \ . \tag{26}$$

for

$$\alpha_{ij} = \frac{<s_{k_j}\Psi_f, \Psi_i>}{<\Psi_i, \Psi_i>} \quad . \tag{27}$$

yields orthogonal multinomials.

2.3.2. The Surface Fit

We now know how to calculate the Ψ_i's such that they are orthogonal with respect to the input data. It remains to define how to preform the least squares fit using these multinomials.

For given data points $\{ \vec{s}^{(n)}, f^{(n)} \}$, where $\vec{s}^{(n)}$ is the vector representing the independent variables, and $f^{(n)}$ is the dependent variable at that point, we want to find the function

$$f(\vec{s}) = \gamma_1\Psi_1(\vec{s}) + \cdots + \gamma_M\Psi_M(\vec{s}) \quad , \tag{28}$$

such that $f(\vec{s}^{(n)})$ "most closely" approximates $f^{(n)}$, the value of the dependent variable at the given point. $f(\vec{s}^{(n)})$ is the sum of the products $\gamma_m\Psi_m(\vec{s}^{(n)})$, where $\Psi_m(\vec{s}^{(n)})$ is the value of the m^{th} orthogonal multinomial at data point n and the γ_m is the coefficient of that term.

$$\gamma_i = \frac{<\Psi_i, f>}{<\Psi_i, \Psi_i>} \quad . \tag{29}$$

This will not necessarily give the optimal numerical solution for the γ's. A better formulation can be found in Cadwell and Williams [Cadwell61]. Their formulation for γ follows. Assuming

$$\begin{aligned} \gamma_i &= \frac{<\Psi_i, f>}{<\Psi_i, \Psi_i>} \\ &= \frac{\sum_{n=1}^{N} (\Psi_i(\vec{s}^{(n)}) f^{(n)})}{\sum_{n=1}^{N} (\Psi_i(\vec{s}^{(n)}) \Psi_i(\vec{s}^{(n)}))} \quad . \end{aligned} \tag{30}$$

and letting $z_i(\vec{s}^{(n)})$ be the residual at data point n after the curve fit of $(i-1)$ basis multinomials has been subtracted, or

$$Z_i(\vec{s}^{(n)}) = f^{(n)} - \sum_{k=1}^{i-1}(\gamma_k \Psi_k(\vec{s}^{(n)})) \quad . \tag{31}$$

We now replace $f^{(n)}$ in (30) with Z_i and the equation for γ_i becomes

$$\gamma_i = \frac{\sum_{n=1}^{N}(Z_i(\vec{s}^{(n)})\ \Psi_i(\vec{s}^{(n)}))}{\sum_{n=1}^{N}(\ \Psi_i(\vec{s}^{(n)})\ \Psi_i(\vec{s}^{(n)})\ )} \quad . \tag{32}$$

These two formula for γ_i can be shown to be algebraically equivalent. However the latter is considered to be better as the Z_i's will have less variability than the data values $f^{(n)}$. Therefore using (32) we are less likely to incur numerical error by having a big term followed by small terms in the summation of equation (30).

2.3.3. Evaluation of $f(\vec{s}^{(n)})$

We now have a multinomial representation of the desired surface. Evaluation of a specific point on this surface requires the values of the independent variables, or $\vec{s}$, at the desired point. These values are used with the stored α_{ji}'s to find the value of the Ψ_i's at that point.

$$\begin{aligned} \Psi_1(\vec{s}^{(n)}) &= 1 \\ \Psi_j(\vec{s}^{(n)}) &= s_{k_j}\Psi_f - \sum_{i=1}^{j-1}\alpha_{ji}\Psi_i \quad . \end{aligned} \tag{33}$$

where

$$\alpha_{ji} = \frac{<s_{k_j}\Psi_f, \Psi_i>}{<\Psi_i, \Psi_i>} \quad . \tag{34}$$

The value of $f(\vec{s})$ is then found by taking the sum of the products of each $\{\gamma_m,\ \Psi_m(\vec{s})\}$ pair as in (32). The γ's are then used with the $\Psi_m(\vec{s})$ to find the value of $f(\vec{s})$ at a specific point according to (28).

2.3.4. Weighting of the $f^{(n)}$

In the general case it may be desirable to weight or emphasize the value, $f^{(n)}$, at a specific point, $s^{(n)}$, more heavily than at the other data points. If this is desired, the formula for γ_i becomes

$$\gamma_i = \frac{< \Psi_i, w \cdot f >}{< \Psi_i, \Psi_i >} = \frac{\sum_{n=1}^{N} (w_n \Psi_n(\vec{s}^{(n)}) f^{(n)})}{\sum_{n=1}^{N} (w_n \Psi_n^2(\vec{s}^{(n)}))} . \quad (35)$$

The local effect of setting $w_n > 1$ for a single point is that it tends to pull the curve or surface towards the specific point. Weighting, however, has a global as well as a local effect on the curve, and the entire surface will be altered if a single weight is adjusted.

2.4. Parametric Surface Fit

We can now define the fit of a parametric surface in three space. A parametric data point consists of a vector of spatial coordinate values and a vector of independent or parametric variables. The surface fit produces a set of three equations, one each for x, y and z. This leads to three different sets of γ_i for one set of Ψ_i's and α_{ji}'s.

The problem now is that data usually comes with three spatial values. A parametric curve is defined in three space by only a single parametric variable and a surface by two parametric variables. Parametric values need to be extracted from the data using some "projection" from three spatial coordinates into two or one parametric dimension. The determination of the best parametric coordinates can have a profound effect on the accuracy and stability of the final curve fit. The next chapter describes a method for the selection of parametric coordinates for a set of data points.

3. Automatic Parameterization of Three Dimensional Data

3.1. Introduction

Most of the standard curve or surface representations, such as those formed from B-splines, Beta-splines or Bernstein polynomials, are arranged so that the input points can be assumed to lie on regular intervals in parametric space. These representations require that data be arranged in a particular order and limit the position and number of data points used to form a given curve or patch. Experimental data points with these restrictions can not always be obtained or are, in the least, tedious to generate.

One of the interests pursued in this research was to free the user from these conventional restrictions. To achieve this, data points are taken in the form of their spatial coordinates, and parametric values that will yield a reasonable fit are automatically assigned. Information about the data can be obtained from a statistical analysis of the data distribution. This information can then be used in the parameterization of the points. If some *a priori* knowledge of what the data represents is available it can guide the choice of parameterization.

3.2. Projection of N Dimensional Data into K Parametric Dimensions

Parameterization of the data points can be viewed as the transformation of the points into a coordinate system that somehow relates to the orientation and distribution of the data. Such a coordinate system may be something as complicated as a curvilinear space or just an alternative Euclidean space. One approach to parameterization is to transform the data to a coordinate space that has been derived from the data so as to make the significant features of the data most apparent. Several methods exist for finding an alternative vector space. The method presented here uses the distribution of the data to find a vector space.

Once an alternative space has been selected a projection into K space can be accomplished by simply ignoring $N - K$ components of the coordinate basis. This is equivalent to projecting the data along the $N - K$ dimensional subspace down onto the K dimensional subspace. The components to be ignored are usually those which show the least variance, since it is hoped that these components hold the least information. Alternatively, any of the axes could be selected based on *a priori* knowledge.

Given a set of data points in N space,

$$\vec{x} = \left[\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(n)} \right] \quad , \tag{36}$$

we wish to find a coordinate space that will give us the least loss of information through the projection. The method presented here is taken from pattern recognition literature [Jernigan79], [Grasselli69], [Patrick72], [Funkunaga72]. This problem is typical in the study of pattern recognition and image processing. In these disciplines it is known as *feature extraction*. Feature extraction involves the selection of a reduced set of measurements for data from the raw data such that the most information can be held in the least number of measurements. The derived measurements are known as *features*.

3.2.1. The Alternate Basis

The derivation of the data transformation relates to the geometric distribution of the data. The data is viewed as a cloud with variances between the measurements, or features, and with each measurement having a probabilistic distribution. We can visualize axes starting at the mean point of the cloud and radiating out to the extremities in the general directions of maximum variance. If these axes are constructed such that they are mutually perpendicular they can be used as the basis vectors of a transformed space. Selecting the subspace formed by the K longest vectors as the dimensionally reduced space, we can project the points into this reduced space preserving the most information about the data. The following section presents the construction of an orthogonal basis based on the data distribution, and also the transformation required to collapse the data into the new space.

3.2.1.1. Construction of the Orthogonal Basis

Let the data points be represented as random vectors in N space. A data point then becomes

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ x_N \end{bmatrix} . \tag{37}$$

The mean or expected vector of n data points is

$$\begin{aligned} \vec{\mu} &= E\left[\vec{x}^{(n)}\right] \\ &= \int \vec{x}^{(n)} p(\vec{x}^{(n)}) d\vec{x} \quad , \end{aligned} \tag{38}$$

where $p(\vec{x})$ is the probability density function of the data measurements. In practice the probability density function of a set of data is unknown, so the sample mean, $\vec{m}$ is substituted,

$$\vec{m} = \frac{1}{N} \sum_{i=1}^{N} \vec{x}^{(i)} \quad . \tag{39}$$

Here N is the number of data points and $\vec{x}^{(i)}$ is the i^{th} data point. $\frac{1}{N}$ is substituted for the probability density function.

The variance of the data points can be found using

$$\Sigma = E\left[(\vec{x}^{(n)}-\vec{\mu})\,(\vec{x}^{(n)}-\vec{\mu})^T\right] \tag{40}$$

where $(\vec{x}^{(n)}-\vec{\mu})$ is the difference between the data vector and the mean vector and ^{T} is the transpose operator. Σ is known as the covariance matrix and is the multivariate analogue of variance. It is a $K \times K$ matrix, the ij^{th} entry of which is the variance of the x_i and x_j features.

$$\Sigma = \begin{bmatrix} \sigma_1\sigma_1 & \sigma_2\sigma_1 & \sigma_3\sigma_1 & . & \sigma_n\sigma_1 \\ \sigma_1\sigma_2 & \sigma_2\sigma_2 & \sigma_3\sigma_2 & & \\ \sigma_1\sigma_3 & \sigma_2\sigma_3 & \sigma_3\sigma_3 & . & . \\ & & & & \\ . & . & . & . & . \\ & & & & \\ \sigma_1\sigma_n & . & . & . & \sigma_n\sigma_n \end{bmatrix} . \tag{41}$$

The diagonal entries of the matrix are $E[(x_j-\mu_i)(x_j-\mu_i)] = \sigma_{ii} = \sigma_i^2$ or the feature variances. The covariance matrix is symmetric, ($\sigma_{ij} = \sigma_{ji}$ for all i and j), and therefore $\Sigma = \Sigma^T$. As with the mean, the sample covariance is usually substituted,

$$S = \frac{1}{N}\sum_{i=1}^{N}(\vec{x}^{(i)}-\vec{m})\,(\vec{x}^{(i)}-\vec{m})^T \quad . \tag{42}$$

Now we introduce the transformation that gives us uncorrelated features. This is equivalent to finding the principle vectors of the data distribution. If $\vec{v}$ and $\vec{w}$ are two random vectors whose features are uncorrelated then

$$E[\vec{v}^T\vec{w}] = E[\vec{v}^T]E[\vec{w}] \quad . \tag{43}$$

This is a necessary but not sufficient condition for the features being independent.

We want a linear transformation **A** such that

$$\vec{x} \xrightarrow{\mathbf{A}} \vec{t} \tag{44}$$

where the features of the transformed data points $\vec{\tau}$ are uncorrelated [Patrick72]. We want **A** to be linear in order to preserve the geometrical relationships between the data points. Therefore the transformation of the mean vector must be equivalent to the mean of the transformed points,

$$\mathbf{A}\, E[\vec{x}^{(n)}] = E[\mathbf{A}\, \vec{x}^{(n)}] \quad . \tag{45}$$

Such a transformation is equivalent to a translation of the mean point to the origin followed by some set of rotations about the mean point.

The transformation **A** is the transformation that diagonalizes the covariance matrix,

$$\Sigma \xrightarrow{\mathbf{A}} \Delta = \begin{bmatrix} \lambda_1 & 0 & 0 & . & . & 0 \\ 0 & \lambda_2 & . & . & . & . \\ 0 & . & \lambda_3 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & . & . & . & . & \lambda_K \end{bmatrix} \quad . \tag{46}$$

A is a linear transformation and therefore the covariance matrix for the transformed data set $\vec{\tau}$ is

$$\Sigma_{\vec{\tau}} = \mathbf{A}\Sigma\mathbf{A}^T \tag{47}$$

where $\Sigma_{\vec{\tau}}$ will be the diagonal matrix Δ. We know that Σ may be diagonalized with a matrix Φ such that,

$$\Phi^{-1}\Sigma\Phi = \Delta \quad \text{or,} \tag{48}$$

$$\Sigma\Phi = \Phi\Delta \quad . \tag{49}$$

Letting ϕ_i be the i^{th} column of Φ,

$$\Sigma\phi_i = \lambda_i\phi_i \quad . \tag{50}$$

This is the eigenvalue problem. The ϕ_i are the eigenvectors of Σ and the diagonal elements of Δ are the eigenvalues associated with the eigenvectors. Since Σ is symmetric ($\sigma_{ij} = \sigma_{ji}$) the eigenvectors are orthogonal,

$$\phi_i^T\phi_j = 0 \quad \text{for all} \quad i \neq j \quad . \tag{51}$$

If we normalize the eigenvectors to be of unit length

$$\phi_i^T \phi_j = 1 \quad \text{for all } i = j \tag{52}$$

and therefore

$$\Phi^T \Phi = I \rightarrow \Phi^T = \Phi^{-1} \quad . \tag{53}$$

Thus Φ is an orthogonal coordinate transformation and therefore preserves the geometric relationships between points, i.e. Euclidean distance is preserved. Since $\Phi^T = \Phi^{-1}$ equation (48) becomes

$$\Phi^T \Sigma \Phi = \Delta \tag{54}$$

and A the transformation we have been looking for is

$$A = \Phi^T \quad . \tag{55}$$

The rows of A are the eigenvectors of Σ, the covariance matrix. Applying A to the data points $\vec{s}^{(n)}$ transforms them into a new coordinate space whose axes are the eigenvectors. These axes are oriented along the principle axes of the data distribution. The new feature variances give a measure of the spread or dispersion of the data along the new axis.

