Procedural Encoding of Scattered Data, Theory and Applications

Chair: Kelly Gaither
The University of Texas at Austin
Texas Advanced Computing Center
Austin, TX
kelly@tacc.utexas.edu
(512) 232-7751
(512) 475-9445
http://www.tacc.utexas.edu/~kelly

Instructors: Greg Nielson, Arizona State University, nielson@asu.edu
Hans Hagen, University of Kaiserslautern, hagen@informatik.uni-kl.de
David Ebert, Purdue University, ebertd@ecn.purdue.edu
Kelly Gaither, The University of Texas at Austin, kelly@tacc.utexas.edu

Length: Full Day

Level: Intermediate/Advanced

Abstract: Procedural encoding of scattered data sets is an active area of research with great potential for reconstructing surface information and compactly representing large data. The reduced storage requirements allow greater flexibility in the methods for manipulating and analyzing this data interactively. In this course, we will cover both the mathematical foundations behind existing encoding techniques, surface reconstruction methods, and volumetric representations. Additionally, we will present methods for feature analysis in the functional domain and conclude with applications and benefits of functional encoding in the scientific and engineering disciplines.

Who Should Attend: Anyone interested in learning how to represent either scattered, surface or volumetric data in a functional form, and anyone interested in learning how to manipulate this functional representation to generate feature information and visualizations.

What Attendees Will Gain: The mathematical foundations for encoding scattered, surface and volumetric data and concrete examples of extracting features from and visualizing the data expressed in a functional representation.

Course Outline:

1. **Introduction (Kelly Gaither 15 minutes, 8:30 am – 8:45 am)**
   The introduction will open up with the motivation behind putting this material together into one course, and the goals that we hope to achieve by presenting the material to the course attendees. This will set the stage for the full day. The presentation of the material is organized such that the mathematical theory is presented in the first half of the day, and applications of the theory and techniques are presented in the second half of the day.

a. Motivation and Goals
b. Overview of the Course

2. Mathematical Foundations (Greg Nielson, Hans Hagen 2 ½ hours, 8:45 am – 11:15 am)

The mathematical foundations will be a survey of several techniques for modeling scalar and vector valued functions that are based upon arbitrarily discrete sample measurements in a plane, a 3D volume, on manifolds, and in a domain that consists of both Euclidean space and time. Scattered data of this type occurs in many science, engineering and medical applications.

a. Data Examples – (Pressure over a wing , CAT, MRI, and fMRI , Rainfall over the Earth, Well Log Data, Big Sur Data, Flame Data, Car Flow Data, Brain Data, Climate Model Data, Stock Market Data, FEM Data, Reservoir Data)
b. Classification of Data
    i. Source: measured/simulated
    ii. Dimension: range/domain
    iii. Structure, Topology and Grids (uniform, rectilinear, curvilinear, triangular, tetrahedral, etc.)
c. Sampling and Brief Overview of Modeling Methods
    i. Basic Ideas – (Motivation, Problem, Basis Functions)
    ii. Methods – (Modified Quadratic Shepard, Volume Splines, Multiquadrics, Volume Minimum Norm Network, Localized Volume Splines)

BREAK (10:00 am – 10:30 am)

    iii. Comparisons
        1. Analytical Comparisons
            a. MQS: Fast, reasonably good fitting properties, very large data sets
            b. Volume Splines: Easy to implement, VG fitting, Conditioning problems
            c. Multiquadrics: Easy, excellent fitting, Conditioning and parameter selection
            d. Volume MNN: Massive Data, VG fitting, Not easy to implement
            e. Local Volume Splines: Massive Data, good fitting, problem with subdivision selection
        2. Results of Empirical Comparison

3. Surface Reconstruction (Greg Nielson, Hans Hagen 1 hour 45 minutes, 10:30 am – 12:15 pm)

Following the presentation of the necessary mathematical theory, the course will cover specific techniques that can be used for surface reconstruction.

a. Triangular Patches
b. Bezier Techniques
c. BBG Methods
d. The Side-Vertex Method for Interpolation in Triangles

e. Variation of Design

LUNCH BREAK (12:15 pm – 1:45 pm)

4. Volume Encoding (David Ebert 1 ½ hours, 1:45 pm – 3:15 pm)

This portion of the course will cover the process and methods for encoding volumetric scalar, vector and multifield data sets. RBF encoding methods are presented to provide specific examples of encoding volumetric data sets. Additionally, the benefits of being able to render this encoded data are presented to the attendees by covering a variety of techniques for direct rendering of the functionally encoded data.

a. RBF Encoding Techniques
   i. Motivation and Survey of Approaches
   ii. Advantages and Comparisons
   iii. Details of One System for Gaussian RBF Encoding of Scalar Data
   iv. Gaussian RBF Encoding of Vector Data
b. Rendering Issues for Interactive Exploration and Visualization
   i. Surface Generation and Visualization
   ii. Direct Rendering from Functional Encoding
      1. Hardware Capabilities and Limits
      2. Splatting Approaches
      3. Texture-based Volume Rendering
      4. Comparison and Trade-Offs

5. Feature Analysis Using Functional Encoding (Kelly Gaither 1 ½ hours, 3:15 pm – 4:45 pm)

The motivation and a brief survey of existing feature detection techniques are presented to the course attendees to provide a basis from which specific feature definitions are presented. These feature definitions are then presented in the functional domain by performing a change of basis on the fundamental operators and directly computing the feature equations in this basis.

a. Motivation and Survey of Existing Techniques
b. Feature Definitions
   i. Nomenclature
   ii. Feature Definitions

BREAK (3:45 pm – 4:15 pm)

c. Computing Features in the Functional Domain
   i. Mathematical Operators
   ii. Computing in the Functional Domain

6. Applications (Kelly Gaither 1 hour, 4:45pm – 5:45pm)

The course will close by presenting examples of using the functionally encoded representation to solve systems of equations, feature definitions, and to analyze and visualize the results.

a. Meshless Methods for Solving Systems of Partial Differential Equations
b. Examples of using functional encoding to analyze computational data sets

Tutorial Instructors:

Gregory M. Nielson is a professor of computer science and affiliate professor of mathematics at Arizona State University where he teaches and does research in the areas of Computer Graphics, Computer Aided Geometric Design, and Scientific Visualization. He has lectured and published widely on the topics of curve and surface representation and design; interactive computer graphics; scattered data modeling; and the analysis and visualization of multivariate data. He has edited several books and authored over 100 scientific articles. He has collaborated with several institutions including NASA, Xerox, General Motors, and LLNL. Professor Nielson received his PhD from the University of Utah. He has been on the editorial boards of ACM's Transactions on Graphics, The Rocky Mountain Journal of Mathematics, IEEE Computer Graphics and Applications, Visualization and Computer Animation Journal. He is currently on the editorial board of Computer Aided Geometric Design and the Editorial Advisory Board of IEEE Transactions on Visualization and Computer Graphics. He is one of the founders and members of the steering committee of the IEEE sponsored conference Visualization. He has previously chaired and is currently a director of the IEEE Computer Society Technical Committee on Computer Graphics. He is the recipient of an IEEE Meritorious Service Award, an IEEE Outstanding Contribution Award and the John Gregory Memorial Award in Geometric Modeling.

Hans Hagen is currently full professor at the Technical University of Kaiserslautern and chairman of the Computer Science Department. He is also the scientific director of the institute on Intelligent Visualization and Simulation at the German Research Center for Artificial Intelligence (DFKI). He holds a Ph.D. in mathematics from the University of Dortmund, a B. S. and M. S. in mathematics and a B. S. in computer science from the University of Freiburg. Prior to his current position, he was an associate professor at the TU Braunschweig and he had several visiting positions, especially in the USA. His research interests include all areas of scientific visualization, computer graphics and geometric modeling. He was editor in chief of the IEEE Transactions on visualization and computer graphics from1999-2003 and is an associated editor of CAGD, Computing and Surveys on Mathematics in Industry. Prof. Hagen has published nearly 200 articles in scientific visualization, computer graphics, geometric modeling and geometry and is a member of ACM, GI, IEEE, and SIAM.

David Ebert's research interests include scientific visualization, volume rendering, and procedural techniques. Ebert received his Ph.D. from The Ohio State University in 1991 and is an Associate Professor with the School of Electrical and Computer Engineering at Purdue. He has taught twelve courses at the ACM SIGGRAPH Conference every year since 1992, has published numerous articles on visualization, volume rendering, volume illustration, multifield visualization, simulating natural phenomena, and is the co-author of Texturing and Modeling: A Procedural Approach, published by Morgan Kaufmann. Ebert has been very active in the visualization and computer graphics community, serving on numerous program committees, serving as papers co-chair for IEEE

Visualization 98 and 99, and is currently Editor-in-Chief of IEEE Transactions on Visualization and Computer Graphics.

Kelly Gaither (Course Organizer) is an Associate Director and Research Scientist at the Texas Advanced Computing Center, The University of Texas at Austin. The combination of her undergraduate and master's degree in Computer Science and her doctoral degree in Computational Engineering make her skilled at developing and researching topics that are heavily rooted in the science and engineering applications. She spent ten years at the National Science Foundation Engineering Research Center for Complex Geometries and Complex Physics where she worked closely with both the Computational Fluid Dynamics group and the Scientific Visualization group. Her research interests include large data visualization, feature detection, and applications of visualization. She is currently serving as the general chair for the Visualization '04 conference. She previously served as the program co-chair ('03), case studies co-chair ('02), works in progress co-chair ('01), late breaking hot topics co-chair ('00), tutorials co-chair ('98,'99), and publicity co-chair ('95,'96).

Mathematical Foundations of Procedural Encoding of Scattered Data

Gregory M. Nielson, nielson@asu.edu

1. Data Examples

Rectilinear, Cartesian Grids, Well Log, Curvilinear Grids, Free hand US, Flame

2. Models and Methods

2.1 Interpolation Methods

2.1.1 Sampling of Methods and Techniques
(i) Inverse Distance and related RBFs
(ii) Volume Splines and related RBFs
(iii) Multiquadrics
(iv) Volume version of Minimum Norm Network
(v) Localization techniques for massive data sets

References:
1) Nielson, Minimum Norm Network, *Math. Comp*. 40:161, 253-271
2) Nielson, Multivar. Smoothing Splines, *SIAM J. Num. Anal*., 11:2, 435-446

2.1.2 Comparisons
(i) Ease of Implementation, (ii) Applicability
(iii) Feature maintenance quality (iv) Efficiency

References
3) Franke, Scattered Data Interpolation, *Math. Comp.* 38:157, 181-200
4) Nielson, Scattered Data Modeling, *CG&A*, 13:1, 60-70

2.2 Approximation Methods

2.2.1 Least Squares
(i) Knot Selection, (ii) Total Fit, (iii) Venetia Criteria

2.2.2 Adaptive/Progressive Models
(i) Refinement strategies (ii) Cracking problems (iii)

References:
5) Roxborough et al. Progressive Models for US, *Vis 2000*, 93-100
6) Nielson, Triangulations & Tetra., *Scientific Visualization*, 429-525
7) Chen et al. *Volume Graphics*, Springer, 29-48.

A Method for Interpolating Scattered Data Based Upon a Minimum Norm Network

Gregory M. Nielson

Mathematics of Computation, Vol. 40, No. 161 (Jan., 1983), 253-271.

Stable URL:
http://links.jstor.org/sici?sici=0025-5718%28198301%2940%3A161%3C253%3AAMFISD%3E2.0.CO%3B2-0

Mathematics of Computation is currently published by American Mathematical Society.

MATHEMATICS OF COMPUTATION
VOLUME 40, NUMBER 161
JANUARY 1983, PAGES 253-271

A Method for Interpolating Scattered Data Based Upon a Minimum Norm Network*

By Gregory M. Nielson

Abstract. A method for interpolating scattered data is described. Given (x_i, y_i, z_i), $i = 1,\ldots,N$, a bivariate function S with continuous first order partial derivatives is defined which has the property that $S(x_i, y_i) = z_i$, $i = 1,\ldots,N$. The method is based upon a triangulation of the domain and a curve network which has certain minimum pseudonorm properties. Algorithms and examples are included.

1. Introduction. In this paper, we present a new method for interpolating scattered data. Given the data (x_i, y_i, z_i), $i = 1,\ldots,N$, we describe the construction of a bivariate function S which has continuous first order partial derivatives and $S(x_i, y_i) = z_i$, $i = 1,\ldots,N$. The method consists of three separate steps:

(i) *Triangulation.* The points $V_i = (x_i, y_i)$, $i = 1,\ldots,N$, are used as the vertices of a triangulation of a domain D.

(ii) *Curve Network.* The approximation S and its first order partial derivatives, S_x and S_y, are defined on the subset consisting of the union of all edges.

(iii) *Blending.* S is extended to D by means of a blending method which will assume arbitrary position and slope on the boundary of a triangular domain.

The basic idea of an interpolant which is defined in a piecewise fashion over triangles is not new. Both Lawson [7] and Akima [1] have described such methods. Lawson's paper contains a good discussion of many of the aspects of triangulating the convex hull of V_i, $i = 1,\ldots,N$. Both of these methods make use of a discrete C^1 interpolant (i.e., a C^1 finite element) for each triangle followed by a local method for estimating certain partial derivatives. Even though our method is based upon the approach of a curve network followed by the use of a triangular blending (transfinite) interpolant, the particular method we eventually propose can be viewed as an assembly of discrete C^1 interpolants along with a technique for estimating partial derivatives.

2. The Curve Network. We assume $N \geq 3$ and that the points V_i, $i = 1,\ldots,N$, are not collinear. Let T_{ijk} denote the triangle with vertices V_i, V_j and V_k, $i \neq j \neq k \neq i$. The list of triple indices which determines the triangulation is denoted by I_t so that $D = \cup_{ijk \in I_t} T_{ijk}$. Let e_{ij} represent the line segment with endpoints V_i and V_j, and let $N_e = \{ij\colon i, j \in \{\alpha, \beta, \gamma\}, \alpha\beta\gamma \in I_t\}$ be a list of the indices representing the edges of the triangulation. In terms of this notation, the domain of the curve network is

Received January 22, 1979; revised April 13, 1981 and June 21, 1982.
1980 *Mathematics Subject Classification.* Primary 65D05; Secondary 41A15.
Key words and phrases. Bivariate interpolation, scattered data interpolation, random data interpolation, splines, multivariate approximation and interpolation.
*This research was supported by the Office of Naval Research under Contract NR 044-443.

0025-5718/82/0000-0751/$05.50

$E = \bigcup_{ij \in N_e} e_{ij}$. The derivative normal to an edge e_{ij} is given by

$$\frac{\partial F}{\partial n_{ij}} = \frac{(y_j - y_i)}{\|e_{ij}\|} \frac{\partial F}{\partial x} - \frac{(x_j - x_i)}{\|e_{ij}\|} \frac{\partial F}{\partial y},$$

where $\|e_{ij}\|$ is the length of e_{ij}. The derivative along an edge is given by

$$\frac{\partial F}{\partial e_{ij}} = \frac{(x_j - x_i)}{\|e_{ij}\|} \frac{\partial F}{\partial x} + \frac{(y_j - y_i)}{\|e_{ij}\|} \frac{\partial F}{\partial y}.$$

Therefore,

$$\frac{\partial F}{\partial x} = \frac{(x_j - x_i)}{\|e_{ij}\|} \frac{\partial F}{\partial e_{ij}} + \frac{(y_j - y_i)}{\|e_{ij}\|} \frac{\partial F}{\partial n_{ij}},$$

$$\frac{\partial F}{\partial y} = \frac{(y_j - y_i)}{\|e_{ij}\|} \frac{\partial F}{\partial e_{ij}} - \frac{(x_j - x_i)}{\|e_{ij}\|} \frac{\partial F}{\partial n_{ij}},$$

and so it is clear that if S and $\partial S/\partial n_{ij}$ are known on each edge, the information required for step (ii) is available. It is more convenient to specify S and its normal derivatives on E since these values can be defined independent of each other at all points except the vertices. We will first define S on E, but prior to this, we review some material concerning univariate cubic splines which motivates our particular choice of the curve network.

Given the data (t_i, s_i), $i = 1, \ldots, n$, where $t_1 < t_2 < \cdots < t_n$, the univariate natural spline of interpolation can be characterized (cf. de Boor and Lynch [4]) as the unique solution to the problem

$$\text{(2.1)} \qquad \begin{aligned} &\operatorname*{Min}_{f \in H[t_1, t_n]} \int_{t_1}^{t_n} [f''(t)]^2 \, dt, \\ &\text{subject to: } f(t_i) = s_i, \end{aligned}$$

where $H[t_1, t_n] = \{f\colon f \in C[t_1, t_n],\ f' \text{ is absolutely continuous},\ f'' \in L^2[t_1, t_n]\}$. From this point of view, the mathematical spline is an analogue of the physical spline. As a result of this minimization, it can be shown that s is a piecewise cubic polynomial which has a continuous second derivative and $s''(t_1) = s''(t_n) = 0$. Towards the definition of a curve network with an analogous characterization, we introduce $C[E] = \{F\colon F \text{ is the restriction to } E \text{ of some } C^1 \text{ function defined on } D$ and the univariate function obtained as the restriction of F to e_{ij} is an element of $H[e_{ij}]\}$. Analogous to the minimum pseudonorm property of univariate splines, we consider the problem of finding an interpolating curve network which minimizes

$$\text{(2.2)} \qquad \sigma(F) = \sum_{ij \in N_e} \int_{e_{ij}} \left[\frac{\partial^2 F}{\partial e_{ij}^2} \right]^2 ds_{ij},$$

where ds_{ij} represents the element of arc length on the curve consisting of the line segment e_{ij}.

We find it convenient to view each $F \in C[E]$ as a collection of univariate functions

$$\text{(2.3)} \qquad f_{ij}(t) = F\big((1-t)V_i + tV_j\big), \qquad ij \in N_e,\ 0 \leqslant t \leqslant 1.$$

With the following parametrization of the curve e_{ij}:

$$x(t) = (1-t)x_i + tx_j, \qquad y(t) = (1-t)y_i + ty_j,$$

and the fact that

$$\frac{1}{\|e_{ij}\|^2} f''_{ij} = \frac{\partial^2 F}{\partial e_{ij}^2}\Big|_{e_{ij}},$$

we have that

$$\begin{aligned}\sigma(F) &= \sum_{ij\in N_e} \int_0^1 \left[\frac{f''_{ij}(t)}{\|e_{ij}\|^2}\right]^2 \sqrt{[x'(t)]^2 + [y'(t)]^2}\, dt \\ &= \sum_{ij\in N_e} \frac{1}{\|e_{ij}\|^3} \int_0^1 [f''_{ij}(t)]^2\, dt.\end{aligned}$$

THEOREM 2.1. *Let $S \in C[F]$ be the unique piecewise cubic network with the properties that $S(V_i) = z_i$, $i = 1,\dots,N$, and*

$$\begin{aligned}(2.4)\quad &\sum_{ij\in N_i} \frac{(x_j - x_i)}{\|e_{ij}\|^3}\Big[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) \\ &\qquad + \tfrac{1}{2}(x_j - x_i)S_x(V_j) + \tfrac{1}{2}(y_j - y_i)S_y(V_j) + \tfrac{3}{2}(z_i - z_j)\Big] = 0, \\ &\sum_{ij\in N_i} \frac{(y_j - y_i)}{\|e_{ij}\|^3}\Big[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) \\ &\qquad + \tfrac{1}{2}(x_j - x_i)S_x(V_j) + \tfrac{1}{2}(y_j - y_i)S_y(V_j) + \tfrac{3}{2}(z_i - z_j)\Big] = 0,\end{aligned}$$

where

$$N_i = \{ij\colon e_{ij} \text{ is the edge of the triangulation with the endpoint } V_i\}.$$

Then, among all functions $F \in C[E]$, $F(V_i) = z_i$, $i = 1,\dots,N$, the function S uniquely minimizes $\sigma(F)$.

Proof. We first define the inner product

$$\langle F, G\rangle = \sum_{ij\in N_e} \frac{1}{\|e_{ij}\|^3} \int_0^1 f''_{ij}(t) g''_{ij}(t)\, dt$$

and note that

$$\sigma(F) - \sigma(S) = \langle F - S, F - S\rangle + 2\langle S, F - S\rangle.$$

For the moment, we assume that (2.4) has a solution and write

$$\begin{aligned}s_{ij}(t) &= t^2(3-2t)z_j + (1-t)^2(2t+1)z_i \\ &\quad + t(1-t)^2\big[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i)\big] \\ &\quad + t^2(t-1)\big[(x_j - x_i)S_x(V_j) + (y_j - y_i)S_y(V_j)\big].\end{aligned}$$

If F is any element of $C[E]$ which interpolates, then

$$\int_0^1 s_{ij}''(t)\big[f_{ij}''(t) - s_{ij}''(t)\big]\,dt$$

$$= s_{ij}''\big[f_{ij}' - s_{ij}'\big]\Big|_0^1 - s_{ij}'''\big[f_{ij} - s_{ij}\big]\Big|_0^1 + \int_0^1 s_{ij}^{\mathrm{iv}}(t)\big[f_{ij}(t) - s_{ij}(t)\big]\,dt$$

$$= s_{ij}''\big[f_{ij}' - s_{ij}'\big]\Big|_0^1.$$

Since

$$\begin{aligned} s_{ij}''(1) &= 6[z_i - z_j] + 2\big[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i)\big] \\ &\quad +4\big[(x_j - x_i)S_x(V_j) + (y_j - y_i)S_y(V_j)\big], \\ s_{ij}''(0) &= 6[z_j - z_i] - 4\big[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i)\big] \\ &\quad -2\big[(x_j - x_i)S_x(V_j) + (y_j - y_i)S_y(V_j)\big], \end{aligned}$$

we conclude that

$$\begin{aligned} (2.5)\quad &\sum_{ij\in N_e} \frac{1}{\|e_{ij}\|^3} \int_0^1 s''(t)\big[f''(t) - s''(t)\big]\,dt \\ &= \sum_{i=1}^{N} \Bigg\{ \sum_{ij\in N_i} \frac{1}{\|e_{ij}\|^3} \big[6(z_i - z_j) + 2(x_j - x_i)\big(S_x(V_j) + 2S_x(V_i)\big) \\ &\qquad +2(y_j - y_i)\big(S_y(V_j) + 2S_y(V_i)\big)\big] \\ &\qquad \times \big[(x_j - x_i)\,\big[F_x(V_i) - S_x(V_i)\big] + (y_j - y_i)\big[F_y(V_i) - S_y(V_i)\big]\big]\Bigg\}. \end{aligned}$$

The change in summation is allowable because $s_{ij}''(1) = s_{ji}''(0)$. This last equation points out the fact that $\langle S, F - S\rangle$ is independent of whether ij or ji is listed in N_e. This is the reason we were not definite about this before. Applying (2.4) to (2.5), we have that

$$\langle S, F - S\rangle = 0, \tag{2.6}$$

and so

$$\sigma(F) - \sigma(S) = \sigma(F - S) \geqslant 0$$

for any curve network F such that $F(V_i) = z_i$. This establishes the minimum property assuming that S exists. The existence of S requires a solution of the $2N$ linear equations of (2.4). We will show that this system has a solution by showing that the homogeneous system ($z_i = 0$, $i = 1,\ldots,N$) has only the trivial solution. This argument will also establish the uniqueness of S. Let $\bar{S}$ be the piecewise cubic curve network associated with the solution $\bar{S}_x(V_i)$, $\bar{S}_y(V_i)$, $i = 1,\ldots,N$, of the homogeneous system. The same line of reasoning which led to (2.6) can be used to conclude that

$$\langle \bar{S}, \bar{S}\rangle = 0.$$

That is, we replace both S and $F - S$ with $\bar{S}$. Therefore, $\int_0^1 [\bar{s}_{ij}''(t)]^2\,dt = 0$, $ij \in N_e$, which implies that $\bar{s}_{ij}$ is linear. But $\bar{s}_{ij}(0) = \bar{s}_{ij}(1) = 0$, and so

$$(x_j - x_i)\bar{S}_x(V_i) + (y_j - y_i)\bar{S}_y(V_i) = 0, \qquad ij \in N_i,\ i = 1,\ldots,N.$$

Since each V_i is the vertex of some nondegenerate triangle, this is sufficient to imply that $\bar{S}_x(V_i) = \bar{S}_y(V_i) = 0$, $i = 1,\ldots,N$, and so the proof is complete.

COROLLARY 2.2. *If the data* (x_i, y_i, z_i), $i = 1,\ldots,N$, *lie on a plane, then the curve network* S *of Theorem* 2.1 *will also lie on this plane.*

Proof. Let P represent the plane, and let

$$\hat{S} = P\,|_E$$

be the restriction of this plane to the edges. Then $\hat{S} \in C[E]$, $\hat{S}$ interpolates and $\sigma(\hat{S}) = 0$. Since the minimum norm network is unique, it must be that case that $S = \hat{S}$.

In order to complete the information required by step (ii), we need to define the normal derivative on each edge. We make a particularly simple choice here and take the normal derivatives to be linear. That is,

$$\frac{\partial S}{\partial n_{ij}}\big((1-t)V_i + tV_j\big) = (1-t)\left[\frac{(y_j - y_i)S_x(V_i) - (x_j - x_i)S_y(V_i)}{\|e_{ij}\|}\right] + t\left[\frac{(y_j - y_i)S_x(V_j) - (x_j - x_i)S_y(V_j)}{\|e_{ij}\|}\right], \qquad ij \in N_e.$$

3. The Blending Method. We now discuss the choice of the triangular blending method to be used to extend the curve network to the domain D. For these purposes, we let T represent an arbitrary triangle with vertices V_i, $i = 1, 2, 3$, and $b_i = b_i(x, y)$, $i = 1, 2, 3$, denote the barycentric coordinates given by

$$\begin{aligned} x &= b_1x_1 + b_2x_2 + b_3x_3,\\ y &= b_1y_1 + b_2y_2 + b_3y_3,\\ 1 &= b_1 + b_2 + b_3. \end{aligned}$$

Let $I = \{(1,2,3), (2,3,1), (3,1,2)\}$.

The first method of approximation to assume predescribed values of a function and its first order derivatives on the boundary of a triangle is due to Barnhill, Birkhoff and Gordon [3]. This method requires the specification and compatibility of the cross partials:

$$\frac{\partial^2 F}{\partial e_{ik}\partial e_{ij}}(V_i) = \frac{\partial^2 F}{\partial e_{ij}\partial e_{ik}}(V_i), \qquad (i, j, k) \in I.$$

While those values are obtainable from the curve network information, in general they will not be compatible.

In fact,

$$\frac{\partial^2 S}{\partial e_{ij}\partial e_{ik}}(V_i) = s_{ij}''(0)c_{kji} + \alpha_{ji}\,d_{kji}, \qquad (i, j, k) \in I,$$

where

$$\alpha_{mn} = (y_m - y_n)\big(S_x(V_m) - S_x(V_n)\big) + (x_m - x_n)\big(S_y(V_n) - S_y(V_m)\big),$$

$$c_{lmn} = \frac{(x_l - x_n)(x_m - x_n) + (y_l - y_n)(y_m - y_n)}{\|e_{nl}\|\,\|e_{mn}\|^3}$$

and

$$d_{lmn} = \frac{(x_l - x_n)(y_m - y_n) - (y_l - y_n)(x_m - x_n)}{\|e_{nl}\|\,\|e_{mn}\|^3}.$$

Therefore $\partial^2 S(V_i)/\partial e_{ij}\partial e_{ik}$ will involve z_j, $S_x(V_j)$ and $S_y(V_j)$, while $\partial^2 S(V_i)/\partial e_{ik}\partial e_{ij}$ will involve z_k, $S_x(V_k)$ and $S_y(V_k)$, and so in general these two partials will not be equal. A recently developed method which does not explicitly involve or require the compatibility of the cross partials is described in [9]. This method is based upon the combination of three interpolants each having certain miminum norm properties. When the boundary values given by the curve network are substituted into this triangular blending method we obtain the following nine-parameter C^1 interpolant defined over T.

$$\begin{aligned}(3.1)\qquad S_T(x, y) = \sum_{(i,j,k)\in I} & S(V_i)\big[b_i^2(3 - 2b_i) + 6wb_i(b_k\alpha_{ij} + b_j\alpha_{ik})\big] \\ & + S_k'(V_i)\big[b_i^2 b_k + wb_i(3b_k\alpha_{ij} + b_j - b_k)\big] \\ & + S_j'(V_i)\big[b_i^2 b_j + wb_i(3b_j\alpha_{ik} + b_k - b_j)\big],\end{aligned}$$

where

$$S_j'(V_i) = (x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i),$$

$$w = \frac{b_1 b_2 b_3}{b_1 b_2 + b_1 b_3 + b_2 b_3},$$

$$\alpha_{ij} = \frac{\|e_{jk}\|^2 + \|e_{ik}\|^2 - \|e_{ij}\|^2}{2\|e_{ik}\|^2}.$$

If S_{ijk} is used to represent the same discrete interpolant for the triangle T_{ijk}, then the final interpolant can be represented as

$$S(x, y) = S_{ijk}(x, y) \quad \text{for } (x, y) \in T_{ijk}.$$

Concerning the degree of algebraic precision of S, it can be shown that S_T will reproduce quadratics but the curve network is limited to linear precision and so the final interpolation operator has precision of degree one.

4. Algorithms and Examples. The first step in applying the approximation S requires a triangulation of D. In those cases where D is the convex hull of V_i, $i = 1,\ldots,N$, we have incorporated an algorithm described by Lawson [7] which selects a particular triangulation on the basis of the max-min angle criterion. The program that implements this algorithm produces information describing the boundary along with three arrays n_1, n_2 and n_3, each of length N_t, which form a list of the vertices of each triangle of the triangulation. An example of a triangulation produced by this program is given in Table 1 and Figure 1.

TABLE 1

Data			Boundary	Triangulation			
i	x_i	y_i	1, yes 0, no	i	$n_1(i)$	$n_2(i)$	$n_3(i)$
1	.21	.88	1	1	7	1	8
2	.46	.93	1	2	1	7	2
3	.83	.89	1	3	7	8	10
4	.97	.54	1	4	7	10	11
5	.67	.71	0	5	2	7	6
6	.53	.74	0	6	6	7	11
7	.28	.77	0	7	10	8	9
8	.07	.70	1	8	11	10	17
9	.06	.43	1	9	17	10	9
10	.25	.56	0	10	11	17	16
11	.48	.61	0	11	11	16	12
12	.67	.54	0	12	6	11	5
13	.77	.45	0	13	2	6	5
14	.90	.31	0	14	5	11	12
15	.66	.35	0	15	17	9	18
16	.50	.47	0	16	17	19	16
17	.32	.44	0	17	19	17	18
18	.25	.31	0	18	16	19	15
19	.46	.33	0	19	12	4	5
20	.57	.20	0	20	2	5	3
21	.75	.25	0	21	12	16	15
22	.94	.05	1	22	12	15	13
23	.46	.07	0	23	18	9	24
24	.18	.19	0	24	18	23	19
25	.14	.06	1	25	23	18	24
				26	4	12	13
				27	15	19	20
				28	15	20	21
				29	13	15	21
				30	24	9	25
				31	19	23	20
				32	23	24	25
				33	13	21	14
				34	3	5	4
				35	20	22	21
				36	22	20	23
				37	4	13	14
				38	23	25	22
				39	14	21	22
				40	4	14	22

The next step of our method requires the solution of (2.4). The coefficient matrix of this linear system is in general quite sparse, but the structure is sufficiently complicated to eliminate the use of a direct method which takes advantage of this sparseness. We have found that an iterative method based upon the following equivalent form of (2.4) works quite well:

$$\begin{pmatrix} Sx_i \\ Sy_i \end{pmatrix} = -\begin{pmatrix} \alpha_i & \beta_i \\ \beta_i & \gamma_i \end{pmatrix}^{-1} \begin{pmatrix} \sum_{ij \in N_i} \alpha_{ij} Sx_j + \sum_{ij \in N_i} \beta_{ij} Sy_j - Zx_i \\ \sum_{ij \in N_i} \beta_{ij} Sx_j + \sum_{ij \in N_i} \gamma_{ij} Sy_j - Zy_i \end{pmatrix}, \qquad i = 1, \ldots, N,$$

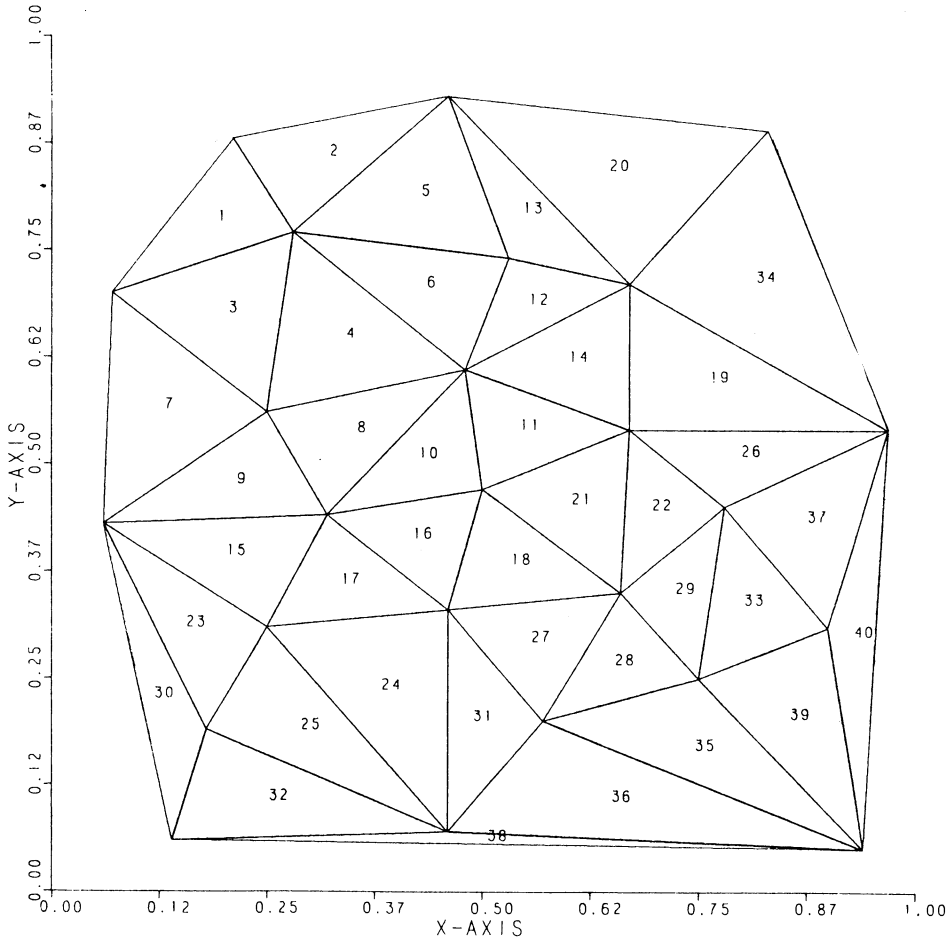


FIGURE 1

where $Sx_j = S_x(V_j)$, $Sy_j = S_y(V_j)$

$$\alpha_{ij} = \frac{(x_j - x_i)^2}{2\|e_{ij}\|^3}, \qquad \alpha_i = 2 \sum_{ij \in N_i} \alpha_{ij},$$

$$\beta_{ij} = \frac{(x_j - x_i)(y_j - y_i)}{2\|e_{ij}\|^3}, \quad \beta_i = 2 \sum_{ij \in N_i} \beta_{ij},$$

$$\gamma_{ij} = \frac{(y_j - y_i)^2}{2\|e_{ij}\|^3}, \qquad \gamma_i = 2 \sum_{ij \in N_i} \gamma_{ij},$$

$$Zx_i = \frac{3}{2} \sum_{ij \in N_i} \frac{(z_j - z_i)(x_j - x_i)}{\|e_{ij}\|^3},$$

and

$$Zy_i = \frac{3}{2} \sum_{ij \in N_i} \frac{(z_j - z_i)(y_j - y_i)}{\|e_{ij}\|^3}.$$

For an initial approximation, we first obtain the first order partial derivatives of each plane which interpolates over each triangle. Then for each vertex, we compute the average of these derivatives for each triangle that involves this vertex. This amounts to the following computation:

$$Sx_i^0 = \frac{1}{|M_i|} \sum_{ijk \in M_i} \frac{f_{ijk}}{A_{ijk}}, \qquad Sy_i^0 = \frac{1}{|M_i|} \sum_{ijk \in M_i} \frac{g_{ijk}}{A_{ijk}},$$

where $M_i = \{abc\colon T_{abc}$ is a triangle with vertex $V_i\}$,

$|M_i|$ is the number of elements of M_i,

$f_{ijk} = (y_j - y_k)z_i + (y_k - y_i)z_j + (y_i - y_j)z_k$,

$g_{ijk} = (x_k - x_j)z_i + (x_i - x_k)z_j + (x_j - x_i)z_i$, and

$A_{ijk} = (x_i - x_j)(y_i - y_k) - (y_i - y_j)(x_i - x_k)$.

The first pass of the following algorithm computes the initial values Sx_i^0, Sy_i^0, $i = 1,\ldots,N$, as well as α_i, β_i, γ_i, Zx_i, Zy_i, $i = 1,\ldots,N$, which remain constant throughout the iteration process.