We have now transformed the data into a coordinate system whose features are uncorrelated, and therefore a projection of the data down one of the axes from N to K dimensions will preserve the maximum information about the data. The projection is done by simply ignoring $N - K$ coordinates of the feature space.

For the purposes of curve and surface representation in computer graphics, N will usually be 2 or 3 and K will be 1 or 2. The code implemented in connection with this research is for general N and K.

3.2.1.1.1. Finding Eigenvectors and Eigenvalues

The method given in the previous section relies heavily on finding the eigenvalues and eigenvectors of a square matrix. Perhaps the best method of doing this on the computer is

some variation of the QR algorithm. In the case of a symmetric matrix, as is the case with the covariance matrix, convergence is fast. The routines used in this research to diagonalize the covariance matrix and to find the eigenvectors were converted to C, from the Fortran versions in the EISPACK math library [Smith76]. Specifically, TRED2 and TQL2 were used.

3.2.2. Parameterization of the Data

In the previous section we found a transformation A that maps the data points into a coordinate system in which the features or measurements that represent the data are uncorrelated. The points are then parameterized in the new coordinate space. The Figures 2, 3 and 4 illustrate this process.

Figure 2 shows a set of points describing a surface in three space. The spatial (xyz) basis is shown as well as the principle vectors of the data distribution. Each point $\vec{x}$ is transformed by the matrix A giving the new points $\vec{t}$,

$$\vec{t} = A\vec{x} \quad .$$

This has the effect of rotating the points about the mean point of the distribution so that the principle vectors are oriented with the spatial basis, as seen in Figure 3.

The simplest parameterization is accomplished by translating the points such that the mean point lies on the origin (Figure 4). The points can be projected down the vertical axis to give their images in the base plane. The coordinates of a points image are then assigned to the points as their parametric coordinates.

The points are sorted for maximum and minimum values in parametric image space. Using these values the point features are normalized. The parametric values of the points now lie within a unit square. The subsequent fit will produce a patch which is valid over a unit square in the parametric space, where bu valid we mean that evaluation of the fitted

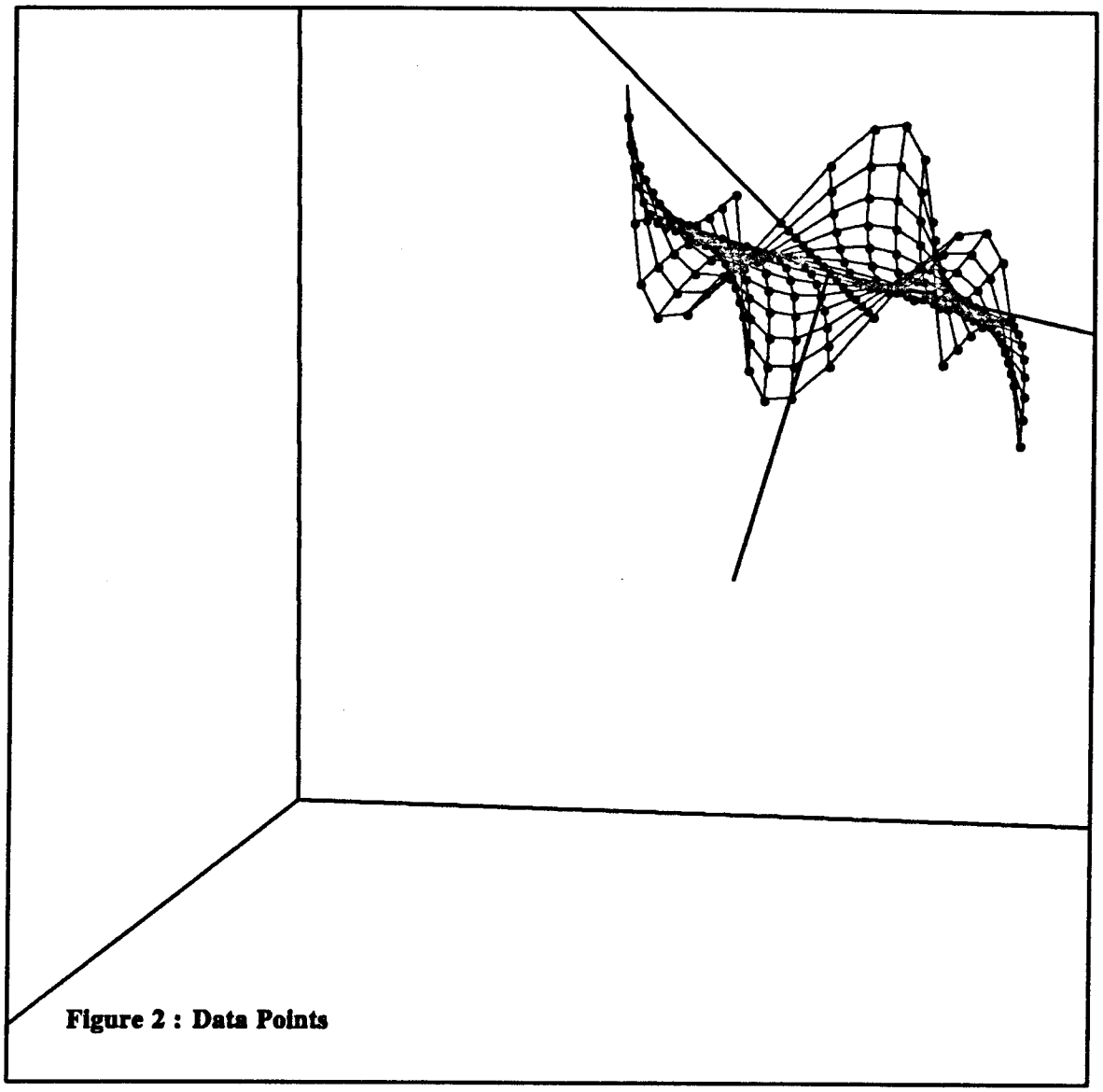


Figure 2 : Data Points

function outside the unit square represents extrapolation rather than interpolation.

3.2.2.1. Alternative Coordinate Systems

The simple approach of projecting the data onto a unit square has some shortcomings. The fit will only give meaningful results if the data is not much different than a slightly distorted sheet. If, for example, the data represents some sort of axi-symmetric distribution the fit will be quite poor in the corners of the unit square and will vary with the angle of rotation around the axis of symmetry.

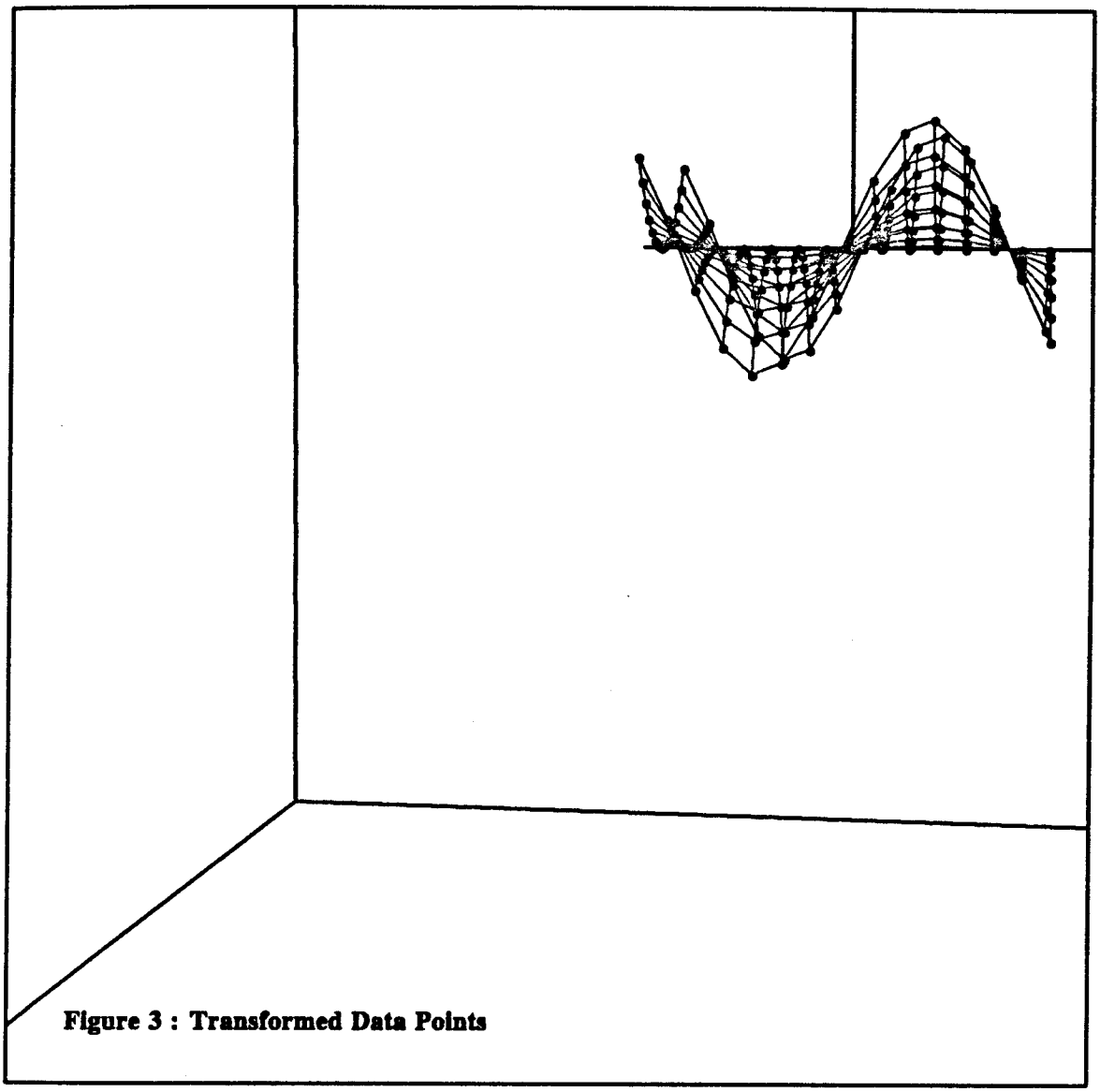


Figure 3 : Transformed Data Points

An alternative parameterization is possible using *a priori* knowledge of the point distribution. In the context of surface representation in three dimensional space, suppose that the points represent the surface of a known object. For example, the fit for the surface of a vase, or a tube, would be much better if the parametric space were derived from a cylindrical coordinate system. Alternatively, if the surface were a closed object such as a ball it would be advantageous to use a spherical coordinate system for the parametric space.