For $l = 1,\ldots,N_t$, do:

  For $(i, j, k) \in I$, do:

    $a = n_i(l), b = n_j(l), c = n_k(l)$

    $|M_a| = |M_a| + 1$

    $Sx_a^0 = Sx_a^0 + f_{abc}/A_{abc}$

    $Sy_a^0 = Sy_a^0 + g_{abc}/A_{abc}$

    $\alpha_a = \alpha_1 + \tilde{\alpha}_{ab} + \tilde{\alpha}_{ac}$

    $\beta_a = \beta_a + \tilde{\beta}_{ab} + \tilde{\beta}_{ac}$

    $\gamma_a = \gamma_a + \tilde{\gamma}_{ab} + \tilde{\gamma}_{ac}$

    $$Zx_a = Zx_a + \frac{3}{2}\left[\frac{\tau_{ab}(x_b - x_a)(z_b - z_a)}{\|e_{ab}\|^3} + \frac{\tau_{ac}(x_c - x_a)(z_c - z_a)}{\|e_{ac}\|^3}\right]$$

    $$Zy_a = Zy_a + \frac{3}{2}\left[\frac{\tau_{ab}(y_b - y_a)(z_b - z_a)}{\|e_{ab}\|^3} + \frac{\tau_{ac}(y_c - y_a)(z_c - z_a)}{\|e_{ac}\|^3}\right]$$

For $i = 1,\ldots,N$, do:

  $$Sx_i^0 = \frac{Sx_i^0}{|M_i|}$$

  $$Sy_i^0 = \frac{Sy_i^0}{|M_i|}$$

  $\delta = \alpha_i\gamma_i - \beta_i^2$

  $$\alpha_i^{-1} = \frac{\gamma_i}{\delta}$$

  $$\beta_i^{-1} = \frac{-\beta_i}{\delta}$$

  $$\gamma_i^{-1} = \frac{\alpha_i}{\delta}$$

We have used the notation

$$\tilde{\alpha}_{ab} = \tau_{ab}\alpha_{ab}, \quad \tilde{\beta}_{ab} = \tau_{ab}\beta_{ab}, \quad \tilde{\gamma}_{ab} = \tau_{ab}\gamma_{ab},$$

where

$$\tau_{ab} = \begin{cases} 1 & \text{if } V_a \text{ and } V_b \text{ are on the boundary,} \\ 1/2 & \text{otherwise.} \end{cases}$$

The factor of τ_{ab} is necessary because we go sequentially through the list of triangles causing each interior edge to be processed twice.

The iterative part of the algorithm can be described as follows:

For $i = 1,\ldots,N$, do:
  $\delta_i = Zx_i, \varepsilon_i = Zy_i$

For $n = 0, 1,\ldots,$ until satisfied, do:
  For $l = 1,\ldots,N_t$, do:
    For $(i, j, k) \in I$, do:
      $a = n_i(l), b = n_j(l), c = n_k(l)$
      $\delta_a = \delta_a - \frac{1}{4}\left[\tilde{\alpha}_{ab}Sx_b^n + \tilde{\alpha}_{ac}Sx_c^n + \tilde{\beta}_{ab}Sy_b^n + \tilde{\beta}_{ac}Sy_c^n\right]$
      $\varepsilon_a = \varepsilon_a - \frac{1}{4}\left[\tilde{\beta}_{ab}Sx_b^n + \tilde{\beta}_{ac}Sx_c^n + \tilde{\gamma}_{ab}Sy_b^n + \tilde{\gamma}_{ac}Sy_c^n\right]$
  For $i = 1,\ldots,N$, do:
    $Sx_i^{n+1} = \alpha_i^{-1}\delta_i + \beta_i^{-1}\varepsilon_i$
    $Sy_i^{n+1} = \beta_i^{-1}\delta_i + \gamma_i^{-1}\varepsilon_i$
    $\delta_i = Zx_i, \varepsilon_i = Zy_i$

In Figure 2, we show an example of the results of the above algorithm. The values z_i, $i = 1,\ldots,N$, are obtained from the function $.25\,\mathrm{EXP}(-16((x - .5)^2 + (y - .5)^2))$ and the triangulation is that of Figure 1. In practice, we have found that, on the average, about a dozen iterations will yield five or six digits of accuracy. Although they are rare, we have encountered cases that take as many as eighteen and as few as one to obtain this same accuracy.

The final step requires the evaluation of S given by (3.1) on the proper triangle. In order to obtain a perspective plot of the surface, we evaluate the approximation on a uniform rectangular grid. These values are stored in rectangular array S with

$$S(i, j) = S(\bar{x}_i, \bar{y}_j), \qquad i = 1,\ldots,NR, j = 1,\ldots,NC,$$

where

$$\bar{x}_i = XL + (i - 1)\,DX, \qquad \bar{y}_j = YL + (j - 1)\,DY,$$

$$DX = \frac{XH - XL}{NR - 1}, \quad \text{and} \quad DY = \frac{YH - YL}{NC - 1}.$$

We take the approximation to be zero for those points which lie in the display rectangle $[XL, XH] \times [YL, YH]$ but outside D. Rather than stepping through the values of $(\bar{x}_i, \bar{y}_j)$ and asking which triangle these points lie in, our algorithm goes through the list of triangles and computes the values of $S(i, j)$ which are defined by

a given triangle. More precisely,

For $l = 1,\ldots,N_t$, do:
- $a = n_1(l), b = n_2(l), c = n_3(l)$
- $IL = (\min(x_a, x_b, x_c) - XL)/DX$
- $JL = (\min(y_a, y_b, y_c) - YL)/DY$
- $IH = (\max(x_a, x_b, x_c) - XL)/DX$
- $JH = (\max(y_a, y_b, y_c) - YL)/DY$
  - For $i = IL, IL + 1,\ldots,IH$, do:
    - For $j = JL, JL + 1,\ldots,JH$, do:
      - $x = XL + (i - 1)\,DX$
      - $y = YL + (j - 1)\,DY$
      - Solve $x = b_1x_a + b_2x_b + b_3x_c$
        $y = b_2y_a + b_3y_b + b_3y_c$
        $1 = b_1 + b_2 + b_3$
      - to obtain b_1, b_2, b_3
      - If $b_i \geqslant 0, i = 1,2,3$ then $S(i, j) = S_{abc}(x, y)$

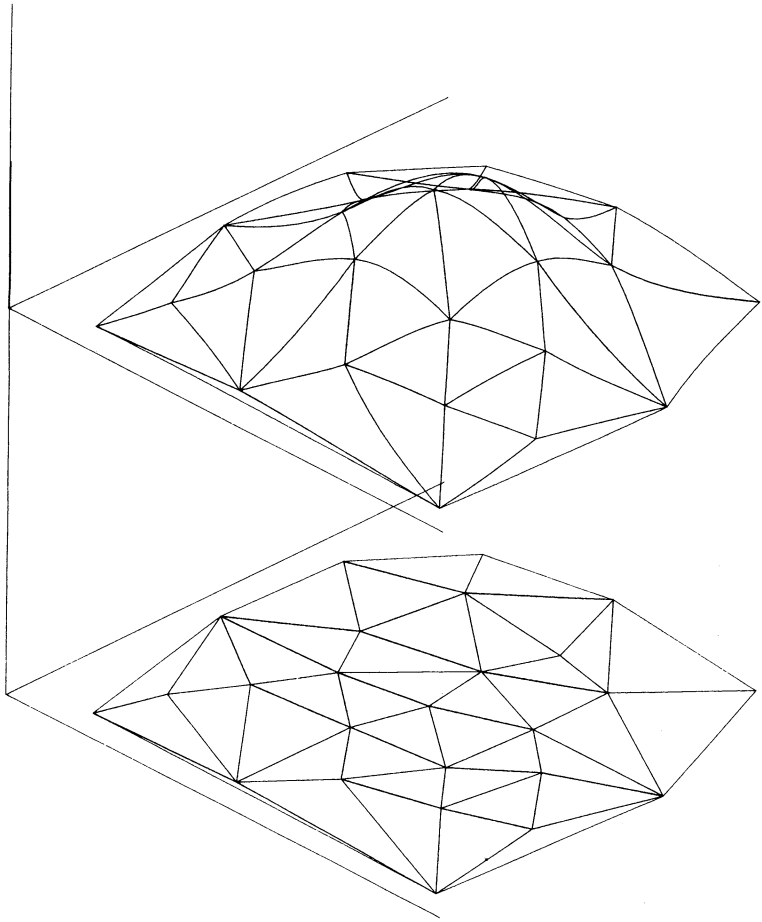


FIGURE 2

In Figure 3a, we show the final interpolant based upon the curve network of Figure 2. For comparison, in Figure 3b we include the interpolant which uses the initial derivatives (Sx_i^0, Sy_i^0), $i = 1,\ldots,N$.

Our next example is comparable to one discussed by Lawson (cf. [7, Figure 7, p. 175]). The values z_i, $i = 1,\ldots,n$, are obtained from the function

$$\mathrm{EXP}\left(-8\left[(x - .5)^2 + (y - .5)^2\right]\right).$$

The 26 data points and a contour plot of the interpolant are shown in Figure 4a. The contours are at the values $S(x, y) = .2, .4, .6$ and .8. A plot of the interpolant using the initial values for the derivatives is shown in Figure 4b.

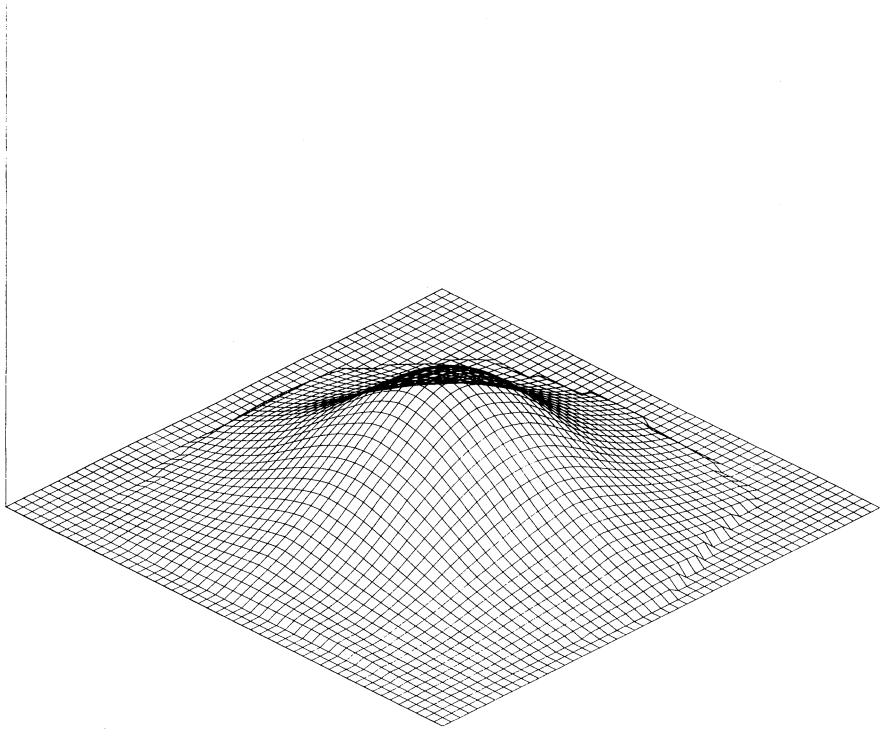


FIGURE 3a

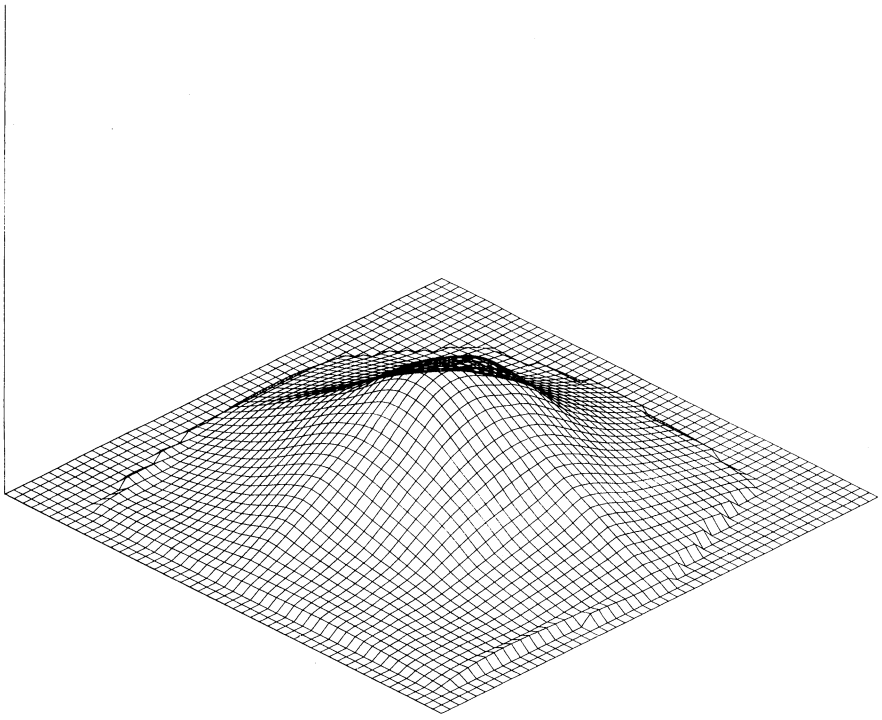


FIGURE 3b

Our third example involves a nonconvex, multiply-connected domain and is included mainly to point out the possibility of using such domains with the present method. Further discussion on the problem of triangulating this type of domain and an algorithm can be found in [8]. In Figure 5a we show the triangulation and the final interpolant. The values z_i, $i = 1,\ldots,N$, are obtained from the same function as used in the previous example. Figure 5b contains a plot of the interpolant using the same data but over a triangulation of the convex hull.

The last example is based upon data provided by the United States Geological Survey [6]. The ordinates represent elevations of a mostly subterranean formation of granite called Hawk Rock which is located in the southeastern desert of Arizona. We found this example particularly interesting because of the unique configuration of the data. Due to the techniques of collection, this data consists of subsets which lie on certain line segments. Figures 6a and 6b show these lines of data along with the triangulation of it. Two views of the interpolant are shown in Figures 6c and 6d.

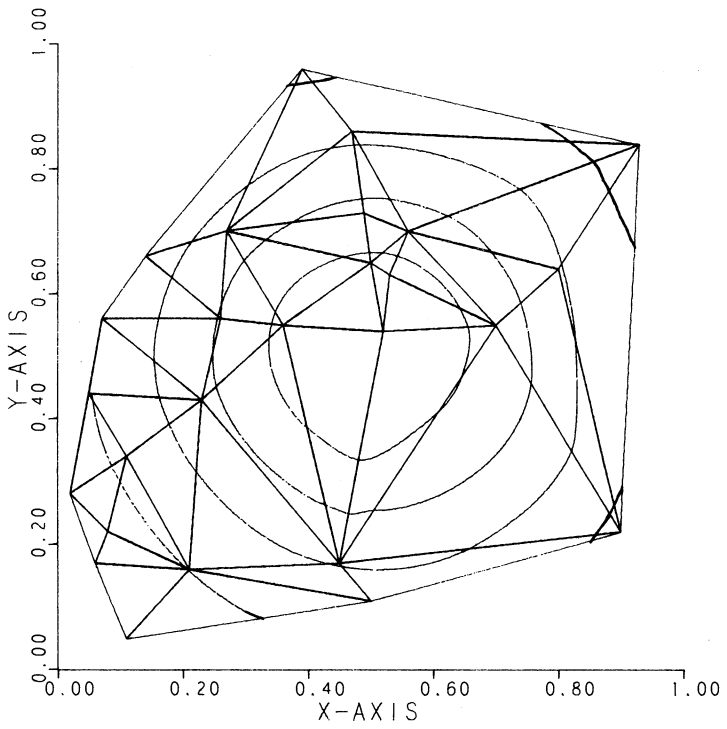


FIGURE 4a

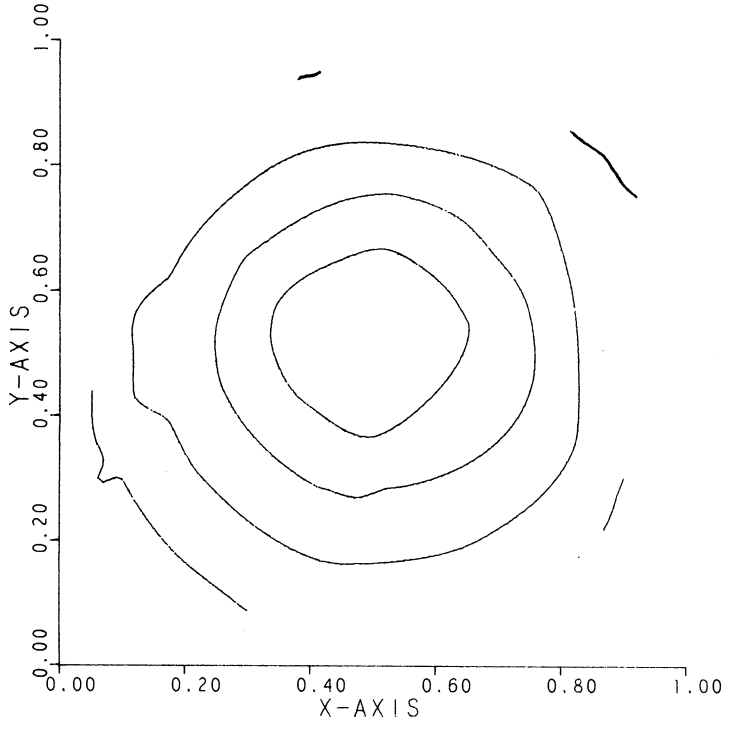


FIGURE 4b

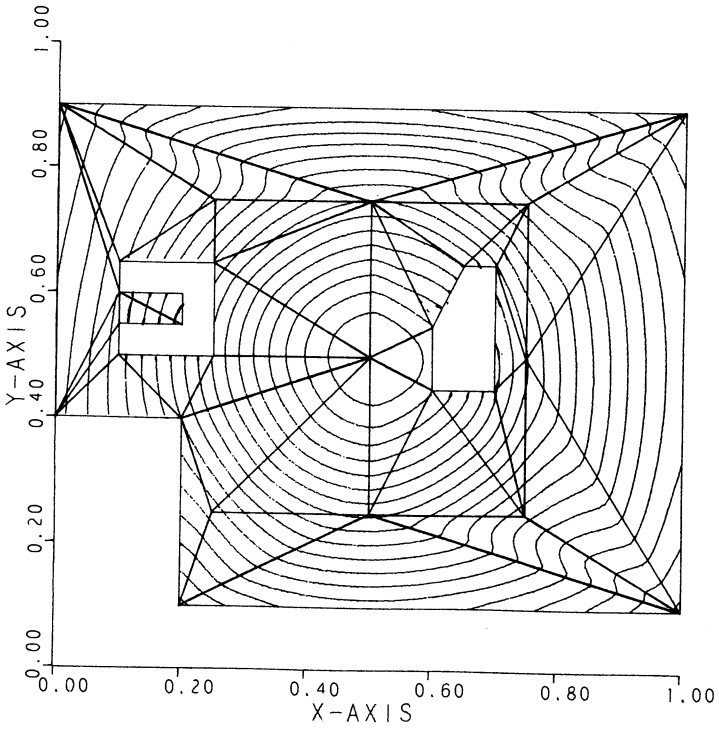


FIGURE 5a

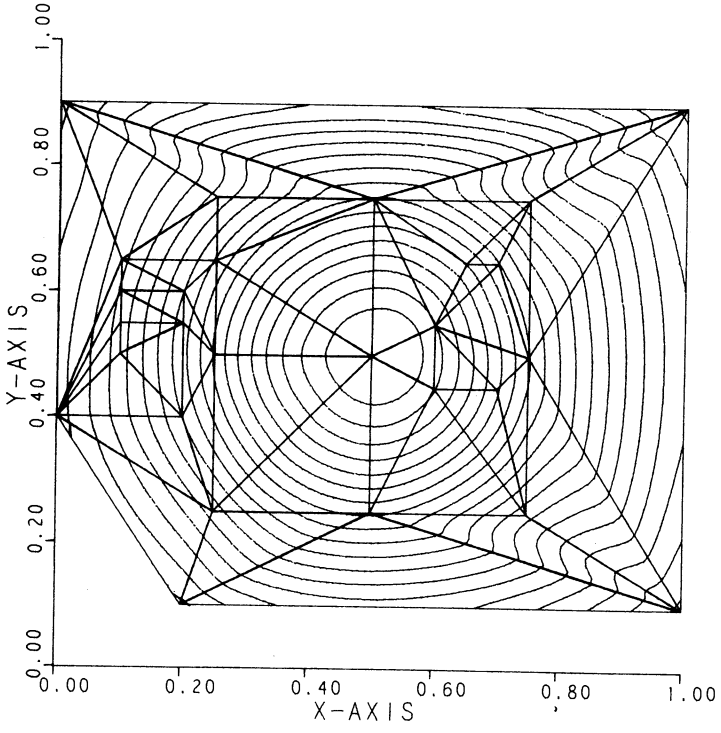


FIGURE 5b

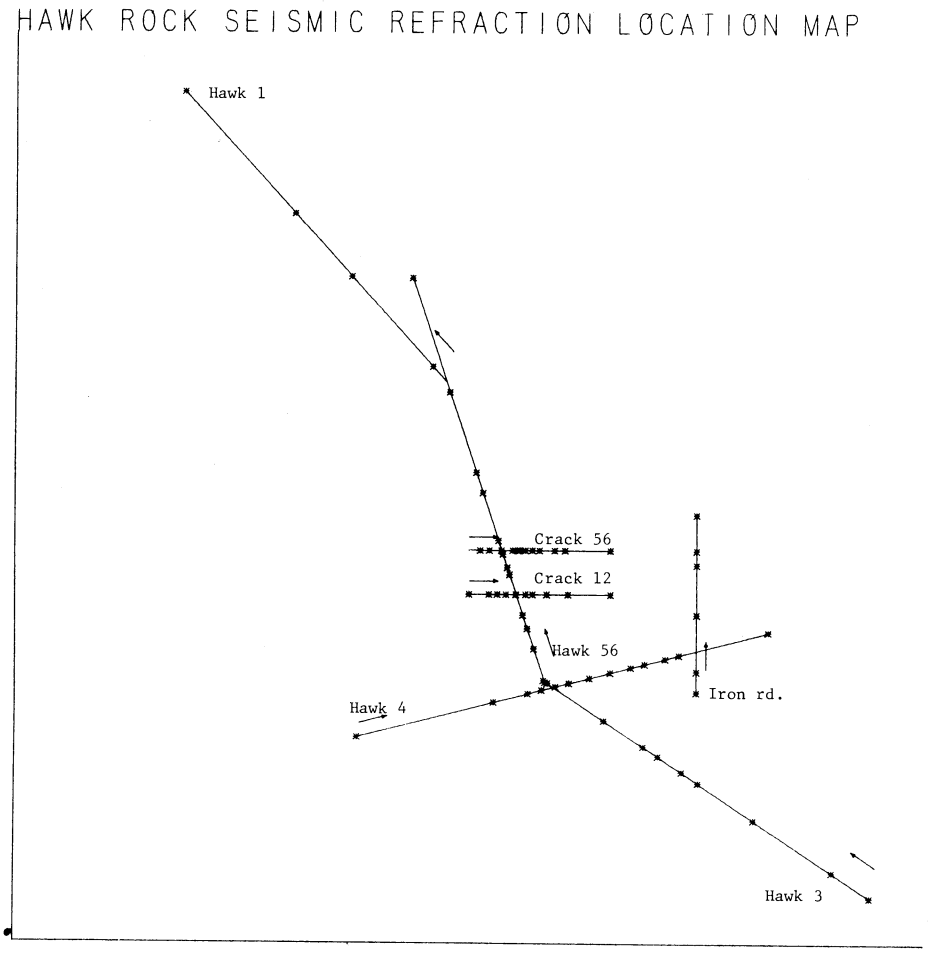

HAWK ROCK SEISMIC REFRACTION LOCATION MAP
Hawk 1
Crack 56
Crack 12
Hawk 56
Iron rd.
Hawk 4
Hawk 3

FIGURE 6a

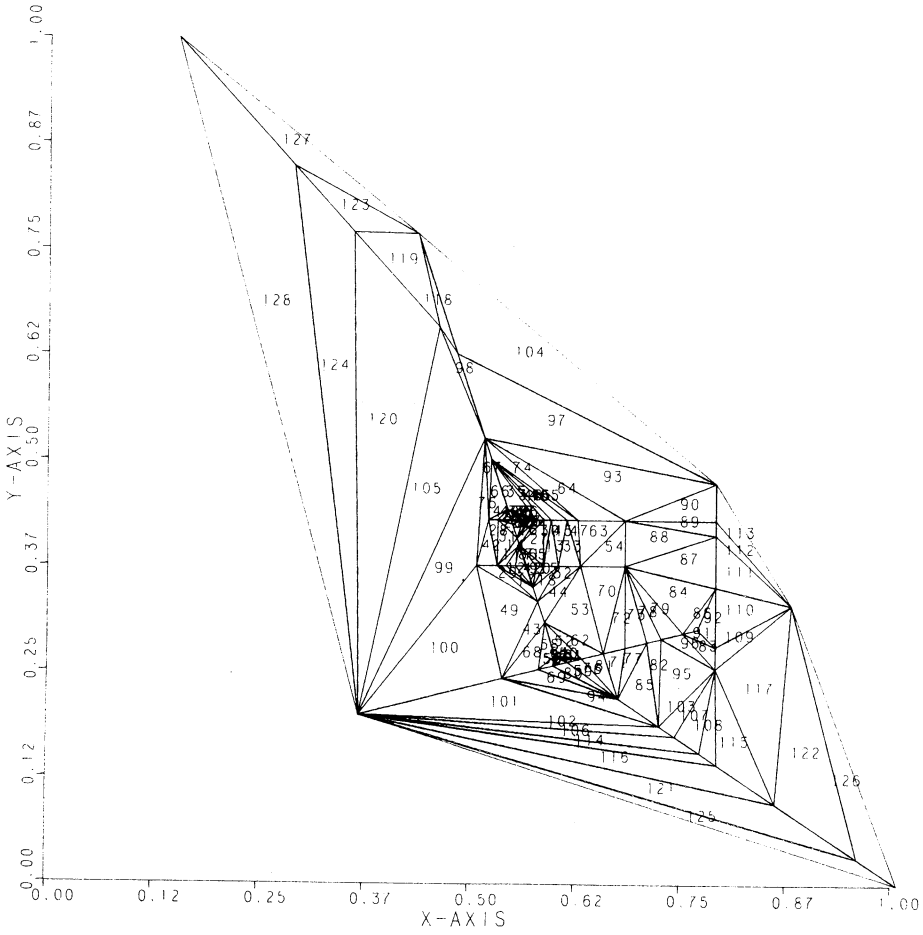

Y-AXIS
0.00
0.12
0.25
0.37
0.50
0.62
0.75
0.87
1.00
X-AXIS
0.00
0.12
0.25
0.37
0.50
0.62
0.75
0.87
1.00

FIGURE 6b

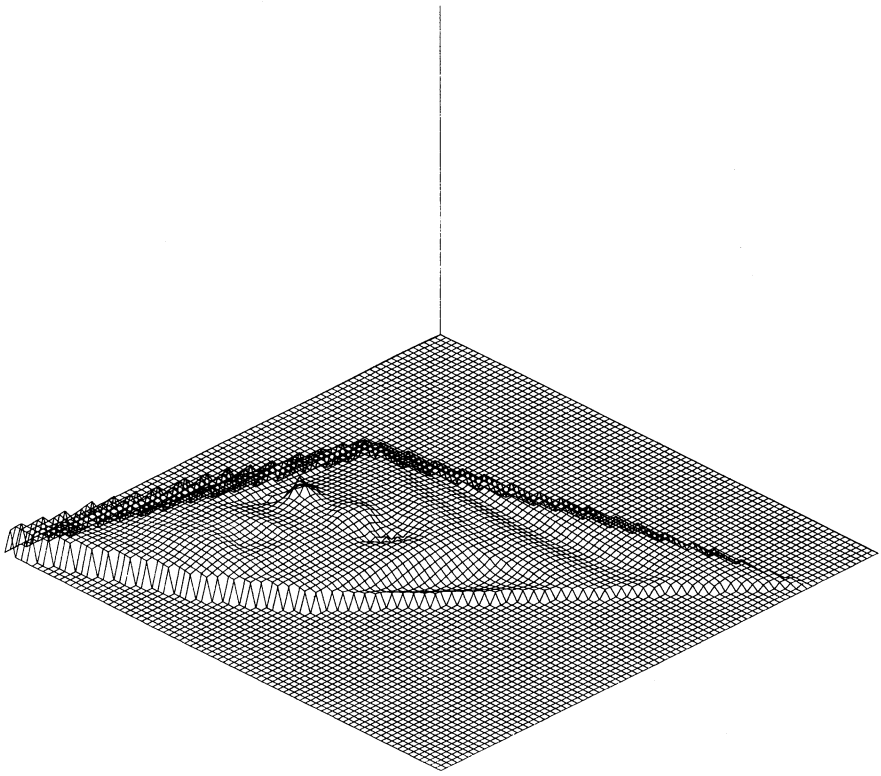


FIGURE 6c

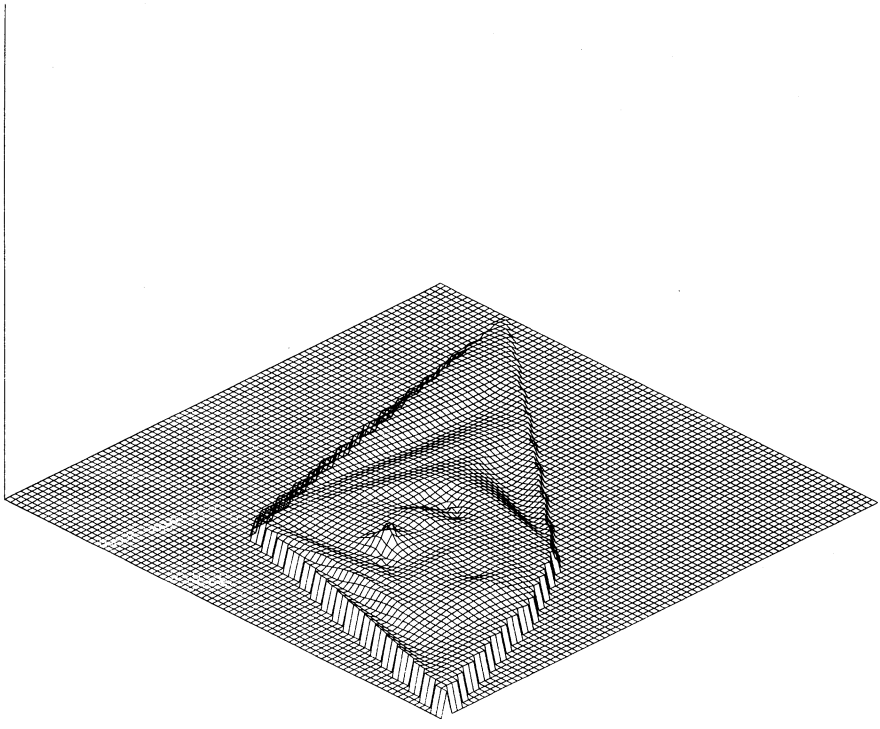


FIGURE 6d

Recently, Franke [5] has reported on the results of a project devoted to the comparison of some 29 methods for interpolating scattered data. Included in this report are the results of our implementation of the present method which is based upon the max-min angle optimization of the triangulation of the convex hull, the algorithm of this section for computing the minimum norm network and the algorithm of this section for evaluating the interpolant on a rectangular grid. In addition to certain assessments of the fitting characteristics of each of the methods, Franke's report includes storage requirements and timing results. The storage requirements tabulated by Franke are given in terms of additional storage required beyond that needed for the data (x_i, y_i, z_i), $i = 1,\ldots,N$, and the output array of evaluations. For the implementation of the present method, this amounts to approximately 32N. Franke's timing results are based on the use of an IBM 360/67. We have run all of our examples on a UNIVAC 1100/42. In Table 2 we give some approximations of the running time (in seconds) required for the three steps of our method. All programs mentioned have been written in FORTRAN.

TABLE 2

N	Triangulation	Minimum Norm Network	Evaluation 40×40 Grid	Evaluation 80×80 Grid
25	.10	.88	.35	1.16
50	.35	2.14	.85	2.88
100	1.24	4.18	1.64	5.56
200	4.65	7.41	3.21	11.32

Department of Mathematics
Arizona State University
Tempe, Arizona 85281

1. HIROSHI AKIMA, "A method of bivariate interpolation and smooth surface fitting for values given at irregularly distributed points," *Trans. Math. Software*, v. 4, 1978, pp. 148–159.

2. R. E. BARNHILL, "Representation and approximation of surfaces," *Mathematical Software* III (J. R. Rice, ed.), Academic Press, New York, 1977, pp. 68–119.

3. R. E. BARNHILL, G. BIRKHOFF & W. J. GORDON, "Smooth interpolation in triangles," *J. Approx. Theory*, v. 8, 1973, pp. 114–128.

4. C. DE BOOR & R. E. LYNCH, "On splines and their minimum properties," *J. Math. Mech.*, v. 15, 1966, pp. 953–969.

5. RICHARD FRANKE, *A Critical Comparison of Some Methods for Interpolation of Scattered Data*, Naval Postgraduate School Technical Report NPS-53-79-003, December 1979.

6. R. L. LANEY, L. W. PANKRATZ & C. ZENONE, Private Communication, Phoenix, Arizona, 1978.

7. C. L. LAWSON, "Software for C^1 surface interpolation," *Mathematical Software* III (J. R. Rice, ed.), Academic Press, New York, 1977, pp. 161–194.

8. B. A. LEWIS & J. S. ROBINSON, "Triangulation of planar regions with applications," *Comput. J.*, v. 21, 1978, pp. 324–332.

9. G. M. NIELSON, "Minimum norm interpolation in triangles," *SIAM J. Numer. Anal.*, v. 17, 1980, pp. 44–62.

Multivariate Smoothing and Interpolating Splines

Gregory M. Nielson

SIAM Journal on Numerical Analysis, Vol. 11, No. 2 (Apr., 1974), 435-446.

Stable URL:
http://links.jstor.org/sici?sici=0036-1429%28197404%2911%3A2%3C435%3AMSAIS%3E2.0.CO%3B2-V

SIAM Journal on Numerical Analysis is currently published by Society for Industrial and Applied Mathematics.

SIAM J. NUMER. ANAL.
Vol. 11, No. 2, April 1974

MULTIVARIATE SMOOTHING AND INTERPOLATING SPLINES*

GREGORY M. NIELSON†

Abstract. A theorem that characterizes spline functions that both smooth and interpolate is given. A bivariate generalization is presented which permits interpolation and smoothing of information which is not necessarily on a rectangular grid. A theorem which involves reproducing kernels for Hilbert spaces unifies this theory.

1. Introduction. The subject of this report is the approximation of functions given certain linear functional values. The approximating function ϕ is chosen from a space of functions on the basis that ϕ is "smooth" and that ϕ interpolates or approximates a function relative to a set of linear functionals.

Both the concepts of smoothing and interpolation are combined into one variational problem. The problem is used to characterize what we will call spline functions. The problem and its solution include as special cases the work of Reinsch [13], Schoenberg [16], Handscomb [8], and Anselone and Laurent [3].

With the Taylor-type expansion with remainders given by Sard [15], we are able to pose and solve a variational problem for bivariate functions. This characterises bivariate spline functions and permits the interpolation and smoothing of data for bivariate functions which do not necessarily lie on a rectangular grid. This latter restriction is present for the bicubic splines of Ahlberg, Nilson and Walsh [1], and de Boor [6]. Mansfield [10], [11] has previously constructed the reproducing kernel for the Hilbert space $T^{p,q}(\alpha, \beta)$ which has Taylor expansions similar to the Sard space $B_{p,q}(\alpha, \beta)$. She has used this reproducing kernel to characterize interpolating spline functions and optimal approximation of linear functionals.

The approach that we take for both univariate and bivariate characterizations can be generalized. A theorem involving reproducing kernels for Hilbert spaces that unifies this theory is given in § 4.

2. Univariate smoothing and interpolating splines. We define the space of functions $H^n[a, b]$ to be all functions f such that $f^{(n-1)}$ is absolutely continuous on $[a, b]$ and $f^{(n)} \in L^2[a, b]$. By $F^n[a, b]$ we mean the space of linear functionals which have the form

$$L(f) = \sum_{j=0}^{n-1} \int_a^b f^{(j)}(t)\, d\mu^j(t),$$

where each μ^j is a function of bounded variation on $[a, b]$.

THEOREM 1. *Suppose* $\{L_j\}_{i=1}^N \subseteq F^m[a, b]$ *and let real numbers* $Y_i, i = 1, \cdots, N$, *be given. Choose* $0 \leqq k \leqq N$ *and let* W *be a symmetric* $k \times k$ *positive definite matrix. If*

(i) $\{1, t, \cdots, t^{m-1}, L_1(x - t)_+^{2m-1}, \cdots, L_N(x - t)_+^{2m-1}\}$ *is a linearly independent set and*

(ii) $L_i(p) = 0, i = 1, \cdots, N$, *implies* $p = 0$ *for all* $p \in \mathscr{P}_{m-1}$,

* Received by the editors October 12, 1971, and in revised form August 2, 1972.