Figure 5 shows data points representing a vase, after the coordinate transformation. If parameterization of the data points is done by a simple projection into the base plane, no

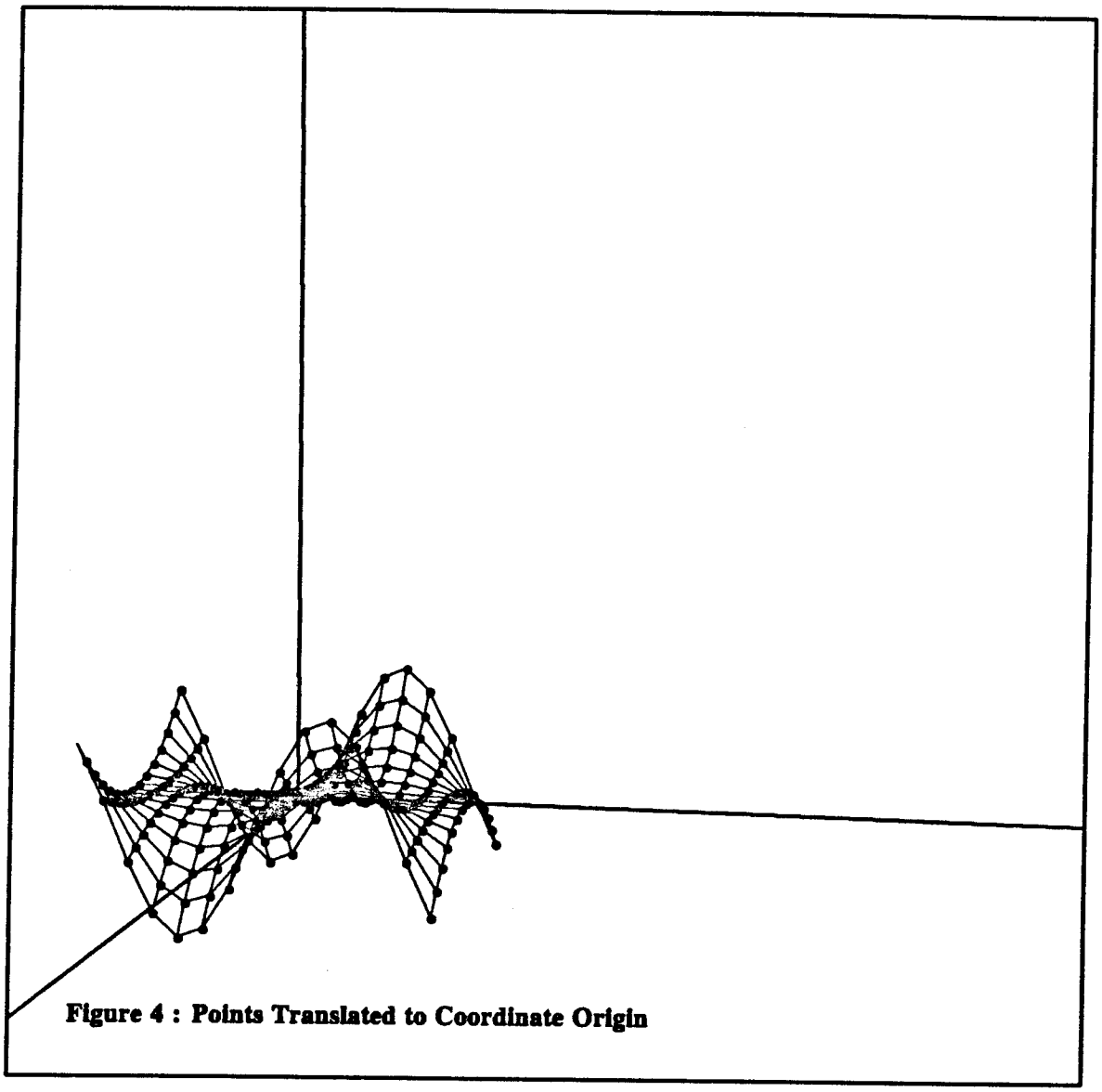


Figure 4 : Points Translated to Coordinate Origin

information about the profile of the vase will be preserved. However, if the points are parameterized by a projection on a unit cylinder the fit will give a much better representation. The major axis of the cylinder is chosen to lie on one of the axes of the transformed space, and the parametric values are chosen as the angle and the height out of the base plane. Normalization of the periodic coordinate is from zero to 2π. Figure 6 shows the resultant curve fit to the vase data points.

In general, the ability to fit more complex surfaces is directly related to the amount of distinct information that can be conveyed in the parametric coordinate system. If complex

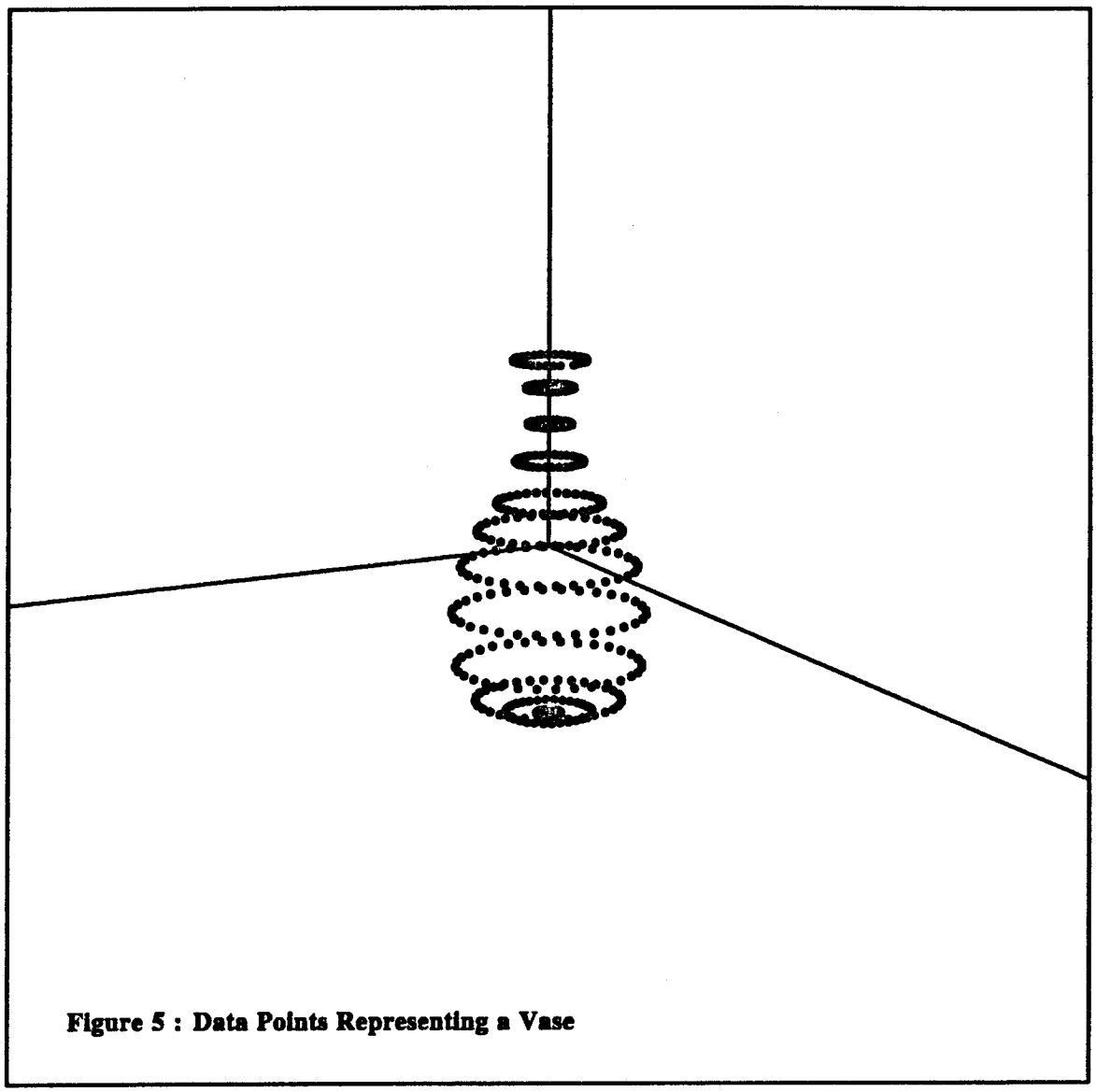


Figure 5 : Data Points Representing a Vase

arrangements of points can be mapped into simple parametric coordinate systems, they can be fit relatively well with simple multinomial surfaces.

For example, data points representing an axi-symmetric single valued function may be fit much better when the data points are parameterized in polar coordinates. Figure 7 shows 900 data points representing the function $z = \frac{\cos(r)}{(r + 1)}$, $r = 0.0$ to 1.0. Figure 8 shows the sixth order fit to the data when it has been parameterized in rectangular coordinates. Figure 9 shows a sixth order fit to the same data. However, the points have been parameterized in polar coordinates and projected onto the unit disk.

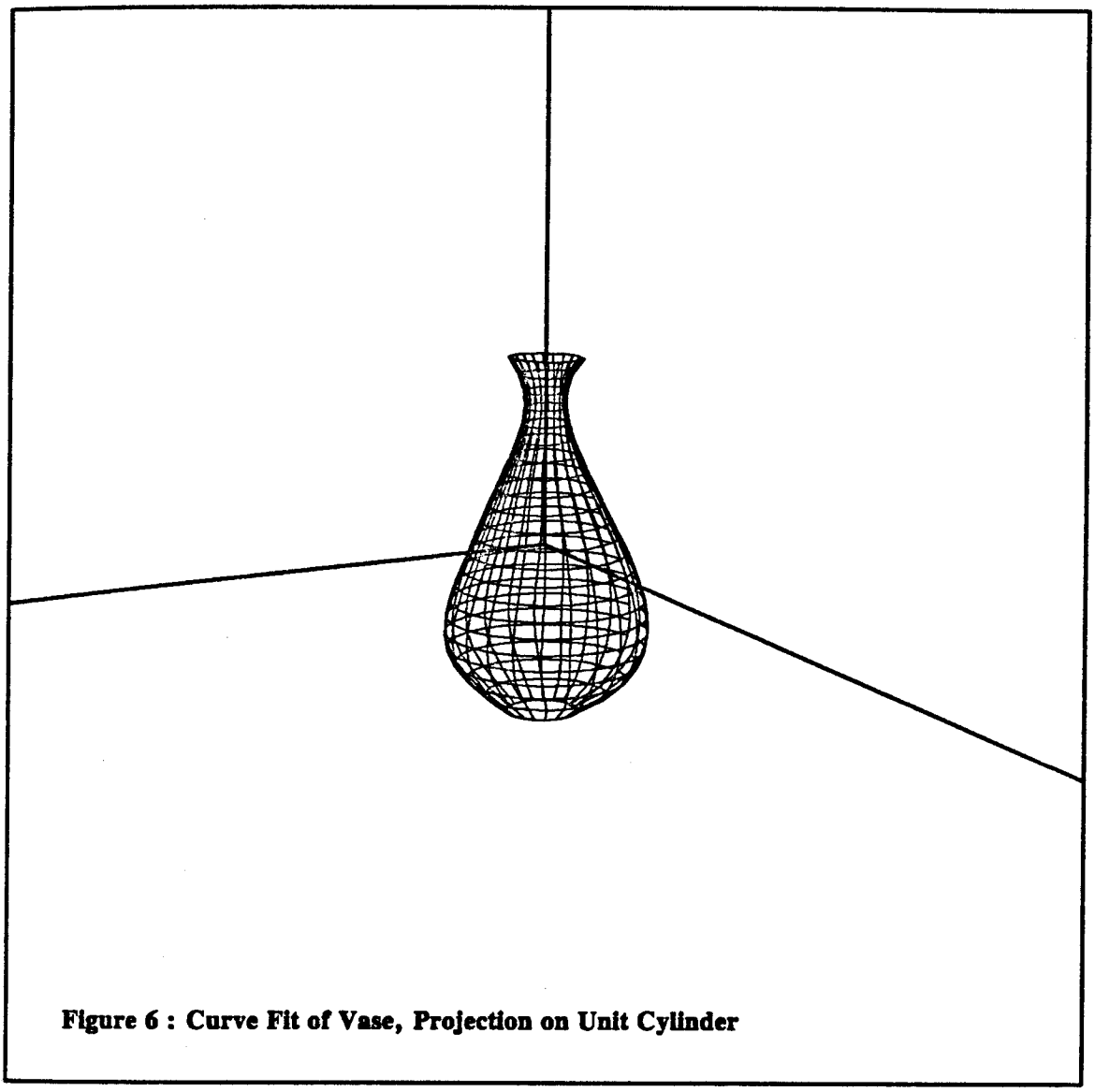


Figure 6 : Curve Fit of Vase, Projection on Unit Cylinder

The transformations used in parameterizing the data need not be based on the mean value and principle vectors of the data points. A parameterization of points describing a hemisphere, for instance, could be parameterized in spherical coordinates using a vector space describing the base of the hemisphere and a point lying on the base plane at the center of the point distribution as the mean point. Renormalizing the parametric coordinates and fitting a surface would then give a patch of roughly hemispherical shape.

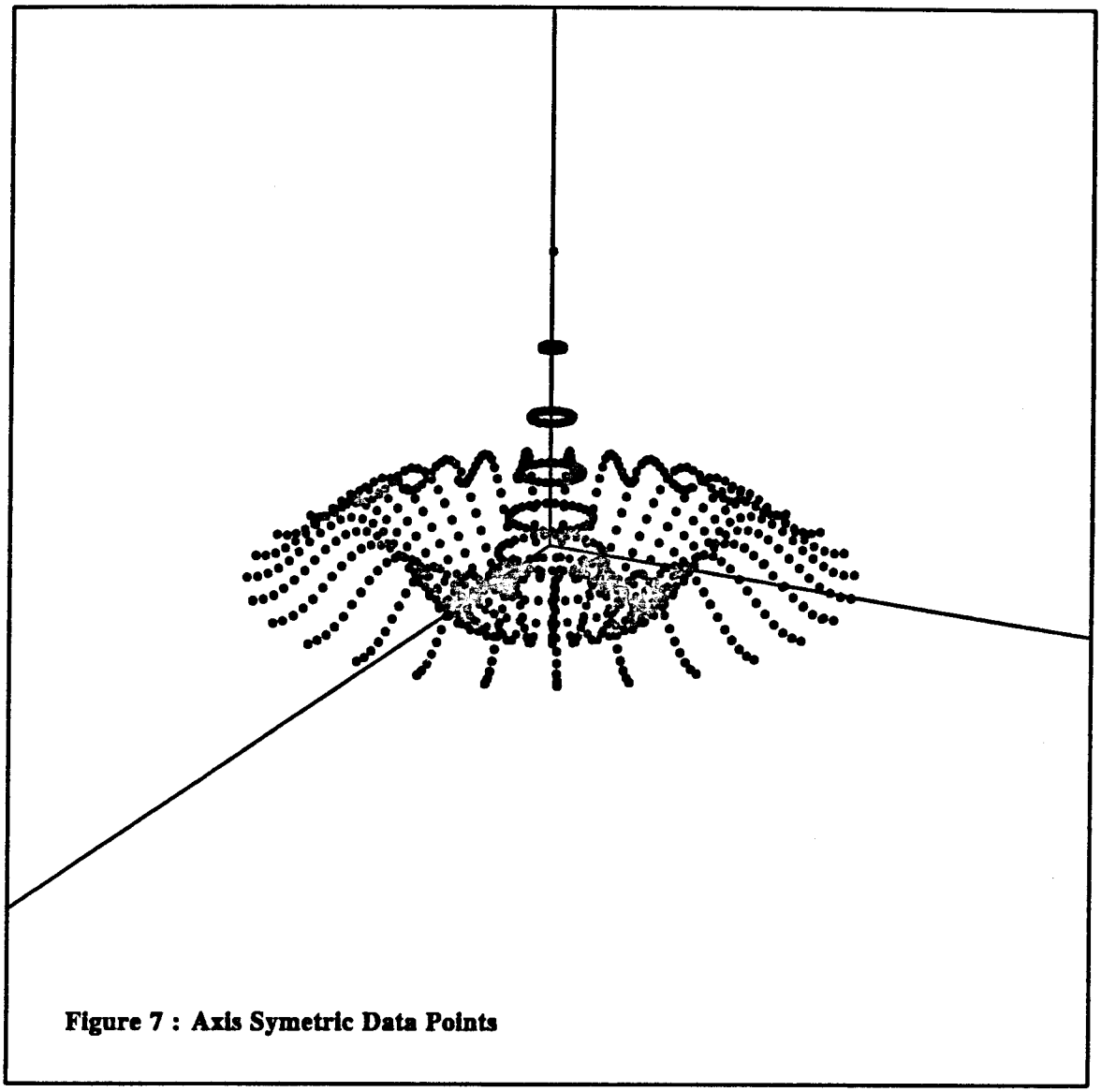


Figure 7 : Axis Symetric Data Points

3.2.2.2. Periodic End Conditions

Periodic coordinate systems allow the fitting of surfaces to shapes that require multiple-valued functions in rectangular coordinate systems. However, using multinomials, problems do occur in matching the edges of the parametric patch. Some method is needed to provide end conditions for the fit.

One way is to provide additional points that constrain the fit at the edge locations. The required values can be provided by using the points that are located on the other side of the periodic boundary. Points that have parametric values close to one edge can be added to the

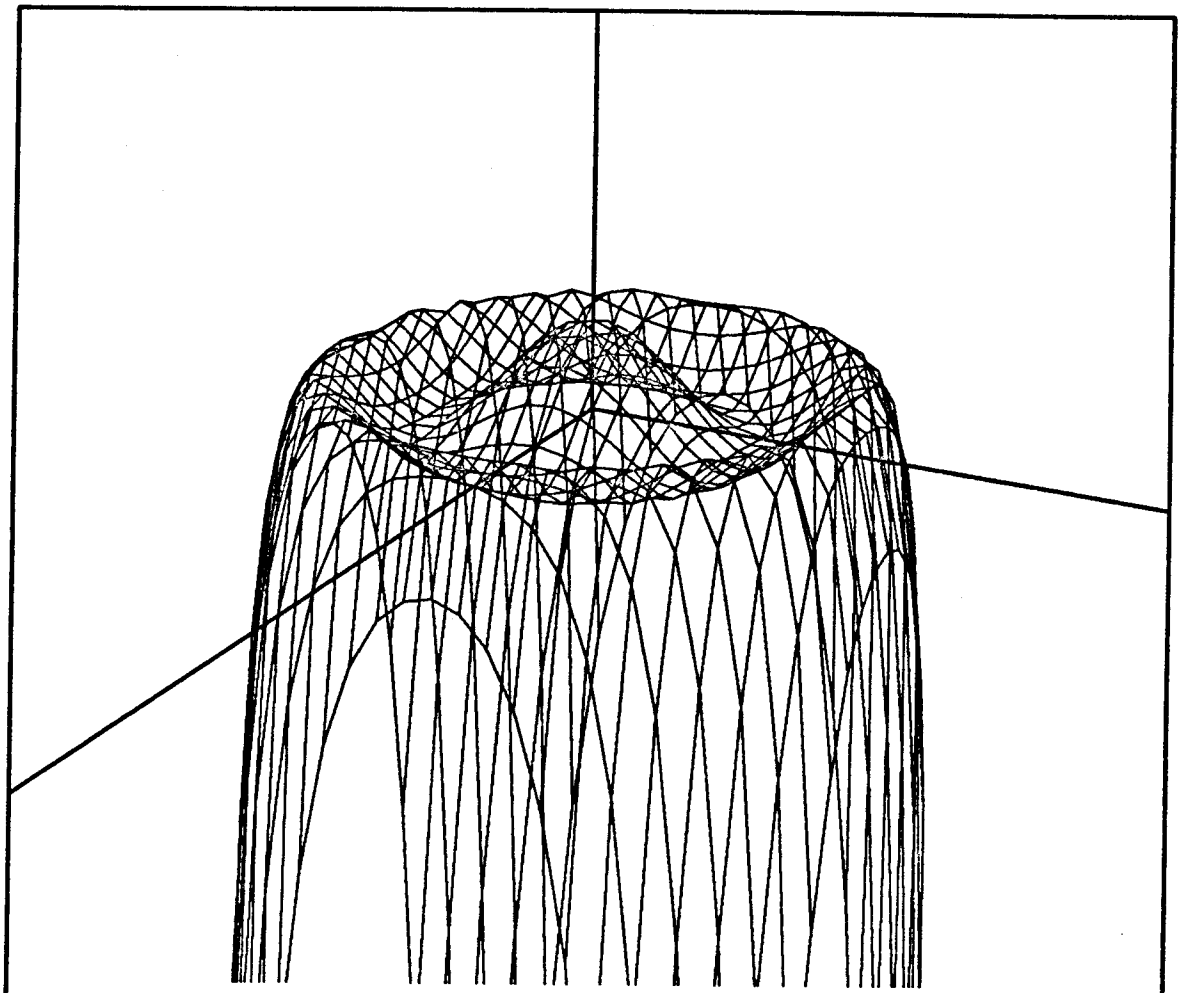


Figure 8 : Sixth Order Fit Parameterized on Unit Square

other edge with their parametric values suitably incremented. This draws the boundaries closer together, but does not join them.

Figure 10 and 11 illustrate this method in cylindrical coordinates. Figure 10 shows the image of a set of data points in parametric space. In this case u is taken as the periodic variable (angle). The parametric variable v is the vertical or z component in the cylindrical coordinate system. The ends of the u coordinate actually meet in Cartesian space. A limit, set empirically, can be used to decide which points from Cartesian space have duplicate images on the parametric plane. The resulting second points have images as in Figure 11.

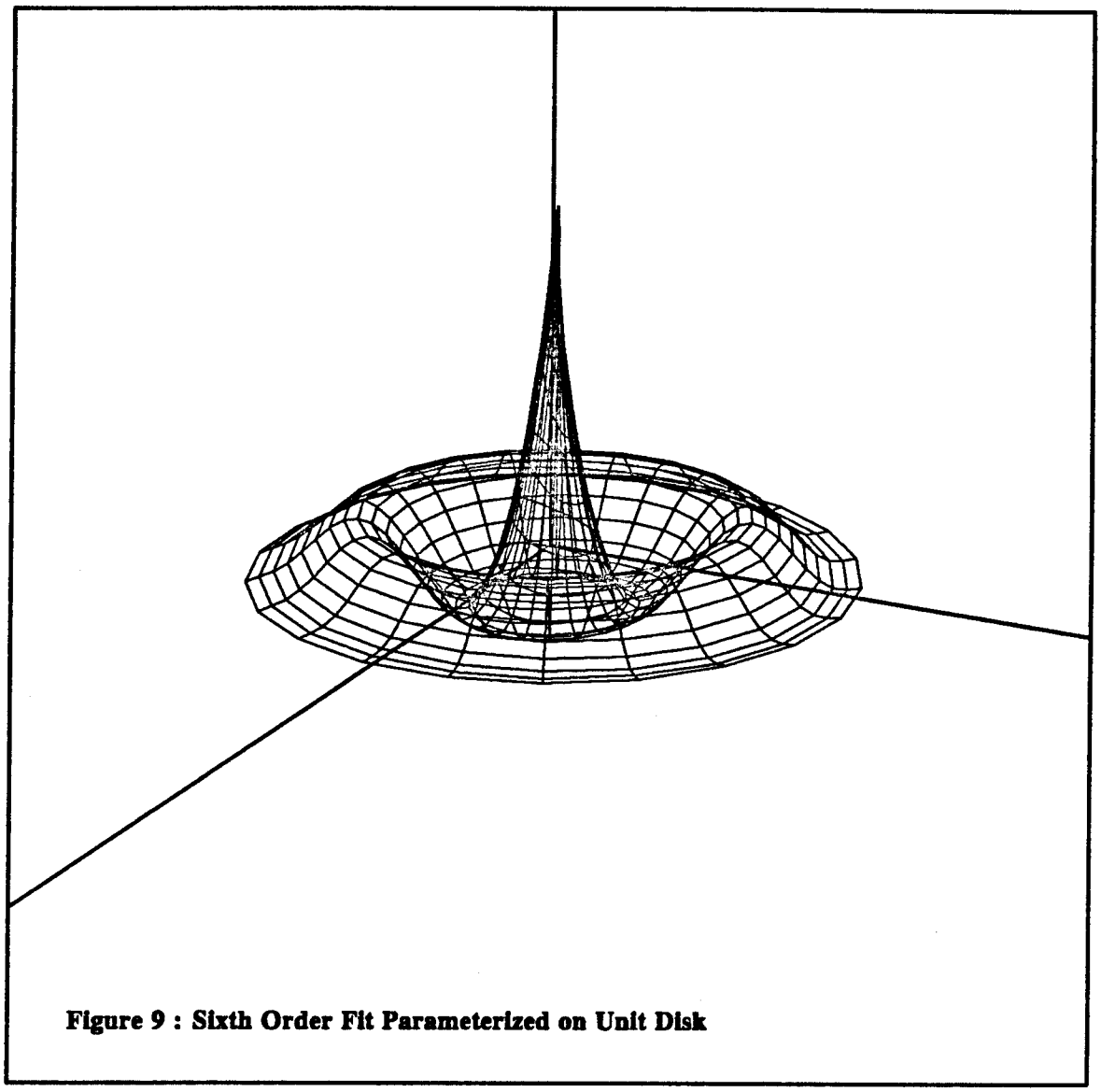


Figure 9 : Sixth Order Fit Parameterized on Unit Disk

The effect of adding end conditions can be significant. Figures 12 and 13 show ninth order curve fits of the vase data points with no overlap and with 40 percent overlap at each edge.

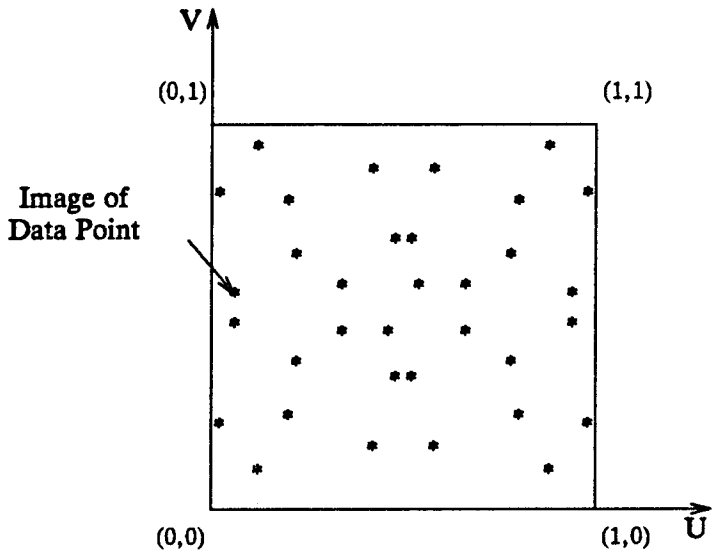


Figure 10 : Data Points on the UV Plane

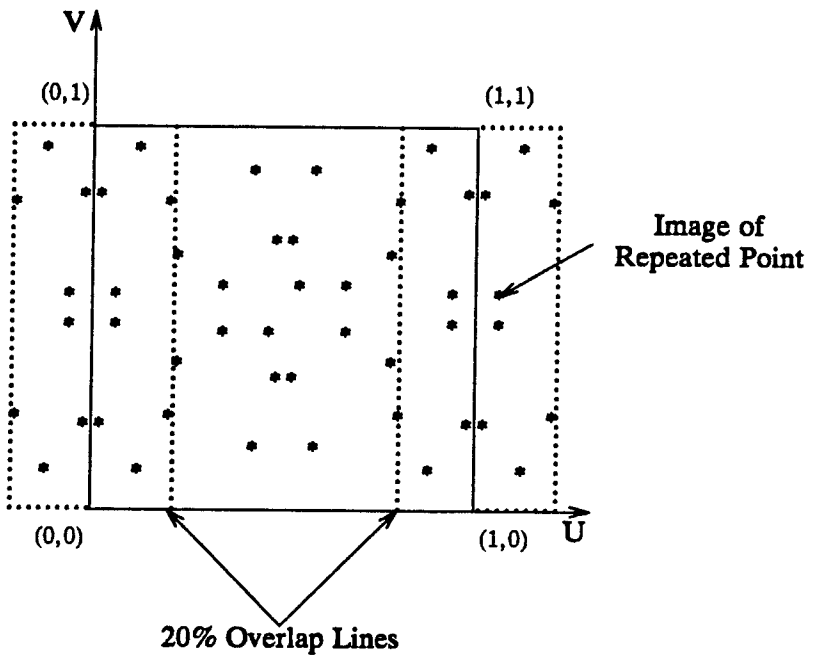


Figure 11 : Additional Constraining Points.