† Department of Mathematics, Arizona State University, Tempe, Arizona 85281.

then the quantity

$$\sigma(\psi) = \int_a^b [\psi^{(m)}(t)]^2\,dt + \sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}[L_i(\psi) - Y_i][L_j(\psi) - Y_j]$$

is minimized, subject to

$$L_j(\psi) = Y_j, \quad \psi \in H^m[a, b], \quad j = k+1, \cdots, N,$$

uniquely by

$$\phi(t) = \sum_{j=1}^{m} a_j t^{j-1} + \sum_{i=1}^{N} \lambda_i L_{i_{(x)}}((x-t)_+^{2m-1}),$$

where the coefficients of ϕ are determined by

$$\sum_{i=1}^{N} L_i(t^{j-1})\lambda_i = 0, \qquad j = 1, \cdots, m,$$

$$\lambda_i = \frac{(-1)^{m+1}}{(2m-1)!}\sum_{j=1}^{k} w_{ij}[L_j(\phi) - Y_j], \qquad i = 1, \cdots, k, \tag{1}$$

$$L_j(\phi) = Y_j, \qquad j = k+1, \cdots, N.$$

Proof. We will not include the existence and uniqueness part of the proof here since this theorem is a special case of Theorem 3 given in § 4. Sard [15, p. 13] has shown that if $f \in H^m[a, b]$ and $L \in F^m[a, b]$ then

$$L(f) = \sum_{j=0}^{m-1} f^{(j)}(a)L\left(\frac{(x-a)^j}{j!}\right) + \int_a^b L\left(\frac{(x-t)_+^{m-1}}{(m-1)!}\right) f^{(m)}(t)\,dt, \tag{2}$$

where

$$(z)_+ = \begin{cases} z, & z \geqq 0, \\ 0, & z \leqq 0. \end{cases}$$

Let $[f, g] \equiv \int_a^b f^{(m)}(t)g^{(m)}(t)\,dt$ and let $\psi \in H^m[a, b]$ such that $L_i(\psi) = Y_i$, $i = k+1, \cdots, N$. Then

$$\begin{aligned} \sigma(\psi) - \sigma(\phi) = {} & [\psi - \phi, \psi - \phi] + \sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}L_i(\psi - \phi)L_j(\psi - \phi) \\ & + 2[\psi - \phi, \phi] \\ & + 2\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}L_i(\psi - \phi)[L_j(\phi) - Y_j]. \end{aligned} \tag{3}$$

We will now show that the last two terms of (3) sum to zero:

$$\begin{aligned} [\psi - \phi, \phi] &= \left[\psi - \phi, \sum_{j=0}^{m-1} a_i t^j + \sum_{i=1}^{N} \lambda_i L_i((x-t)_+^{2m-1})\right] \\ &= \sum_{i=1}^{N} \lambda_i[\psi - \phi, L_i((x-t)_+^{2m-1})] \\ &= \left\{\sum_{i=1}^{N} \lambda_i L_i(\psi - \phi) - \sum_{j=0}^{m-1} (\psi - \phi)^{(j)}(a) \sum_{i=1}^{N} \lambda_i L_i\left(\frac{(x-a)^j}{j!}\right)\right\} \\ &\qquad \cdot (2m-1)!_{(-1)^m} \\ &= (-1)^m(2m-1)! \sum_{i=1}^{k} \lambda_i L_i(\psi - \phi). \end{aligned} \tag{4}$$

We have used (2), the first set of equations in (1), and the assumption that $L_i(\psi) = Y_i, i = k + 1, \cdots, N$. We now use (4) to obtain

$$[\psi - \phi, \phi] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} L_i(\psi - \phi)[L_j(\phi) - Y_j]$$

$$= \sum_{i=1}^{k} L_i(\psi - \phi)\left[(-1)^m(2m-1)!\lambda_i + \sum_{j=1}^{k} w_{ij}(L_j(\phi) - Y_j)\right],$$

which is zero by (1). Thus

$$\sigma(\psi) - \sigma(\phi) = [\psi - \phi, \psi - \phi] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij}(\psi - \phi) L_j(\psi - \phi) \geqq 0,$$

which implies that if ϕ exists, it does minimize σ subject to the interpolation conditions.

The following example illustrates the combination of both smoothing and interpolating possible by Theorem 1. For comparison, we have computed the interpolating spline for the very same data.

Example 1. We choose $m = 2$, $N = 6$, $k = 5$, and

$$\begin{aligned}
L_1(f) &= f(0), & Y_1 &= 0.40,\\
L_2(f) &= f(0.5), & Y_2 &= 0.35,\\
L_3(f) &= f(1.0), & Y_3 &= 0.24,\\
L_4(f) &= f(1.5), & Y_4 &= 0.13,\\
L_5(f) &= f(2.0), & Y_5 &= 0.00,\\
L_6(f) &= \int_0^2 f(t)\,dt, & Y_6 &= 0.5.
\end{aligned}$$

Table 1 gives the coefficients for one selection of weights, and Fig. 1 shows the graphs of η, the interpolating spline, and ϕ, the combined smoothing and interpolating spline.

TABLE 1

Weight	Smoothing Spline ϕ	Interpolating Spline η
$w_{1,1} = 100$	$a_1 = 0.603$	$a_1 = 0.126\,(1)$
$w_{2,2} = 50$	$a_2 = 0.295$	$a_2 = 0.630$
$w_{3,3} = 75$	$\lambda_1 = 0.175$	$\lambda_1 = 0.406\,(1)$
$w_{4,4} = 50$	$\lambda_2 = 0.889\,(-1)$	$\lambda_2 = 0.111\,(2)$
$w_{5,5} = 60$	$\lambda_3 = 0.325$	$\lambda_3 = 0.956\,(1)$
$w_{i,j} = 0, i \neq j$	$\lambda_4 = 0.176$	$\lambda_4 = 0.113\,(2)$
	$\lambda_5 = 0.132$	$\lambda_5 = 0.200\,(2)$
	$\lambda_6 = 0.449$	$\lambda_6 = 0.200\,(2)$

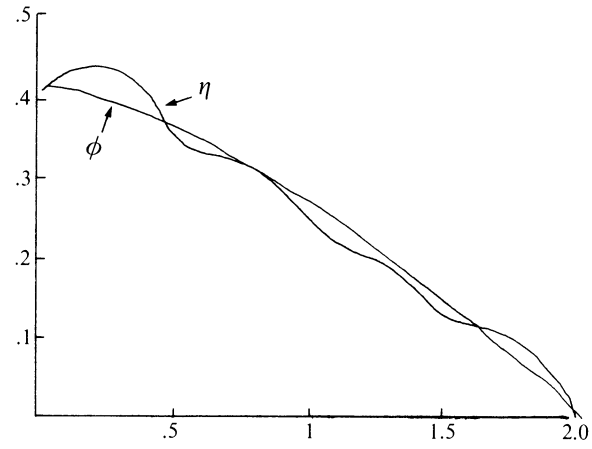


FIG. 1

3. A case of bivariate smoothing and interpolating splines. Let $R = \{(x, y): a \leqq x \leqq b \text{ and } c \leqq y \leqq d\}$ and $(\alpha, \beta) \in R$. Let p and q be given nonnegative integers and $m = p + q$. We define the space $H_{p,q}$, which has properties similar to the Sard spaces $B_{p,q}$ [15, pp. 160, 172], to be all functions f of the form

$$
\begin{aligned}
f(x, y) = & \sum_{i+j<m} \frac{(x-\alpha)^i}{i!} \frac{(y-\beta)^j}{j!} f^{i,j} \\
& + \sum_{j<q} \frac{(y-\beta)^j}{j!} \int_\alpha^x \frac{(x-s)^{m-j-1}}{(m-j-1)!} f^{m-j,j}(s)\, ds \\
& + \sum_{i<p} \frac{(x-\alpha)^i}{i!} \int_\beta^y \frac{(y-t)^{m-i-1}}{(m-i-1)!} f^{i,m-i}(t)\, dt \\
& + \int_\alpha^x \int_\beta^y \frac{(x-s)^{p-1}}{(p-1)!} \frac{(y-t)^{q-1}}{(q-1)!} f^{p,q}(s, t)\, ds\, dt,
\end{aligned} \tag{5}
$$

where

$$
\begin{aligned}
& f^{i,j} \text{ are constants}, && i + j < m; \\
& f^{m-j,j} \in L^2[a, b], && j < q; \\
& f^{i,m-i} \in L^2[c, d], && i < p; \\
& f^{p,q} \in L^2[R].
\end{aligned}
$$

We can conclude from the fundamental theorem of calculus for one- and two- dimensional Lebesgue measure that

$$
\begin{aligned}
& f_{i,j}(a, b) = f^{i,j}, \quad i + j < m; \\
& f_{m-j,j}(s, \beta) = f^{m-j,j} \quad \text{a.e.}, \quad j < q; \\
& f_{i,m-i}(\alpha, t) = f^{i,m-i} \quad \text{a.e.}, \quad i < p; \\
& f_{p,q} = f^{p,q} \quad \text{a.e.};
\end{aligned}
$$

where

$$f_{i,j} = D_t^{j-j_0} D_s^i D_t^{j_0} f, \qquad j_0 = \min(j, q).$$

With the bilinear form

$$[f, g] = \sum_{j<q} \int_a^b f_{m-j,j}(s, \beta) g_{m-j,j}(s, \beta)\, ds + \sum_{i<p} \int_c^d f_{i,m-i}(\alpha, t) g_{i,m-i}(\alpha, t)\, dt + \int_a^b \int_c^d f_{p,q}(s, t) g_{p,q}(s, t)\, ds\, dt$$

defined on $H_{p,q}$, we have a situation analagous to that of § 2.

We now wish to construct a function analogous to $(x - t)_+^{2m-1}/(2m - 1)!$.

This "pseudo-reproducing kernel function" must be a function K defined on $R \times R$ such that:

(6a) for each fixed point $(x, y) \in R$, $K(s, t; x, y) \in H_{p,q}$ as a function of (s, t);

(6b) $f(x, y) - [f(s, t), K(s, t; x, y)]_{(s,t)} \in \Pi_{m-1} = \left\{ g : g(s, t) = \sum_{i+j<m} C_{i,j} s^i t^i \right\}$,

the null space of $[\cdot, \cdot]$ in $H_{p,q}$.

For a fixed $(x, y) \in R$ the function

$$\begin{aligned} K(s, t; x, y) = {} & \sum_{j<q} \frac{(t - \beta)^j}{j!} \int_\alpha^s \frac{(s - u)^{m-j-1}}{(m - j - 1)!} \left(\frac{(x - u)^{m-j-1}}{(m - j - 1)!} \psi(\alpha, u, x) \frac{(y - \beta)^j}{j!} \right) du \\ & + \sum_{i<p} \frac{(s - \alpha)^i}{i!} \int_\beta^t \frac{(t - v)^{m-i-1}}{(m - i - 1)!} \left(\frac{(x - \alpha)^i}{i!} \frac{(y - v)^{m-i-1}}{(m - i - 1)!} \psi(\beta, v, y) \right) du \\ & + \int_\alpha^s \int_\beta^t \frac{(s - u)^{p-1}}{(p - 1)!} \frac{(t - v)^{q-1}}{(q - 1)!} \\ & \cdot \left(\frac{(x - u)^{p-1}}{(p - 1)!} \psi(\alpha, u, x) \frac{(y - v)^{q-1}}{(q - 1)!} \psi(\beta, v, y) \right) du\, dv, \end{aligned} \tag{7}$$

where

$$\psi(x, y, z) = \begin{cases} 1, & x \leqq y < z, \\ -1, & z \leqq y < x, \\ 0 & \text{otherwise}, \end{cases}$$

is in $H_{p,q}$ as a function of (s, t) and

$$\begin{aligned} & [f(s, t), K(s, t; x, y)]_{(s,t)} \\ & = \sum_{i<q} \int_a^b f_{m-j,j}(u, \beta) \left[\frac{(x - u)^{m-j-1}}{(m - j - 1)!} \psi(\alpha, u, x) \frac{(y - \beta)^j}{j!} \right] du + \text{dual} \\ & \quad + \int_a^b \int_c^d f_{p,q}(u, v) \left[\frac{(x - u)^{p-1}}{(p - 1)!} \psi(\alpha, u, x) \frac{(y - v)^{q-1}}{(q - 1)!} \psi(\beta, v, y) \right] du\, dv \\ & = f(x, y) - \sum_{i+j<m} \frac{(x - \alpha)^i}{i!} \frac{(y - \beta)^j}{j!} f_{i,j}(\alpha, \beta). \end{aligned}$$

The function K of (7) was first constructed by Mansfield [10], [11]. She also gives a somewhat simpler form of K by carrying out the integrations. Since each of the integrals is of the same form, the formula

$$g^k(\alpha, s, x) = \int_\gamma^\delta \frac{(s-u)^{k-1}}{(k-1)!}\psi(\alpha, u, s)\frac{(x-u)^{k-1}}{(k-1)!}\psi(\alpha, u, x)\,du$$

$$= \begin{cases} (s-a)^k \sum_{j=0}^{k-1} \frac{(\alpha-s)^j}{(k+j)!}\frac{(x-\alpha)^{k-1-j}}{(k-1-j)!}, & \alpha \leqq s \leqq x, \\ 0, & s \leqq \alpha \leqq x, \end{cases}$$

along with the fact that $g^k(\alpha, s, x) = g^k(\alpha, x, s)$ and $g^k(-\alpha, -s, -x) = -g^k(\alpha, s, x)$, can be used to eliminate the integrals.

If the integrals of (7) are replaced by integrals over $[a, b]$ and $[c, d]$ with the use of ψ, it is easy to see that $K(s, t; x, y) = K(x, y; s, t)$. This symmetry is not necessary; for example, in the case where $(\alpha, \beta) = (a, c)$ the following function can be used:

$$\sum_{i<p} \frac{(s-a)^i}{i!}\frac{(x-a)^i}{i!}\frac{(t-y)_+^{2(m-i)-1}}{(2(m-i)-1)!} + \sum_{j<q} \frac{(t-b)^j}{j!}\frac{(y-b)^j}{j!}\frac{(s-x)_+^{2)(m-j)-1}}{(2(m-j)-1)!}$$

$$+ g^p(a, s, x)g^q(b, t, y).$$

As a choice of a set of linear functionals analogous to F^m we can take $F_{p,q}$ which consists of all functionals L of the form

$$L(f) = \sum_{\substack{i<p \\ j<q}} \int_a^b \int_c^d f_{i,j}(s, t)\,d\mu^{i,j}(s, t)$$

$$+ \sum_{\substack{i+j<m \\ i \geqq p}} \int_a^b f_{i,j}(s, \beta)\,d\mu^{i,j}(s)$$

$$+ \sum_{\substack{i+j<m \\ j \geqq q}} \int_c^d f_{i,j}(\alpha, t)\,d\mu^{i,j}(t),$$

where each $\mu^{i,j}$ is of bounded variation on its domain [15, p. 524].

For $L \in F_{p,q}$ and $f \in \mathbf{B}_{p,q}$, Sard [15, p. 175] has proved that

$$L(f) = \sum_{i+j<m} C^{i,j} f_{i,j}(\alpha, \beta)$$

$$+ \sum_{j<q} \int_a^b f_{m-j,j}(u, \beta)k^{m-j,j}(u)\,du$$

$$+ \sum_{i<p} \int_a^b f_{i,m-1}(\alpha, v)k^{i,m-i}(v)\,dv$$

$$+ \int_a^b \int_c^d f_{p,q}(u, v)k^{p,q}(u, v)\,du\,dv,$$

where

$$C^{i,j} = L\left[\frac{(x-\alpha)^i}{i!}\frac{(y-\beta)^j}{j!}\right]$$

and the kernel functions, which are of bounded variations, can be taken so that

$$k^{m-j,j}(u) = L_{(x,y)}\left[\frac{(x-u)^{m-j-1}}{(m-j-1)!}\psi(\alpha,u,x)\frac{(y-\beta)^j}{j!}\right] \quad \text{a.e.,} \quad j<q;$$

$$k^{i,m-i}(v) = L_{(x,y)}\left[\frac{(x-\alpha)^i}{i!}\frac{(y-v)^{m-i-1}}{(m-i-1)!}\psi(\beta,v,y)\right] \quad \text{a.e.,} \quad i<p; \tag{8}$$

$$k^{p,q}(u,v) = L_{(x,y)}\left[\frac{(x-u)^{p-1}}{(p-1)!}\psi(\alpha,u,x)\frac{(y-v)^{q-1}}{(q-1)!}\psi(\beta,v,y)\right] \quad \text{a.e.}$$

Since $B_{p,q} \subset H_{p,q} \subset \mathbf{B}_{p,q}$ we can apply the Sard kernel theorem to $K(s,t;x,y)$ as a function of (x,y) and obtain

$$\begin{aligned}
&L_{(x,y)}(K(s,t;x,y)) \\
&\quad = \sum_{j<q}\int_a^b \frac{(s-u)^{m-j-1}}{(m-j-1)!}\psi(\alpha,u,s)\frac{(t-\beta)^j}{j!}k^{m-j,j}(u)\,du \\
&\qquad + \sum_{i<p}\int_c^d \frac{(s-\alpha)^i}{i!}\frac{(t-v)^{m-i-1}}{(m-i-1)!}\psi(\beta,v,t)k^{i,m-i}(v)\,dv \\
&\qquad + \int_a^b\int_c^d \frac{(s-u)^{p-1}}{(p-1)!}\psi(\alpha,u,s)\frac{(t-v)^{q-1}}{(q-1)!}\psi(\beta,v,t)k^{p,q}(u,v)\,du\,dv \qquad (9)\\
&\quad = \sum_{j<q}\frac{(t-\beta)^j}{j!}\int_\alpha^s \frac{(s-u)^{m-j-1}}{(m-j-1)!}k^{m-j,j}(u)\,du \\
&\qquad + \sum_{i<p}\frac{(s-\alpha)^i}{i!}\int_\beta^t \frac{(t-v)^{m-i-1}}{(m-i-1)!}k^{i,m-j}(v)\,dv \\
&\qquad + \int_\alpha^s\int_\beta^t \frac{(s-u)^{p-1}}{(p-1)!}\frac{(t-v)^{q-1}}{(q-1)!}k^{p,q}(u,v)\,du\,dv,
\end{aligned}$$

where the kernel functions $k^{i,j}$ are given by (8). From (9) and the fact that the kernels are of bounded variation it is clear that $L_{(x,y)}(K(s,t;x,y))$ is an element of $H_{p,q}$ as a function of (s,t).

Using the Sard kernel theorem we can see that

$$\begin{aligned}
&[f(s,t), L_{(x,y)}K(s,t;x,y)](s,t) \\
&\qquad = \sum_{j<q}\int_a^b f_{m-j,j}(u,\beta)k^{m-j,j}(u)\,du \\
&\qquad\quad + \sum_{i<p}\int_c^d f_{i,m-i}(\alpha,v)\,k^{i,m-i}(v)\,dv \\
&\qquad\quad + \int_a^b\int_c^d f_{p,q}(u,v)k^{p,q}(u,v)\,du\,dv
\end{aligned}$$

$$= L_{(x,y)}(f(x, y)) - \sum_{i+j<m} f_{i,j}(\alpha, \beta) L_{(x,y)}\left[\frac{(x-\alpha)^i}{i!}\frac{(y-\beta)^j}{j!}\right]$$

$$= L_{(x,y)}\left[f(x, y) - \sum_{i+j<m} f_{i,j}(\alpha, \beta)\frac{(x-\alpha)^i}{i!}\frac{(y-\beta)^j}{j!}\right]$$

$$= L_{(x,y)}([f(s, t), K(s, t; x, y)]_{(s,t)}).$$

We now give a bivariate generalization of Theorem 1. Since this theorem is a special case of Theorem 3 we will not include the proof.

THEOREM 2. *Suppose* $\{L_i\}_{i=1}^N \subset K_{p,q}$ *and let the real numbers* Y_i, $i = 1, \cdots, N$, *be given. Choose* $0 \leqq k \leqq N$ *and let* W *be a* $k \times k$ *symmetric positive definite matrix. If*

(i) $\{L_{i(x,y)}[K(s, t; x, y)], i = 1, \cdots, N; s^i t^i, i + j < m\}$ *is a linearly independent set and*

(ii) $L_i(g) = 0, i = 1, \cdots, N,$ *implies* $g = 0$ *for all* $p \in \Pi_{m-1}$ *where*

$$\Pi_{m-1} = \{p : p(s, t) = \sum_{i+j<m} a^{i,j} s^i t^j\},$$

then

$$\sigma(\psi) = [\psi, \psi] + \sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij}(L_i(\psi) - Y_i)(L_j(\psi) - Y_j)$$

is uniquely minimized over $H_{p,q}$; *subject to*

$$L_j(\phi) = Y_j, \qquad j = k + 1, \cdots, N;$$

by

$$\phi(s, t) = \sum_{i+j<m} a^{i,j} s^i t^j + \sum_{i=1}^{N} \lambda_i L_i K(x, y; s, t),$$

where the coefficients of ϕ *are determined by the conditions*

$$\sum_{i=1}^{N} \lambda_i L_i(x^n y^j) = 0, \qquad n + j < m,$$

$$\lambda_i = \sum_{j=1}^{K} w_{ij}(Y_j - L_j(\phi)), \qquad i = 1, \cdots, k,$$

$$L_j(\phi) = Y_j, \qquad j = k + 1, \cdots, N.$$

4. General smoothing and interpolation splines. The previous two sections motivate the following theorem which unifies this theory.

THEOREM 3.

(a) *Let* $[\cdot, \cdot]$ *be an inner product (not necessarily definite) defined on the linear space of functions* H *with domain* D.

(b) *Let* $\{g_i\}_{i=1}^m$ *be a basis for* $N = \{g \in H : [g, g] = 0\}$.

(c) *Let* K *be a function defined on* $D \times D$ *such that*

(i) *for each fixed* $y \in D$, $K(x, y) \in H$ *as a function of* x;

(ii) $f(y) - [f(x), K(x, y)]_{(x)} \in N$.

(d) *Let* F *be a set of linear functionals defined on* H *such that*

(i) $L_{(y)}[K(x, y)] \in H$ *as a function of* x *for all* $L \in F$;

(ii) $L_{(y)}([f(x), K(x, y)]_{(x)}) = [f(x), L_{(y)}(K(x, y))]_{(x)}$ *for all* $L \in F$.

Suppose $\{L_i\}_{i=1}^N \subset F$ *and let the real numbers* Y_i, $i = 1, \cdots, N$, *be given. Choose* k *such that* $0 \leqq k \leqq N$ *and let* W *be a* $k \times k$ *symmetric positive definite matrix. If*

(i) $\{L_{i(y)}[K(x, y)], i = 1, \cdots, N; g_i(x), i = 1, \cdots, m\}$ *is a linearly independent set, and*

(ii) $L_i(g) = 0, i = 1, \cdots, N$, *implies* $g = 0$ *for all* $g \in N$,

then

$$\sigma(\psi) \equiv [\psi, \psi] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij}(L_i(\psi) - Y_i)(L_j(\psi) - Y_j)$$

is uniquely minimized over H, *subject to*

$$L_j(\psi) = Y_i, \qquad i = k + 1, \cdots, N,$$

by

$$\phi(x) = \sum_{i=1}^{m} a_i g_i(x) + \sum_{i=1}^{N} \lambda_i L_{i(y)}[K(x, y)],$$

where the coefficients of ϕ *are determined by*

$$\sum_{i=1}^{N} \lambda_i L_i(g_j) = 0, \qquad j = 1, \cdots, m,$$

$$\lambda_i = \sum_{j=1}^{k} w_{ij}(Y_j - L_j(\phi)), \qquad i = 1, \cdots, k, \tag{10}$$

$$L_j(\phi) = Y_j, \qquad j = k + 1, \cdots, N.$$

Proof. Minimal variation property:

Let $\psi \in H$ such that $L_i(\phi) = Y_i$, $i = k + 1, \cdots, N$,

$$\sigma(\psi) - \sigma(\phi) = [\psi - \phi, \psi - \phi] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} L_i(\psi - \phi) L_j(\psi - \phi)$$

$$+ 2[\psi - \phi, \phi] \tag{11}$$

$$+ 2 \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} L_i(\psi - \phi)(L_j(\phi) - Y_j).$$

We will now show (as we did with Theorem 1) that the last two terms of (11) sum to zero:

$$
\begin{aligned}
[\psi - \phi, \phi] &= [\psi - \phi, \sum_{j=1}^{m} a_j g_j + \sum_{i=1}^{N} \lambda_i L_{i(y)}(K(x, y))]_{(x)} \\
&= \sum_{i=1}^{N} \lambda_i [\psi - \phi, L_{i(y)}(K(x, y))]_{(x)} \\
&= \sum_{i=1}^{N} \lambda_i L_i(\psi - \phi) - \sum_{j=1}^{m} \alpha_j \sum_{i=1}^{N} \lambda_i L_i(g_j) \\
&= \sum_{i=1}^{k} \lambda_i L_i(\psi - \phi),
\end{aligned} \tag{12}
$$

where we have used $\sum_{i=1}^{m} \alpha_i g_i$ to denote an element of N. Now using (12) and the second set of equations in (10), we obtain

$$
\begin{aligned}
&[\psi - \phi, \psi] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} L_i(\psi - \phi)[L_j(\phi) - Y_j] \\
&\quad = \sum_{i=1}^{k} L_i(\psi - \phi)\left[\lambda_i + \sum_{j=1}^{k} w_{ij}[L_j(\phi) - Y_j]\right] = 0.
\end{aligned}
$$

This implies that

$$
\sigma(\phi) - \sigma(\psi) = [\psi - \phi, \psi - \phi] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} L_i(\psi - \phi) L_j(\psi - \phi) \geqq 0.
$$

Uniqueness and existence: Assume that ϕ_1 and ϕ_2 are both solutions (i.e., assume that the linear system (10) has two solutions). Then

$$
\begin{aligned}
0 &= \sigma(\phi_1) - \sigma(\phi_2) \\
&= [\phi_1 - \phi_2, \phi_1 - \phi_2] + \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} L_i(\phi_1 - \phi_2) L_j(\phi_1 - \phi_2).
\end{aligned}
$$

Therefore, $\phi_1 - \phi_2 \in N$ and $L_i(\phi_1 - \phi_2) = 0$, $i = 1, \cdots, N$, and consequently $\phi_1 = \phi_2$. This implies that the system of equations (10) has a nonsingular coefficient matrix.

If we let the linear functionals α_i, $i = 1, \cdots, m$, be defined by

$$
f(y) - [f(x), K(x, y)]_{(x)} = \sum_{i=1}^{m} \alpha_i(f) g_i(y), \tag{13}
$$

then we can observe that

$$\alpha_i(g_j) = \delta_{i,j} \quad \text{and} \quad \alpha_{i(y)}([f(x), K(x, y)]_{(x)}) = 0, \qquad i, j = 1, \cdots, m. \tag{14}$$

LEMMA 4. *Assume* (a), (b), *and* (c) *of Theorem* 3. *Let* $\hat{H}$ *consist of the elements of* H *with the inner product*

$$(f, g) = [f, g] + \sum_{i=1}^{m} \alpha_i(f)\alpha_i(g),$$

and assume $\hat{H}$ *is a Hilbert space. Then the reproducing kernel for* $\hat{H}$ *is*

$$\hat{K}(x, y) = [K(t, y), K(t, x)]_{(t)} + \sum_{i=1}^{m} g_i(x)g_i(y).$$

Proof. Using (14) we have that

$$\begin{aligned}(f(x), \hat{K}(x, y))_{(x)} &= [f(x), \hat{K}(x, y)]_{(x)} + \sum \alpha_i(f)\alpha_{i(x)}[\hat{K}(x, y)] \\ &= [f(x), \hat{K}(x, y)]_{(x)} + \sum_{i=1}^{m} \alpha_i(f)g_i(y).\end{aligned} \tag{15}$$

We can now employ (14) to obtain

$$K(x, y) = [K(t, y), K(t, x)]_{(t)} + \sum_{i=1}^{m} \alpha_{i(x)}[K(x, y)]g_i(x). \tag{16}$$

Therefore,

$$\begin{aligned}(f(x), \hat{K}(x, y))_{(x)} &= [f(x), K(x, y)]_{(x)} + \sum_{i=1}^{m} \alpha_i(f)g_i(y) \\ &= f(y),\end{aligned}$$

and so $\hat{K}$ has the reproducing property. From (16) it is clear that $K(x, y)$ is an element of $\hat{H}$ as a function of x.

COROLLARY 5. *Assume* $\hat{H} = (H, (\cdot, \cdot))$ *is a Hilbert space. Then Theorem* 3 *holds with* $F = \hat{H}^*$.

Proof. We need only show that (d)(i) and (d)(ii) hold for $L \in \hat{H}^*$. Since

$$\begin{aligned}L_{(y)}[\hat{K}(x, y)] &= L_{(y)}[K(x, y)] - \sum_{i=1}^{m} L_{(y)}[\alpha_{i(x)}[K(x, y)]]g_i(x) \\ &\quad + \sum_{i=1}^{m} g_i(x)L_{(y)}(g_i(y)) \in \hat{H},\end{aligned}$$

it is clear that $L_{(y)}[K(x, y)] \in H$. Using (13) and Lemma 4 we have

$$
\begin{aligned}
&L_{(y)}([f(x), K(x, y)]_{(x)}) \\
&\quad = L(f) - \sum_{i=1}^{m} \alpha_i(f) L(g_i) \\
&\quad = (f(x), L_{(y)}\hat{K}(x, y))_{(x)} - \sum_{i=1}^{m} \alpha_i(f) L(g_i) \\
&\quad = [f(x), L_{(y)}K(x, y)]_{(x)} + \sum \alpha_i(f)\alpha_{i(x)} L_{(y)}\hat{K}(x, y) - \sum \alpha_i(f) L(g_i) \\
&\quad = [f(x), L_{(y)}K(x, y)]_{(x)} \\
&\qquad + \sum \alpha_i(f) L_{(y)}\alpha_{i(x)}\hat{K}(x, y) - \sum \alpha_i(f) L(g_i).
\end{aligned}
$$

The third equality follows because $L_{(y)}\hat{K}(x, y) - L_{(y)}K(x, y) \in N$.

Since $\alpha_{i(x)}[K(t, y), K(t, x)]_{(t)} = 0$ by (14), we have that

$$\alpha_{i(x)}(\hat{K}(x, y)) = g_i(y)$$

and (d)(ii) follows.

REFERENCES

[1] H. AHLBERG, E. N. NILSON AND J. L. WALSH, *The Theory of Splines and their Applications*, Academic Press, New York, 1967.

[2] J. H. AHLBERG AND E. N. NILSON, *The approximation of linear functionals*, this Journal, 3 (1966), pp. 173–187.

[3] P. M. ANSELONE AND P. J. LAURENT, *A general method for the construction of interpolating or smoothing spline-functions*, Numer. Math., 12 (1968), pp. 68–82.

[4] P. J. DAVIS, *Interpolation and Approximation*, Blaisdell, New York, 1963.

[5] C. R. DE BOOR, *Bicubic spline interpolation*, J. Math. and Phys., 41 (1962), pp. 212–218.

[6] C. R. DE BOOR AND R. F. LYNCH, *On splines and their minimum properties*, J. Math. Mech., 15 (1966), pp. 953–969.

[7] M. GOLOMB AND A. F. WEINBERGER, *Optimal approximation and error bounds*, On Numerical Approximation, R. E. Longer, ed., Univ. of Wisconsin Press, Madison, 1959, pp. 117–190.

[8] D. E. HANDSCOMB, ed., *Methods of Numerical Approximation*, Pergamon Press, New York, 1966.

[9] J. W. JEROME AND L. L. SCHUMAKER, *On Lg-splines*, J. Approximation Theory, 2 (1969).

[10] L. E. MANSFIELD, *On the optimal approximation of linear functions in spaces of bivariate functions*, this Journal, 8 (1971), pp. 115–126.

[11] ———, *Optimal approximation and error bounds in spaces of bivariate functions*, J. Approximation Theory, 5 (1972), pp. 77–96.

[12] G. M. NIELSON, *Surface approximation and data smoothing using generalized spline functions*, Ph.D. thesis, Dept. of Math., Univ. of Utah, Salt Lake City, 1970.

[13] C. H. REINSCH, *Smoothing by spline functions*, Numer. Math., 10 (1967), pp. 177–183.

[14] K. RITTER, *Two-dimensional spline functions and best approximation of linear functionals*, J. Approximation Theory, 3 (1970), pp. 352–368.

[15] A. SARD, *Linear Approximation*, Mathematical Surveys No. 9, American Mathematical Society, Providence, R.I., 1963.

[16] I. J. SCHOENBERG, *Spline functions and the problem of graduation*, Proc. Acad. Sci. U.S.A., 52 (1964), pp. 947–950.

[17] M. H. SCHULTZ AND R. S. VARGA, *L-splines*, Numer. Math., 19 (1967), pp. 345–369.

Scattered Data Interpolation: Tests of Some Method

Richard Franke

Mathematics of Computation, Vol. 38, No. 157 (Jan., 1982), 181-200.

Stable URL:
http://links.jstor.org/sici?sici=0025-5718%28198201%2938%3A157%3C181%3ASDITOS%3E2.0.CO%3B2-W

Mathematics of Computation is currently published by American Mathematical Society.

MATHEMATICS OF COMPUTATION
VOLUME 38, NUMBER 157
JANUARY 1982

Scattered Data Interpolation: Tests of Some Methods*

By Richard Franke

Abstract. This paper is concerned with the evaluation of methods for scattered data interpolation and some of the results of the tests when applied to a number of methods. The process involves evaluation of the methods in terms of timing, storage, accuracy, visual pleasantness of the surface, and ease of implementation. To indicate the flavor of the type of results obtained, we give a summary table and representative perspective plots of several surfaces.

1.0. Introduction. The basic problem which is being addressed here is evaluation of methods for obtaining a smooth (at least continuous first partial derivatives) bivariate function, $F(x, y)$, which takes on certain prescribed values, $F(x_k, y_k) = f_k$, $k = 1, \ldots, N$. The points (x_k, y_k) are not assumed to satisfy any particular conditions as to spacing or density, hence the term "scattered." It is usually convenient to think of the values f_k as arising from some underlying (not necessarily known) function $f(x, y)$, so that $f_k = f(x_k, y_k)$, $k = 1, \ldots, N$.

The problem of interpolation of scattered data in two or more independent variables has been addressed by numerous authors, as can be seen by the bibliography. Many of the basic ideas involved are discussed in two survey papers (both over a wider class of approximations than we consider here) due to Schumaker [52] and Barnhill [4]. A recent review of methods for contouring, which treats many of the same ideas from that point of view, is given by Sabin [51]. Many ideas put forth have not previously been explored computationally, or only to a limited extent. Thus, the capabilities of some plausible ideas were unexplored. In addition, most of the methods involve one or more ad hoc assumptions requiring a user to specify parameters (one or more). Generally only cursory attention has been paid to the appropriate choice of these parameters, and their overall effect on the interpolant has usually not been determined.

Out of this situation arose a desire to attempt to answer a number of questions, basically all related: Which of these many methods deserve further study and development, and which should be discarded? Some methods require the user to specify an ad hoc parameter, and we have investigated the possibility of using a standard default value. The default value should give reasonably good results over a number of different sets of data, and preferably the interpolant should be rather stable with respect to changes in the parameter. Additionally, it is convenient for the user if the parameter is related to something about the data which can be easily

Received July 21, 1980; revised May 6, 1981.
1980 *Mathematics Subject Classification.* Primary 65D05; Secondary 65D15.
* Supported in part by the Foundation Research Program at the Naval Postgraduate School.

0025-5718/82/0000-0479/$06.00

estimated. In many cases (perhaps all), subjective judgements must be made about these matters, although some firm information can be obtained.

Some previous fairly extensive work had been done by McLain [41] which inspired a somewhat similar study of another class of ideas by the current investigator [16]. The initial thrust of the investigation was to compare a few "local" methods to determine which seem to work reasonably well. As the investigation proceeded, more ideas were supplied by colleagues and others so that in the end, more than a few methods are tested and compared here, including "global" methods. The total number of programs involved in this study is 32, some of which are fairly minor variations of others.

The concept of a "global" method is easily understood. The interpolant is dependent on all data points, and addition or deletion of a data point, or a change of one of the coordinates of a data point, will propagate throughout the domain of definition. The idea of a "local" method is not so clear. Typically one thinks of it as meaning that addition or deletion of a point, or a change of one of the coordinates of a data point, will affect the interpolant only at nearby points, that is, the interpolant will be unchanged at distances greater than some given distance. There are some difficulties here. If the data (the (x_k, y_k) points) are "random", one must inspect (in some way) all the data to determine which are "nearby". Does this mean there is no such thing as a "local" method? (Rosemary Chang first mentioned this idea.) We have taken a somewhat more liberal view of "local" and take it to mean that the interpolant involves only "nearby" points and one or more parameters. We allow the parameters to have been globally determined as a matter of user convenience, even though a (successful) argument can be made that then the method is not local. Thus, we classify methods as local or global without regard to how parameters are chosen or computed.

The use of global methods is not feasible for very large N since they often involve the solution of a system of $O(N)$ equations (often exactly N) and in any case involve processing all points. When systems of equations must be solved, the systems are often full and not necessarily well conditioned. While our primary aim was to investigate local methods suitable for very large data sets (several hundred points up to some millions, say), in many instances local methods involve the use of global methods on smaller sets which are then "blended" together to obtain a locally defined global interpolant. Thus it makes sense to to test global methods on moderately sized sets of data. By the same token, it is not necessary to test local methods on sets of 10,000 points (say) by virtue of the fact that they are local. If very large sets of data were to be considered, it is clear that a different implementation approach might be necessary, one which would involve a larger amount of preprocessing and perhaps additional storage.

This paper is essentially a condensed version of technical report [18]. The full documentation consists of some 370 pages, nearly 300 pages being devoted to comparative tables and perspective plots obtained by applying 29 algorithms for solution of the scattered data interpolation problem. Each of the methods is described there in some detail along with discussions of its performance in the tests.

1.1. Tested Characteristics of Methods. The characteristics on which various methods are to be compared, and how they are to be weighted in the final analysis,

are somewhat subjective. While no representation is made that the list is exhaustive (or even close to it), nor that everyone will be in agreement on it, the following items are the ones considered here. We give them and discuss them in order of decreasing importance. In the presentation of information in the summary (tables and perspective plots) each reader may weigh various aspects to suit his own needs.

Accuracy. Accuracy in reproducing a known surface is certainly one important aspect of comparison. In the usual application no representation of the underlying surface $z = f(x, y)$ is known; however, if the method approximates a variety of surface behavior faithfully, we expect it to give reasonable results in other instances. Numbers can be put on the performance of a method tested in this fashion, and we have used this idea extensively.

Visual Aspects. It has developed during the course of this project that the appearance of the interpolant is very important. The most useful representation of the surface is a dynamic one, where different viewing angles can easily be obtained. This could be achieved by building models, as well. Neither of these capabilities is available to the author, and in any case, wide distribution of such representations is impossible. Perspective plots of 3-dimensional surfaces were available and have been used extensively. The resolution and viewpoint of a perspective plot could obscure the fact that a surface is bad, but it is doubtful any truly bad surface has escaped detection.

Visual ratings are often closely related to the accuracy with which an interpolant reproduces test surfaces. There seems to be a closer relationship when accuracy is high since there is less chance for the interpolant to misbehave. At moderate accuracies one interpolant may be visually pleasing while another with similar accuracy is not.

The visual aspect is quite subjective, and ratings by different persons will give somewhat different results, although probably not contradictory ones. While it is felt that the visual aspect is quite important, exactly how this information is integrated into the overall assessment of a method is also a subjective matter.

Sensitivity to Parameters. Many of the tested methods involve the choice of one or more parameters. These choices have generally been converted to ones which are related to mean distances to nearest neighbor, although precisely that idea is never directly used. Here we are talking of nearest neighbor in the set of points $\{(x_k, y_k)\}$. Sometimes the parameter takes the form of an anticipated number of points in the region which defines a local interpolant.

Methods which involve parameters underwent informal testing for suitable values of the parameters. For fixed sets of data, the parameter was varied to find a suitable range for its value. Some methods were quite sensitive to the parameter value. Some methods were apparently sensitive to the dependent-variable values, as well as the (x_k, y_k) values. Thus, a parameter value giving good results for one function might yield poor results for a different function sampled at the same points. It is desirable that a method be stable with respect to perturbations in the parameter and that its value not be highly dependent on the function sampled. Such methods were found.

Timing. The computational effort required is generally not of great interest, unless it is very high. In this respect, only one of the methods tested was downrated

for this reason. Some methods are quite efficient in terms of time required for the calculations. These methods have generally been found deficient in other categories, unfortunately. For methods which involve a preprocessing phase, distinct from an evaluation (of the interpolant) phase, the two times for standard problems are given separately. Execution times were taken from the multiprogramming environment on the IBM 360/67 and as such may vary 10–20% with exactly the same data. Thus, execution times must be viewed as a guide rather than as precise measurements.

Storage Requirements. As with computational effort, storage requirements are not crucial, unless they are very high. For very large problems this may be altered, of course. We count storage requirements only in terms of additional *arrays* needed to store data beyond the (x_k, y_k, f_k) points. No account is taken of simple variables or program length.

Ease of Implementation. Ease of implementation is of no great concern if one obtains a working program. In other instances it may be of considerable importance. The judgement is again subjective. Further, it could be different depending on the philosophy behind the implementation. The form of the implementation could involve trade-offs between timing and storage and would doubtlessly alter the ease of implementation.

Implementation of programs specifically for this project generally was done with a lack of frills. Reasonable care was taken to assure that a grossly inefficient algorithm was not coded, but no doubt it is possible to improve on most of them. In particular, use of some preprocessing and additional storage was not used to increase efficiency during the evaluation phase. For a general purpose program this should probably be done. Some of the documented programs did use these devices. Ease of implementation is generally meant to take into account the complexity of the ideas involved in the method and the amount of code required.

1.2. The Testing Process. The initial tests performed on a few methods eventually gave rise to a standard set of test problems and a set of supporting subprograms to generate statistics from the tests and generate perspective plots of surfaces. Due to the evolution of ideas as the study progressed, some aspects of the process are not as simple as they might have been. This is particularly true of some of the test functions, but this has no bearing on the validity of the tests.

To enable testing many different methods in a consistent manner, and with a minimum of effort, a set of standard subprograms was developed which generate the test cases, compute deviation statistics for known test surfaces, obtain timing statistics, and generate and label perspective plots of the surfaces. With the current set of supporting subprograms it is generally quite easy to test a new method which is typically supplied as a subprogram (or several) which generates the values of the interpolant on a grid of x-y points. Typically all that is required is to set certain parameters, reserve any required workspace, and call the subroutine, all of which can be done with a few statements added to the prototype driver program.

There were six different test functions selected. These exhibit a variety of behavior, and, when sampled over three different x-y data sets of 100, 33, and 25 points, gave a total of 18 data sets. In addition to these, two sets of data were obtained from the literature (from [2] and [13]). One of these [13] was scaled in one

variable, which revealed something of the effects of scaling variables differently. A fourth x-y-z data set was a cardinal function, giving a total of 22 different data sets. Not all methods were tested on all sets of data; only those readily available methods, or those which performed well in initial test, have complete test results reported.

2.0. Descriptions of Tested Methods and Some Results. For description purposes the methods are classed into six groups: (1) Inverse distance weighted methods, (2) Rectangle based blending methods, (3) Triangle based blending methods, (4) Finite element based methods, (5) Foley's methods and (6) Nodal basis function methods. While there is necessarily a blurring of distinctions across these group lines, they constitute fairly distinct ideas and it is convenient to group them this way. In addition to methods which fall into those groups, a variation of Maude's method [40] has been tested since [18] appeared. While it is somewhat similar to methods of group (1), and while Maude's method also led to the methods of group (2), it will be discussed separately as group (7) Modified Maude methods.

2.1. Inverse Distance Weighted Methods. The original inverse distance weighted interpolation method is due to Shepard [53]. All methods of this type which we consider may be viewed as generalizations of Shepard's method, or variations of such generalizations. The basic Shepard's method is

$$F(x,y) = \sum_{k=1}^{N} w_k(x,y) f_k \Big/ \sum_{k=1}^{N} w_k(x,y), \tag{1}$$

where $w_k(x,y) = d_k^{\mu}$, and typically $\mu = 2$, although other values may be used. Here $d_k = ((x - x_k)^2 + (y - y_k)^2)^{1/2}$. μ may be replaced by μ_k and could possibly be different for each k. Several authors have considered various aspects of Shepard's method [4], [5], [21], [52].

Shepard's method is a global method, and the original paper suggested a scheme for localizing it by piecing together a parabolic segment with d_k^{-2} in such a way as to obtain a w_k which is zero outside some disk, say of given radius R, centered at (x_k, y_k), and which is still C^1. A simpler and more natural scheme suggested by Franke and Little [4, p. 112] is used in much of this work, that is,

$$w_k(x,y) = \left[\frac{(R - d_k)_+}{R d_k} \right]^2. \tag{2}$$

Shepard's method has an undesirable property for general use in that a flat spot occurs at each data point. Use of information about derivatives, either given or generated from the data, was suggested by Shepard and resulted in an approximation of the form

$$F(x,y) = \sum_{k=1}^{N} w_k(x,y) \left[f_k + \left(\frac{\partial f}{\partial x} \right)_k (x - x_k) + \left(\frac{\partial f}{\partial y} \right)_k (y - y_k) \right] \Big/ \sum_{k=1}^{N} w_k(x,y). \tag{3}$$

More generally, one may consider approximations of the form

$$F(x,y) = \sum_{k=1}^{N} w_k(x,y) L_k f(x,y) \Big/ \sum_{k=1}^{N} w_k(x,y), \tag{4}$$

where $L_k f$ is an approximation to f such that $L_k f(x_k, y_k) = f_k$. This is the basis for several of our methods. In this context we refer to the $L_k f$ as nodal functions.

Another way in which Shepard's method can be generalized is to view the method as an inverse distance weighted least squares approximation to $f(x, y)$ by a constant. One can then generalize to an approximation taking the form

$$F(x, y) = \tilde{F}(a_0, a_1, \ldots, a_n; x, y), \tag{5}$$

where $a_0, \ldots, a_n$ are parameters chosen by taking them to minimize (for a given (x, y)) the expression

$$\sum_{k=1}^{N} \left[f_k - \tilde{F}(a_0, a_1, \ldots, a_n; x_k, y_k) \right]^2 w_k(x, y). \tag{6}$$

This approach was taken by McLain [41] in evaluating a number of methods where $\tilde{F}$ was taken as a linear combination of low order monomials and $w_k(x, y)$ as d_k^{-2} or $\exp(-\alpha d_k^2) d_k^{-2}$. McLain also considered some approximations where f entered nonlinearly. We have considered one of McLain's methods and a variation of another. All of the methods of this class may be derived as variations of the above formula for $\tilde{F}$ [19].

Some papers discussing theoretical aspects of the above generalizations of Shepard's method have appeared recently [34], [33]. During revision of this paper, the details of two papers came to the attention of the author. Each gives, at an earlier publication date, a method previously attributed to others. Crain and Bhattacharyya [8] give the simplest version of Shepard's method, while Pelto, et al. [48], give the inverse distance weighted quadratic method credited to McLain.

The performance of methods in this group is very dependent on an appropriate weight function, $w_k(x, y)$ in (4) or (6). $w_k = d_k^{-2}$ is unacceptable since it allows too much influence by far away points, even when, for example, the $L_k f(x, y)$ in (4) are reasonably good local approximations. The use of polynomials of degree < 2 for the $L_k f(x, y)$ is inadequate to describe the local behavior of the surface. McLain's quadratic version of (6), with $w_k = \exp(-\alpha d_k^2) d_k^{-2}$, performs well, but is extremely time consuming. Best performance in the group is achieved by a version of (4) using quadratic approximations for the $L_k f$ and w_k, given by (2), for an appropriate R. We have called this the Modified Quadratic Shepard's Method. It is developed from (6) in [19], and pertinent theoretical results are given in [34].

2.2. Rectangle Based Blending Methods. The basis for this class of methods is discussed in [16] and was inspired by a short paper by Maude [40] which generalized the idea of deficient quintic splines to several variables. Unfortunately, the original interpolation function exhibits rather poor behavior and has not even been included in our tests. The original idea was to represent the interpolation function as

$$F(x, y) = \sum_{k=1}^{N} w_k(x, y) Q_k(x, y) \Big/ \sum_{k=1}^{N} w_k(x, y), \tag{7}$$

where $Q_k(x, y)$ is the quadratic polynomial interpolating $f(x, y)$ at (x_k, y_k) and the five nearest neighbors to (x_k, y_k) from the set $\{(x_j, y_j)\}$, and

$$w_k(x, y) = \begin{cases} 1 - \left(\frac{d_k}{R_k}\right)^2 \left(3 - 2\frac{d_k}{R_k}\right), & d_k \leqslant R_k, \\ 0, & d_k > R_k, \end{cases}$$

where R_k is the distance between (x_k, y_k) and its fifth closest neighbor. This idea was generalized to include any $w_k(x, y)$ which have finite support (to make the method local) so long as the $Q_k(x, y)$ interpolate $f(x, y)$ at all (x_j, y_j) where $w_k(x_j, y_j) \neq 0$. Use of approximations $Q_k(x, y)$ in Hilbert spaces, particularly in Sard spaces, was suggested and implemented [17]. One of the chief advantages of this approach is that instead of taking w_k with disks centered at the (x_k, y_k) as support regions, it is easy to use a smaller number of overlapping rectangles in such a fashion that at most four terms in the sum are nonzero, and $w_k(x, y) \equiv 1$. Use of rectangles also simplifies the problem of determining which terms are nonzero and thus results in a faster algorithm.

The set of rectangles is chosen to attempt to make each rectangle contain a given fixed number of points. Suppose the rectangles are defined by grid lines at $x = \tilde{x}_0$, $\tilde{x}_1, \ldots, \tilde{x}_{n_x+1}$ and $y = \tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_{n_y+1}$. Then weight functions with support $[\tilde{x}_{i-1}, \tilde{x}_{i+1}] \times [\tilde{y}_{j-1}, \tilde{y}_{j+1}] = R_{ij}$ are formed from piecewise Hermite polynomials, local interpolation functions Q_{ij} are constructed so that $Q_{ij}(x_k, y_k) = f_k$ whenever $(x_k, y_k) \in R_{ij}$, and then the overall approximation takes the form

$$F(x, y) = \sum_{i,j} w_{ij}(x, y) Q_{ij}(x, y). \tag{8}$$

Any type of local interpolation function Q_{ij} could be used. The author previously suggested Sard type approximations [17]. These have some undesirable properties in that they depend on factors other than relative position of (x_k, y_k) points. A second implementation using "thin plate splines" (see Section 2.6) was also tested. Neither of the methods performs as well as the author expected. It would seem that the method should be nearly as good as the underlying local approximation, however, this was not quite borne out by the tests, although the version using "thin plate splines" performs well.

Recently, some work due to Jancaitus, Junkins, and coworkers [30]–[32] has come to the investigator's attention. This work involves the idea of weighted local approximations in a similar fashion and was applied to the problem of terrain modeling. In their case the local interpolation functions were replaced by least squares approximations by polynomials and thus interpolation was not achieved.

2.3. Triangle Based Blending Methods. These methods are conceptually the same as those given by Eq. (4), but a significant difference is that the weight functions are based on a triangulation of the convex hull of the point set $\{(x_k, y_k)\}$. Several such schemes have been proposed, e.g., [7], [19], [20], and [42]. One of those considered here is the one described in [19].

Assume a triangulation of the convex hull, and suppose $(x, y) \in \bar{T}_{ijk}$, where $\bar{T}_{ijk}$ is the triangle with vertices (x_i, y_i), (x_j, y_j), and (x_k, y_k). We then take

$$F(x, y) = w_i(x, y) Q_i(x, y) + w_j(x, y) Q_j(x, y) + w_k(x, y) Q_k(x, y), \tag{9}$$

where the weight functions are finite element "shape" functions satisfying $w_m(x_n, y_n) = \delta_{mn}$ and the nodal functions Q_n satisfy $Q_n(x_n, y_n) = f_n$ for $m, n = i, j, k$. In all previously referenced methods the weight functions may be viewed as nine-parameter cubic shape functions with a rational correction to obtain normal derivatives equal to zero, and hence a C^1 approximation overall. There are many ways to obtain such correction terms, all of which appear to lead to the possibility of negative values being taken on by one of the weight functions if the triangle is very obtuse. This is probably not serious, although one has no control over the shape of the triangle in the sense that very obtuse angles cannot be avoided, especially near the boundary of the convex hull. The weight functions used here are obtained from a minimum norm problem [45]. Let b_i, b_j, b_k be the barycentric coordinates of (x, y) in $\overline{T}_{ijk}$, and let l_i, l_j, l_k be the lengths of the sides opposite vertices i, j, and k, respectively. Then the weight function is given by

$$w_k(x, y) = b_k^2(3 - 2b_k) + 6b_ib_jb_k[\alpha_{kj} + \alpha_{ki}],$$

with

$$\alpha_{kj} = \frac{b_kb_j(1 + b_i)}{(1 - b_i)(1 - b_k)}\left[\frac{l_k^2 + l_i^2 - l_j^2}{2l_i^2}\right],$$

and the others are obtained by a cyclic permutation of the indices.

While the basic method is defined only on the convex hull of the point set, it is easily extended to a globally defined function by the following idea. The exterior of the convex hull is divided into semi-infinite rectangles and semi-infinite triangles by constructing perpendiculars to the exterior edges of the convex hull at each exterior vertex. The value of the interpolant at an exterior point is obtained from the nodal function values at one (triangular area) or two (rectangular area) nearest points.

The Q_n in (9) can be taken to be any function having the required property. As with the inverse distance weighted methods, linear functions are inadequate. Use of appropriate quadratic functions yields results similar to those obtained from (4) in that case. Certain advantages accrue here. The evaluation phase is very fast since only three terms appear in (9), and the algorithm for determination of which triangle a point lies in is fast. Disadvantages are that a large amount of auxiliary storage is required for the triangulation (incidentally the triangulation algorithm itself is very fast), and long slim triangles sometimes yield surfaces which appear to have discontinuities along these triangles because of very rapid changes in function value across the narrow part.

2.4. Finite Element Based Methods. These methods are based on the concept of using C^1 finite element functions on a triangulation of the convex hull of the point set. This requires a scheme for estimating some derivatives (which derivatives depends on the element used by the method) at the data points. Our test results indicate that accurate estimates of the derivatives are very important and have a pronounced effect on the visual aspects of the surface as well as the accuracy. Three methods of this type, each using a different element, were tested. One was tested with several variations in the way partial derivatives are estimated. An additional scheme has been tested since [18] appeared, and we mention it here as well.

Akima's method [2], [3] is readily available. It uses the C^1 18 parameter quintic finite element. Extrapolation outside the convex hull is provided. The element requires estimates of first and second partial derivatives at the data points. In standard form a certain average of slopes of planes through the data point and each pair of several nearest neighbors is used to determine first derivative estimates. Second derivatives are estimated by applying the process to the derived data. Two variations of this scheme (by varying the weights in the average) were tested, as well as a version which obtained the derivatives from a local quadratic approximation. Performance of the method depends greatly on the estimates of the derivatives. The latter version gives the best results but at a considerable time penalty in the preprocessing phase. The published version is by far the fastest algorithm tested here, but gives poor results in some instances due to poor derivative estimates, generally, and sometimes due to long slim triangles in the triangulation. The latter is unavoidable in triangle based methods and often occurs. It cannot be avoided without abandoning the convex hull, or adding fictitious points.

Since the appearance of [18], Akima has proposed a variation in the computation of derivatives. Instead of using nearest neighbors in the usual sense, the neighbors in the triangulation are used. This scheme generally gives poorer surfaces than the original method, especially near the boundaries of the convex hull, where extraneous bumps often occur. This version is available in edition 8 of the IMSL library as subroutine IQHSCV.

Lawson's method [35] is similar in spirit to Akima's except that the Clough-Tocher element is used. First partial derivatives are required, and these are obtained from a quadratic approximation. Results are generally better than for Akima's method, although execution times are greater. Lawson's program does not extrapolate outside the convex hull.

Nielson's minimum norm network [46] uses a cubic element with a rational correction to achieve a C^1 function. The element is the solution of a certain minimum norm problem [45] and requires first partial derivatives in its discretized form. These are obtained by assuming a cubic variation along each edge in the triangulation and minimizing the integral (over all edges in the triangulation) of the second derivative squared. This gives the best results in this class of methods. It is somewhat slower than the other methods, but could probably be improved considerably in the evaluation phase. The method does not provide extrapolation outside the convex hull, although the investigator provided C^0 extrapolation for the tested version. Nielson's method is global as opposed to Akima's and Lawson's, which are local. The system of equations for the partial derivatives is solved by an iterative process which converges rapidly.

Since the appearance of [18], Little's method [36] has been tested and performed very well. It is based on the use of a cubic element with a rational correction term. Partial derivatives are estimated using a weighted average of the slopes of planes through neighboring points in the triangulation. One significant difference from other schemes in this group usually results in better control over long slim triangles. That difference is abandonment of the convex hull by extrapolating for a function value at some added exterior points. These points are then added to the set, which is retriangulated. This eliminates the usual edge effect, but, depending on the

extrapolated function value, can distort the surface near the edge if it is not representative of its behavior near the boundary of the convex hull.

Other finite elements could be used. One which might be appropriate is the piecewise quadratic due to Powell and Sabin [49]. This element was designed for contouring, hence the desirability of a quadratic. For general application the large number of subtriangles involved would seem to be a detriment. The author has not had access to a program based on this scheme, but it is likely it would perform on about a par with others considered here.

2.5. Foley's Methods. Foley's methods [14], [15] involve several ideas. The use of a generalized Newton type interpolant is involved in them prominently. Another idea which is exploited successfully is that of using one interpolant to generate a grid of points on which product type approximations can be constructed. The product approximation will not, in general, interpolate the given data. Hence a correction based on the original approximation is made to the error. This process is termed a "delta sum" by Foley, written $P\Delta Q$, defined by $P\Delta Q = P \oplus QP$, and implemented as $(P\Delta Q)f = P(I - QP)f + QPf$.

The idea has greater generality than considered by Foley, but the application of it seems to be the appropriate one. He considers cases where the product type approximation (taking the part of Q) is either the bivariate product Bernstein polynomial or the bivariate product natural bicubic spline. The first interpolant (taking the part of P) is taken as either the generalized Newton interpolant, or a form of Shepard's method. The delta sum idea is applied in iterated form for two methods.

The generalized Newton interpolant takes the form

$$T_N(x, y) = \sum_{k=1}^{N} a_k w_k(x, y), \quad \text{where } a_k = \frac{f_k - T_{k-1}(x_k, y_k)}{w_k(x_k, y_k)},$$

and $w_k(x, y)$ has the property $w_k(x_i, y_i) = 0$, $i = 1, 2, \ldots, k - 1$. This function is dependent on the order of the points, and so Foley's scheme involves an ordering process.

The best performance is provided by the iterated delta sum method using the generalized Newton polynomial with natural bicubic splines. The method performs reasonably well, but sometimes exhibits "polynomial-like" ripples in the surface, although it generally gives quite smooth surfaces.

2.6. Global Basis Function Type Methods. These methods can be characterized by the following idea. For each (x_k, y_k) simply choose some function $G_k(x, y)$, and then determine coefficients A_k so that $F(x, y) = \Sigma_k A_k G_k(x, y)$ interpolates the data. Schemes which work are not so simple in that appropriate choices of functions G_k are not particularly easy to make. Even if the functions G_k have only local support, the methods are global and further they require solution of a system of N linear equations. In all instances we consider, the systems have a symmetric coefficient matrix $(G_i(x_j, y_j))$, but this need not be the case. Usually the G_k are really functions of the one variable d_k. Numerous colleagues have suggested (among others) B-splines, Gaussian distributions, and other basis functions which seem to have an at best shaky mathematical justification. These schemes involve parameters to be specified by the user. For a Gaussian distribution function it is

the variance, while for rotated B-splines it is the radius at which the function becomes zero. These two methods are quite sensitive to the parameter, and, while good results are possible, the appropriate value of the parameter seems to depend on the function value as well as the (x_k, y_k) points, which is an undesirable characteristic. A potentially undesirable feature of many of these schemes is that they usually have no polynomial precision, e.g., not even constant functions are reproduced exactly. Based on practical experience, however, it is this author's opinion that incorporating polynomial precision does not, in itself, yield significant improvement. This observation has also been made elsewhere [14].

In terms of fitting ability and visual smoothness, the most impressive method included in the tests is the "multiquadric" method, due to Hardy [23]–[29]. In this method the G_k's are taken to be the upper sheet of a hyperboloid of revolution, $G_k = (d_k^2 + r^2)^{1/2}$. Here r is a parameter to be specified by the user. The method is quite stable with respect to this parameter and yields consistently good results, often giving the most accurate results of all tested methods. The surfaces are usually pleasing and very smooth. Results nearly as good are obtained with the "reciprocal multiquadric" method, $G_k = (d_k^2 + r^2)^{-1/2}$. However, here the choice of r is somewhat more crucial since small values of r will lead to a surface of peaks and dips at each data point.

Two methods which have basis functions similar to the multiquadric method are due to Duchon [9]–[12] and are also treated by Meinguet [43]–[44]. Unlike Hardy's, which as yet has no theoretical basis, these methods have an elegant theory in a Hilbert space setting. In one case $G_k = d_k^3$, while in the other $G_k = d_k^2 \log d_k$. The latter minimizes the thin plate functional

$$\int_{R^2} \left(\left| \frac{\partial^2 f}{\partial x^2} \right|^2 + 2 \left| \frac{\partial^2 f}{\partial x \partial y} \right|^2 + \left| \frac{\partial^2 f}{\partial y^2} \right|^2 \right) dx\, dy$$

in a certain Hilbert space and is termed a "thin plate spline". In each case the approximation contains a linear combination of functions in the kernel of the functional (that is, a linear function), along with side conditions, the geometric effect being to remove terms which grow faster than linear as one moves far away from the data. The thin plate splines had previously been discovered by Harder and Desmarais [22], where they are called surface splines. The two methods generally perform in comparable fashion, but the thin plate spline leads to coefficient matrices with smaller condition numbers, and hence was the more extensively tested of the two. The thin plate splines generally give approximations nearly as good as the multiquadric method, pleasant visually, and very smooth. This method has no parameter and, like other methods tested in this class, has the desirable properties of translation and rotation invariance.

It would seem that functions G_k which diminish as one moves away from the point (x_k, y_k) would yield better results than the ones which increase with distance. The reasons for thinking this is that a large value far away means a basis function has more influence far away than at the point with which it is associated. Also, the coefficient matrix for the system giving the weights is full, with its largest elements off the diagonal. Nonetheless, methods which performed the best have basis functions which are unbounded.

TABLE 1

Program Number	Description	Global/Local Type	Continuity (Continuous Derivatives)	Precision (Polynomial)	Storage	Domain	Sensitivity	Complexity	Accuracy	Visual	Timing (Eval/100 point total)
1	Franke's Method - 3	L 2	2	1	≃11N	R^2	C	D	B	B	C/B
4	Akima's Method	L 4	1	1	<33N	R^2	C	D	B	C	A/A
10	Akima's Method - Mod I	L 4	1	1	<33N	R^2	C	D	B	C	A/A
13	Nielson-Franke Quad.	L 3	1	2^a	<32N	R^2	B	D	A^-	A^-	A/B^-
14	Mod. Quad. Shepard	L 1	1	2^a	5N	B^d	B	B	A^-	A^-	C/D
16	Akima's Method - Mod III	L 4	1	2^a	<33N	R^2	B	D	B^+	B^+	A/B^-
24	Franke's Method - TPS	L 2	1	1	≃13N	R^2	C	D	B^+	B^+	B/B
28	Lawson's Method	L 4	1	2^a	<18N	$CH\{(x_k,y_k)\}^c$	N^g	D	B	B	A/A
19	Nielson's Min Norm Net.	G 4	1	1	<32N	$CH\{(x_k,y_k)\}^f$	N	D	A^-	A^-	B/B
21	Hardy's Multiquadrics	G 6	∞	-1^e	$\frac{1}{2}N(N+4)$	R^2	B	A	A	A	B^-/C^-
23	Duchon's Thin Plate Splines	G 6	1^b	1	$\frac{1}{2}(N+3)(N+7)$	R^2	N	A	A	A	C/D
27	Hardy's Recip. Multiquadric	G 6	∞	-1	$\frac{1}{2}N(N+4)$	R^2	B	A	A	A	B^-/C^-
30	Foley's TFΔ Cubic Spline	G 5	2	-1	≃8N	R^2	C	C	B^+	B	B/D
2	Mod. Shepard ⊕ Plane	L 1	1	1	0	B	C	B	C	C	D/D
3	Mod. Linear Shepard	L 1	1	1	2N	B	C	B	C	C	C/C
5	McLain's Method M_{10}	G 1	∞	1	0	R^2	C	B	B^+	B^+	F/F
6	Franke's Method - 1	L 2	1	-1	≃10N	R^2	C	D	B	B	C/B
7	Mod. Shepard's Method	L 1	1	0	0	B	C	A	D	D	B^-/B^-
8	Mod. McLain's M_8	L 1	1	1	0	B	C	B	C	C^-	C/C
11	Akima's Method - Mod II	L 4	1	1	<33N	R^2	C	D	B	C	A/A

Program Number	Description	Global/Local Type	Continuity (Continuous Derivatives)	Precision (Polynomial)	Storage	Domain	Sensitivity	Complexity	Accuracy	Visual	Timing (Eval/100 point total)
12	Nielson-Franke Linear	L 3	1	1	<32N	R^2	C	D	C	C	A/A
17	Quad. Shepard's Method	G 1	∞	2	5N	R^2	N	B	F	F	D/F
18	Shepard's Method	G 1	∞	0	0	R^2	N	A	F	F	C/C
20	Rotated Gaussians	G 6	∞	-1	$\frac{1}{2}$N(N+4)	R^2	D	A	B^+	B^+	B^-/C^-
22	Duchon's Radial Cubic	G 6	2[b]	1	$\frac{1}{2}$(N+3)(N+7)	R^2	N	A	A	A	C/D
25	Foley's Generalized Newton	G 5	∞	-1	$\approx$8N	R^2	N	C	B^-	B	B^-/C
26	Foley's TFΔ Bernstein	G 5	∞	-1	$\approx$8N	R^2	B	C	B	B	D/C^-
29	Rotated B-Splines	G 6	2	-1	$\frac{1}{2}$N(N+4)	R^2($\equiv$0 outside B)	D	A	B^+	B	D/D^-
31	Foley's Shep. Δ Cub. Spl.	G 5	2	0	$\approx$8N	R^2	C	C	B	B	B/C
32[j]	Akima's Method - Mod 80	L 4	1	1	$\approx$33N	R^2	N	D	C^+	C^-	A/A
33[j]	Little's Method	L 4	1	1	$\approx$15N	H^h	N	D	A^-	A^-	B/B[i]
34[j]	Vittitow's Adaptive Maude	L 7	1	2	$\approx$27N	$\approx G^k$	C	D	C	C	B/B[i]

[a] except for certain possible dispositions of points
[b] c^∞ except at data points
[c] convex hull of $\{(x_k, y_k)\}$
[d] $B = \{(x, y)\colon d_k(x, y) < R_w, \text{ some } k\}$
[e] no polynomial precision
[f] program modified by this investigator to extrapolate to all of R^2
[g] no parameter
[h] extrapolates beyond convex hull, not all of R^2
[i] estimated (360/67 no longer available)
[j] these methods tested since the appearance of [18]
[k] at least a rectangle given by user

We have not directly tested any method based on the idea of "kriging", or "regionalized variables", due to Matheron [38], [39] and discussed by numerous others, e.g., [47], [1], [37], [50]. However, it appears that kriging methods are related to global basis function methods, and indeed are identical to them under certain conditions. The statistical assumptions and approach taken in Kriging make the method appear harder, computationally, although this viewpoint allows estimation of the goodness of fit. The assumptions made seem to this author to be related to choice of a good parameter value in global basis function methods.

2.7. Modified Maude Methods. We briefly discussed Maude's method [40] in Section 2.2 and noted that it did not perform very well. This is primarily due to poor behavior of interpolating quadratics in two variables. Vittitow [54] has developed some modifications of the idea which attempt to alleviate this problem, as well as to overcome the possibility of "holes" appearing in the domain due to varying sparseness of the data.

Poor behavior of the local interpolation functions (the Q_k's in Section 2.2, Eq. (7)) is improved by (1) reducing the number of interpolation points, and (2) increasing the total number of points used to define the local interpolation function. This is achieved by calculating a constrained (to interpolation at a reduced number of points) least squares fit to a larger set of nearby points. Quadratic, cubic, or quartic functions can be used.

Complete coverage of a specified domain is achieved by adaptively determining the disks on which $w_k(x, y)$ (in Section 2.2, Eq. (7)) is nonzero. In the process, disks are no longer centered at the data points and fewer than N are usually needed. The actual number of interpolation points varies from disk to disk, but is no greater than a specified number. The number of points included in the least squares process (in addition to the interpolation points) is also specified by the user.

3.0. Summary. Numerous tables in [18] summarize the results of the study. In particular, there are tables giving

—maximum, mean, and rms deviations of surfaces generated by data taken from known functions;

—best performance in the accuracy tests among local methods, and overall;

—effect of varying the parameter, if any;

—times for preprocessing, interpolant evaluation, and total time;

—a summary, giving an overall "quick look" at the results.

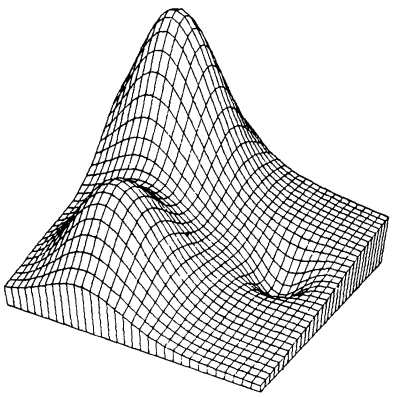
The summary table is reproduced here as Table 1, including results for the three subsequently tested programs. We briefly describe each column in the table. Footnotes are referenced by small letters. *Program number* is a number assigned to the program and used to identify it in plots and tables. *Description* is a brief pointer to the person or ideas involved. *Global/Local* tells whether the method depends globally on the data (G) or locally (L). *Type* gives the subsections of article 2 into which the method falls. *Continuity* indicates the highest order derivatives of the interpolant which are all continuous. *Precision* refers to the highest degree polynomial which is reproduced exactly by the interpolant. *Storage* refers to estimated size of storage arrays required in addition to the given data. No account of scalar variables or program size is included. *Domain* is the domain of definition of the

interpolant. *Sensitivity* to parameters is a purely subjective score, based on informal testing of the scheme. Included were whether some value of the parameter worked well for a variety of surfaces for a given set of (x, y) points, and whether the interpolant was stable with respect to changes in the parameter from that value. *Complexity* simply reflects the investigator's perception as to the complexity of ideas involved and the ease of implementation into a computer program. *Accuracy* is again subjective and is based on the relative amount of deviation one might expect from the true surface for a given method. Of course, perusal of the deviations tables will reveal that some methods do well on some surfaces and not so well (relatively speaking) on others. *Visual* pleasantness is a subjective rating based on perspective plots of the interpolant. *Timing* is relatively well defined. The first letter represents the sum of the evaluation times for three cases of 100, 33, and 25 data points. Ranges for A, B, C, D, and F, respectively, are (0, 7], (7, 21], (21, 30], (30, 50], and (50, ∞). The second letter represents the total time for 100 data points and 1089 evaluation points. Ranges are (0, 4], (4, 12], (12, 20], (20, 30], and (30, ∞). The first 13 lines in the table give the results for the extensively tested methods. The remaining lines give results for less extensively tested methods and the three subsequently tested methods.

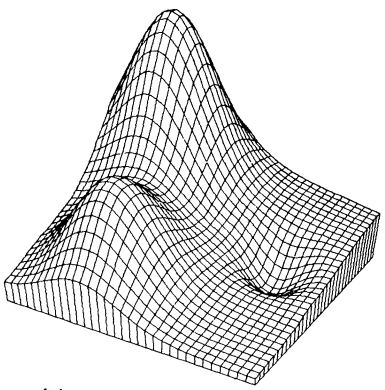
To give the flavor of the type of visual information included in the report, two pages are reproduced here in Figures 1 and 2. Figure 1 gives the test surface in part (a) and reconstructions of it by the multiquadric method for three different data sets with 100, 33, and 25 points in parts (b), (c), and (d), respectively. Figure 2 shows surfaces generated by the rectangle based blending method due to the author, using thin plate splines as the local approximations. Part (a) is a cardinal function, part (b) was generated from Akima's data, and parts (c) and (d) were generated from Ferguson's data. As a general rule, the best global methods seem to result in surfaces which are visually more pleasant than those obtained from local methods, as though localizing the surface loses something, which, while small, is still significant in that respect. Poor behavior near edges of the data set is more prevalent for local methods. For data sets of up to 100–200 points, global methods are feasible and should be considered. Nielson's minimum norm network can probably be used on somewhat larger sets of data since the sparse system of equations is solved by iteration, while other global methods generally require solution of a full system of N or more equations. Choice of a method for a large number of points is to a certain extent a personal matter, but the previously mentioned Modified Quadratic Shepard's Method performs well, requires moderate storage and computation time, and is relatively easy to implement. It is also easily extended to more independent variables. The triangle based programs, of which Akima's is the most readily available, require considerable machinery and storage for the triangulation, but in the end they are quite fast (Akima's being by far the fastest of all tested methods). These methods are extremely difficult (if not impossible) to extend to more than two independent variables and have other previously mentioned potential defects.

Despite the number of ideas explored and programs written or obtained from authors, and tested, there are still some which were not investigated. In addition to the two methods from the CAGD group at Utah, which were recently obtained,

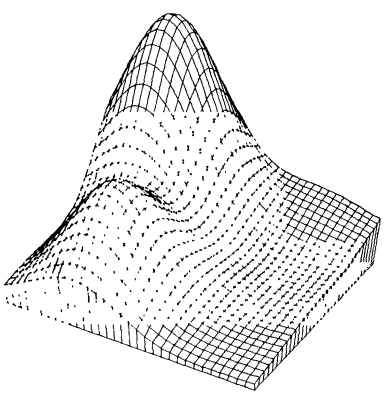
there are still more ideas which have arisen there. Many of these are based on triangulations, which the investigator feels are more suited to the design problem (where long slim triangles can be avoided) rather than the interpolation problem. Another idea which was not tested has its genesis in Briggs [6], and is available commercially [55]. The user's manual contains some impressive material, but no tests of the software have been conducted. There are no doubt more ideas worthy of investigation appearing in the literature.