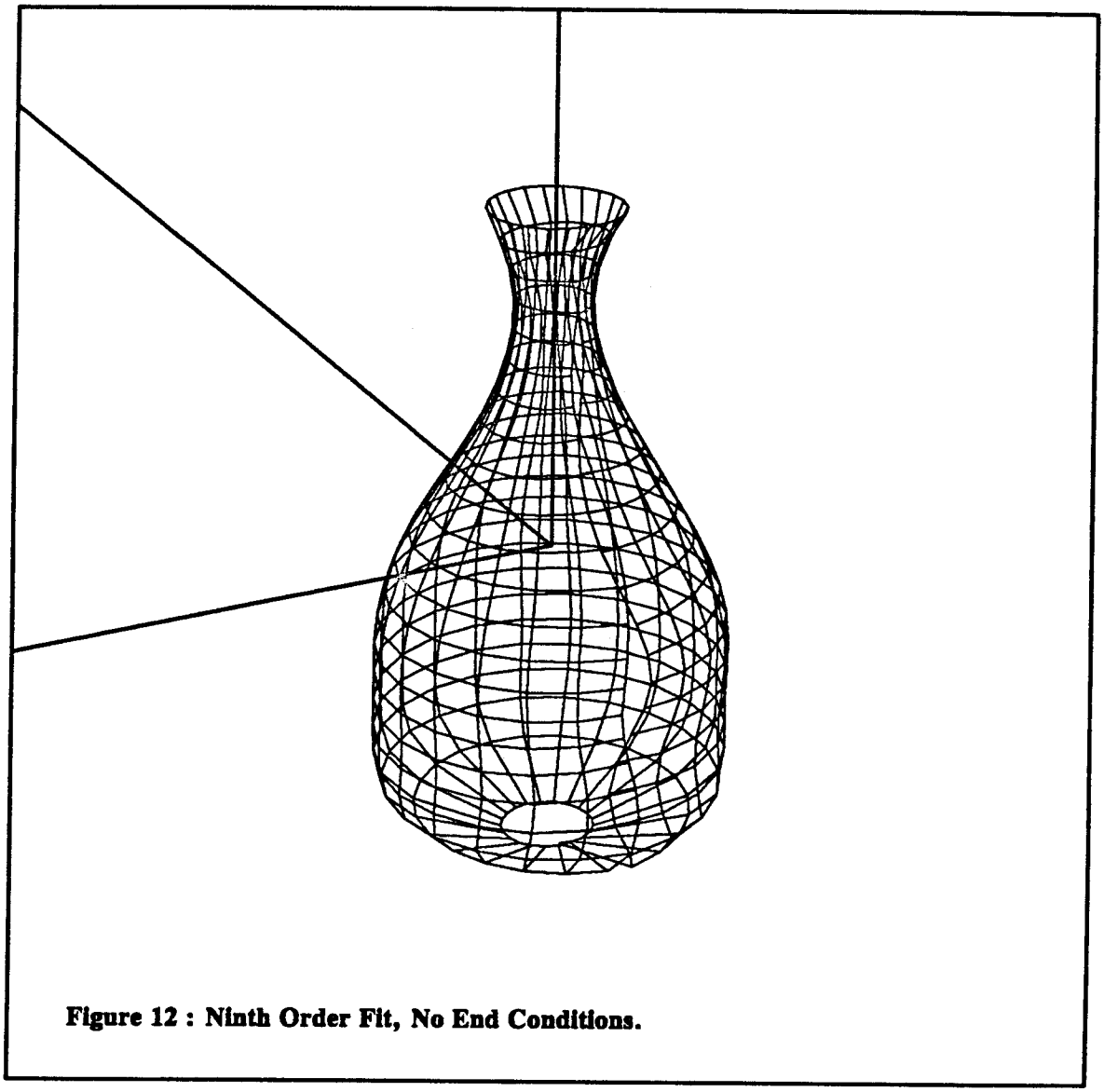


Figure 12 : Ninth Order Fit, No End Conditions.

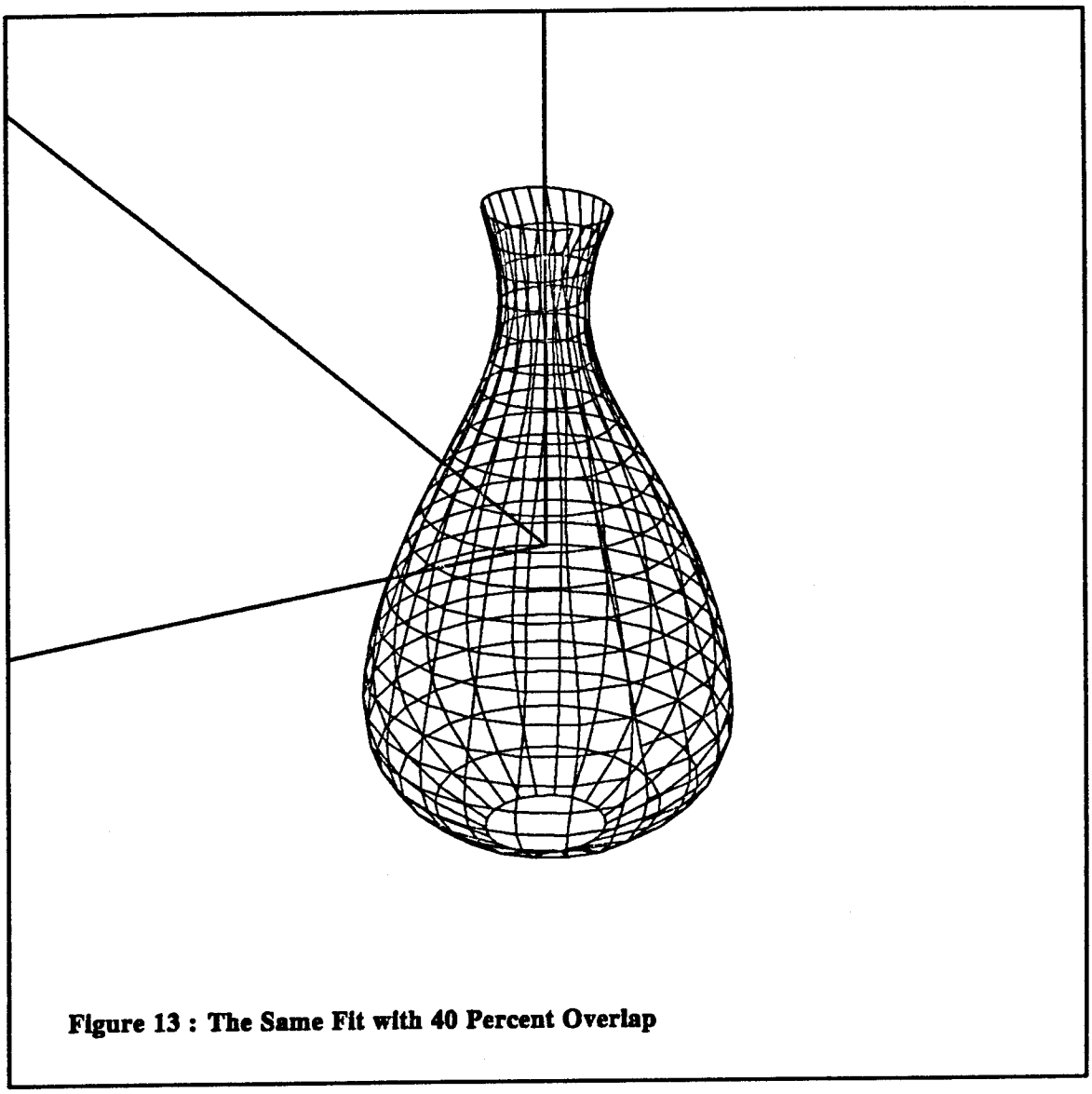


Figure 13 : The Same Fit with 40 Percent Overlap

4. Surface Rendering and Subdivision

4.1. Introduction

The main objective of this research was to develop tools that could be used to investigate parametric surface fits with orthogonal multinomials in computer graphics. An obvious use for these surfaces is simple object modeling. Rendering routines for the display of these parametric surfaces as objects were needed. The rendering method chosen needed to give reasonably good images of the surfaces.

The initial method for rendering parametric surfaces consisted of dividing the surface into discrete polygonal facets of a fixed size. These facets were drawn using standard polygonal drawing algorithms. The resulting images, however, still have visibly piecewise linear edges and corners on silhouettes. Better methods for rendering parametric surfaces are available. However, these methods are computationally expensive when compared to polygonal approximation.

In 1975, Catmull [Catmull74] introduced a method that gave results superior to standard polygonal division. His idea was to subdivide the surface until the drawing of a polygonal facet would only effect a single pixel (on raster devices). This effectively meant that any shortcomings associated with drawing polygons were submerged in the resolution of the display device.

Two other rendering methodologies have been developed, namely ray tracing and scan line algorithms. Ray tracing algorithms are based on the idea of firing a ray from the eye point, through the image of a pixel and intersecting it with the objects of the scene. A set of rays emanates from all intersection points and must be intersected with each object in the scene [Kajiya82]. Ray tracing gives at least as good and usually better results than subdivision. This is due in part to the sophisticated light and shadow calculations that are possible with ray tracing.

Scan line algorithms carry out object rendering in scan line order. Scan line algorithms for non-piecewise linear surfaces were presented by Lane, Carpenter, Whitted, and Blinn [Lane80] and later by Schweitzer and Cobb [Schweitzer82]. These algorithms really represent methods for ordering the rendering of the scene from top to bottom, side to side. Parametric patches can be rendered using a method of subdivision that reduces the patch in scan line order.

4.1.1. Rendering Methods: Advantages and Disadvantages

4.1.1.1. Catmull Subdivision

Catmull's algorithm gives reasonably good images. However, it requires a large amount of computation. For example, in a 512×512 raster display, the worst case is given by a patch which covers all of the pixels in the display. Nine levels of binary subdivision, dividing each patch in half, ($2^9 = 512$) are required to display the patch; or in the case of quad subdivision, dividing each patch into 4 subpatches, five levels of subdivision ($4^5 > 512$). In all, a quarter of a million polygons are rendered.

Hidden surface elimination techniques such as Z-buffer algorithms are expensive when combined with subdivision, since a depth buffer of at least the device resolution must be stored throughout the drawing of the surface. This is partially alleviated if a hardware Z-buffer is available.

4.1.1.2. Ray Tracing

The ray tracing of arbitrary parametric patches is also costly. It involves the intersection of rays with all the objects in the scene, whether they are parametric patches or polygons. This is equivalent to finding the roots of the parametric functions which define the surfaces and is, as a result, very time intensive.

Antialiasing is a problem with ray tracing algorithms as each ray samples the scene in the center of each pixel and there is no information available as to what portion of the pixel is actually the sampled colour. Possible solutions for this are to send more rays when an edge is detected, or to take some sort of an area sample rather than a point sample. Ray tracing has the advantage that depth buffering is not necessary, since only the visible parts of the scene are ever drawn.

4.1.1.3. Scan Line Algorithms

Parametric patches can be rendered using subdivision in scan line order. This is done by controlling the order of subpatch division such that the subpatch closest to the current scan line is divided next. One modification to subdivision often seen in implementations of this type is premature termination of subdivision when the current subpatch is "flat enough". A polygon covering the area of the subpatch is then drawn. The criterion for flatness depends on the desired appearance of the final image. Once a patch has been converted to a polygon it is displayed in scan line order. This has the advantage of not requiring the same depth of patch division as Catmull subdivision. However, it has all the problems associated with polygonal representations of surfaces.

An additional problem results from the subdivision of neighbouring subpatches to different levels: in this case spaces open between patches, leaving glitches in the scene. Work on this problem is reported by Clark [Clark79].

The big advantage of scan line algorithms is that the Z-buffer is only required for one scan line.

4.1.1.4. Conclusions

Since we wished to implement routines that would give reasonably good renderings of multinomial surfaces, it was decided to implement a Catmull subdivision rendering algorithm. A ray tracing algorithm was discarded because of ray tracing's high computational cost. Scan line algorithms, although easier to implement than ray tracing, also have the additional overhead required to sort the components of the scene: scenes with multiple objects need to have all objects rendered simultaneously in y priority order. The GR graphics package used to support this research does not implement such a bucket sort of image primatives. Since the subdivision software is intended to be an addition to the package, it can not rely on such information being available.

As a result of these considerations a space efficient and surface representation-independent algorithm for subdivision has been developed. The following section presents this algorithm.

4.2. Subdivision

The essence of subdivision is that a patch is divided into subpatches until some termination criterion is met. A patch is some section of a parametric surface with parametric variables u and v sweeping out some region of the surface. As the subdivision proceeds, new values for the points at the vertices of the subpatches need to be calculated. Upon termination each subpatch is rendered as a small polygon.

Previous implementations of subdivision have used a technique of dividing the current patch into subpatches, pushing all but one subpatch onto a stack, "solving" the current subpatch and then popping the top patch from the stack to become the new current patch. This was an artifact of the language that was being used as much as anything else (Catmull's original program was written in Fortran).