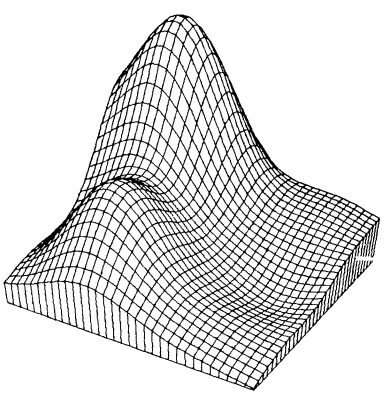
(a) Test surface



(b) 100 point sample

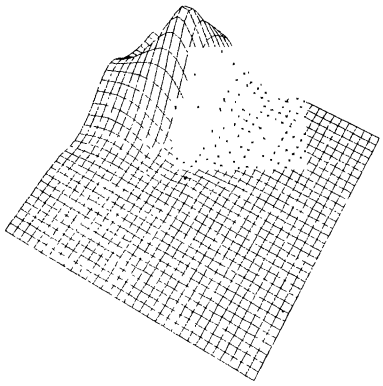


(c) 33 point sample



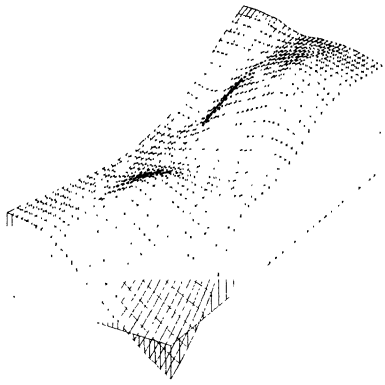
(d) 25 point sample

FIGURE 1
Hardy's multiquadric method

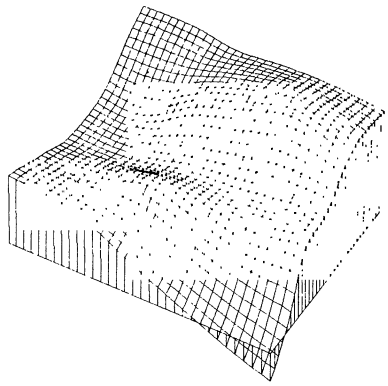


(a) Cardinal function, 25 points

(b) Akima's data, 50 points



(c) Ferguson's data, 25 points



(d) Ferguson's data, yx3, 25 points

FIGURE 2
Franke's local thin plate splines

In terms of the data considered here, it was for the most part rather nice data, even though some effort was made to include some data with varying densities. Real data exists which is very sparse in certain regions or lies in clumps. Some methods will not work in a reasonable fashion for this type of data, although we have not tried to determine which methods will and which will not. Methods based on quadratic approximations will likely misbehave for such data. In addition, local methods based on distance weighting may have holes in the domain of definition when density varies greatly or when data appears in clumps. Some additional work is necessary to see if there are suitable local methods for such data.

Acknowledgements. This investigation has consumed a great deal of time and effort. Thanks are due to numerous colleagues, among them Greg Nielson, Bob Barnhill, Frank Little, Tom Foley, Rosemary Chang, and others with whom I

discussed many ideas and who made valuable suggestions (which were sometimes followed!). Thanks are also due to those who supplied working programs, among them Greg Nielson, Hiroshi Akima, Charles Lawson, Tom Foley, and Frank Little.

Very extensive use was made of the computer facility at the Naval Postgraduate School. More recent computations have been done at the Center for Scientific Computation and Interactive Graphics at Drexel University during the author's sabbatical leave.

The author expresses thanks to the referees for suggestions which improved the paper.

Department of Mathematics
Naval Postgraduate School
Monterey, California 93940

1. HIROSHI AKIMA, "Comments on 'Optimal contour mapping using universal kriging' by Ricardo A. Olea," *J. Geophysical Res.*, v. 80, 1975, pp. 832–836 (with reply).

2. HIROSHI AKIMA, "A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points," *ACM Trans. Math. Software*, v. 4, 1978, pp. 148–159.

3. HIROSHI AKIMA, "Algorithm 526: Bivariate interpolation and smooth fitting for irregularly distributed data points," *ACM Trans. Math. Software*, v. 4, 1978, pp. 160–164.

4. R. E. BARNHILL, "Representation and approximation of surfaces," in *Mathematical Software* III (J. R. Rice, Ed.), Academic Press, New York, 1977, pp. 69–120.

5. R. E. BARNHILL, R. P. DUBE & F. F. LITTLE, *Shepard's Surface Interpolation Formula: Properties and Extensions*, CAGD report, University of Utah, 1980.

6. IAN C. BRIGGS, "Machine contouring using minimum curvature," *Geophysics*, v. 39, 1974, pp. 39–48.

7. JIM BROWN, PETER DUBE & FRANK LITTLE, *Smooth Interpolation with Vertex Functions* (manuscript).

8. I. K. CRAIN & B. K. BHATTACHARYYA, "Treatment of nonequispaced two dimensional data with a digital computer," *Geoexploration*, v. 5, 1967, pp. 173–194.

9. JEAN DUCHON, *Fonctions—Spline du Type Plaque Mince en Dimencion* 2, Report #231, Univ. of Grenoble, 1975.

10. JEAN DUCHON, *Fonctions—Spline à Energie Invariate par Rotation*, Report #27, Univ. of Grenoble, 1976.

11. JEAN DUCHON, "Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces," *R.A.I.R.O. Anal. Numér.*, v. 10, 1976, pp. 5–12.

12. JEAN DUCHON, "Splines minimizing rotation invariant semi-norms in Sobolev spaces," in *Constructive Theory of Functions of Several Variables* (W. Schempp and K. Zeller, Eds.), Lecture Notes in Math. Vol. 571, Springer-Verlag, Berlin and New York, 1977, pp. 85–100.

13. JAMES C. FERGUSON, "Multivariable curve interpolation," *J. Assoc. Comput. Mach.*, v. 11, 1964, pp. 221–228.

14. THOMAS ALFRED FOLEY, JR., *Smooth Multivariate Interpolation to Scattered Data*, Ph. D. Dissertation, Arizona State University, 1979.

15. THOMAS A. FOLEY & GREGORY M. NIELSON, "Multivariate interpolation to scattered data using delta iteration," in *Approximation Theory* III (E. W. Cheney, Ed.), Academic Press, New York, 1980, pp. 419–424.

16. RICHARD FRANKE, "Locally determined smooth interpolation at irregularly spaced points in several variables," *J. Inst. Math. Appl.*, v. 19, 1977, pp. 471–482.

17. RICHARD FRANKE, *Smooth Surface Approximation by a Local Method of Interpolation at Scattered Points*, Naval Postgraduate School, NPS-53-78-002, 1978.

18. RICHARD FRANKE, *A Critical Comparison of Some Methods for Interpolation of Scattered Data*, Naval Postgraduate School, TR #NPS-53-79-003, 1979. (Available from NTIS, #AD-A081 688/4.)

19. RICHARD FRANKE & GREGORY NIELSON, "Smooth interpolation of large sets of scattered data," *Internat. J. Numer. Methods Engrg.*, v. 15, 1980, pp. 1691–1704.

20. C. M. GOLD, J. D. CHARTERS & J. RAMSDEN, "Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain," *Comput. Graphics*, v. 11, 1977, pp. 170–175.

21. WILLIAM J. GORDON & JAMES A. WIXOM, "Shepard's method of "metric interpolation" to bivariate and multivariate interpolation," *Math. Comp.*, v. 32, 1978, pp. 253–264.

22. R. L. HARDER & R. N. DESMARAIS, "Interpolation using surface splines," *J. Aircraft*, v. 9, 1972, pp. 189–191.

23. ROLLAND L. HARDY, "Multiquadric equations of topography and other irregular surfaces," *J. Geophys. Res.*, v. 76, 1971, pp. 1905–1915.

24. ROLLAND L. HARDY, "Analytical topographic surfaces by spatial intersection," *Photogrammetric Engineering*, v. 38, 1972, pp. 452–458.

25. ROLLAND L. HARDY, "Research results in the application of multiquadric equations to surveying and mapping problems," *Surveying and Mapping*, v. 35, 1975, pp. 321–332.

26. ROLLAND L. HARDY, *Geodetic Applications of Multiquadric Equations*, Iowa State Univ. TR # 76245 (NTIS PB 255296), 1976.

27. ROLLAND L. HARDY, "Least squares prediction," *Photogrammetric Eng. and Remote Sensing*, v. 43, 1977, pp. 475–492.

28. ROLLAND L. HARDY, *The Application of Multiquadric Equations and Point Mass Anomaly Models to Crustal Movement Studies*, NOAA TR NOS 76, NGS 11, 1978.

29. ROLLAND L. HARDY & W. M. GOPFERT, "Least squares prediction of gravity anomalies, geoidal undulations, and deflections of the vertical with multiquadric harmonic functions," *Geophys. Res. Letters*, v. 10, 1975, pp. 423–426.

30. J. R. JANCAITUS & J. L. JUNKINS, "Modeling irregular surfaces," *Photogrammetric Eng. and Remote Sensing*, v. 39, 1973, pp. 413–420.

31. J. R. JANCAITUS & J. L. JUNKINS, "Modeling in n dimensions using a weighting function approach," *J. Geophys. Res.*, v. 79, 1974, pp. 3361–3366.

32. J. L. JUNKINS, G. W. MILLER & J. R. JANCAITUS, "A weighting function approach to modeling of irregular surfaces," *J. Geophys. Res.*, v. 78, 1973, pp. 1794–1803.

33. P. LANCASTER, "Moving weighted least-squares methods," in *Polynomial and Spline Approximation* (B. N. Sahney, Ed.), Reidel, Dordrecht, 1979, pp. 103–120.

34. P. LANCASTER & K. SALKAUSKAS, *Surfaces Generated by Moving Least Squares Methods*, Research Paper No. 438, Dept. of Math. and Stat., The Univ. of Calgary, Calgary, Alberta, Canada, 1979.

35. C. L. LAWSON, "Software for C^1 surface interpolation," in *Software* III (J. R. Rice, Ed.), Academic Press, New York, 1977, pp. 159–192.

36. FRANK LITTLE, CAGD report, University of Utah. (Forthcoming.)

37. A. MARECHAL & J. SERRA, "Random kriging," in *Geostatistics* (Daniel F. Merriam, Ed.), Plenum Press, New York, 1970, pp. 91–112.

38. G. MATHERON, "Random functions and their applications in geology," *in Geostatistics* (Daniel F. Merriam, Ed.), Plenum Press, New York, 1970, pp. 79–87.

39. G. MATHERON, "The intrinsic random functions and their applications," *Adv. in Appl. Probab.*, v. 5, 1973, pp. 439–468.

40. A. D. MAUDE, "Interpolation—Mainly for graph plotters," *Comput. J.*, v. 16, 1973, pp. 64–65.

41. DERMOT H. MCLAIN, "Drawing contours from arbitrary data points," *Comput. J.*, v. 17, 1974, pp. 318–324.

42. DERMOT H. MCLAIN, "Two dimensional interpolation from random data," *Comput. J.*, v. 19, 1976, pp. 178–181; also errata, *ibid.*, v. 19, 1976, p. 384.

43. JEAN MEINGUET, "Multivariate interpolation at arbitrary points made simple," *Z. Angew. Math. Phys.*, v. 30, 1979, pp. 292–304.

44. JEAN MEINGUET, "An intrinsic approach to multivariate spline interpolation at arbitrary points," in *Polynomial and Spline Approximation* (B. N. Sahney, Ed.), Reidel, Dordrecht, 1979, pp. 163–190.

45. G. M. NIELSON, "Minimum norm interpolation in triangles," *SIAM J. Numer. Anal.*, v. 17, 1980, pp. 44–62.

46. GREGORY M. NIELSON, *A Method for Interpolating Scattered Data Based Upon a Minimum Network*. (Manuscript.)

47. RICARDO O. OLEA, "Optimal contour mapping using universal kriging," *J. Geophys. Res.*, v. 79, 1974, pp. 695–702.

48. CHESTER R. PELTO, THOMAS A. ELKINS & H. A. BOYD, "Automatic contouring of irregularly spaced data," *Geophysics*, v. 33, 1968, pp. 424–430.

49. M. J. D. POWELL & M. A. SABIN, "Piecewise quadratic approximation on triangles," *ACM Trans. Math. Software*, v. 3, 1977, pp. 316–325.

50. JEAN-MICHEL RENDU, "Disjunctive kriging: Comparison of theory with actual results," *Math. Geol.*, v. 12, 1980, pp. 305–320.

51. M. A. SABIN, "Contouring—A review of methods for scattered data," in *Mathematical Methods in Computer Graphics and Design* (K. W. Brodlie, Ed.), Academic Press, New York, 1980, pp. 63–85.

52. L. L. SCHUMAKER, "Fitting surfaces to scattered data," in *Approximation Theory* II (G. G. Lorentz, C. K. Chui & L. L. Schumaker, Eds.), Academic Press, New York, 1976, pp. 203–268.

53. DONALD SHEPARD, *A Two-Dimensional Interpolation Function for Irregularly Spaced Data*, Proc. 23rd Nat. Conf. ACM, 1968, pp. 517–523.

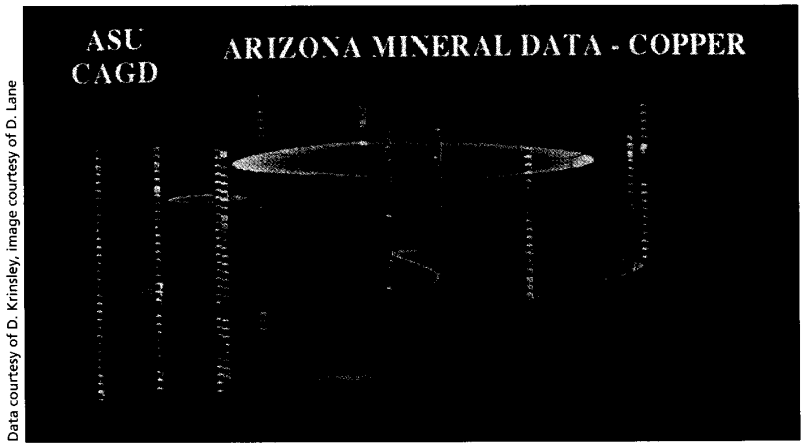
54. W. L. VITTITOW, *Interpolation to Arbitrarily Spaced Data*, Ph. D. Dissertation, Dept. of Math., Univ. of Utah, 1978.

55. *User Manual for "Surface Gridding Library"*, Dynamic Graphics, 2150 Shattuck Avenue, Berkeley, Calif., 1978.

Scattered Data Modeling

Gregory M. Nielson
Arizona State University

By comparing a variety of methods, this article provides the basis for selecting or customizing a method for modeling scattered 3D data.



Data courtesy of D. Krinsley, image courtesy of D. Lane

Figure 1. Mineral concentrations from well-log data.

This article explains modeling scattered data. Particularly interesting is the case where the independent data is 3D and the dependent data is a simple scalar. We can represent samples of this type of data by $(x_i, y_i, z_i; F_i)$, $i = 1, \ldots, N$ where $P_i = (x_i, y_i, z_i)$ represents the independent data variables and F_i is the dependent data variable. This type of data arises often in scientific studies. Examples include

1. Measurements of temperature at various locations in a furnace.
2. Mineral concentrations known at various depths of scattered bore hole locations (see Figure 1).
3. Density measurements at various locations inside a human body.
4. Economic performance levels known at various times, interest rates, and unemployment levels.
5. Pressure values computed or measured at various points on the surface of a wing (see Figure 2).
6. Precipitation measurements at various weather stations.
7. Electroencephalogram (EEG) measurements from electrodes attached to a scalp (see Figure 3).

To analyze or visualize the relationships implied by the data, we can obtain a mathematical modeling function, $F(x, y, z)$ such that $F(x_i, y_i, z_i)$ matches or approximates F_i. In this article, I discuss methods that lead to "smooth" approximations, $F(x, y, z)$, that have at least continuous first-order derivatives (that is, they are C^1 continuous). While all the examples mentioned above have data samples with the same representation, there are fundamental differences between them. In one case the domain is a 3D region, and in other cases the domain is restricted to a 2D region of a 3D space. We call the first *volumetric* scattered data and the second, *surface-on-surface* or *manifold* scattered data. Examples 1, 2, 3, and 4 are of the first type, while 5, 6, and 7 are of the second type. I separate this discussion of methods for modeling these two types of data, taking up the case of volumetric scattered data first. I should mention that other authors also use the phrases "random," "irregular," "unstructured," and "arbitrarily located" to refer to scattered data.

Volumetric scattered data

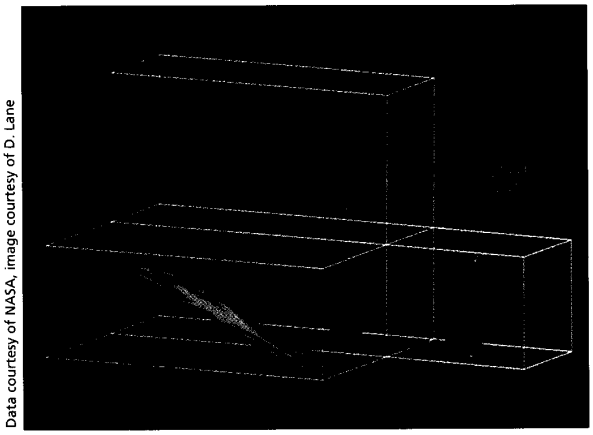
The methods described here, although not exhaustive, are intended to be representative of all trivariate scattered data interpolation methods. The first method starts with the basic Shepard's method, an inverse distanced weighted approximation that is easy to describe and implement. We can write it in the form

$$S(P) = \frac{\sum_{i=1}^{N} \frac{F_i}{\|P - P_i\|^2}}{\sum_{i=1}^{N} \frac{1}{\|P - P_i\|^2}}$$

where $\|P - P_i\|$ represents the distance from P to P_i. While simple, this basic method has some shortcomings that eliminate it from practical application. Its main deficiency is that it does not reproduce any of the local shape properties implied

0272-17-16/93/0100-0060$03.00 © 1993 IEEE

Figure 2. Hypersurface projection map of modeled scattered data representing pressure measurements taken over the surface of a wing.



Data courtesy of NASA, image courtesy of D. Lane

by the data because it typically has local extrema at the data sites. Figure 4 shows an example in the bivariate case.

In the case of bivariate data, Franke and I[2] developed a modification that eliminates the deficiencies of the basic Shepard's method. We call this the *modified quadratic Shepard's method* (MQS). Primarily, we modified the weight function $\|P - P_i\|$ to localize the overall approximation and replaced F_i with a suitable local approximation $Q_i(x, y)$. This method has the general form

$$Q(P) = \frac{\sum_{i=1}^{N} \frac{Q_i(P)}{\rho_i^2(P)}}{\sum_{i=1}^{N} \frac{1}{\rho_i^2(P)}}$$

where

$$\frac{1}{\rho_i(P)} = \frac{\left(R_w - \|P - P_i\|\right)_+}{R_w \|P - P_i\|}$$

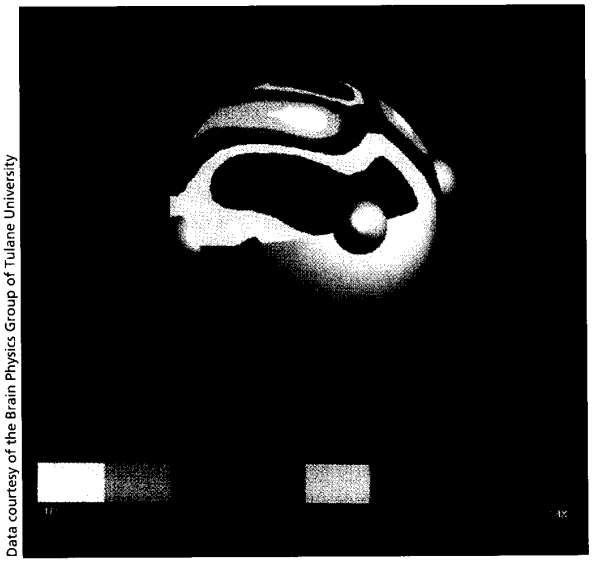
for some constant R_w. Here the subscript + denotes the truncated power function, hence the weight is zero at distances greater than R_w from the data point. We take the $Q_i(P)$ to be quadratic polynomials, obtained by a weighted least-squares fit and constrained to take on the value F_i at P_i. The weights in the least-squares process are of the same form as the weight functions of the interpolant, but with R_w replaced by another value, R_q. The ideas of this bivariate MQS method extend directly to the trivariate case. A formal description of the method consists of first selecting N_q and N_w to define

$$\frac{1}{\rho_i(P)} = \frac{\left(R_w - \|P - P_i\|\right)_+}{R_w \|P - P_i\|}, \quad R_w = \frac{D}{2}\sqrt{\frac{N_w}{N}}$$

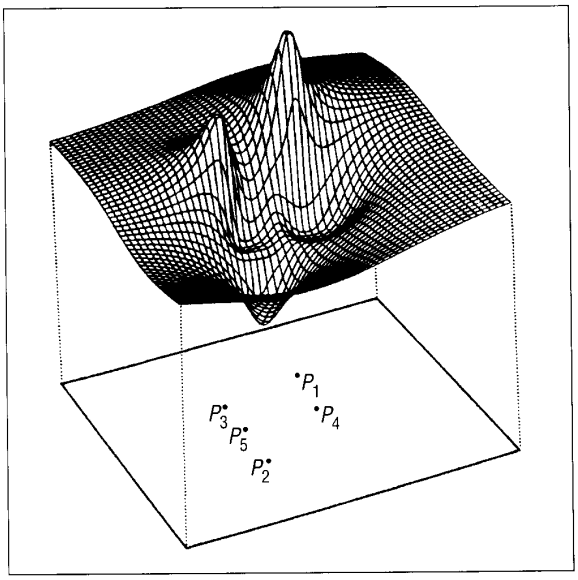
$$\frac{1}{\mu_i(P)} = \frac{\left(R_q - \|P - P_i\|\right)_+}{R_q \|P - P_i\|}, \quad R_q = \frac{D}{2}\sqrt{\frac{N_q}{N}}$$

where

$$D = \max_{i,j} \|P_i - P_j\|$$

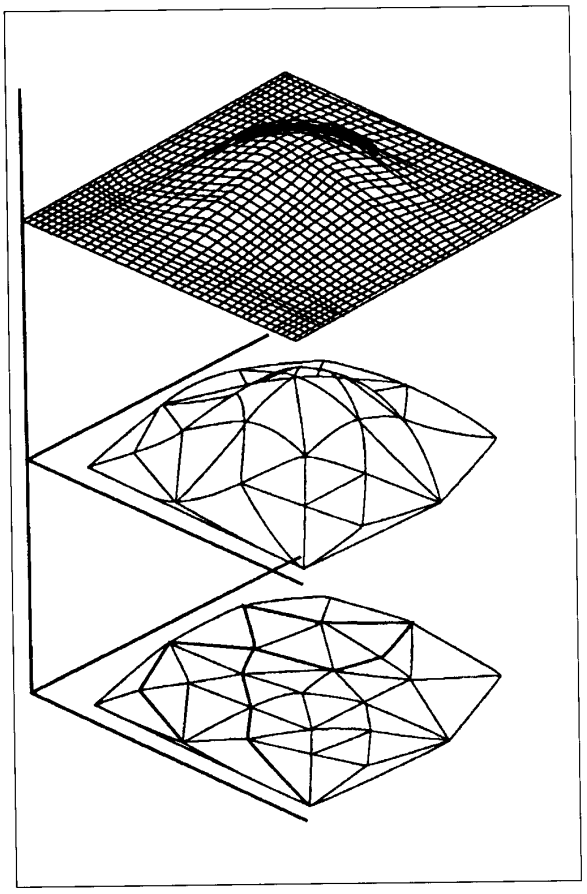


Data courtesy of the Brain Physics Group of Tulane University



Center: Figure 3. Choropleth graph of modeled EEG data.
Bottom: Figure 4. Bivariate Shepard's method. (Derived from Wixom and Gordon[1].)

Figure 5. The three steps of the MNN method in the case of bivariate scattered data.



and the default values of N_q and N_w are 54 and 27, respectively. Next, we solve the following least-squares problem:

$$\min_{a_{kj}, j=2,\ldots,10} \sum_{\substack{i=1 \\ i\neq k}}^{N} \frac{1}{\mu_i^2(x_k, y_k, z_k)} \Big[f_k + a_{k2}(x_i - x_k) + a_{k3}(y_i - y_k) + a_{k4}(z_i - z_k) + a_{k5}(x_i - x_k)^2 + a_{k6}(y_i - y_k)^2 + a_{k7}(z_i - z_k) + a_{k8}(x_i - x_k)(y_i - y_k) + a_{k9}(x_i - x_k)(z_i - z_k) + a_{k10}(y_i - y_k)(z_i - z_k) - f_i \Big]^2$$

to define

$$Q_k(x,y,z) = f_k + a_{k2}(x\text{-}x_k) + a_{k3}(y\text{-}y_k) + a_{k4}(z\text{-}z_k) + a_{k5}(x\text{-}x_k)^2 + a_{k6}(y\text{-}y_k)^2 + a_{k7}(z\text{-}z_k)^2 + a_{k8}(x\text{-}x_k)(y\text{-}y_k) + a_{k9}(x\text{-}x_k)(z\text{-}z_k) + a_{k10}(y\text{-}y_k)(z\text{-}z_k),\ k = 1, \ldots, N$$

which completes the definition of the final interpolant, $Q(P)$. Note that the function is locally determined, the influence of any point not extending further than a distance $R_w + R_q$ from each data point. Assuming somewhat uniform data density, the constant values for R_w and R_q are appropriate. If the data density is not reasonably uniform, then we might want to let the radii R_w and R_q depend on i.

The next method is a straightforward generalization to volumetric data of the distance function approach to natural cubic splines. The form of the modeling function is

$$V(P) = \sum_{i=1}^{N} c_i \|P - P_i\|^3 + a + bx + cy + dz$$

We obtain the unknown coefficients from the following system of equations, which represent the interpolation requirements and constraints analogous to end conditions:

$$\begin{pmatrix} & & & 1 & P_1 \\ & \|P_i - P_j\|^3 & & 1 & P_2 \\ . & . & . & . & . \\ & & & 1 & P_N \\ 1 & 1 & 1 & 0 & 0 \\ P_1^t & P_2^t & P_N^t & 0 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ . \\ c_N \\ a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ . \\ F_N \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

We call this the *volume spline* method. To implement this method, we need a routine to solve a linear system of equations. As long as the points P_i, $i = 1, \ldots, N$ are distinct, the coefficient matrix is nonsingular and we can solve the system. But this is a theoretical result; in the practical sense, there are limits to the value of N we can use because of the conditioning of the coefficient matrix. Even in the case of fairly uniformly distributed points, the condition number can swamp single-precision calculations for data sets of size N = 300 to 500. So, unless we take some other measures, this method is limited to fairly moderate sized data sets.

In many ways the next method, the *multiquadric* method, is similar to the volume splines of the previous paragraph. Both methods fall in the category of radial basis function methods. (For more discussion on both of these topics, see the works by Franke and Nielson[3] and Nielson and others.[4]) The general form of the multiquadric modeling function is

$$H(P) = \sum_{i=1}^{N} c_i \sqrt{R^2 + \|P - P_i\|^2}$$

The interpolation requirements lead to the system of equations

$$\begin{pmatrix} \sqrt{R^2 + \|P_i - P_j\|^2} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ . \\ c_N \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ . \\ F_N \end{pmatrix}$$

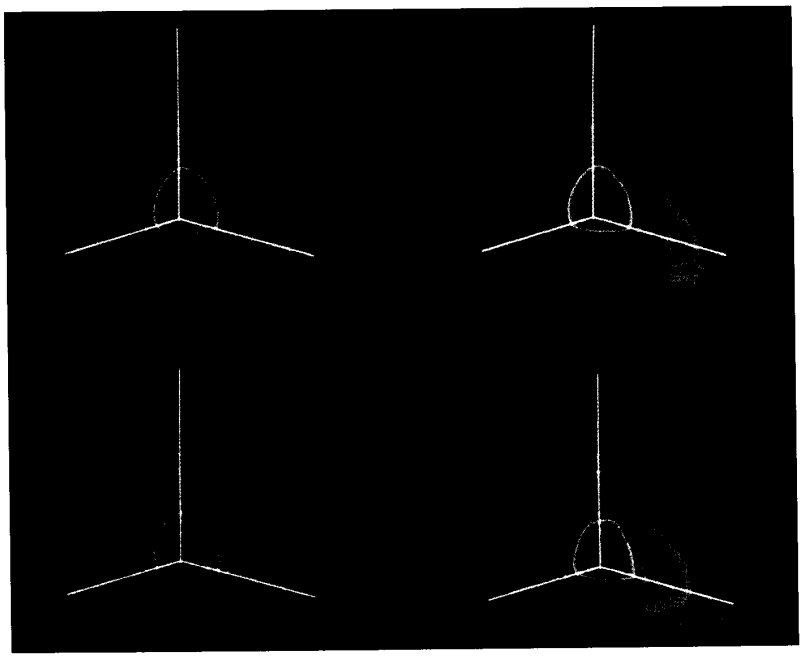


Figure 6. The face-vertex method of interpolation in tetrahedra: color coded-planar slices of the test function (upper left), the transfinite interpolant (upper right), the interpolant to transfinite data on edges (lower left), and the interpolant to discrete data (lower right).

The user must provide the parameter R^2. In the case of bivariate data, Carlson and Foley[5] have studied the selection of R^2.

The next method is considerably more complicated to describe than the previous ones. It is a generalization of the minimum norm network (MNN) method[6] for interpolating scattered bivariate data. I will briefly describe this bivariate method, then indicate how we can extend it to the case of volumetric data. There are three steps:

1. Decompose the convex hull of the points $P_i = (x_i, y_i)$ into a collection of triangles.
2. Compute an interpolating curve network defined over the edges, which has certain minimization properties.
3. Fill in the curve network to complete the definition of the modeling function by the use of a C^1 triangular interpolant.

Each of these three steps requires some explanation. A triangulation of the convex hull of a set of points $P_i = (x_i, y_i)$, $i = 1, \ldots, N$ consists of a list of triple indices, $(i, j, k) \in N_t$. Each triple (i, j, k) represents the triangle T_{ijk} with vertices P_i, P_j, P_k. Assume that no two triangles intersect and that the union of all triangles is the convex hull. There are many possible triangulations of the convex hull. Usually we prefer some type of optimal triangulation, which avoids long skinny triangles. A popular choice is the max-min triangulation, which selects the triangulation with the minimum angle as large as possible. (Schumaker provides more discussion on triangulations.[7] Nielson and Tvedt[8] cover the details of how to obtain an analogous decomposition into tetrahedra for volumetric data.)

We now take up the second step of this method. In the bivariate MNN, a network is defined over the collection of all edges. This network is characterized in a manner similar to the minimum norm characterization of standard, cubic interpolating splines. Recall that a cubic spline is the unique minimizer of

$$\int_a^b \left[F''(x)\right]^2 dx$$

subject to the interpolation requirements.

In the case of the MNN, the quantity

$$\sigma(F) = \sum_{ij \in N_e} \int_{e_{ij}} \left(\frac{d^2 F}{d^2 e_{ij}} \right)^2 ds_{ij}$$

is minimized. Here, ds_{ij} represents the element of arc length on the line segment e_{ij} and $N_e = \{ ij$ or ji (but not both) : V_i to V_j is an edge of the triangulation$\}$. It turns out that the solution is a piecewise cubic network determined by solving a sparse system of linear equations of size $2N \times 2N$.

Figure 5 shows the three steps of the MNN method in the case of bivariate scattered data.

These ideas extend to volumetric data. We minimize a quantity similar to $\sigma(F)$ except that now N_e is the collection of all edges of the triangular faces of the tetrahedral decomposition of the convex hull, and the linear system we must solve is $3N \times 3N$. Once we have computed the curve network, we can move to the third step, filling in the model with a C^1 tetrahedral interpolant. We define an interpolant on each tetrahedron that will match given position and derivative information on all edges. The method we use is a 3D generalization of the side-vertex interpolant. Called the face-vertex method, it is described elsewhere.[9] Figure 6 shows an example of this method where the test function was sampled at 125 randomly placed positions. This leads to 683 tetrahedra. The derivatives are taken from the test function in all cases.

We now move to the discussion of the last method, called *localized volume splines.* As mentioned earlier, there are definite limits to the size of the data sets for volume splines. Often the problems in scattered volume data exceed these limits.

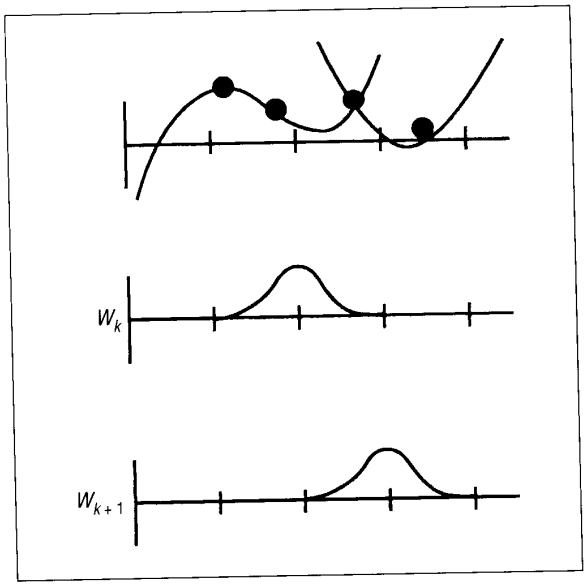


Figure 7. Localizing functions illustrated in the univariate case.

To make a particular method usable and apply it to very large data sets, we can localize it. This requires "localizing" functions, which are smooth and have a small region of support. We can easily understand the basic ideas by considering the univariate case illustrated in Figure 7. The functions w_k are nonzero only on two intervals and have the property that $\sum w_k(x) = 1$ for any x in the domain. We compute local scattered data interpolants F_k so that $F_k(x_i) = F_i$ for all data points x_i in the support (non-zero region) of w_k. From this, we can see that

$$R(x) = \sum w_k(x) F_k(x)$$

has the property that $R(x_i) = F_i$ for all x_i in the union of the support of the w_k, which serves as the domain of the approximation. In general, we can use any method to obtain the local interpolants F_k, but for this particular method we used the volume splines mentioned above. We can easily construct the localizing function w_k with piecewise cubics (bicubics or tricubics). (More details on this method are discussed elsewhere.[8])

Comparisons

To demonstrate the performance of these various methods, I include some results from an empirical study I did with Tvedt.[8] We considered the methodology of comparing methods of trivariate scattered data. In part of our work we generalized Franke's previous work on bivariate scattered data interpolation[10] to trivariate interpolation. We used test functions and data sets. We used a test function $F(x, y, z)$ to generate the dependent data and to serve as a base for evaluating method's performance. Given an independent data set, (x_i, y_i, z_i), $i = 1, \ldots, N$ and a test function $F(x, y, z)$, we used a particular method of scattered data interpolation to produce an approximation $A(x, y, z)$, which we then compared to $F(x, y, z)$. Comparisons consist of numerical statistics and subjective assessments based upon the analysis of the "graphs" of F and A. Figure 8 shows data set configurations.

We chose six data sets to represent the entire gamut of data sets. We selected three to be uniformly and randomly placed in the unit cube domain: one small, one moderate, and one large. Referred to as R125, R200, and R1000, these data sets contain 125 points, 200 points, and 1,000 points respectively. We chose the remaining three data sets to represent certain types of data likely to occur in real-world situations. The data set L200 consists of samples taken from a given number of vertical lines that were randomly perturbed. The data set

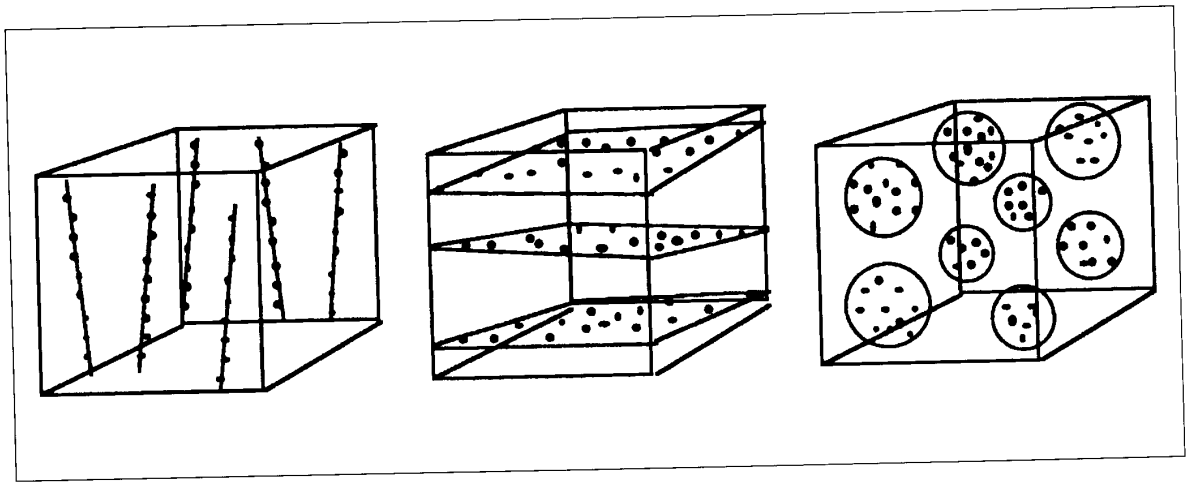


Figure 8. Data set configurations.

Table 1. Root mean square error for data sets versus test functions for local volume spline.

	F_1	F_2	F_3	F_4	F_5	F_6
R125	0.026	0.015	0.011	0.001	0.006	0.002
R200	0.013	0.012	0.008	0.001	0.003	0.002
R1000	0.001	0.003	0.001	0.0001	0.0003	0.0005
L200	0.023	0.012	0.006	0.002	0.013	0.001
P200	0.022	0.013	0.007	0.001	0.003	0.001
C200	0.025	0.015	0.016	0.002	0.040	0.001

P200 consists of scattered samples taken from horizontal planes that were randomly perturbed. This type of sampling is analogous to samples taken from slices of an object. Cluster sampling has many analogies to real-world sampling. It is a set of densely sampled areas with large gaps where no samples are taken. We can think of it as a collection of random samples. This data set, which has 200 points, is denoted by C200.