By nature subdivision is recursive. Recursive solutions for problems tend to be more elegant than iterative solutions. The algorithm for subdivision using stacks is nothing but a simulation of recursion.

The issues of implementation associated with subdivision are to organize the information retained from subpatch to subpatch and to minimize the information passed down through the layers of the recursion. When a subpatch has been rendered, the points internal to that patch are no longer required; however, points on the outside edges may still be needed. A data structure for subdivision must retain the points that are still needed but allow for the release of points that are no longer required. At the same time, lower levels in the recursion should be passed only the parts of the data structure that are required to render the subpatch.

4.2.1. Data Retention

Since subdivision is recursively structured, with each level being a subimage of the overall problem, hierarchical data structures based on trees are the simplest method for storing the information required. As subdivision proceeds the tree is grown, each level storing the information generated by a different level of subdivision. As subdivision is a two dimensional process, information is arranged hierarchically in two directions.

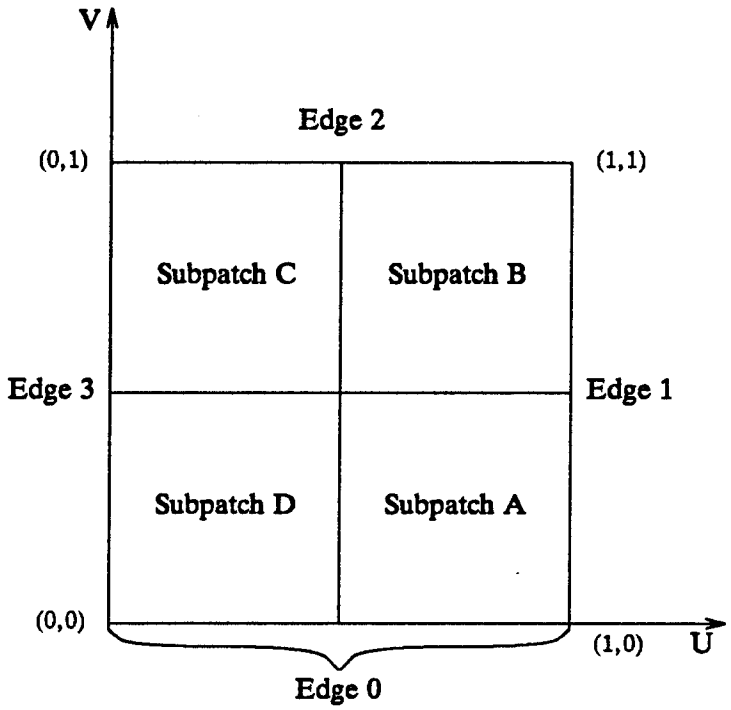


Figure 14 : Patch with four Subpatches.

Octrees and quadtrees are two data structures that are used to store multidimensional information [Clark76, Dyer80a, Dyer80b, Meagher82]. Both have been used to store information about scene structuring.

The subdivision process is slightly different from those in which a quadtree structure would be applicable. The data that must be saved in subdivision is associated with the edges of the subpatches and not with the bodies of the patches. A subpatch shares each one of its edges with a different neighbour. Figure 14 shows a patch with its 4 immediate subpatches. In the following discussion the present level patch is referred to as the patch while its

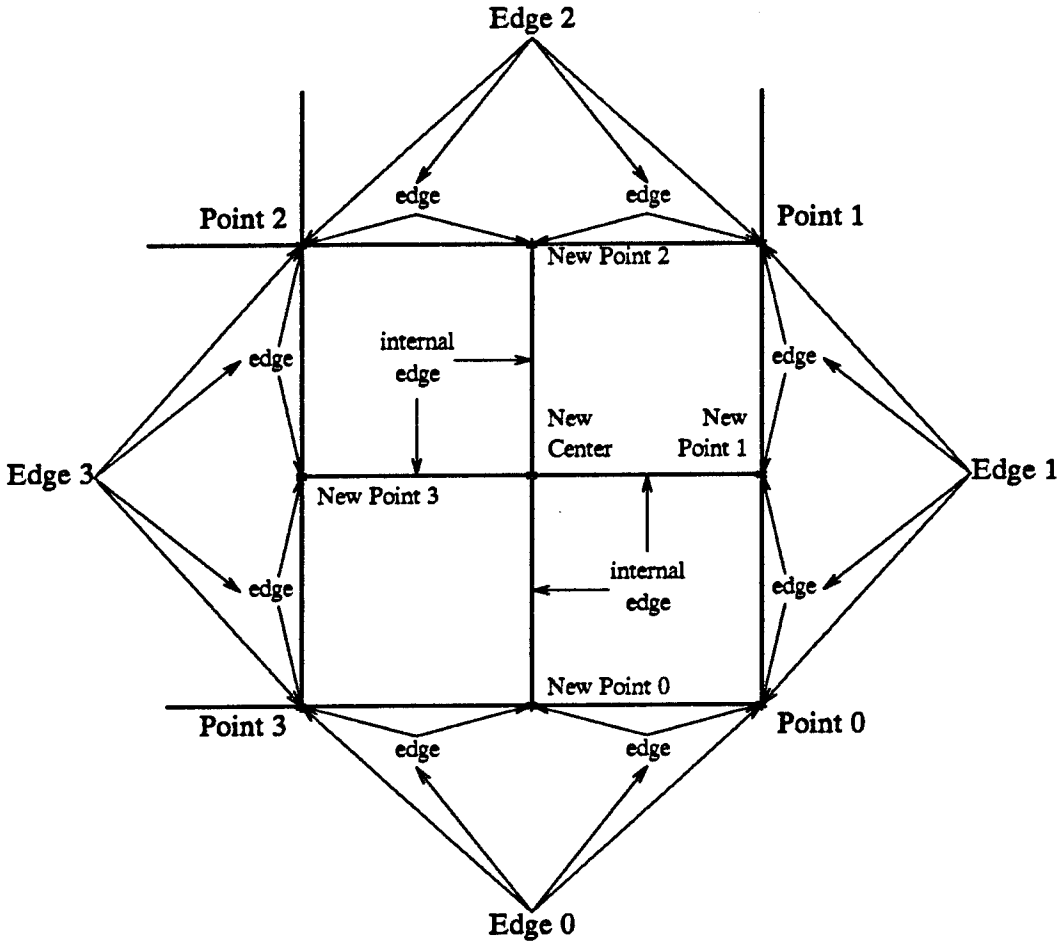


Figure 15 : Growth of Data Structure.

children are subpatches. In reality all but the uppermost patch is the subpatch of a patch.

As a subpatch is subdivided the edges are split in half, producing two new edges. These edges are the descendants of the first edge. The data structure used is associated with the edges of each subpatch. The edge data structure consists of pointers to the points of either end of the edge and 2 pointers for the edge descendants. Figure 15 demonstrates subdivision of subpatch A of Figure 14. As the patch is divided, twelve new edges are generated, four internal and eight external. The four internal edges are shown in Figure 15.

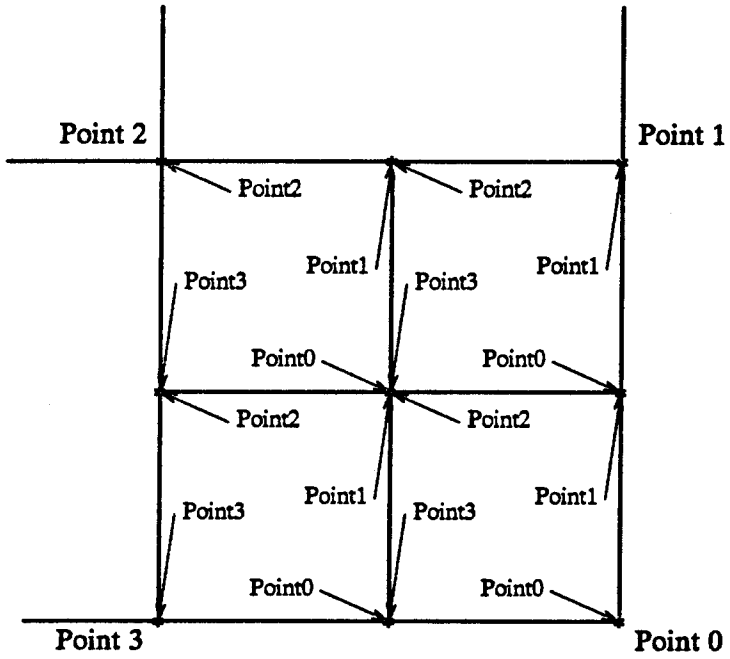


Figure 16 : Ordering of Subpatch Points for Termination.

The eight external edges are the children of the patches' external edges. The recursion proceeds on each subpatch. Each patch is passed its own external edges as parameters (edges 0 to 3 in Figure 15). During the evaluation of a subpatch new edges are built in the edge tree. As each subpatch is finished, control is returned to the parent patch and the next subpatch is processed. Before the patch is finished all points and edges associated with its internal area are deleted, as no subsequent patch will require the values internal to the present patch. Edges that have been created on the current patch's outside edges are saved.

Each successive subpatch can take advantage of the fact that its sibling has already computed at least one shared edge.

Each edge has two points associated with it, and also a direction. This means that an edge shares an end point with a neighbouring edge. An edge is oriented either in the direction of increasing u or of increasing v; the other parameter has a constant value. The point at one end of an edge has a greater value of the varying parameter than the other. As the recursion proceeds the termination test, as well as the creation of new edges, requires the end point values in both u and v as well as the spatial values x, y and z. In order to insure that four distinct end points are used, the edges of each subpatch are ordered in the way they are passed down into the recursion. The ordering is demonstrated in Figure 16. In the figure, subpatch A has four external edges and four external points. Edge 0 and edge 1 are both associated with point 0. The values at the four corners of a subpatch are used in the termination test. Each edge is associated with the point counter clockwise from the center of the edge, in order to assure uniqueness. In addition, the point at the center of the subpatch may be useful in the termination test as well as in the rendering of the object. The subdivision of each edge uses the values of the end points to determine the new values needed for the subpatch's subpatches.

4.2.2. Rendering

Termination of subdivision is based on two criteria: size and clipping. As a patch is divided it is checked against these two criteria. If the patch is entirely outside the field of view it is discarded as being invisible. If it is partially within the field of view or entirely within it is checked for size. Once a patch is "small enough" it is rendered and the process repeats on the next subpatch.

Catmull in his subdivision scheme approximates a patch as a polygon defined by the corner points of the patch. Termination of subdivision based on a size criterion occurs when

the projection of this polygon covers at most one sample point. Sample points are the points at the center of each raster element. If a patch covers no sample point Catmull assigns the patch to the nearest sample point [Catmull74] , page 12. This is not strictly a size test, but is rather a test of coverage.

Once subdivision is terminated on a patch, the polygon that represents that patch needs to be assigned a colour and written into the frame buffer. The actual colour of the polygon depends on the colour of the surface as well as the colour of the lights in the scene. A variety of lighting rules are possible; all are based on the position of the eye, the position of the lights and the normal vector of the polygon.

5. Implementation of 3-D Multinomial Fitting

5.1. Introduction

The ideas presented so far have been implemented as three different sets of routines. A program using these three sets for the fitting and subsequent display of three dimensional surfaces has also been written. Emphasis, however, has been put on developing tools for further study rather than on creating an object modeling program with a friendly user interface.

The first set of routines performs the analysis of raw data points and forms parameterized data points from them. The second set includes the least squares fitting routines. The third is a set of routines for the representation independent rendering of surfaces using a subdivision algorithm. All three sets are detailed in the next sections.

5.1.1. The GR Graphics Package and UNIX

The software presented in this chapter was written in C on a VAX-11/780 running Berkley UNIX† 4.1. The GR graphics package used by the routines was developed at the

† UNIX is a Trademark of Bell Laboratories.

University of Waterloo. This graphics package grew out of a set of assignments used for a fourth year graphics course at the university. These assignments were patterned after assignments used at the University of Utah by Jim Blinn. The package is modeled losely after the CORE graphics standard. The GR package used for research is written in C and is an expanded version of the GR Pascal teaching package.

The emphasis in the design of the GR graphics package was on modularity and device independence. A variety of output devices are accessible from the package including both vector and raster displays. The software itself is organized in levels, each level dealing with a specific part of the classical display pipeline. The bottom level contains all the device dependent software while the fourth, or top level, provides an interface to programs written using the package. User programs form the fifth level of the hierarchy.

Further information on the GR graphics package can be found in a thesis by Sylvia Lea [Lea83].

5.1.2. Hardware used in this Research

The CGL (Computer Graphics Lab.) VAX-11/780 at the University of Waterloo was the computer used in this research. In its current configuration it has 8 Mbytes of interleaved memory, a floating point accelerator and 580 Mbytes of disk storage. The major output device connected to the VAX is an Ikonas RDS 3000 frame buffer system. The Ikonas is configured as a 512 × 512 by 32 bit frame buffer. Normally the frame buffer is organized so that 8 bits are used each of red, green and blue, with 8 bits remaining for overlay information. This is software modifiable via various control registers and a crossbar switch. In this research the frame buffer was used to store Z-buffer information in 15 bits of each frame buffer word, the remaining bits being used for colour information.

Communication between the VAX and the Ikonas is by DMA to the frame buffer for complete scanline writes, or by programmed I/O writes for single word access. When the

frame buffer is used as a Z-buffer access is usually by single words.

The memory image in the Ikonas is read from the frame buffer by the frame buffer controller and passed through the crossbar switch to three 8 bit colour look up tables (one for each primary colour). Using the crossbar switch, bits of the frame buffer word may be selected and permuted. Output from the crossbar switch is passed as three bytes to the colour lookup tables. Output from the colour look up tables goes to the input of the digital to analog converters that produce the RGB video signals. The final display is on an Aydin colour monitor.

5.2. Parameterization

Parameterization of raw data points requires some *a priori* knowledge of the class of shape the points represent. If the desired shape of the final representation and the distribution of the data points are taken into consideration significant simplification and improvement can be found in the resulting curve fits. This is true whether the surfaces fit to the data are least square surfaces, spline representations or any other type of surface representation.

The routines described in this section attempt to provide tools for the optimal parameterization of the data points representing a diversified set of surface types.

5.2.1. Analyses of Data Distribution

Chapter 3 discussed a method of analyzing how data is distributed in space by finding the directions and relative lengths of the principle vectors of the data distribution. Eigenvectors and eigenvalues of the covariance matrix correspond to the lengths and directions of these principle axes.

The routine *eigen*( ) takes a set of data points as one of its arguments and returns the

mean vector of the data as well as the eigenvalues and vectors in matrix form. The K dimensional covariance matrix of the data points is calculated from formula (42) in 3.2.1.1. The routines for finding the eigenvectors and eigenvalues of the matrix were translated to C from the Fortran routines TRED2 and TQL2 in the EISPACK, Eigen System Routines library [Smith76].

The routine *tred2( )* reduces a real symmetric matrix to a symmetric tridiagonal matrix using and accumulating orthogonal similarity transformations. Since the covariance matrix is symmetric the process is greatly simplified. The routine *tql2( )* takes the tridiagonal matrix and determines the eigenvalues and eigenvectors using the QL iteration with shifts.

For the case of three spatial dimensions and two parametric dimensions the routine *param( )* produces the parametric coordinates in a selected coordinate system. The routine takes as arguments the final type of parametric coordinate system, the number of data points, the raw data points, eigenvalues, eigenvectors and mean vector and returns the final parameterized data points. Available types of parameterization mappings are: (1) projection onto a unit square, (2) projection onto a unit disk, (3) projection onto the sides of a unit cylinder and (4) projection onto a unit sphere. A routine, called *parameterize( )* is provided for straight projection from N to K dimensions in Euclidean space.

In Chapter 3 it was mentioned that, when using periodic coordinate systems for the parameterization of data, control over the boundaries of the fit patch can be gained by the duplication of points. The routine *perio( )* was implemented for this purpose. The routine is given the amount of overlap required and it returns the rebuilt point data structure. Section 3.2.2.2 and Figures 7 and 8 illustrate this process.

Additional routines are provided to manipulate matrices of arbitrary dimension. These can be used to build up transformation matrices other than those produced with *eigen( )*. Thus useful parameterizations of data points into a coordinate systems other than that of the principle distribution of the data are possible.

5.3. The Surface Fit

The fitting routines are capable of fitting N dimensional parametric surfaces with K parametric variables. The order of the fit is also arbitrary with the proviso that there must be at least as many data points as there are terms in the fitting multinomial. The routine *curve_fit*( ) controls the fitting process. This routine takes the parameterized data points and returns the equation data structure.

An equation is built as a data structure shown in Figure 17. It contains the values for the order of fit, the number of spatial dimensions, and the number of parametric variables. The fitting process works with the pyramid elements, which hold the information pertinent to each multinomial term in the fit. The name pyramid element suggests the appearance of the recurrence relation as expressed in section 2.3.1. Each pyramid element holds the scalar indices that indicate its position in the recursion (j, j' and j''). Each element also holds the index of the parametric variable that is used to produce that element's multinomial from previous terms (*k_of_element*). The *Alpha's* are the coefficients of the term's multinomial and are stored in an array. The number of *Alpha's* is equal to the number of previous multinomials in the pyramid that contribute to an element's multinomial, which is equal to the difference between this element's scalar index j and j'' the index of the last contributing multinomial. The fitting process produces one parametric equation for each spatial variable. These parametric equations are linear combinations of the fitting multinomials. The *Gamma's* are the coefficients of the surface fit, one set for each spatial dimension, and in each set one for each multinomial. The contribution of each point in the fit is stored in the array of *Psi's* associated with each element. Once the fit is completed the *Psi's* are deallocated, as they are no longer needed.

During the fitting process a large amount of storage is needed. For example, in a fit of three spatial dimensions and two parametric dimensions, with 1000 data points to be fit with a surface of order 6, about 2 Mbytes of space are required. However, once the fit is completed and the intermediate values are released, only about 300 bytes of storage is

required to hold the equation itself.

5.4. Rendering

As mentioned in Chapter 4, parametric surfaces can be rendered in several ways. Often the surface is approximated as a set of polygonal facets of varying sizes. These facets are then rendered as desired. If the facets are large they can be solidly shaded or simply outlined. Methods of smooth shading can be applied during polygonal rendering to improve the quality of the final image. Subdivision involves the division of the surface into facets so small that they cover only a single pixel of the raster output device.

Three different types of rendering were implemented for multinomial surfaces in three spatial dimensions. The first two, rendering as vectors and as polygons, were done using the GR graphics package. The third, subdivision, was implemented with a set of additional routines designed to be added to the GR package. Any rendering algorithm is called from the routine *crunch*( ). This routine takes the desired type of rendering as well as the required rendering information and calls the appropriate routine.

5.4.1. Vectors and Polygons

The simplest way of rendering a parametric patch is to divide its parametric domain into a grid, defined by equally spaced parameter values, and to evaluate the patch at the grid points to produce points in 3 space. These points are then joined with vectors. Though this does not result in a very realistic looking surface, it is a quite acceptable form of rendering for some applications. The figures in this thesis were rendered in just this way.

A routine *vectors*( ) is provided for such a rendering of three dimensional parametric surfaces. The number of divisions in u and v is specified, and a regular grid on the parametric plane is generated using these divisions. The resultant patch points, arranged

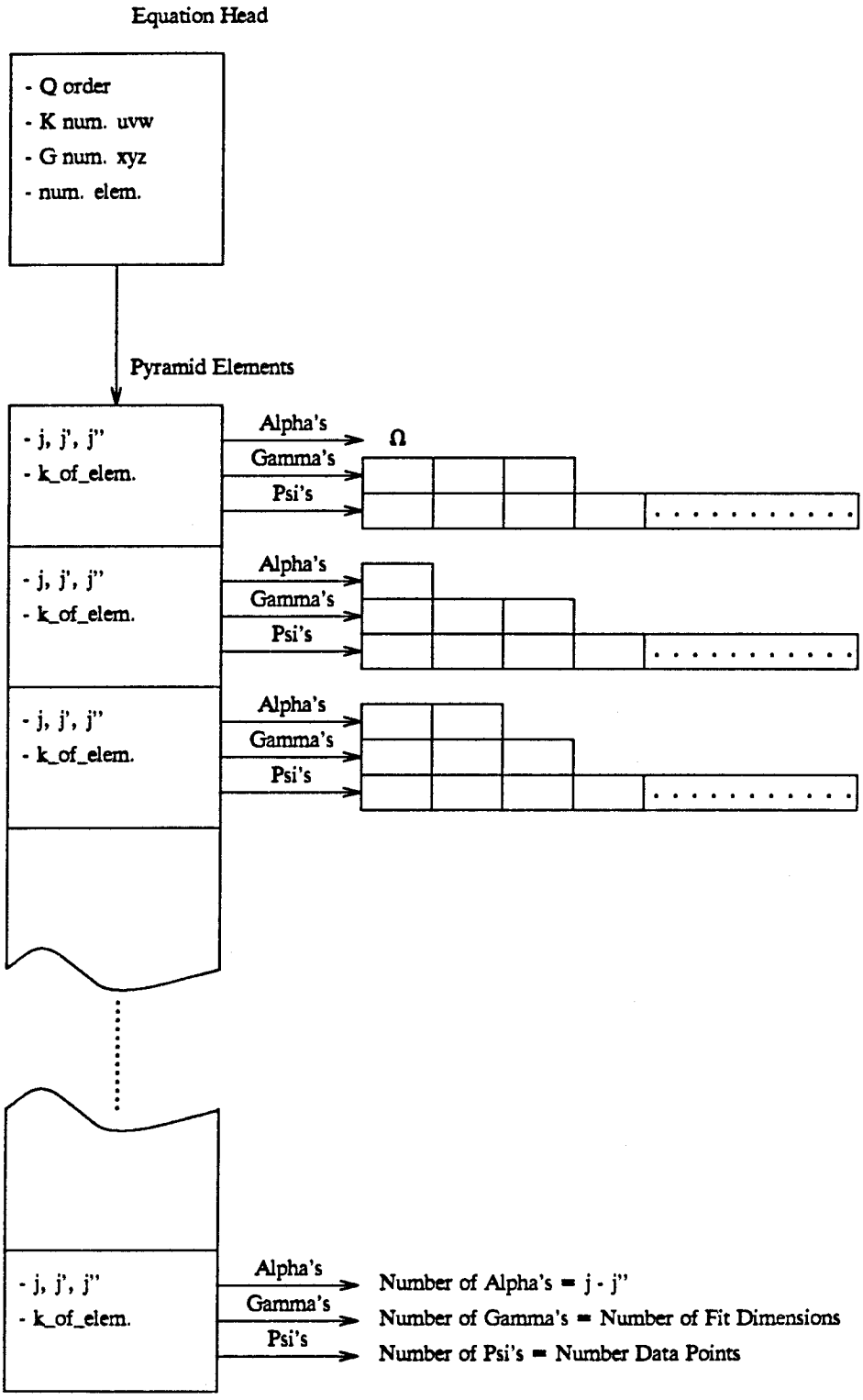


Figure 17 : Data Structure for an Equation

along lines of constant u or constant v, are then drawn on the desired output device with calls to *Move*( ) and *Draw*( ) in the GR package.

Polygonal rendering is done in much the same way, except that the GR routine *Poly*( ) is invoked rather than *Move*( ) and *Draw*( ). *Poly*( ) can provide Gouraud smooth shading when it is provided with the polygon vertex normals. The routine *polys*( ) carries out the polygonal rendering.

5.4.2. Subdivision

As discussed in Chapter 4, subdivision is recursive. At each stage a check is first made as to whether the patch meets some termination criteria. If it does the patch is rendered directly. If not the patch is subdivided and each subpatch is considered in turn.

The subdivision routines implemented are representation and application independent. Subdivision has mostly been applied to the rendering of three dimensional surfaces. It can also be used to divide a surface to some specified tolerance for use in another application. A possible example would be dividing a multinomial surface up in order to provide a control grid for a spline surface representation. The subdivision routines organize the order of subdivision and the storage of required data points.

The calculation of new data points, which is representation dependent, is done by a user-provided function. The other user-provided routine which deals with termination conditions is application-dependent.

The interface to the subdivision is through a single function *subdivide*( ). This routine takes the minimum and maximum parametric values, the minimum depth for subdivision, and the user functions *new*( ) and *term*( ). The minimum depth is a tuning constant which allows the user to speed up the subdivision process by setting a minimum limit on the number of subdivisions before termination is checked. This is provided on the assumption that checking termination is costly and that the user may wish to avoid checking it for some

number of subdivisions.

Communication between subdivision and user routines is accomplished by passing pointers to the data structure describing a patch corner point. New corner points are allocated with the routine *new*( ), which takes the present u and v values and returns a new corner point.

The routine *term*( ) takes five point data structures: the four corners of the patch and the center point of the patch. *term*( ) renders the patch if termination takes place and if the patch lies in the visible area of the scene.

In order to disassociate the surface representation from subdivision a user may select his own data structure for information that pertains to data points. This structure may include the spatial and parametric coordinates of the corner points and also attributes that are important to the patch rendering. In the case of multinomial surfaces the dependent variables may be of any number, not just the three spatial dimensions. Using this scheme, any number of dimensions could be stored in the structure.

5.4.2.1. Termination Criteria

One aim in this work was the graphical display of three dimensional multinomial surfaces. In this application termination of subdivision is based on two criteria, size and clipping. As a patch is divided it is checked against these two criteria. If the patch is entirely outside the field of view it is discarded as being invisible. If it is partially within the field of view or entirely within it is checked for size. Once a patch is small enough it is rendered and the process is repeated on the next subpatch.

The routine *Sub_term*( ) has been added to the GR graphics package for both the termination test and the eventual rendering of a subpatch. The *term*( ) routine makes use of *Sub_term*( ) for subdivision termination. An appropriate data structure is passed to *Sub_term*( ) so that it can determine what colour will be assigned to the target output pixel. The structure includes information about lighting in the scene as well as information about

the colour characteristics of the surface. Sample points are the points at the center of each raster element. If a patch covers no sample point he assigns the patch to the nearest sample point. This is not strictly a size test, but rather a test of coverage.

The termination procedure used by *Sub_term( )* is very similar to Catmull's. The polygon that approximates the subpatch is transformed from world coordinates, or object space, to a homogeneous coordinate space, where it is clipped against the of the viewing frustum. Once clipped the polygon and is eventually projected onto the image plane and the pixel closest to the patch's center is set to the appropriate colour.

Clipping takes place as fallows and is based on the polygon as well as the center point of the patch. If the center point is within the field of view the patch is checked for size. If the center point is not visible but its patch, defined by the polygon, is within the field of view, the patch is checked for size. If the patch is small enough and still not visible, it will never become visible.