The test functions used are extensions to three dimensions of the the test functions Franke[10] used:

$$F_1(x,y,z) = 0.75 \cdot \exp\left[-\frac{(9x-2)^2+(9y-2)^2+(9z-2)^2}{4}\right]$$
$$+0.75 \cdot \exp\left[-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10} - \frac{(9z+1)}{10}\right]$$
$$+0.5 \cdot \exp\left[-\frac{(9x-7)^2+(9y-3)^2+(9z-5)^2}{4}\right]$$
$$-0.2 \cdot \exp\left[-(9x-4)^2-(9y-7)^2-(9z-5)^2\right]$$

$$F_2(x,y,z) = \left\{\tanh(9z-9x-9y)+1\right\}/9$$

$$F_3(x,y,z) = \left[(1.25+\cos(5.4y))\cos(6z)/\left\{6+6(3x-1)^2\right\}\right]$$

$$F_4(x,y,z) = \exp\left\{-\frac{81}{16}\left[(x-0.5)^2+(y-0.5)^2+(z-0.5)^2\right]\right\}/3$$

$$F_5(x,y,z) = (\exp\left\{-\frac{81}{4}\left[(x-0.5)^2+(y-0.5)^2+(z-0.5)^2\right]\right\}/3$$

$$F_6(x,y,z) = \sqrt{64-81\left[(x-0.5)^2+(y-0.5)^2+(z-0.5)^2\right]}/9-0.5$$

The approximate ranges of these functions over the domain $C = \{(x, y, z) : 0 \le x \le 1, 0 \le y \le 1, 0 \le z \le 1\}$ are F_1: [–0.1, 1.1], F_2: [0.0, 0.22], F_3: [–0.37, 0.37], F_4: [0.007, 0.33], F_5: [0.0, 0.32], F_6: [–0.3, 0.39]

Table 2. Root mean square error for method versus test function for data set R200.

	F_1	F_2	F_3	F_4	F_5	F_6
Q	0.0237	0.0134	0.0127	0.0033	0.0089	0.0012
V	0.0120	0.0014	0.0068	0.0006	0.0020	0.0015
H	0.0109	0.0126	0.0042	0.0003	0.0007	0.0018
N	0.0190	0.0121	0.0128	0.0019	0.0045	0.0030
R	0.0131	0.0117	0.0078	0.0008	0.0026	0.0017

Table 3. Root mean square error for data set versus method for test function F_1.

	R125	R200	R1000	L200	P200	C200
Q	0.0293	0.0237	0.0150	0.0328	0.0332	0.0454
V	0.0265	0.0120	–	0.0237	0.0206	0.0268
H	0.0218	0.0109	–	0.0177	0.0135	0.0349
N	0.0327	0.0190	0.0040	0.0346	0.0257	0.0429
R	0.0259	0.0131	0.0014	0.0235	0.0217	0.0257

The root mean square error is defined by

$$RMS = \sqrt{\frac{\sum_{i=0}^{N_i}\sum_{j=0}^{N_j}\sum_{k=0}^{N_k}\left[F\left(\frac{i}{N_i},\frac{j}{N_j},\frac{k}{N_k}\right)-A\left(\frac{i}{N_i},\frac{j}{N_j},\frac{k}{N_k}\right)\right]^2}{(N_i+1)(N_j+1)(N_k+1)}}$$

where F represents the test function and A represents one of the scattered data fitting methods being studied. The results given in Tables 1, 2, and 3 use the resolution values for this root mean square error of $N_i = N_j = N_j = 20$.

Q — modified quadratic Shepard
V — volume splines
H — multiquadric
N — volume minimum norm network
R — local volume splines

While numerical error statistics are useful for comparing scattered data interpolants, they do not allow subjective visual evaluation of the results. In addition, error statistics do not convey much information about the local behavior of an ap-

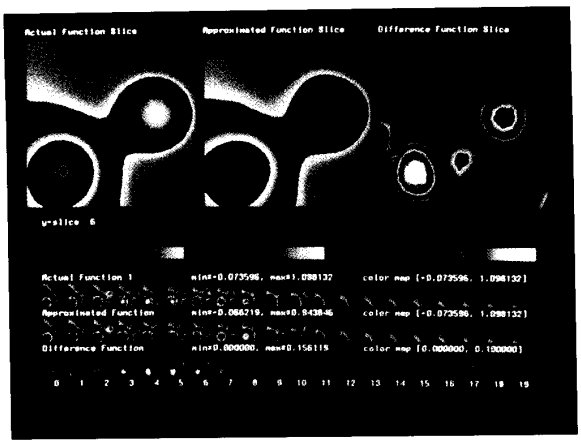


Figure 9. Typical screen image of Slice Viewer.

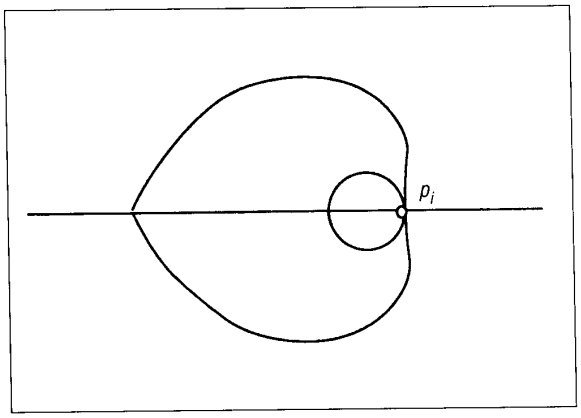


Figure 10. The C^1 discontinuity at an antipodal point of the geodesic function raised to the third power.

proximation; they only provide an overall error estimate. A major component of Tvedt's and my study[8] is Slice Viewer, an interactive program for visually evaluating the performance of various methods. The program lets the user interactively change the method, data set, or test function. In this way, the user can browse around to gain insight into the overall performance of a method and compare it to other methods. We wrote Slice Viewer with the idea that the application domain was not static. Users can add new methods, data sets, or test functions and delete old ones. A typical screen image appears in Figure 9.

Slice Viewer lets the user view slices of the volume with the x, y, or z value held constant. Across the bottom of the screen are 20 small slices showing the function values of the actual, approximated, and difference functions. At the top of the screen are three enlarged slices chosen by the user from the array below. The user points at slices, then an enlarged version appears at the top of the screen. In between the large and small slices, a color map shows the mapping of color values to function values. At the right side of each row of small slices a range ([lower, upper]) indicates the numeric values of the lower and upper bounds of the color map. The user can pick which slice to enlarge by pointing a small slice with the mouse and pressing a button. The user can show or hide the convex hull, load a new actual function, load a new approximation function, load a new color map, change the joint color mapping of the actual and approximated functions, or change the color mapping of the difference function.

I have used Slice Viewer extensively and found it valuable for learning about the performance of volumetric scattered data interpolants. To convey some sense of the experience, I include my own comments and also those of Richard Franke, who has spent considerable time using the program.

Modified quadratic Shepard method

The modified quadratic Shepard method is an extension to volume data of a well known and effective bivariate method.[2] It usually reproduces the qualitative features of the test function quite well. Near the boundaries it is sometimes performs poorly. It is only C^1. We can apply it to very large data sets, and it operates reasonably fast. In general, implementing this method takes more effort than most. The user must provide the parameters N_q and N_w or accept default values.

Volume splines

The volume splines method is a direct generalization to volumetric data of the univariate cubic interpolating spline when represented with distance functions. In most cases it gives results similar to the multiquadric method, but sometimes noticeably poorer. The implementation requires only a routine for solving a linear system of equations. The method reproduces linear functions and is C^2. In general, and without modifications, the method is limited to data sets smaller than N = 300 to 500.

Multiquadric method

As in the bivariate case, the multiquadric method generally reproduces the qualitative features of the test function quite well. The implementation consists of solving a linear system of equations. The conditioning of the coefficient matrix limits the size of the data set used. Typically, data sets of 300 to 500 or more points will yield condition numbers that will swamp single-precision accuracy. The user must provide the single parameter R^2.

Volume MNN method

This method is a volumetric generalization of an efficient and popular bivariate method.[6] The generalization is not as straightforward as we would hope, and the method proves more difficult than most others to implement. On the positive side, it is a global method that we can apply to extremely large data sets because the sparsity of the equations that we must solve lets us use some effective iterative methods. If the

tetrahedral decomposition is affine invariant, then the method will also be affine invariant. As implemented here, the method has no parameters that the user must provide. Its overall qualitative performance is very good.

Local volume splines

We can use this excellent method for very large data sets. Localization sometimes deteriorates the scheme when compared to the global method. On the R1000 data set with F_1, it works very well. On the hypersphere F_6 it seems to be on a par with the multiquadric method. The user must provide the values that partition the domain or accept uniform spaced default values. This can be a problem in that a subdomain might not have enough points to determine a volume spline. A robust implementation would recognize this situation and take appropriate evasive action. In general, the implementation is a little more complex than multiquadrics or volume splines, but overall quite easy to use.

Surface-on-surface

We now take up the case where the domain itself is a surface in 3D space. Space constraints limit me to a sampling of material in this subarea. Also, I limit the discussion to the case where the domain is a sphere. The sphere is not only one of the simplest manifold domains, but it is also important in some applications because the earth is approximated by a sphere. This is not as restrictive as it might seem, since many of the ideas for a general manifold domain have their origins in spherical methods. Also, a general approach called the *domain mapping technique* often lets us extend a method defined on a sphere to a general manifold domain.[11]

We begin with distance-based methods. A natural generalization of the distance function $|x - x_i|$ to the domain of a sphere—$r(P, P_i)$—represents the distance from P to P_i as measured on the sphere. This distance is the length of the shortest path on the sphere between P and P_i. In general, we call the shortest path on a surface between two points the geodesic curve. We take the length to be the geodesic distance between two points on a surface. For the sphere, a simple formula computes the geodesic distance: $r(P, P_i) = \cos^{-1}(P, P_i)$, where (P, P_i) is the dot product. With this in mind, it seems that one natural choice for basic functions would be $[r(P, P_i)]^3$, analogous to $|x - x_i|^3$ used for cubic splines, but a problem arises that requires some modification.

This problem has to do with the continuity of the first derivative of these functions. We can understand this better if we look at a cross section as shown in Figure 10. I took the point p_i to be (1, 0) and graphed $[r(P, P_i)]^3$ as a radial function. We can see problems at the antipodal point to p_i. From both the top and bottom the function is increasing in distance from the origin. Thus, clearly only the function (and not its higher order derivatives) is continuous. If continuity beyond the function itself is not a problem, then we can use these basis functions without modification. But if we need C^1, then we must make some adjustments. One possibility is to "round off" the backside with a piecewise definition

$$\rho(P,P_i)=\begin{cases}\cos^{-1}(P,P_i) & (P,P_i)\geq 0\\ H\left(\cos^{-1}(P,P_i)\right) & (P,P_i)<0\end{cases}$$

where $H(\pi/2) = \pi/2$, $H(\pi) = \pi$, $H'(\pi/2) = 1$, $H'(\pi) = 0$. One possible choice is

$$H(s)=\pi-4s+\frac{8}{\pi}s^2-\frac{4}{\pi^2}s^3$$

Table 4 shows some root mean square errors for two methods based on this "rounded off" distance function. The first method, *sphere splines*, has the general form

$$SS(P)=\sum_{i=1}^{N}c_i\rho^3(P,P_i)$$

The second method, *spherical multiquadrics*, has the form

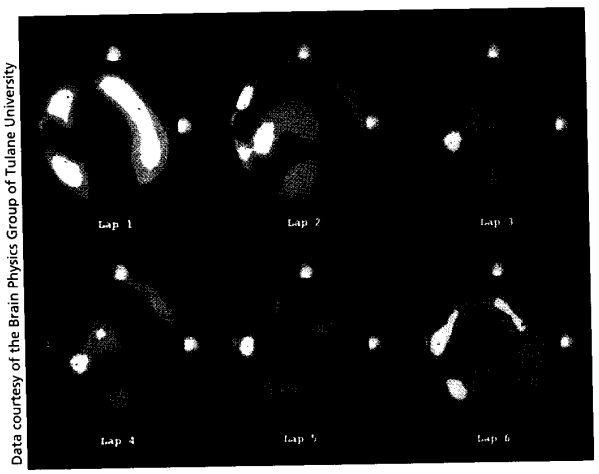
$$SM(P)=\sum_{i=1}^{N}c_i\sqrt{R^2+\rho^2(P,P_i)}$$

Table 4. Root mean square errors for three methods of modeling data from a spherical domain. (The values for this table were provided by H. Wolters.)

	F_1	F_2	F_3	F_4	F_5	F_6
SS	0.034	0.244	0.106	0.077	0.038	0.045
SM	0.406	0.116	0.261	0.045	0.781	0.147

The domain points are 74 points scattered on the surface of the sphere and the value of R = 1.0. Choosing a good value of R is a research question currently under investigation.

For small data sets, I have found the sphere spline to work quite well. Figure 11 shows the results of modeling electroencephalogram (EEG) data measured at 32 locations over the surface of the scalp. Each image corresponds to measurements taken at fixed time steps during a subject's auditory



Data courtesy of the Brain Physics Group of Tulane University

Figure 11. Modelling of EEG data measured at 32 locations and 6 time intervals.

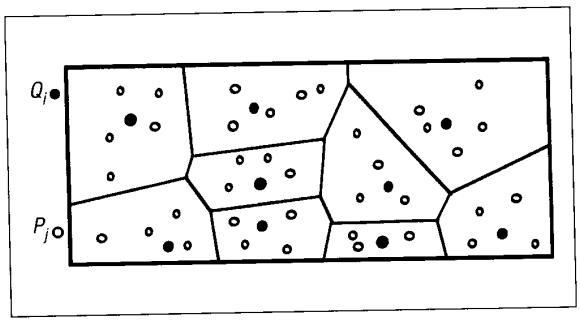


Figure 12. The Venezia criteria for bivariate scattered data.

evoked response. Observe that electrical activity propagates through the brain. Experience shows that we can understand this propagation much better when the data is animated through time.

As with their counterparts, the volume splines and multiquadrics discussed above, these distance-based manifold methods do not work with large data sets. Usually, conditioning problems set in with values of N at about 300. To circumvent this problem, we could consider a least-squares type of approximation. This requires solution of the minimization problem

$$\min_{c_i} \sum_{j=1}^{M} \left(F(P_j) - F_j\right)^2$$

where, for example,

$$F(P_j) = \sum_{i=1}^{N} c_i \rho^3(P_j, Q_i)$$

But how do we select the Q_i? In the case of bivariate scattered data, Franke and McMahon used the "Venezia crite-

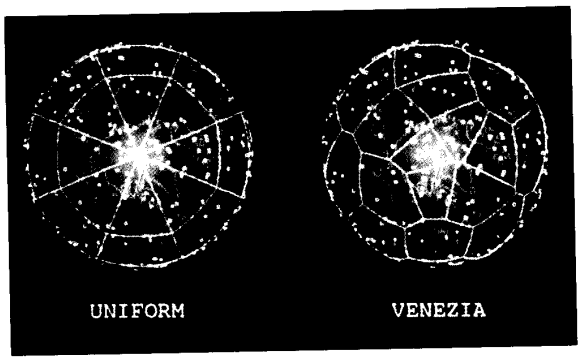


Figure 13. The Venezia points for data over a sphere.

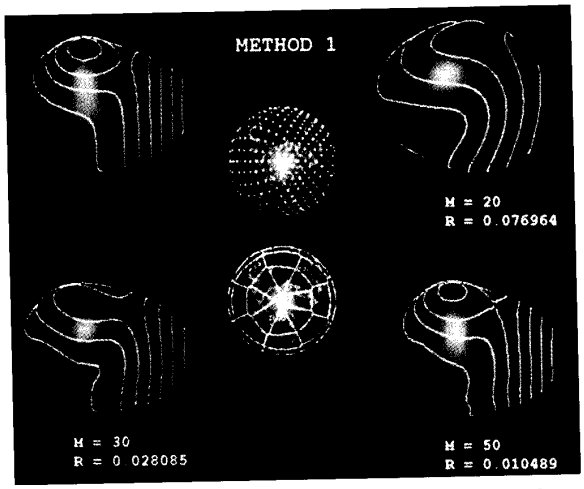
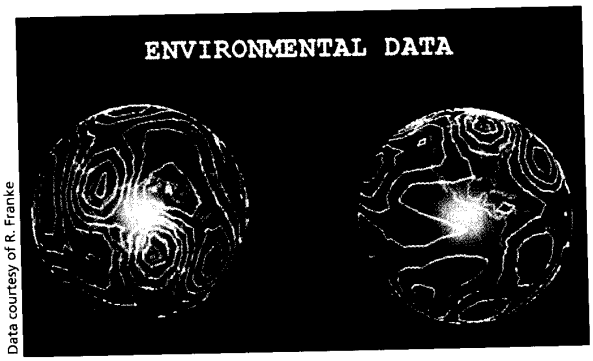
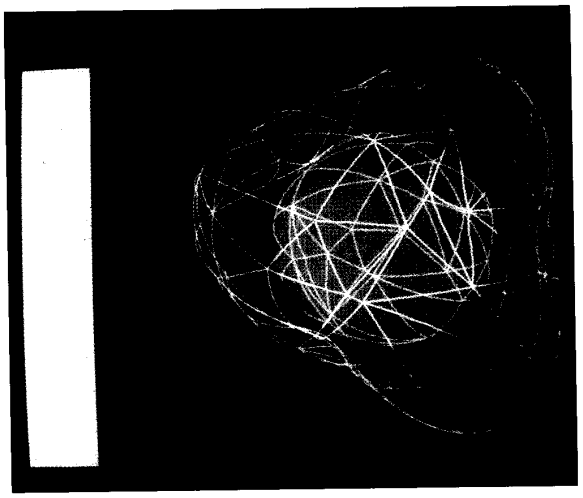


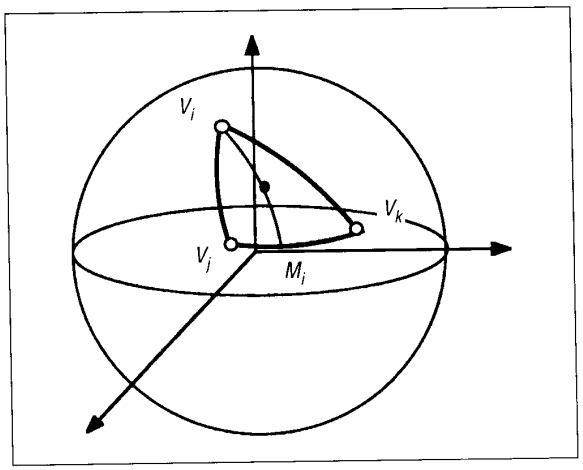
Figure 14. The spherical spline method applied to data obtained from the test function F_1. The test function is evaluated at 452 "uniformly" distributed points shown in the upper center image, and the knots for the modeling function are the vertices of the Dirichlet tessellation in the lower center image (for the case $M = 50$ only). The RMS errors are denoted by R=XXXX.



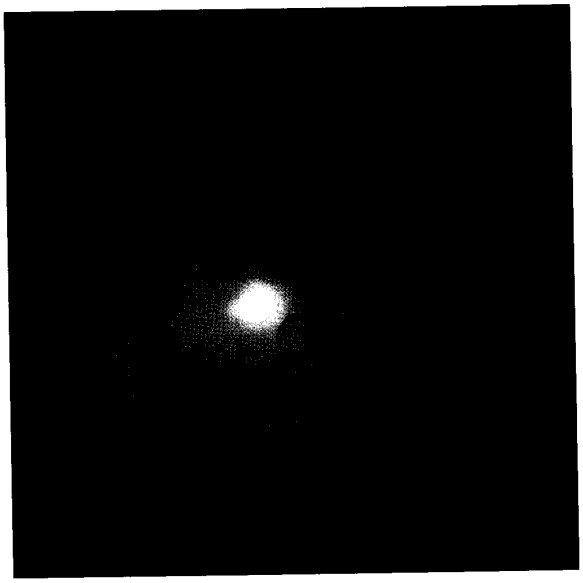
Data courtesy of R. Franke

Figure 15. Data measurements taken at 628 weather locations distributed over the earth are fitted with the spherical spline method using 50 knots determined by the Venezia criterion. Two views of the model appear in the left and right images.





Left: Figure 16. The network step of the spherical MNN method.
Lower Left: Figure 17. Spherical side-vertex triangular interpolant used to fill in the network.
Below: Figure 18. The spherical minimum norm network method applied to data from the function $F(x, y, z) = \sin(x) \sin(y) \sin(z)$.



rion," best explained by referring to Figure 12. This criterion characterizes the Q_i (represented by solid dots) by requiring them to be the centroids of all points π (represented by open dots) lying in the associated Dirichlet region. The Dirichlet region for a point Q_k is the set of points closer to Q_k than any other Q_j, $j \neq k$. These regions are bounded by the perpendicular bisectors of the Q_k. Dierks and I[12] extended this concept to the case of volumetric scattered data. The extension to the spherical domain is straightforward. Simply replace the notion of distance with geodesic distance measured on the surface of the sphere. An example appears in Figure 13. Dierks and I tested and compared these methods. Figure 14 shows typical results. We also applied these methods to "real" data sets, and Figure 15 shows some results where we used the spherical spline method.

The next method we discuss for a spherical domain is an extension of the MNN method mentioned above. Here I give only a brief description. More details appear elsewhere.[13]

The first step requires decomposition of the sphere into a collection of spherical triangles consisting of edges that are geodesic arcs. We can accomplish this by using the triangulation that is dual to the Dirichlet tessellation of the sphere. We extend the Dirichlet tessellation naturally to the sphere by using the notion of geodesic distance. Similarly, the ideas of the interpolating curve network extend quite naturally to the spherical case. Over each geodesic arc of the triangulation, the model is a cubic polynomial of the argument consisting of geodesic distance. A similar characterization to that of the original MNN method yields a similar system of linear equations, which we can solve to obtain the first-order derivatives of the model at each data site. See Figure 16 for an example of a network.

Next we fill in the network with a spherical version of the side-vertex interpolant, which uses univariate Hermite interpolation along rays emanating from a vertex and joining to the opposing edge. See Figure 17.

Figure 18 shows the results of applying this method to the test function $F(x, y, z) = \sin(x) \sin(y) \sin(z)$. We use only 56 "uniformly spaced" points, but the fit is quite good. For larger (approximately 500) data sets, the fit is indistinguishable from the test function. This is one of the advantages of this method; we can use it for very large data data sets. Even though we must solve a large $(2N \times 2N)$ linear system of equations, it is sparse, and iterative methods work well.

Conclusions

I discussed methods for modeling scattered data emphasizing two types of data: volumetric and spherical. In general, there is no single best method for all applications. Each of the

methods has pros and cons, and some are more important than others depending on the application. I have tried to provide information to aid in selecting a type of method or to form the basis for customizing a method for a particular application. ❑

Acknowledgments

This work was supported by the North Atlantic Treaty Organization under grant RG 0097/88. Several colleagues and students at Arizona State University have been an integral part of much of the work described here. In particular, Tom Foley, Karl Sun, Karsten Opitz, Il-Hong Jung, John Tvedt, Hans Jurgen Wolters, and Jong Yoon all have made significant contributions. I would also like to thank Bernd Hamann, David Lane, and Rama Ramaraj for their help and valuable contributions. Richard Franke deserves a special thanks for his contributions to this work and in particular for his comments about the various trivariate scattered data methods based on his experience with Slice Viewer.

References

1. J. Wixom and W.J. Gordon, "On Shepard's Method of Metric Interpolation to Scattered Bivariate and Multivariate Data," *Mathematics of Computation*, Vol. 32, 1978, pp. 253-264.
2. R. Franke and G. Nielson, "Smooth Interpolation of Large Sets of Scattered Data," *Int'l J. Numerical Methods in Engineering*, Vol 15, 1980, pp. 1,691-1,704.
3. R. Franke and G. Nielson, "Scattered Data Interpolation and Applications: A Tutorial and Survey," in *Geometric Modelling: Methods and Their Application*, H. Hagen and D. Roller, eds., Springer, Berlin, 1990, pp. 131-160.
4. G. Nielson et al., "Visualization and Modeling of Scattered Multivariate Data," *IEEE CG&A*, Vol. 11, No. 3, May 1991, pp. 47-55.
5. R. Carlson and T. Foley, "The Parameter R^2 and Multiquadric Interpolation," *Computers and Mathematics with Applications*, Vol. 21, No. 9, 1991, pp. 29-42.
6. G. Nielson, "A Method for Interpolating Scattered Data Based upon a Minimum Norm Network," *Mathematics of Computation*, Vol. 40, 1983, pp. 253-271.
7. L. Schumaker, "Triangulations in CAGD," *IEEE CG&A*, Vol. 13, No. 1, Jan. 1993, pp. 47-52.
8. G. Nielson and J. Tvedt, "Comparing Methods of Interpolation for Scattered Volumetric Data," to appear in *State of the Art in Computer Graphics*, R.A. Earnshaw and D.F. Rogers, eds., Springer, Berlin, 1993.
9. G. Nielson and K. Opitz, *The Face-Vertex Method for Smooth Interpolation in Tetrahedra*, Arizona State University Computer Science Tech. Report, TR-92-013, Arizona State University, Tempe, Ariz., 1992.
10. R. Franke, "Scattered Data Interpolation: Tests of Some Methods," *Mathematics of Computation*, Vol. 38, 1982, pp. 181-200.
11. T. Foley et al., "Interpolation of Scattered Data on Closed Surfaces," *CAGD*, Vol. 7, No. 1-4, 1990, pp. 303-312.
12. G. Nielson and T. Dierks, "Modeling and Visualization of Scattered Volumetric Data," *SPIE Conf. Proc.* 1459, Feb. 1991, pp. 39-52.
13. G. Nielson and R. Ramaraj, "Interpolation over a Sphere," *CAGD*, Vol. 4, Nos. 1-2, July 1987, pp. 41-57.

Gregory M. Nielson is a professor at Arizona State University, where he teaches and does research in computer graphics, computer-aided geometric design, and scientific visualization. He is one of the founders, and serves on the steering committee for, the IEEE-sponsored conference series on visualization. He is currently a director of the IEEE Computer Society Technical Committee on Computer Graphics. He serves on the editorial boards of several professional journals, including *IEEE CG&A*, and is a guest scientist at Lawrence Livermore National Laboratory. Nielson received his PhD from the University of Utah in 1970.

Readers may contact Nielson at Computer Science and Engineering, Arizona State University, Tempe, AZ 85287-5406. His e-mail address is nielson@asuvax.eas.asu.edu.

Tetrahedron Based, Least Squares, Progressive Volume Models with Application to Freehand Ultrasound Data

In: *Proceedings of Visualization 2000*, IEEE CS Press, pages 93-100, 2000

Tom Roxborough and Gregory M. Nielson

Arizona State University, Tempe AZ 85287-5406
tomrox|nielson@asu.edu

Abstract

In this paper we present a new method for the modeling of freehand collected three-dimensional ultrasound data. The model is piece-wise linear and based upon progressive tetrahedral domains created by a subdivision scheme which splits a tetrahedron on its longest edge and guarantees a valid tetrahedrization. Least squares error is used to characterize the model and an effective iterative technique is used to compute the values of the model at the vertices of the tetrahedral grid. Since the subdivision strategy is adaptive, the complexity of the model conforms to the complexity of the data leading to an extremely efficient and highly compressed volume model. The model is evaluated in real time using piece-wise linear interpolation, and gives a medical professional the chance to see images which would not be possible using conventional ultrasound techniques.

1. Introduction and Background

Two dimensional ultrasound imaging has been used for over 30 years in medicine. It has the advantages of being inexpensive, non-intrusive, real-time and safe. In conventional 2-D ultrasound echography a clinician uses a hand held probe to acquire a series of grayscale images, known as B-scans. These B-scans are viewed on a CRT as they are being acquired, and may be saved to media for further investigation. In order to get a three-dimensional feel for the patient's interior, the clinician must move the scanner around the area of interest while viewing the monitor. By training and experience the operator is able to mentally construct a 3-D model of the region being scanned, and can concentrate his scanning accordingly. Since the 1970's, there have been attempts to construct ultrasound systems that can give actual three-dimensional volumes [4, 9]. There are two major approaches for this. One is to construct an actual mechanical device that will acquire all the B-scans into a known volume. The other approach is to allow the clinician to freely probe the patient with a traditional ultrasound machine, while recording each B-scan's position and orientation in space. In this approach, known as freehand 3-D ultrasound, the position and orientation information may be attained by any number of means. There have been systems developed which use mechanical arms, acoustic trackers, image to image registration, and electromagnetic trackers. See [4, 9] for surveys of the history of three-dimensional ultrasound. Whichever system is used for freehand 3-D ultrasound, the result of a scanning session is a series of two-dimensional B-scans, along with their corresponding position and orientation (POSE) information. Figure 1 shows a freehand system that can be inexpensively assembled from readily available hardware and software.

In this paper we only consider freehand 3-D ultrasound. Each B-scan acquired during a scanning session may be thought of as a collection of grayscale intensity values located in space. In order to model these scans many researchers have imposed a regular rectilinear grid around them, and then filled in individual voxel values from the ultrasound scan information [1,17,13]. One problem with these methods is that one must be able to choose an appropriate voxel size to fit the data. If the voxels are too large then much of the data acquired from the scans is ignored. However, if too small of a size is used then there will be many empty voxels. Since the scanned data will not always fall exactly on a voxel, some method of interpolation must be used in order to assign intensities to each voxel. Various methods used include nearest neighbor interpolation [1] and distance weighted interpolation [13]. To avoid the problem of fitting the B-scan data to a regular grid, Prager et al. [15] developed a system that can produce arbitrary 2-D slices from the B-scans independent of any voxels. Because this method is based upon the intersections of an arbitrary plane with the B-scans it will fail when there are no such intersections. A plane halfway between two parallel B-scans would then show up as empty.

In this paper we present a new method for the modeling of freehand 3-D ultrasound scanning data. We approach the problem as a trivariate scattered data approximation problem. See [10] and [12]. The domain is an arbitrary rectilinear box, which is broken down into a tetrahedral grid. A function approximating the B-scan data is constructed over this mesh. An approximated intensity value can be calculated at any location in the volume by means of linear interpolation within the desired point's enclosing tetrahedron. An iterative process is used to calculate the values at the vertices making up the tetrahedral grid. This method has the advantage that no a priori knowledge is needed about voxel resolution. Standard volume visualization methods can be used to view the model [9,8], such as ray casting, isosurface extraction, and arbitrary slice plane extraction. The method is adaptive, based upon error tolerance criteria set by the user. This allows hierarchical and multiresolution models to be constructed. Also, the adaptivity property can help to guide the clinician to scan more in regions where more data may be needed.

2. Progressive Tetrahedral Domains

At all times the domain must be a valid tetrahedrization. A tetrahedrization is valid if the union of all tetrahedra is the domain of interest and any two tetrahedra only intersect at a vertex, edge, face or not at all [11]. For this method the initial domain is represented as a unit cube consisting of the six tetrahedra resulting from adding an edge from the origin (0, 0, 0) to its opposite corner

(1, 1, 1), and adding additional diagonal edges across each of the cube's six faces (See Figure 2).

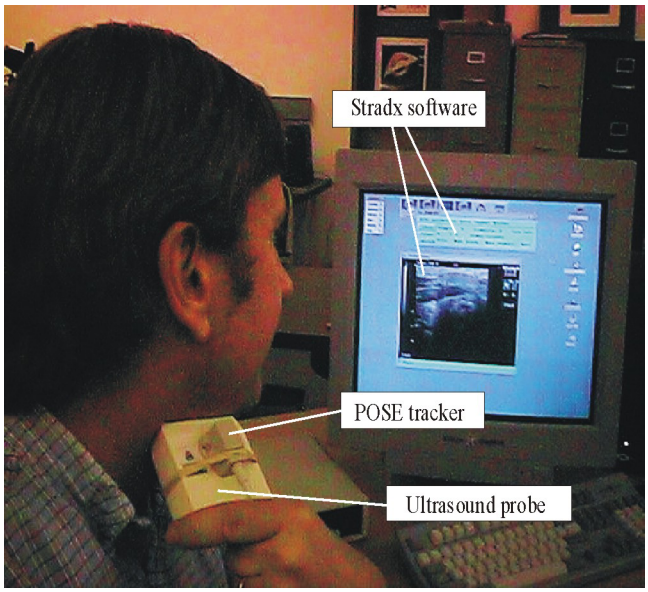


Figure 1. Collecting free-hand ultrasound data. In our lab, we use an Ascension Flock of Birds electromagnetic tracker to get position and orientation for each of the B-scans. We use the Stradx software provided by Cambridge University [14] running on an SGI O2 to simultaneously sample the tracker and to collect an image from the video signal of conventional ultrasound scanning device.

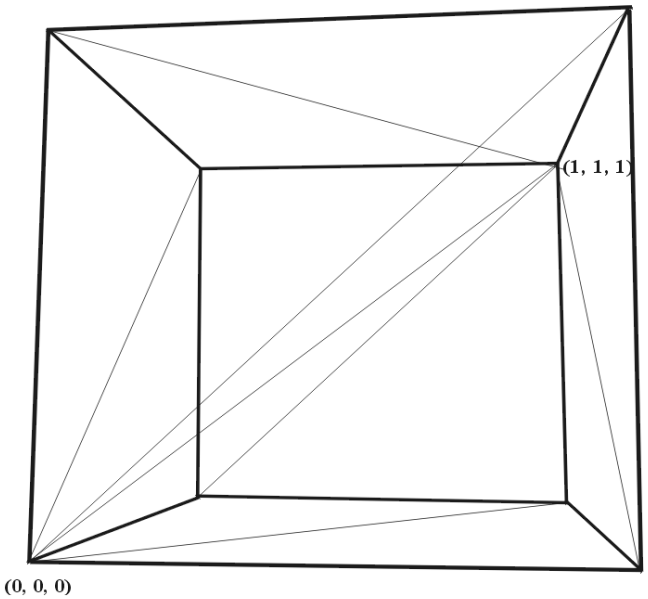


Figure 2. The unit cube initially subdivided into six congruent tetrahedra.

There have been various methods developed for subdividing simplicial grids [16, 7, 2, 3]. Grosso, et al [5] used a common splitting technique known as red-green [2, 3]. In this paper we have used a technique based on bisecting a tetrahedron along its longest edge [16, 7]. The longest edge split method was developed by Rivara [16]. Maubach [7] has adapted the longest edge split method for the special case of the initial tetrahedral grid described above. This is the subdivision method which we use (See Figure 3).

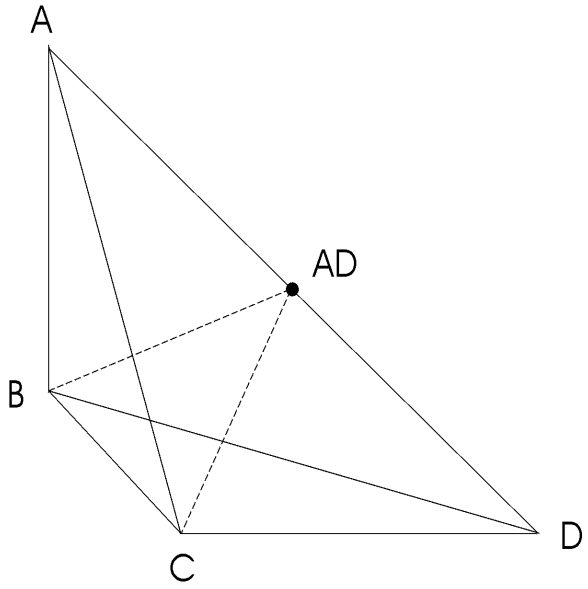


Figure 3. A tetrahedron is to be split along its longest edge at the point AD, halfway between vertices A and D.

The following terminology will be used. Two tetrahdera are *neighbors* if they share a common face. A *split neighbor* of a tetrahedron *t* is a neighbor that shares *t*'s longest edge. Note that there can be at most two split neighbors for any tetrahedron in a proper tetrahedrization. Two tetrahedra are *compatibly divisible* if they are mutually split neighbors, and if their common edge is the longest edge for both tetrahedra. Every tetrahedron belongs to a *generation*, where the generation is an integer referring to the tetrahedron's level of subdivision. Each of the six original tetrahedra making up the initial mesh belongs to generation *0*. When a tetrahedron of generation *n* is split, it will produce two tetrahedra each belonging to generation *n+1*. The tetrahedron to be split is known as the *parent*, and the two resulting tetrahedra are called its *daughters*. The two daughters resulting from one parent are *twins*. Two tetrahedra are *congruent* if one of them can be made to exactly cover the other after any combination of the following affine transformations are performed: uniform-scaling, translation, rotation. A *congruency class* is a set of tetrahedra that are all mutually congruent.

Using the initial subdivision into six congruent tetrahedra described above, and with the longest edge splitting method, the following properties must hold [7]:

- All tetrahedra of a single generation belong to the same congruency class.
- No matter how many subdivisions are performed, all tetrahedra will belong to one of only three congruency classes.
- These three congruency classes are cyclic. Initially the tetrahedra of generation *0* belong to congruency class *0*, those of generation *1* belong to congruency class *1* and those of generation *2* belong to congruency class *2*. Then those of

generation *(0+x)* belong to congruency class *0*, those of generation *(1+x)* belong to congruency class *1*, and those of generation *(2+x)* belong to congruency class *2*. See Figure 5.

So when a tetrahedron is split, it is bisected into two equal-volume tetrahedra, each of which is congruent to its twin. In order to avoid cracks within the mesh a refinement step must be taken after each tetrahedral bisection. The cracking results when there is a violation of the triangulation criteria stated above for a proper tetrahedrization. A tetrahedron might intersect more than one other tetrahedron across a single face or single edge (See Figure 4). This will give discontinuities when the function is evaluated across these tetrahedra. To prevent this from occurring Maubach devised a recursive refinement process.