The decision in image space as to whether a patch is small enough is based on the size of the polygon image. The test uses the largest diagonal measurement of the projected polygon and compares it against the diagonal measurement of a pixel in image space. The size of a pixel in the virtual coordinate system varies for different output devices depending on the actual physical resolution of the output device, which is known only by the lowest level device specific routines. Once a patch is visible and small enough, the pixel that underlies the patch is coloured appropriately after a Z-buffer check.

The termination routine has been implemented in the same style as the GR graphics package, namely as a graphical pipeline starting off in world coordinates and proceeding down through subroutine calls to specified output device coordinates. Windowing, clipping, viewporting and eventual rendering take place at different levels of the pipeline. The colour of a pixel in image space is determined using the normal to the approximating polygon and information about the nature of the patch surface such as colour and reflectance as well as

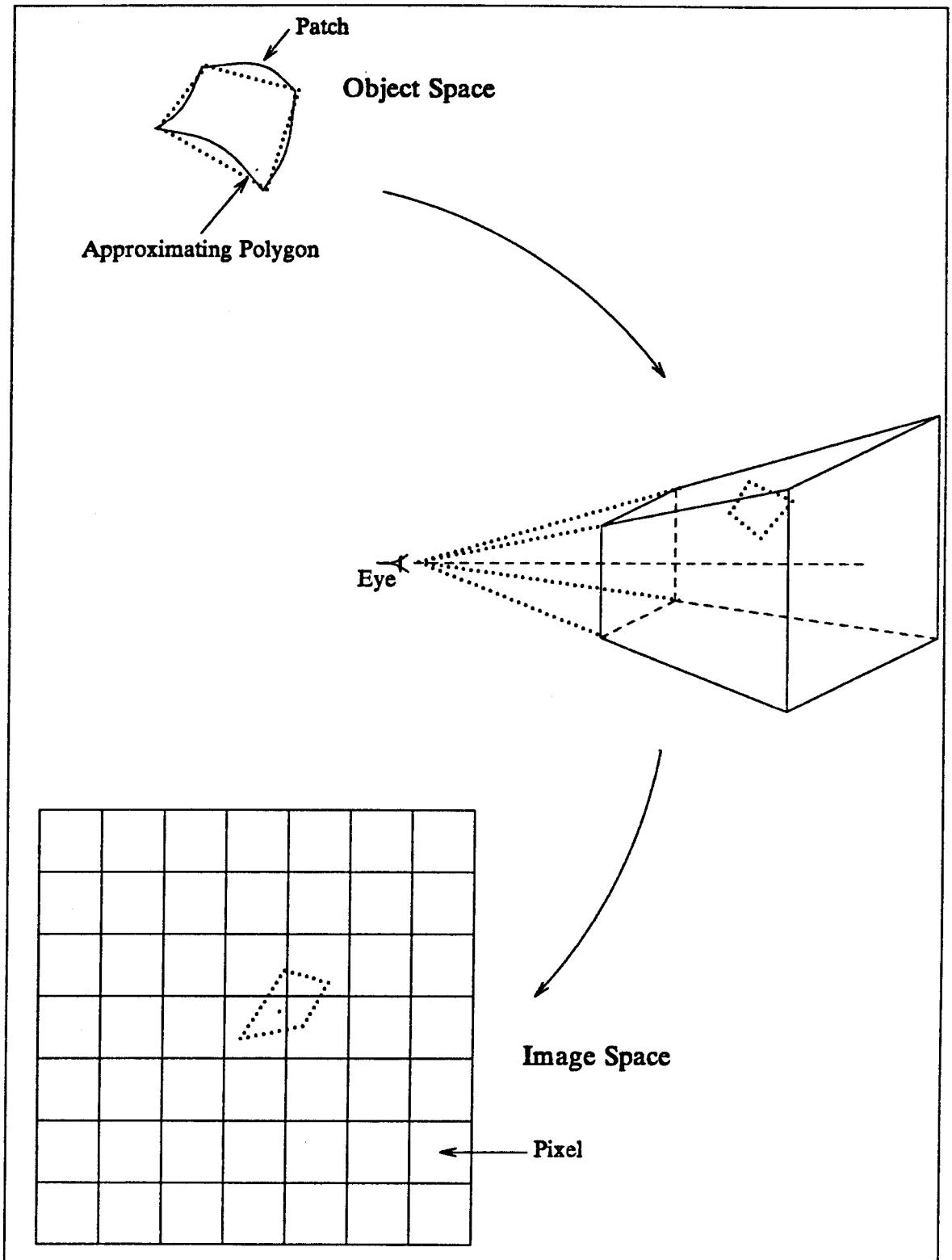


Figure 18 : Transformation of a Patch from Object Space to Image Space.

information about the scene in general, such as the position and colour of lights in the scene. The colour calculation is based on equation (16.9) of J. D. Foley and A. Van Dam [Foley82]. This equation is derived from the shading model developed by Phong Bui-Tuong [Bui-Tuong75].

6. Conclusions

A least squares fitting scheme for multinomial N dimensional surfaces has been developed for applications in computer graphics. It is capable of fitting surfaces in N dependent variables and K independent variables with multinomials of arbitrary order. The implementation has been used to fit parametric surfaces in three dimensions.

The fitting multinomials are derived from the input data using a recursive method for the construction of orthogonal multinomials. The coefficients of the least squares fit are determined using these orthogonal multinomials.

An original method of automatically parameterizing raw input data has also been implemented. It analyzes the data distribution and parameterizes the data in an attempt to maximize the information preserved in the feature selection process. The parameterization method is derived from techniques in statistical pattern recognition.

The parameterization and surface fitting utilities have been combined with the GR graphics package to model and display parametric surfaces. The following picture shows some results of this work.

Innovative aspects of this research are:

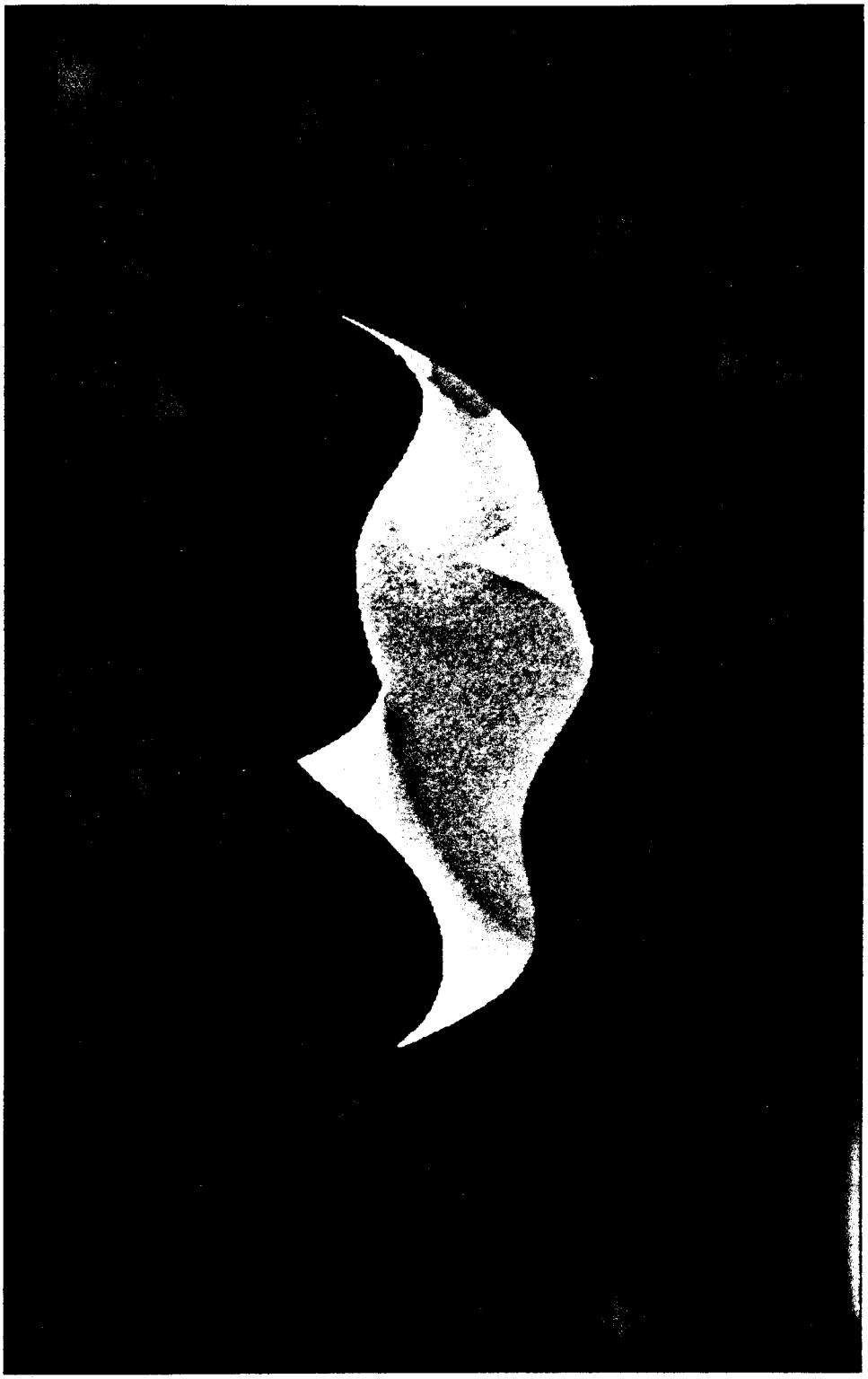
(1) the use of general multinomials to define graphical objects;

(2) the employment of a separate least squares fit in each spatial coordinate to create a parametric surface from arbitrarily given data, which is a technique not restricted to the

use of multinomials;

(3) the use of feature extraction techniques from pattern recognition to aid in the selection of a parametric coordinate system;

(4) a general subdivision process defined specifically in terms of recursion.



References

[Bartels83]
R.H. Bartels and J.J. Jezioranski (1983), On Least Squares Fitting Using Orthogonal Multinomials, University of Waterloo.

[Bui-Tuong75]
Phong Bui-Tuong (1975), Illumination for computer-Generated Pictures, *CACM* **18**(6).

[Cadwell61]
J.H. Cadwell and D.E. Williams (1961), Some Orthogonal Methods of Curve and Surface Fitting, *Computer Journal* **4**.

[Catmull74]
E.E. Catmull (1974), A Subdivision Algorithm For Computer Display of Curved Surfaces, Department of Computer Science, University of Utah.

[Clark76]
J. Clark (1976), Hierarchical Geometric Models for Visible Surface Algorithms, *CACM* **19** (10).

[Clark79]
J.H. Clark (Aug.1979), A Fast Scan-Line Algorithm for Rendering Parametric Surfaces, *Supplement to Proceedings of SIGGRAPH'79*, ACM Inc..

[Conte80]
S.D. Conte and Carl de Boor (1980), *Elimentary Numerical Analysis, third edition*, McGraw-Hill Inc., New York.

[Dyer80a]
C. Dyer, A. Rosenfeld, and H. Sammet (1980), Region Representation: Boundary Codes from Quadtrees, *CACM* **23**(3).

[Dyer80b]
C. Dyer (1980), The Space Efficiency of Quadtrees, *Computer Graphics and Image Processing* **14**.

[Foley82]
J.D. Foley and A. Van Dam (1982), *Fundamentals of Interactive Computer Graphics*, Addison-Weley Publishing Company, Reading, Mass..

[Forsythe57]
G.E. Forsythe (1957), Generation and Use of Orthogonal Polynomials for Data-Fitting with a Digital Computer, *J. SIAM* **5**, 74-87.

[Funkunaga72]
K. Funkunaga (1972), *Introduction to Statistical Pattern Recognition*, Academic Press, New York.

[Grasselli69]
A. Grasselli (1969), *Automatic Interpretation and Classification of Images*, Academic Press, New York.

[Jernigan79]
M.E. Jernigan (1979), *Pattern Recognition, Course Notes*, University of Waterloo.

[Kajiya82]
J.T. Kajiya, H. Fuchs, Z.M. Kedem, and S.P. Uselton (Oct.,1977), Optimal Surface Reconstruction from Planar Contours, *CACM* **20**(10), ACM Inc., 693-702.

[Lane80]
J.M. Lane, L.C. Carpenter, T. Whitted, and J.F. Blinn (Jan.1980), Scan Line

Methods for Displaying Parametrically Defined Surfaces, *Comm.ACMr* **23**(1).

[Lea83]
S.C. Lea (1983), Fecit A Structured Language for Describing Computer Generated Scenes, Department of Computer Science, University of Waterloo.

[Meagher82]
D. Meagher (1982), Geometric Modelling using Octree Encoding, *Computer Graphics and Image Processing* **19**.

[Patrick72]
E.A. Patrick (1972), *Fundamentals of Pattern Recognition*, Prentice-Hall, Inc., Englewood Cliffs, N.J..

[Schweitzer82]
D. Schweitzer and E. Cobb (July.,1982), Scanline Rendering of Parametric Surfaces, *Proceedings of SIGGRAPH'82* **16**, ACM Inc..

[Smith76]
B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V.C. Klema, and C.B. Moler (1976), Matrix Eigensystem Routines, EISPACK Guide, *Lecture Notes in Computer Science* **6**, Springer Verlag.

[Weisfeld59]
M. Weisfeld (1959), Orthogonal polynomials in Several Variables, *Numerische Mathematik* **1** , 38-40 .