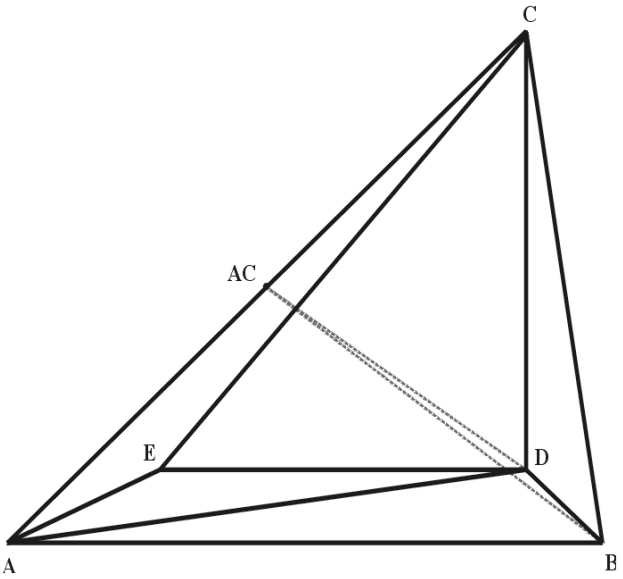


Figure 4. The cracking problem. The tetrahedron $T_{A,B,C,D}$ consisting of vertices A,B,C,D has been bisected into the two tetrahedra $T_{A,B,D,AC}$ and $T_{C,D,B,AC}$ when the edge A,C is split at AC. The tetrahedron $T_{A,E,C,D}$ has not been split. This results in $T_{A,E,C,D}$ intersecting with two distinct tetrahedra across the face made up of vertices A,C,D, violating a valid-tetrahedrization criterion.

Here is pseudo-code for a routine called Refine adapted from [7], which works on a single tetrahedron *t*, and uses the procedure Bisect(*t*), which is the simple bisection of *t* along its longest edge as shown above.

```
Refine(t)
  BEGIN
    WHILE A split neighbor n_i of t is not compatibly divisible
      DO
        Refine(n_I)
      END
    Bisect(t)
    FOR Each split neighbor n_i of t
      DO
        Bisect(n_i)
      END
  END
```

This process will terminate, giving a finite number of additional bisections. In addition, the split neighbors of *t* must be of the same generation of *t*, or one lower [7].

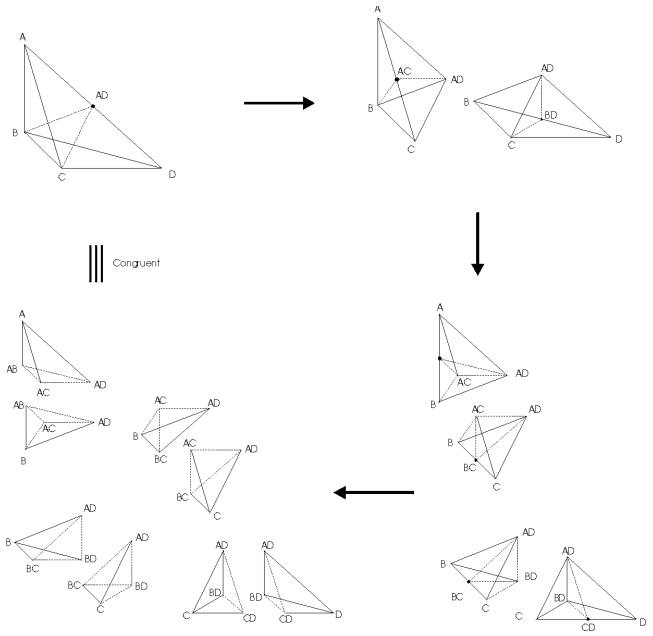


Figure 5. A single tetrahedron is bisected three times. Four generations of tetrahedra are shown, with the eight tetrahedra in the final generation being congruent to the original tetrahedron.

3. Calculation of Vertex Values

The intensity values of the vertices making up the tetrahedrization may be calculated at any time by using a global least square error approximation.
Let

M = number of B-scan data points,
N = number of vertices in mesh,
I_i = unknown intensity at vertex v_i,
$d_j = (x_j, y_j, z_j)$ = B-scan pixel location,
$S(v_i)$ = collection of tetrahedra having v_i as one of their vertices,
ϕ_i = is the piece-wise linear, basis function
such that $\phi_i(v_j) = \delta_{ij} = (1, \text{if } i=j; 0 \text{ if } i \neq j)$
$F(d_j)$ = B-scan intensity value at d_j.

Then the function to be minimized over all vertex intensities I_i, is

$$\sum_j [\sum_i (I_i \, \phi_i(d_j) - F(d_j))]^2$$

The normal equations that characterize the optimal solution are

$$\boldsymbol{AI} = \boldsymbol{b}$$

where $\boldsymbol{A}$ is an N x N Gram matrix, with elements

$$\boldsymbol{A}_{i,j} = \sum_k \phi_i(d_k)\ \phi_j(d_k)$$

And $\boldsymbol{b}$ is an N x 1 vector with elements

$$\boldsymbol{b}_i = \sum_k \phi_i(d_k)\ F(d_k)$$

and $\boldsymbol{I} = (I_1, I_2, \ldots, I_N)$ is the N x 1 vector of unknown vertex intensities.

The location of any three dimensional point with respect to a given tetrahedron may be represented in barycentric coordinates [11]. In barycentric form the 3-D Cartesian point d_j is represented as a linear combination of the four vertices making up any tetrahedron.

$$d_j = b_{v_1}^{T_i} v_1^{T_i} + b_{v_2}^{T_i} v_2^{T_i} + b_{v_3}^{T_i} v_3^{T_i} + b_{v_4}^{T_i} v_4^{T_i}, \text{and}$$
$$b_{v_1}^{T_i} + b_{v_2}^{T_i} + b_{v_3}^{T_i} + b_{v_4}^{T_i} = 1.$$

where $v_1^{T_i}, v_2^{T_i}, v_3^{T_i}, v_4^{T_i}$ are the four vertices of tetrahedron T_i, and the $b_{v_i}^{T_k}, i \in \{1,2,3,4\}$, are barycentric coordinates of point d_j corresponding to v_i with respect to T_k. If any of the barycentric coordinates are negative then the point does not lie within the tetrahedron.

We may use a slightly modified version of barycentric coordinates as a substitute for the tent function ϕ that was defined in Cartesian coordinates above. Let

$$B_{v_i}^{T_k}(p) = \begin{cases} b_{v_i}^{T} \text{ if } p \text{ is inside } T_k \\ 0 \text{ otherwise} \end{cases}.$$

Using this definition we now have for the matrix elements above:

$$A_{i,j} = \sum_{\substack{k=1, \\ T_l \in (S(v_i) \cap S(v_j))}}^{M} B_{v_i}^{T_l}(d_k) B_{v_j}^{T_l}(d_k), \text{ and}$$

$$b_i = \sum_{\substack{k=1, \\ T_l \in (S(v_i) \cap S(v_j))}}^{M} B_{v_i}^{T_l}(d_k) F(d_k).$$

Often, in typical applications, there may be approximately a million vertices making up a tetrahedrization. That means the dimensions of the matrix might be on the order of 10^6 by 10^6. Therefore it is impractical to try to solve this system of equations using direct matrix inverse methods. Instead we use a modified version of the Gauss-Seidel iterative method to solve for the vertex values [6].

Although the matrix A might be very large it will be very sparse. The matrix itself never has to be stored. Instead all values may be calculated on an as needed basis, and all information needed to calculate the elements of A and B are stored within the mesh's data structure. By keeping a list with each vertex containing pointers to the tetrahedron which contain the vertex, and storing a list of pointers to the data point structures within each tetrahedron, all needed elements may be quickly calculated.

4. The Algorithm

The unit cube is initially subdivided into six congruent and equal sized tetrahedra by adding an edge from the corner at the origin of the cube, (0, 0, 0) to its far opposite corner at (1, 1, 1), plus diagonal edges across each of the cube's faces [Figure 2]. All B-scan values are then added to this cube. In order to place the three-dimensional position values from the ultrasound B-scans into a unit cube simple affine transformations are needed. This just involves uniform scaling and transformation of the sensor's position readings, so that all desired B-scan positions will fall within the unit cube. As each B-scan value is encountered it will be added to one of the six tetrahedra making up the cube. At this point its Cartesian coordinates with respect to the cube will be converted to barycentric coordinates with respect to its enclosing tetrahedron. After adding all the B-scan image values each tetrahedron will contain a list of structures, where the structure contains the four barycentric coordinate values of the data points and the corresponding intensity value. The modeling process is now ready to begin.

Any position within the mesh may be evaluated by simple interpolation, using barycentric coordinates with respect to the tetrahedron enclosing the position point. The approximated value for a point p within tetrahedron T having barycentric coordinates b_1, b_2, b_3, b_4 is

$$I(p) = \sum_{i=1}^{4} b_i I_i.$$

The mesh subdivision process is adaptive. This means we only want to split those tetrahedra that need to be split in order to satisfy some tolerance criteria. To decide upon the tetrahedra to be subdivided we do an initial solve of the least squares system, using the Gauss-Seidel method described above. The tetrahedra are then sorted according to their mean square error values, where the mean square error is calculated as

$$mse_{T_i} = \frac{\sum_{j=1}^{N_{T_i}} (I(d_j) - F(d_j))^2}{D_{T_i}},$$

Where D_{T_i} is the number of data points within tetrahedron T_i, and $I(d_j)$ is the interpolated value at d_j.

In order to avoid excessively solving the least squares system we have found that marking the worst five percent of the tetrahedra to be split works well. These tetrahedra are each bisected, the data points from the parent tetrahedron are added to the proper daughter tetrahedron, then the least squares solution is again calculated. If the global error is within a prescribed tolerance then the process is complete, and the model is done. Otherwise the process is repeated.

After each set of tetrahedral subdivisions is performed a global root mean square error is calculated. If this error is less than a prescribed tolerance than the model is done.

$$rms = \sqrt{\frac{\sum_{i=1}^{N_T} mse_{T_i} D_{T_i}}{D}},$$

Where N_T is the number of tetrahedra in the mesh, and D is the number of B-scan data points in the mesh.

If the given global rms error tolerance is set too low, the subdivisions might never produce a model within the tolerance bound. To prevent this an upper bound on the number of tetrahedra produced should be given. Also, a limit on the smallness of tetrahedra is given, not allowing tetrahedra of more than a prescribed maximum generation to be subdivided. This minimum size of tetrahedra should be set according to the actual resolution achievable by the scanning device. Since the tetrahedron refinement process will only subdivide tetrahedra of the same or lower generation we do not have to worry about inadvertently splitting tetrahedra which do not satisfy the maximum generation bound during the refinement process.

The following is a pseudo-code description of the algorithm:

Initialize:
- set tol = user-defined global rms error tolerance
- set tetLim = user-defined maximum number of tetrahedra allowed in mesh
- set genLim = user-defined maximum generation allowed for a tetrahedron
- uniformly subdivide unit cube into 6 equal-volume, congruent tetrahedra
- add all B-scan data points and intensities to their respective tetrahedra within the mesh. Store position as barycentric coordinates
- solve least squares system for vertex intensity values
- calculate grms = calculated global rms error for model

Start:
- done = false
- while (done == false)
  - if (grms < tol) OR (number of tetrahedra > tetLim)
    - done = true
  - else
    - sort tetrahedra by mean square error in descending order
    - split the first five percent of sorted tetrahedra which satisfy:
      - tetrahedron's generation < genLim
      - tetrahedron contains at least 2 data points
    - if (no tetrahedra can be split in above step)
      - done = true
    - else
      - solve least squares system for vertex intensity values
      - calculate grms

When a parent tetrahedron is split into its twin daughter tetrahedra its data points must be correctly assigned to these children, and the new barycentric coordinates calculated. Due to the fact that a tetrahedron is always bisected across its longest edge into two equal-volume, congruent tetrahedra this is a simple process. Figure 6 illustrates how this process works. We only need to compare the two barycentric coordinates corresponding to the vertices on the longest edge, b_0b_k. If $b_0 > b_k$ the data point will belong to the daughter tetrahedron coming from the b_0 side of the parent tetrahedron. Otherwise it will belong to the other child tetrahedron. To update the barycentric coordinates within the new tetrahedra, we only need to perform one subtraction and one multiplication. See Figure 6 for an illustration of this.

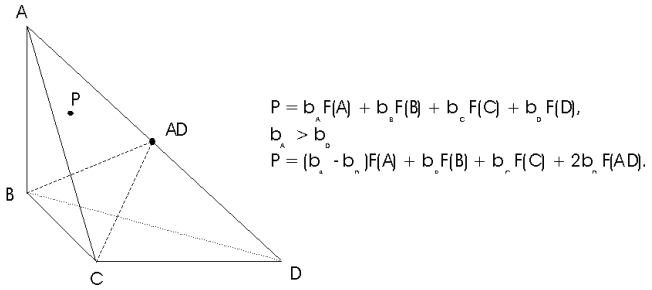


Figure 6. The calculation of new barycentric coordinates for a point P in tetrahedra $T_{A,B,C,D}$ after bisection. If $b_A > b_D$ P will belong to tetrahedron $T_{A,B,C,AD}$ and will have the new barycentric coordinates ($b_A - b_D$), b_B, b_C, $2b_D$. Otherwise it will belong to $T_{D,C,B,AD}$ and have the barycentric coordinates ($b_D - b_A$), b_C, b_B, $2b_A$.

If a vertex has no support, meaning that no data points are contained within any of the tetrahedra sharing that vertex, then during the Gauss-Seidel solution phase there will occur a divide by zero situation. To avoid this we exclude all such vertices from the iterative solution process. This condition may occur when the B-scan data points are not distributed densely enough. We do not choose tetrahedra with fewer than two data points to be split. However, during the refinement process this condition may not be enforced, resulting in occasional unsupported vertices. Also, the volume being scanned does not need to cover the entire unit cube after transformation. During the subdivision process these vertices will not contribute to the solution, and will have no value. In order to have a complete model something must be done. These unsupported vertices could be given intensity values based on a weighted average of the nearest supported vertices. Alternatively, their values could be set by performing another global interpolation method using the supported vertices. We decided to take another approach here. Since ultrasound is used for medical diagnostic purposes it might be dangerous to set these intensity values when there is little actual data around them. So these vertices are marked as "bad". By coloring them red within the unit cube a clinician doing the ultrasound scanning will know to collect more data in those areas, and a new fit can be calculated.

When visualizing the data as arbitrary planes those parts that have no support can be blacked out. These measures will help prevent unwanted artifacts from showing up during the viewing phase.

5. Results

We now present some results of our methods applied to actual data. We choose to base our examples presented here on a data set provided by Robert Rohling because this data set is typical of freehand ultrasound data. A similar data set is available at [14]. The data set used here consists of 462 B-scan images being generated, each of size 84 by 102. (The original data was 335 by 408. We shrunk the image sizes by using a mean filter in because of memory constraints.) Using the position and orientation information from the six degrees of freedom tracker, each grayscale intensity value from the scan images was added to the domain unit cube. Models were then computed using various global rms error tolerances. Figure 7 demonstrates the adaptive nature of the subdivision process, where the grid is finer where the data is present. In figure 8 five of the original B-scans are shown next to the images extracted from the various models at the exact same positions in space. Notice that the modeling process has smoothed out the noisiness of the original B-scans, acting as a low-pass filter.

Next we show images which result from taking an arbitrary two-dimensional slice through the model. These images do not need to correspond to any of the original B-scans, but are the result of evaluating the model by interpolating within the tetrahedra. These are images that a medical professional could not have seen without using 3-D modeling methods. In figure 9 one can plainly see a cut away view of the carotid artery, which would not have been possible with normal 2-D ultrasound echography. From these images it is apparent that 3-D ultrasound could become a very powerful tool in medical diagnostics.

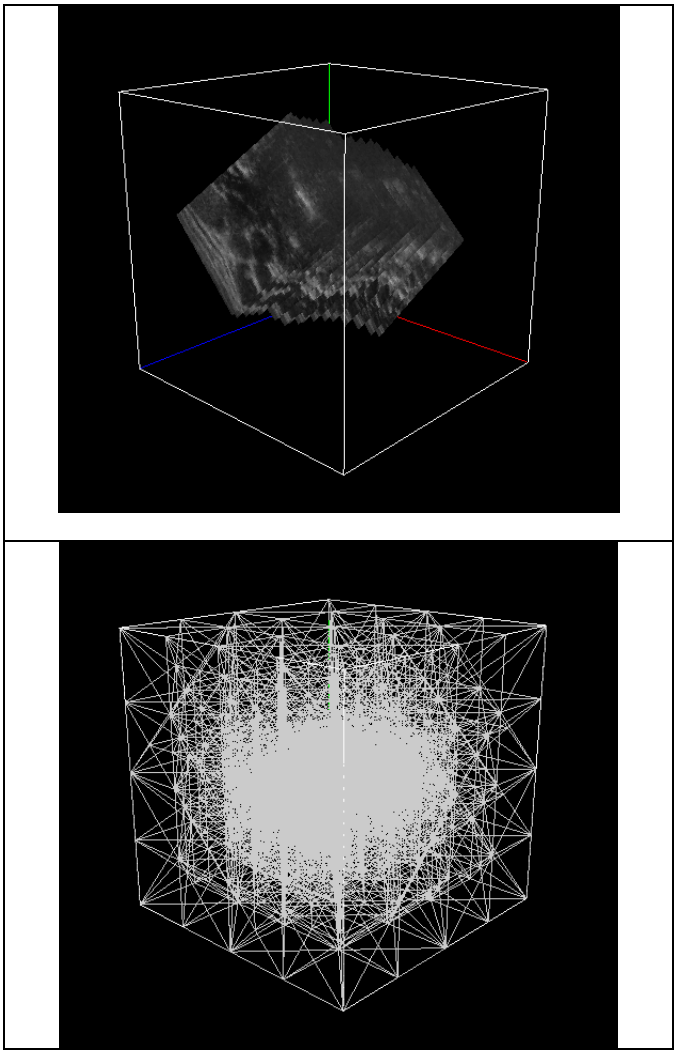


Figure 7. The top figure shows a sampling of the actual B-scans within the domain cube. The bottom figure shows the tetrahedral grid after subdividing. There are 26,552 tetrahedra, with 4,861 vertices.

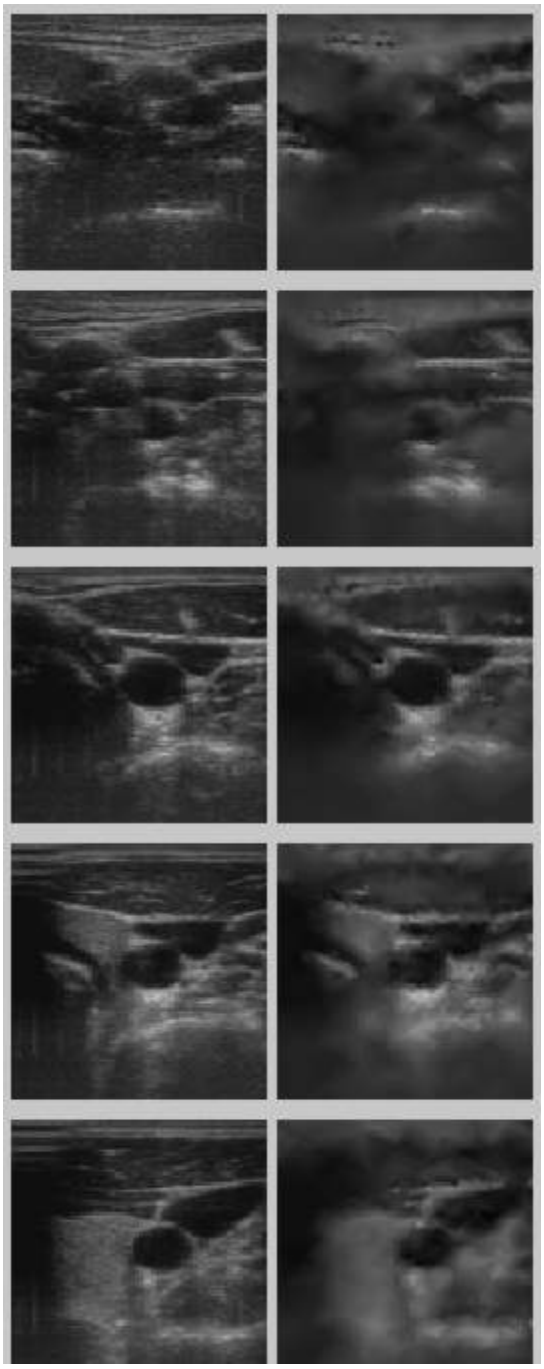


Figure 8. On the left are five actual B-scan images. Next to each is the corresponding image from the model. The rms error is 10.83. The grid contains 252,222 tetrahedra, with 40,578 vertices.

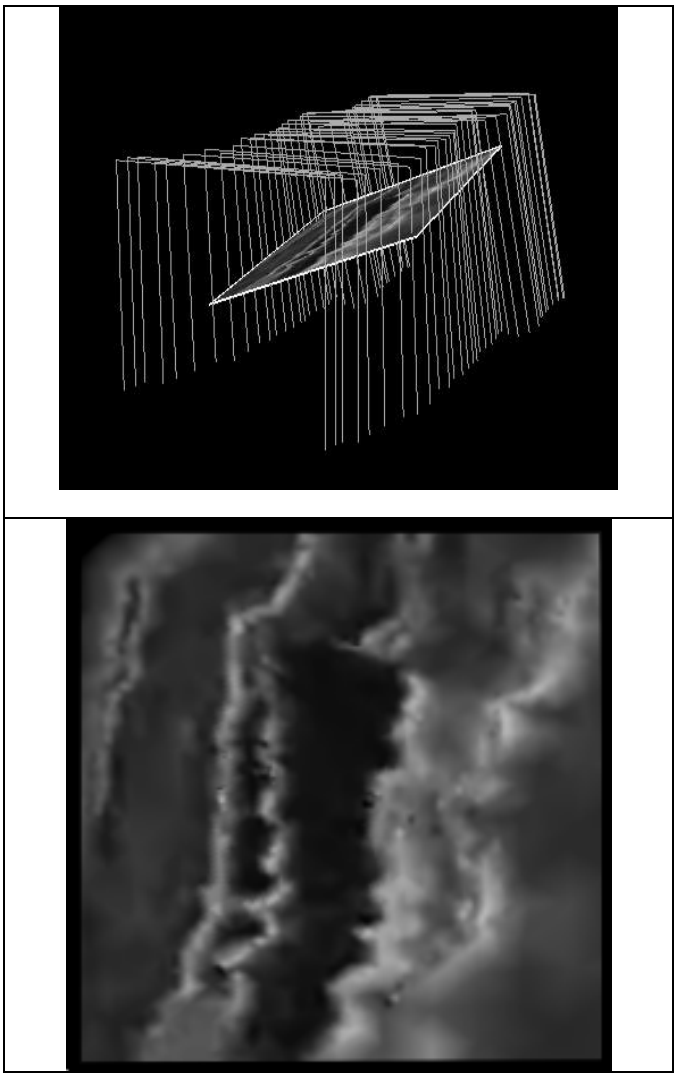


Figure 9. The top image shows the orientation of a slice in relation to the original B-scan images (for clarity only one in four B-scan outlines is shown). On the bottom the slice image is shown. This illustrates the ability to interrogate the data (model) as desired. The original B-scans are collected approximately perpendicular to the carotid artery while the slice here shows an approximate lateral view.

6. Conclusion

Three dimensional ultrasound echography is poised to become a common diagnostic tool within the medical community. Freehand methods used to collect the data from existing ultrasound machinery could provide an inexpensive, non-obtrusive means of collecting this data. We have presented a unique way to model this data, using tetrahedral grids. This method has the advantage over other methods of being adaptive to the given data. Therefore no prior knowledge of voxel size is needed, and there is no wasted memory needed to save or transport the 3-D model.

We have presented this modeling method as a way to model freehand three dimensional ultrasound data. However there is no reason that the same method could not be used on other three

dimensional scattered data problems. Some possible applications are seismic, oceanographic or weather data. We have not given any performance data on the present ultrasound application as our implementation is being continuously improved with the target of real-time rates on currently available PC's. .

7. Acknowledgments

We wish to acknowledge the support of the Office of Naval Research under grants N00014-97-1-0243 & N00014-00-1-0281 and the support of the National Science Foundation under grant IIS 9980166. We also wish to thank Robert Rohling for sharing his data sets.

References

[1] D. Barry, C. P. Allott, N. W. John, P. M. Mellor, P. A. Arundel, D. S. Thomson and J. C. Waterton, "Three-Dimensional Freehand Ultrasound: Image Reconstruction and Volume Analysis", *Ultrasound in Medicine and Biology*, vol. 23, no. 8, pp. 1209-1224, 1997.

[2] R. E. Bank, A. H. Sherman and A. Weiser, "Refinement Algorithms and Data Structures for Regular Local Mesh Refinement", R. Stepleman, ed., *Scientific Computing*, pp.3-17, Amsterdam, 1983.

[3] J. Bey, "Tetrahedral Mesh Refinement", *Computing,* vol. 55, pp. 355-378.

[4] A. Fenster, and D. B. Downey, "3-D Ultrasound Imaging: A Review", *IEEE Engineering in Medicine and Biology*, vol. 15, pp. 41-51, Nov. 1996.

[5] R. Grosso, C. Lurig and T. Ertl, "The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data", *Proceedings of Visualization 97*, Phoenix, AZ, 1997.

[6] M. L. James, G.M. Smith, J. C. Wolford, *Applied Numerical Methods for Digital Computation*, 3rd Edition, Harper & Row, 1985.

[7] J. M. Maubach, "Local Bisection Refinement for *N*-Simplicial Grids Generated by Reflection", SIAM Journal of Scientific Computing, vol. 16, no. 1, pp. 210-227, 1995.

[8] T. R. Nelson and T. T. Elvins, "Visualization of 3D Ultrasound Data", IEEE Computer Graphics and Applications, vol. 13, pp. 50-57, 1993.

[9] T. R. Nelson, Pretorious, A. Fenster and D. B. Downey, *Three Dimensional Ultrasound*, Lippincott Williams & Wilkins, 1999.

[10] G. M. Nielson, "Scattered Data Modeling", *Computer Graphics and Applications*, Vol. 13, N. 1, January 1993, pp. 60-70.

[11] G. Nielson, "Tools for Triangulations and Tetrahedrazations", *Scientific Visualization*, G. Nielson, H. Hagen, and H. Muller, eds., pp. 429-526. Academic Press, 1997.

[12] G. M. Nielson, "Volume Modelling", in *Volume Graphics*, Min Chen, Arie Kaufman and Roni Yagel, eds, Springer, 1999, pp. 29-50.

[13] R. Ohbuchi, D. Chen and H. Fuchs, "Incremental Volume Reconstruction and Rendering for 3D Ultrasound Imaging", SPIE 1808, pp. 312-322, 1992.

[14] R. Prager and A. Gee, "The Stradx 3D Ultrasound Acquisition and Visualisation System", http://svr-www.eng.cam.ac.uk/~rwp/stradx/

[15] R. W. Prager, A. H. Gee and L. Berman, "3D Ultrasound Without Voxels", *Proceedings of Medical Image Understanding and Analysis*, pp. 93-96, 1998.

[16] M. C. Rivara, "Local Modification of Meshes for Adaptive and/or Multigrid Finite Element Methods", Journal of Computational and Applied Mathematics, vol 36, pp. 79-89, 1991.

G. M. Nielson, Tools for Triangulations and Tetrahedrizations, in: *Scientific Visualization: Overviews, Methodologies , Techniques*, G. Nielson, H. Hagen & H. Mueller, eds., CS Press, pp.429-525, 1997.

Tools for Triangulations and Tetrahedrizations

and

Gregory M. Nielson
nielson@asu.edu

1 Introduction
2 Triangulations
  2.1 Basics
    2.1.1 Definitions, Data Structures and Formulas for Triangulations
    2.1.2 Some Special Triangulations
  2.2 Optimal Triangulations
    2.2.1 Types and Characterizations
    2.2.2 Algorithms for Delaunay Triangulations
  2.3 Visibility Sorting of Triangulations
  2.4 Data Dependent Triangulations
  2.5 Affine Invariant Triangulations
  2.6 Interpolation in Triangles
    2.6.1 C^0, Discrete Interpolation in Triangles
    2.6.2 C^0, Transfinite Interpolation in Triangles
    2.6.3 C^1, Discrete Interpolation in Triangles
    2.6.4 C^1, Transfinite Interpolation in Triangles
3 Tetrahedrizations
  3.1 Basics
    3.1.1 Definitions, Data Structures and Formulas for Tetrahedrizations
    3.1.2 Some Special Tetrahedrizations
  3.2 Delaunay Tetrahedrizations
  3.3 Visibility Sorting of Tetrahedra
  3.4 Data Dependent Tetrahedrizations
  3.5 Affine Invariant Tetrahedrizations
  3.6 Interpolation in Tetrahedra
    3.6.1 C^0, Discrete Interpolation in Tetrahedra
    3.6.2 C^0, Transfinite Interpolation in Tetrahedra
    3.6.3 C^1, Transfinite Interpolation in Tetrahedra
    3.6.4 C^1, Discrete Interpolation in Tetrahedra

1 Introduction

This paper is about triangulations and tetrahedrizations. The original and main motivation was to provide some information about tetrahedra and tetrahedrizations only, but it was quickly realized that many of these topics are easier to describe and understand with some background on their two dimensional analogs. Therefore, it was decided to

also include material on triangulations. While much of the material exists elsewhere in the literature, much is new and appears here for the first time. The intended purpose for this paper is to serve as a survey/tutorial in the area of data modeling and visualization. As data modeling and visualization becomes more sophisticated in its application domains and begins to deal with data sets which are more complex than Cartesian grids, there will be the need for tools to deal with these data sets. We feel that the tools and techniques covered here are very basic and will prove to be useful in a variety of contexts in data visualization.

And now some comments about the organization of this paper. While tetrahedrizations are the goal, researchers have dealt with triangulations much longer than tetrahedrizations and so triangulations and related matters are much better understood. The next section is a survey of triangulations and related matters of interest in modeling and visualization. The following section is on tetrahedrizations and we attempt to follow the same flow of information as in the section on triangulations as best possible. We use the phrase "as best possible" because some aspects of triangulations do not generalize to tetrahedrization and some facts known about triangulations and triangular domains are yet to be known about tetrahedrizations and tetrahedral domains. On the other hand there are topics of interest to tetrahedrization which have no 2D counterpart of interest. For example, visibility sorting for tetrahedrizations. The outline of this paper is very simple. In Section 2 we go through a list of topics on triangulations and triangular domains and then in Section 3 we repeat these topics with reference to tetrahedrizations and tetrahedral domains.

2 Triangulations

2.1 Basics

2.1.1 Definitions, Data Structures, and Formulas for Triangulations

In order to avoid any possible confusion and problems latter, it is usually best to be a little precise and formal about the definition of a triangulation. We start with a collection of points in the plane, $\mathbf{P} = \{ p_i = (x_i, y_i), i = 1, \ldots, N\}$ and a domain of interest, D, which contains all of the points of P. We assume that the boundary of D is a simple (does not intersect itself), closed polygon. Often D is the convex hull of P, but in general, it need not be convex. In fact the boundary does not have to be a single polygon so that the domain is not even simply connected. (*Connected* means that there is path joining any two points and *simply connected* means that the compliment is connected.) Roughly speaking, a triangulation is a decomposition of D into a collection of triangles which are formed from vertices of P. Since we are eventually interested in defining functions over D in a piecewise manner over each triangle, we must require that the triangles do not overlap so as not to have any ambiguities. Thus we require the collection of triangles of the triangulation to be mutually exclusive and collectively exhaustive. In order to continue this formalism to a precise definition, we need some additional notation. A single triangle with vertices p_i, p_j and p_k is denoted by T_{ijk} and the list of triples which represents the triangulation is denoted by I_t. A triangle T_{ijk} is a closed 2D

point set that includes its three edges which comprise it boundary. The interior of T_{ijk}, denoted by $Int(T_{ijk})$ is open and does not include the boundary. The edge joining p_i and p_j is denoted by e_{ij} and $N_e = \{ij : ijk \text{ in } I_t \text{ for some } k\}$ is used to refer to the collection of all edges. Formally, the definition of a triangulation requires:

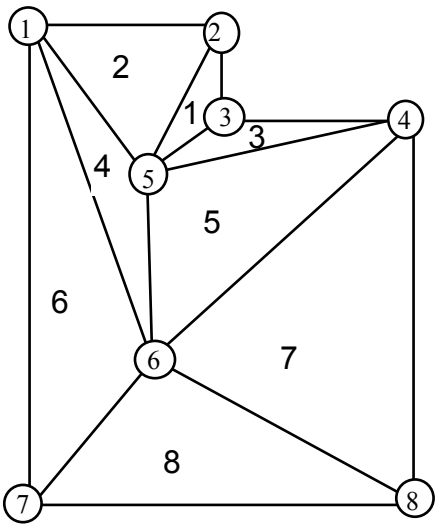
i) No triangle T_{ijk} , $ijk \in I_t$ is degenerate. That is, if $ijk \in I_t$ then p_i, p_j and p_k are not collinear.

ii) The interior of any two triangles do not intersect. That is if $ijk \in I_t$ and $\alpha\beta\gamma \in I_t$ then $Int(T_{ijk}) \cap Int(T_{\alpha\beta\gamma}) = \phi$.

iii) The boundary of two triangles can only intersect at a common edge.

iv) The union of the all triangles is the domain $D = \cup_{ijk \in It} T_{ijk}$.

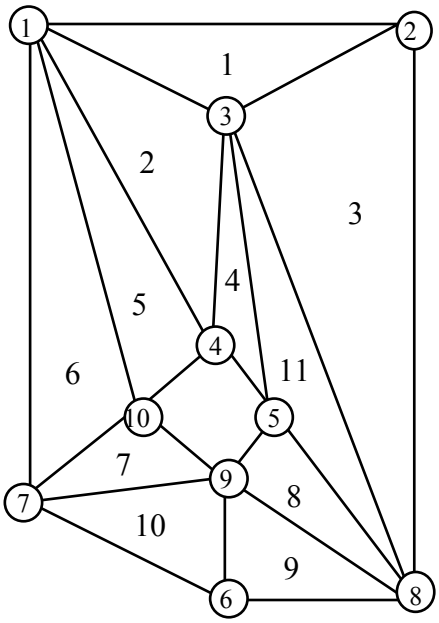
Examples of valid triangulations are shown in Figure 2.1.1 and Figure 2.1.2. Note that the example of Figure 2.1.1 is not convex and that of 2.1.2 is not simply connected. Even though the diagrams of Figure 2.1.3 and Figure 2.1.4 look alright, the actual triangulation given by the corresponding I_t's do not represent valid triangulations. In the case of Figure 2.1.3 the triangle T_{465} is degenerate. Even if this triangle is eliminated, what remains is not a valid triangulation because condition iii) would then be violated since edge e_{46} contains p_5. This example would become a valid triangulation if the point p_5 were to be moved slightly to the right so as not to be on the edge e_{46}. The information of Figure 2.1.4 is not a valid triangulation because condition ii) is violated.



I_t

i	j	k
3	5	2
5	1	2
4	5	3
6	1	5
4	6	5
6	7	1
8	6	4
7	6	8

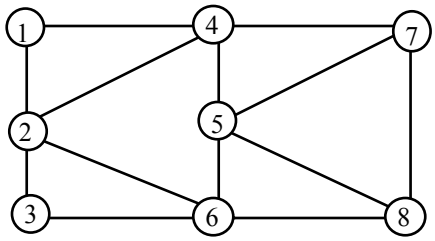
Figure 2.1.1. A triangulation of a non convex domain



I_t

i	j	k
3	1	2
4	1	3
2	8	3
5	4	3
10	1	4
7	1	10
7	10	9
8	9	5
6	9	8
6	7	9
5	3	8

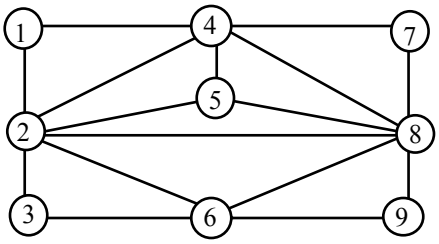
Figure 2.1.2. A triangulation of a domain which is not simply connected.



I_t

i	j	k
1	4	2
2	4	6
2	6	3
4	6	5
4	7	5
5	7	8
5	8	6

Figure 2.1.3. Not a valid Triangulation.



I_t

i	j	k
1	4	2
2	4	5
2	5	8
4	8	5
4	7	8
2	4	8
2	6	3
2	8	6
8	9	6

Figure 2.1.4. Not a valid triangulation.

We now want to make some assertions about the possibility of triangulating a domain containing a collection of data points that is bounded by a simple, closed polygon. First we note that in the case the domain contains no interior data points it is always possible to form a triangulation. Just for the sake of interest, we mention two ways that this can be accomplished. The first way is based upon the fact that every simple closed polygon with more that three vertices can be split into two polygons. This leads to an algorithm which recursively splits each subpolygon until only triangles are left. The following argument which guarantees that each simple closed polygon has a diagonal has been discussed in [16]. A diagonal is an edge between two vertices that lies inside the polygon and does not intersect the polygon except at the endpoints.

Splitting a polygon: Let b be the vertex with minimum x-coordinate and ab and bc be its two incident edges. If ac is not cut by the polygon, then ac is a diagonal. Otherwise there must be at least one polygon vertex inside T_{abc}. Let d be the vertex inside abc furthest from the line through a and c. Now edge bd cannot be cut by the polygon, since any edge intersecting bd must have one endpoint further from line ac.

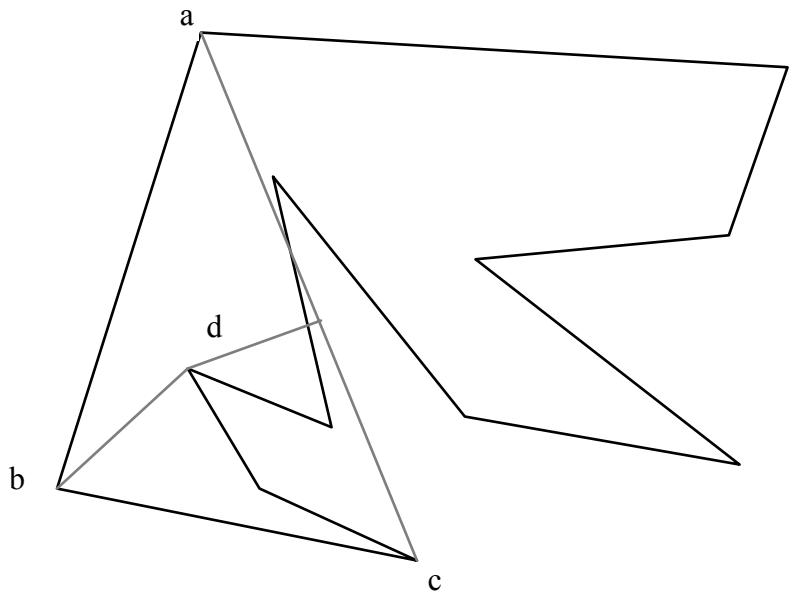


Figure 2.1.5. Any polygon with more than three vertices can be split.

The second approach leads to an iterative algorithm. We first give a definition. A vertex, p_i, of a simple, closed polygon is called *protruding* provided the following conditions hold:

i) The interior angle Θ_i, between the edges, $p_{i-1}p_i$ and p_ip_{i+1} is less than or equal to π. (Cyclic notation is used here so that $p_{N+1} = p_1$)

ii) The triangle $T_{i-1,i,i+1}$ contains no other vertices of the polygon than p_{i-1}, p_i or p_{i+1}.

iii) The interior of $T_{i-1,i,i+1}$ is contained in the interior of D.

It is an easy matter to prove that every simple, closed polygon has at least one protruding vertex. (The proof is left to the reader. Some people call them ears and so there must be two of them!). We can triangulate the polygon bounded domain by successively removing protruding vertices. This approach to triangulating the region bounded by a simple closed polygon is called the "boundary stripping algorithm." It is easy to implement, but in a theoretical sense, it is not competitive with other algorithms (see, for example, the papers of Narkhede & Manocha [175] and Fournier & Montuno [94] among others.).

Once the boundary of D has been triangulated, it is relatively simple matter to build a triangulation including the interior points. This can be done by simply inserting them sequentially in a manner which we now describe:

Insertion of an interior point: If the point to be inserted, p, lies in the interior of the triangle T_{abc}, we replace T_{abc} with the three triangles: T_{abp}, T_{bcp}, T_{cap}. If p lies on an edge shared by T_{abc} and T_{bad}, then replace the two triangles T_{abc} and T_{bad} with the four triangles T_{bcp}, T_{dbp}, T_{pca}, T_{pad}.

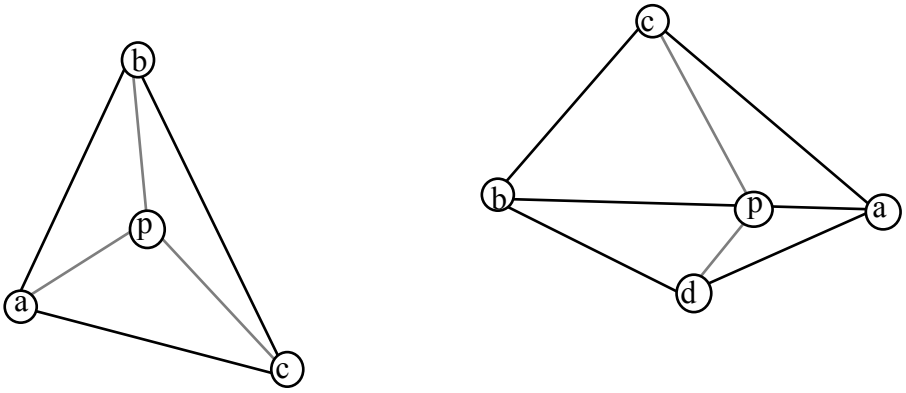


Figure 2.1.6. Insertion of an interior point.

It is also possible to generalize the insertion idea to include an edge. Once we are armed with this capability, we know that we can triangulate any polygon bounded domain: simple connected or multiply connected (i.e. with holes).

Insertion of an interior edge: Assume that the one endpoint, p, lies in the triangle T_{abc} and that the other endpoint, q, lies in the triangle T_{xyz}. Collect all of the triangles from T_{abc} to T_{xyz} which are intersected by edge pq and form a region R with polygon boundary D. We can split D with polygon apqw, where a is the vertex of T_{abc} not on the edge common with the other triangles whose union is R and w is the analogous vertex of T_{xyz}. Now we know that each of these two domains can be triangulated. The union of these two triangulation, which each contain the edge pq, can replace the previous triangulation of D.

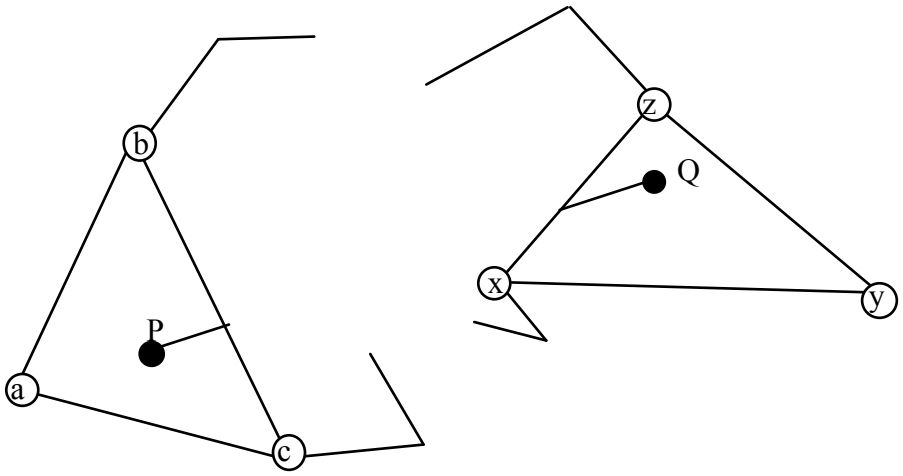
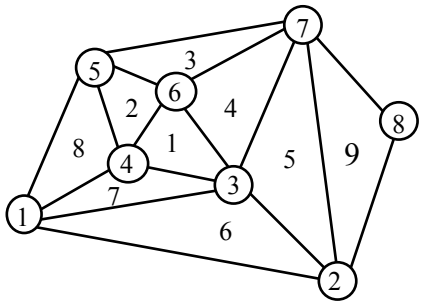


Figure 2.1.7. Insertion of an interior edge.

In addition to I_t, which represents the triangulation, it is often worthwhile to generate and maintain some auxiliary information about the neighbors of each triangle. This information is useful for traversal algorithms and evaluation algorithms which have a searching component that determines the particular triangle containing a point where a function defined piecewise over the triangulation is to be evaluated. One very common and particularly useful data structure is that which is illustrated in Figure 2.1.8. The first three columns contain the data of I_t with the additional constraint that the reading from left to right (cyclically), the vertices of each triangle are traversed in a clockwise order. The next three columns contain the indices of the triangles which are neighbors to this triangle. The character ϕ indicates that the triangle has an edge that is part of the boundary of D. The entries of these three columns are also in a special order. The forth column contains the index of the triangle which shares the common edge with vertex indices specified in the second and third column. Similar relationships hold for the 5th and 6th columns. The information represented by this data structure is called a "triangular grid". The neighborhood information contained in the last three columns does not contain any "new" information over that of I_t, but it is often (and this depends of course on the application) the case that it is useful data which is worth generating a priori.



Triangles			Neighbors		
p_i	p_j	p_k	N_{jk}	N_{ki}	N_{ij}
3	4	6	2	4	7
4	5	6	3	1	8
6	5	7	ϕ	4	2
3	6	7	3	5	1
2	3	7	4	9	6
2	1	3	7	5	ϕ
1	4	3	1	6	8
1	5	4	2	7	ϕ
2	7	8	ϕ	ϕ	5

Figure 2.1.8. An example that defines a *triangular grid structure*.

Another data structure for representing a triangulation which is useful for some applications is illustrated by the example shown in Figure 2.1.9 which represents the same triangulation as that of Figure 2.1.8. Here, for each vertex a list of all vertices which are joined by an edge of the triangulation is given. This list is given in counter clockwise order around each vertex. This is called the *data point contiguity list*. We mention this particular data structure because of its convenience for dealing with the optimal Delaunay triangulation discussed in the next section. Also, it is very useful for computing the parameters of the Minimum Norm Network method [179] which is one of the most effective C^1 interpolation methods for scattered data.

Vertex	Joining Vertices
1	2, 3, 4, 5
2	8, 7, 3, 1
3	1, 2, 7, 6, 4
4	3, 6, 5, 1
5	1, 4, 6, 7
6	3, 7, 5, 4
7	6, 3, 2, 8, 5
8	2, 7

Figure 2.1.9. The data that defines the *data point contiguity* list.

Even though there are a number of possible triangulations for any given domain D, the number of triangles is fixed once the boundary has been specified. More precisely, if N_b represents the number of vertices on the boundary and N_i the number of interior vertices so that $N = N_b + N_i$, then the following formulas hold:

$$N_t = 2N_i + N_b - 2$$

and

$$N_e = 3N_i + 2N_b - 3,$$

where N_t is the total number of triangles and N_e is the total number of edges. The importance of these formulas (not so much what the values in the formulas are, but more the fact that some fixed formula holds) will show up in the next section. If we let M_i represent the number of points joining to p_i then it is easy to see that

$$\sum_{i=1}^{N} M_i = 2N_e$$

and so we have that the "average valence" of a point is given by

$$\bar{M} = \frac{\sum_{i=1}^{N} M_i}{(N_i + N_b)} = 6 - 2\frac{(N_b + 3)}{N}$$

which is approximately 6. For a sphere (or any domain homeomorphic to a sphere) we have no boundary points and so $N = N_i$ and the analogous formulas are

$$N_t = 2(N - 1), \qquad N_e = 3(N - 1), \qquad \bar{M} = 6 \textbf{ Error!}) .$$

2.1.2 Some Special Triangulations

One of the simplest triangulations results from splitting the rectangles of a Cartesian grid. A Cartesian grid involves two monotonically increasing sequences, x_i, $i = 1, \ldots, n$ and y_j, $j = 1, \ldots, m$. The grid points have coordinates (x_i, y_j) and these points mark out a cellular decomposition of the domain consisting of rectangles. See Figure 2.1.10. Forming an edge with one of the diagonals of these rectangular cells leads to a triangulation of the domain. In Figure 2.1.11 is shown a triangulation where a consistent choice for the diagonal is made. In Figure 2.1.12 is shown a triangulation with mixed choices for the diagonals. In some applications where dependent ordinate values are known, it is possible to base the choice of the diagonal upon some criteria such as minimum jump in normal vector (see Section 2.4) or whether or not the diagonal vertices are separated on connected based upon the hyperbolic contours at the mean value (see the asymptotic decider criteria discussed in [186]). In general for this type of triangulation which results from a Cartesian grid, it is not necessary to maintain the triangular grid structure (see Figure 2.1.8) as this information can be directly inferred from the natural

labeling of $p_{ij} = (x_i, y_j)$. Only the information which indicates which diagonal is selected needs to be made available.

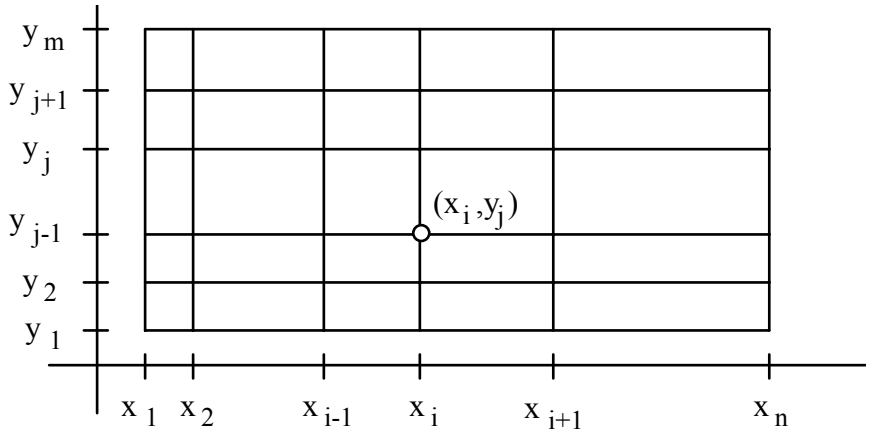


Figure 2.1.10. Cartesian Grid.

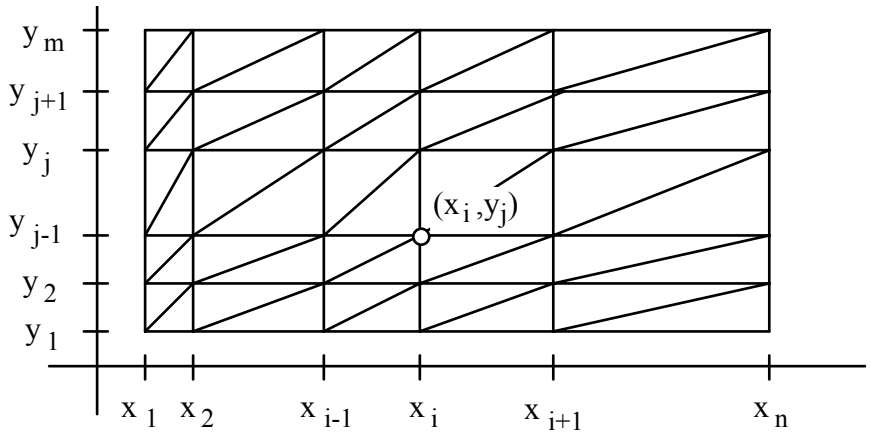


Figure 2.1.11. Triangulation from Cartesian grid with uniform diagonal choice.

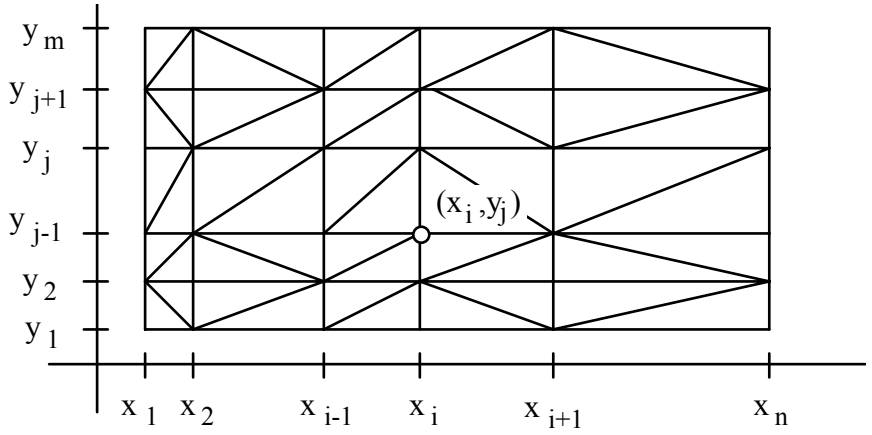


Figure 2.1.12. Triangulation from Cartesian grid with mixed diagonals.

We now want to discuss some special triangulations which result from curvilinear grids. A curvilinear grid is specified with two "geometry arrays" (x_{ij}, y_{ij}), i = 1, . . . , M; j = 1, . . . , N. A cell C_{ij} consists of the quadrilateral with the boundary delineated by (x_{ij}, y_{ij}) to (x_{i+1j}, y_{i+1j}) to (x_{ij}, y_{ij}) back to (x_{ij}, y_{ij}). It is assumed that these four points form a simple (non intersecting) polygon so that the quadrilateral is actually well-defined. This condition obviously puts some geometric constraints on the values of the geometry arrays that specify a curvilinear grid.

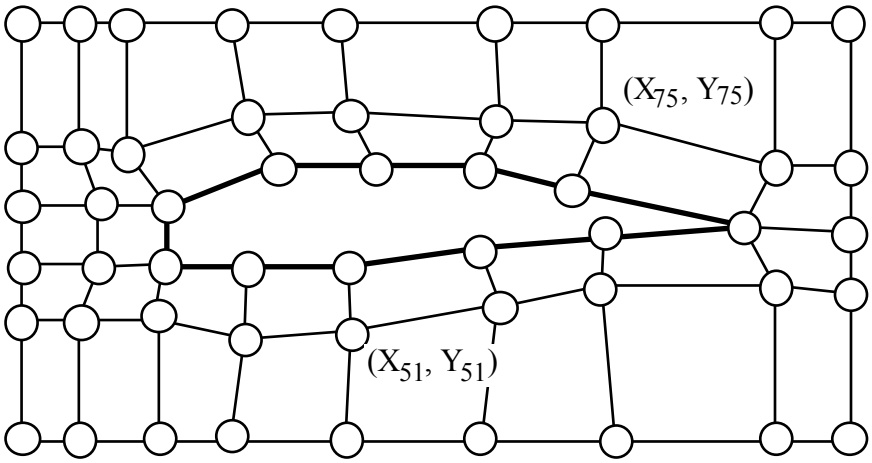


Figure 2.1.13. An example of a curvilinear grid.

An example of a curvilinear grid is shown in Figure 2.1.13. In this case the cell C_{73} degenerates to a triangle because (X_{83}, Y_{83}) and (X_{84}, Y_{84}) are the same point and the cell C_{83} degenerates to an edge because, in addition, (X_{93}, Y_{93}) and (X_{94}, Y_{94}) are the same point. The cells C_{33}, C_{43}, C_{53}, C_{63} and C_{73} have been removed from the domain creating the hole in the interior.

The domain (the union of all of its cells) can be triangulated by simply triangulating each of the cells by choosing a diagonal to an edge of the triangulation. An example related to the grid of Figure 2.1.13 is shown in Figure 2.1.14. Here we have modified the grid by moving the point (x_{72}, y_{72}) a little. This serves to point out that if the cell is not convex, then there may be only one choice for the diagonal.

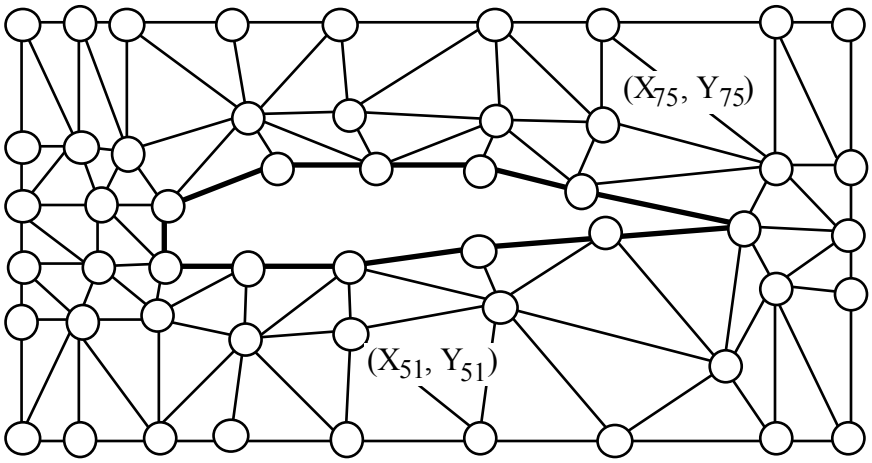


Figure 2.1.14. Triangulation resulting from curvilinear grid.

We now discuss some special triangulations obtained by subdividing an existing triangulation. We briefly mention a couple of possibilities. The first is based upon inserting an additional point into the interior of an existing triangle and thereby forming three new triangles. This is illustrated in Figure 2.1.15. This particular type of subdivision is sometimes referred to as the Clough-Tocher split because of its association with a very well known finite element shape function defined over a triangular domain.

Another way to subdivide an existing triangulation is to insert a new point on an existing edge and split the two triangles (unless the edge is on the boundary) which share this edge. If all edges are split simultaneously we obtain yet another triangulation where each previous triangle is replaced by four new ones. Two different ways for forming triangles from these points is shown in Figure 2.1.16 and Figure 2.1.17 respectively. These types of subdivision are particularly interesting due to the nested properties of function spaces which are defined in a piecewise manner over the embedded subdivisions. This can lead to wavelets and their related multiresolution analysis. For the efficient application of these triangulations, it is important to have a method of labeling the triangles which allows an efficient algorithm for finding the labels of all neighbors of a triangle. The labeling scheme illustrated in Figure 2.1.17 has these properties. We call it the *divide and flip* scheme and have found it to be very useful for implementations. It is related to the spherical quadtrees discussed by Fekete [85].

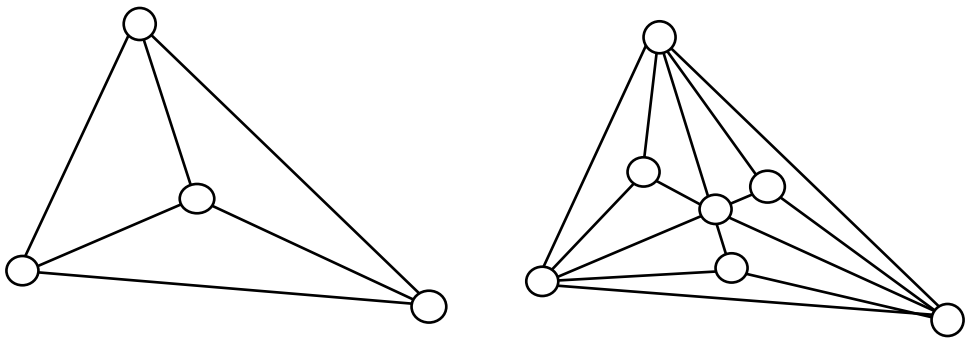


Figure 2.1.15. Subdivision by inserting a new point that is interior to an existing triangle.

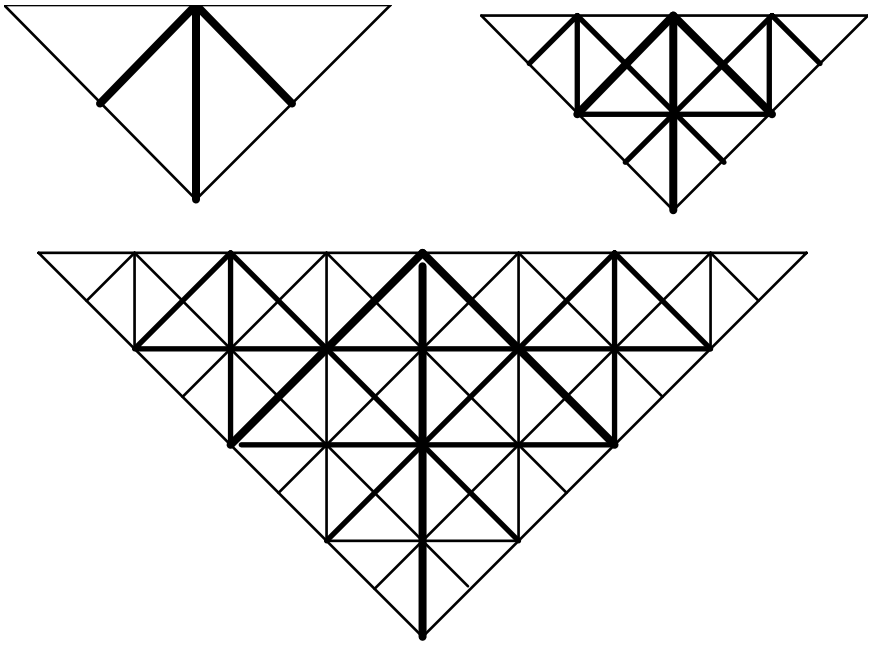


Figure 2.1.16. Nested subdivision triangulation.

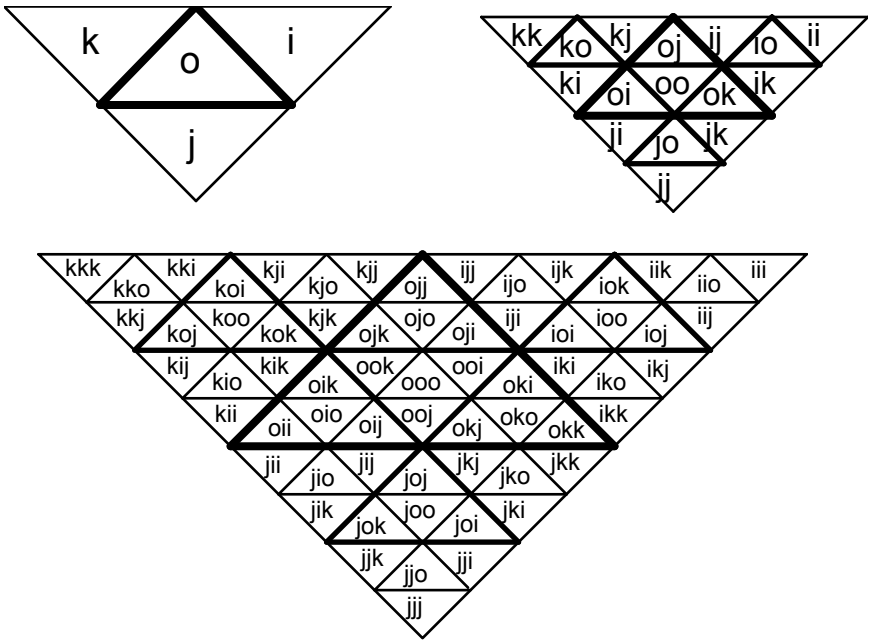


Figure 2.1.17. The *divide and flip* labeling scheme for a nested subdivision triangulations.

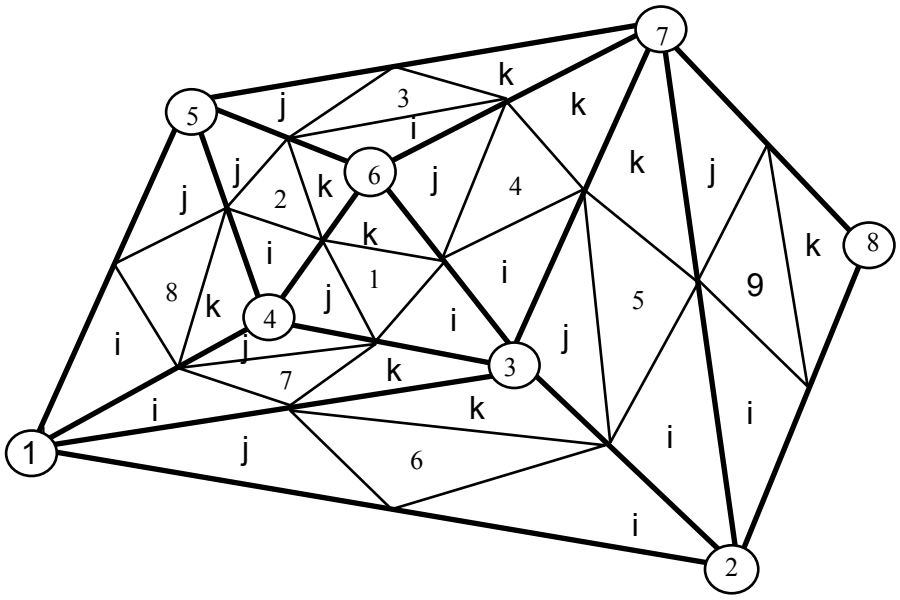


Figure 2.1.18. A triangulation obtained by splitting each edge of an existing triangulation and forming triangles as indicated in Figure 2.1.17.

2.2 Optimal Triangulations

2.2.1 Types and Characterizations

Figure 2.2.1. Examples of poorly shaped triangles

There are many possible triangulations of a given, polygon bounded domain D. For some applications (but not all) it is desirable to avoid poorly shaped triangles. These are triangles with very large angles or ones with very small angles. This give rise to two types of optimal triangulations which have been discussed quite widely: the MaxMin and MinMax. Both of these optimal triangulations have a similar method of characterization. Associated with each triangulation there is a vector with N_t entries representing either the largest or smallest angle of each triangle. The entries of each vector are ordered and then a lexicographic ordering of the vectors is used to impose an ordering on the set of all triangulations. In the case of the MinMax criterion, A_i is the largest angle of a triangle and the entries of each vector, A_t, are ordered so that

$$A_t = (A_1, A_2, \ldots, A_{n_t}),\ A_i \geq A_j,\ i < j.$$

The smallest of these vectors based on their lexicographic ordering associates with the optimal triangulation. In the case of the MaxMin criteria, a_i, is the smallest angle and the entries of each vector are ordered the other way so that

$$a_t = (a_1, a_2, \ldots, a_{n_t}),\ a_i \le a_j,\ i < j.$$

The largest of these vectors represents the optimal triangulation in the MaxMin sense. In Figure 2.2.2, six data points are shown which have a total of ten possible triangulations which are shown in Figure 2.2.3. The associated vectors for MinMax criterion are

$$\begin{aligned}
A_{\tau 0} &= (2.84, 2.36, 1.99, 1.77, 1.57)\\
A_{\tau 1} &= (2.98, 2.84, 1.99, 1.91, 1.57)\\
A_{\tau 2} &= (2.98, 2.42, 1.91, 1.88, 1.57)\\
A_{\tau 3} &= (2.84, 2.36, 2.32, 1.99, 1.40)\\
A_{\tau 4} &= (2.42, 2.36, 1.88, 1.77, 1.57)\\
A_{\tau 5} &= (2.98, 2.42, 1.95, 1.91, 1.27)\\
A_{\tau 6} &= (2.42, 2.36, 2.32, 1.88, 1.40)\\
A_{\tau 7} &= (2.42, 2.36, 2.32, 1.50, 1.50)\\
A_{\tau 8} &= (2.42, 2.36, 1.95, 1.74, 1.50)\\
A_{\tau 9} &= (2.42, 2.36, 1.95, 1.77, 1.27)
\end{aligned}$$

which we rearrange into decreasing order to obtain

$$\begin{aligned}
A_{\tau 1} &= (2.98, 2.84, 1.99, 1.91, 1.57)\\
A_{\tau 5} &= (2.98, 2.42, 1.95, 1.91, 1.27)\\
A_{\tau 2} &= (2.98, 2.42, 1.91, 1.88, 1.57)\\
A_{\tau 3} &= (2.84, 2.36, 2.32, 1.99, 1.40)\\
A_{\tau 0} &= (2.84, 2.36, 1.99, 1.77, 1.57)\\
A_{\tau 6} &= (2.42, 2.36, 2.32, 1.88, 1.40)\\
A_{\tau 7} &= (2.42, 2.36, 2.32, 1.50, 1.50)\\
A_{\tau 9} &= (2.42, 2.36, 1.95, 1.77, 1.27)\\
A_{\tau 8} &= (2.42, 2.36, 1.95, 1.74, 1.50)\\
A_{\tau 4} &= (2.42, 2.36, 1.88, 1.77, 1.57)
\end{aligned}$$

which implies the following ordering

$$\tau_4 < \tau_8 < \tau_9 < \tau_7 < \tau_6 < \tau_0 < \tau_3 < \tau_2 < \tau_5 < \tau_1$$

and so τ_4 is the optimal triangulation in MinMax sense. On the other hand, the associated vectors for MaxMin criteria sorted in increasing order are

$$\begin{aligned}
a_{\tau 1} &= (0.02, 0.04, 0.35, 0.46, 0.50)\\
a_{\tau 2} &= (0.02, 0.11, 0.42, 0.46, 0.50)
\end{aligned}$$

$$a_{\tau 5} = (0.02, 0.11, 0.50, 0.58, 0.88)$$
$$a_{\tau 3} = (0.04, 0.14, 0.35, 0.37, 0.66)$$
$$a_{\tau 0} = (0.04, 0.14, 0.35, 0.46, 0.62)$$
$$a_{\tau 6} = (0.11, 0.14, 0.37, 0.42, 0.66)$$
$$a_{\tau 7} = (0.11, 0.14, 0.37, 0.46, 0.70)$$
$$a_{\tau 4} = (0.11, 0.14, 0.42, 0.46, 0.62)$$
$$a_{\tau 8} = (0.11, 0.14, 0.57, 0.58, 0.70)$$
$$a_{\tau 9} = (0.11, 0.14, 0.58, 0.62, 0.88)$$

which results in the following ordering

$$\tau_1 < \tau_2 < \tau_5 < \tau_3 < \tau_0 < \tau_6 < \tau_7 < \tau_4 < \tau_8 < \tau_9$$

and so τ_9 is the optimal triangulation in the case of the MaxMin criterion.

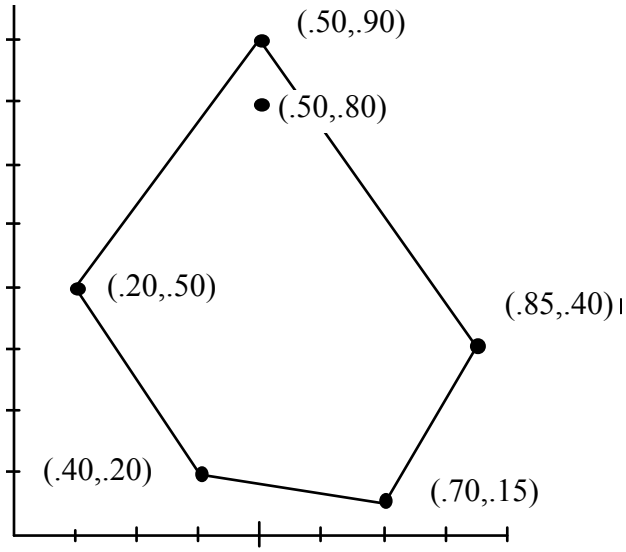


Figure 2.2.2. Six data points.

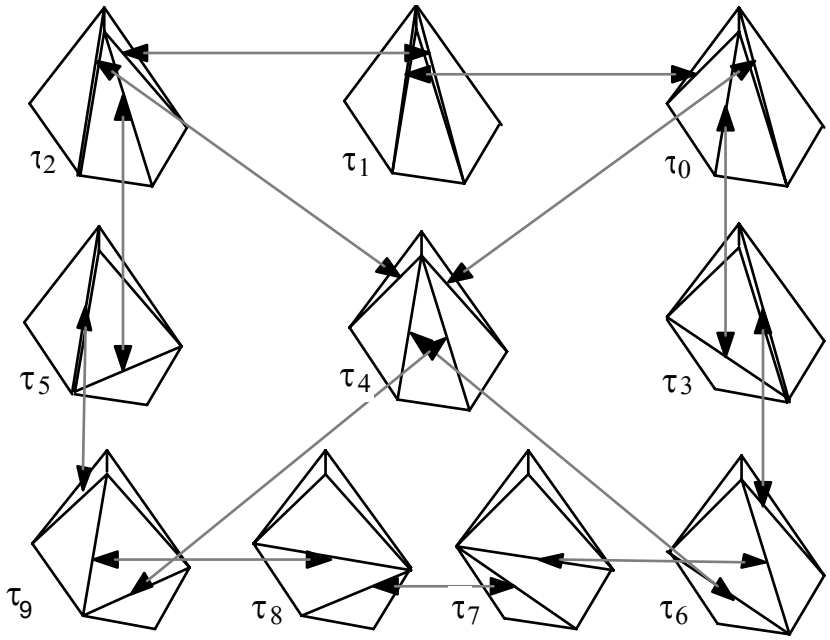


Figure 2.2.3. Ten triangulations of six data points.

In the case where D is the convex hull of the points of P, there is an important relationship between the MaxMin triangulation and the Dirichlet tessellation. The Dirichlet tessellation is a partition of the plane into regions R_i, i = 1, . . . , N called Thiessen regions. The Thiessen region R_k consists of all points in the plane whose closest point among p_i, i = 1, . . . , n is p_k. A Dirichlet tessellation is usually illustrated by drawing the boundaries of the Thiessen regions. The collection of these edges is sometimes referred to as the Voronoi diagram (see [252]) An example is shown in the left image of Figure 2.2.6. In the right image of Figure 2.2.6 is shown the MaxMin triangulation which is also called the Delaunay triangulation. It is dual to the Dirichlet tessellation in that the edges of this optimal triangulation join vertices which share a common Thiessen region boundary. We have included the great circles in the left image of this figure so as to point out another important property of the Dirichlet tessellation and its companion Delaunay triangulation. By definition, the edges of the Thiessen regions meet at triads (possibly more than three edges meet in some special, neutral/cyclic cases) which are equally distant to three points. These three points will form a triangle of the optimal triangulation and the great circle will not contain any other data points.

We can be a little more formal about this properties if we introduce some notation. Recall that I_t = { (i(m), j(m), k(m)), m = 1, ... , Nt} so that the three data points $p_{i(m)}$, $p_{j(m)}$, $p_{k(m)}$ will be the vertices of a triangle of the triangulation. We assume that the neighbor information of the triangular grid is given by three arrays ij(m), jk(m), and ki(m), m = 1, . . . , N_t. Let V_m be the point which is equidistant from $p_{i(m)}$, $p_{j(m)}$ and $p_{k(m)}$ and C_m = { p : $||p - V_m|| \le ||V_m - p_{a(m)}||$, a = i, j or k} be the circumcircle (disk) for this triangle which has V_m as it center. The Delaunay triangulation is characterized by the fact that C_m does not contain any other data points p_i, i = 1, . . . , N other than $p_{i(m)}$,

$p_{j(m)}$ and $p_{k(m)}$. The points V_m are the vertices of the Voronoi diagram. In order to draw the Voronoi diagram we simply start with some V_m and draw the edges to the three points that are joined to it; namely $V_{ij(m)}$, $V_{jk(m)}$ and $V_{ki(m)}$. If anyone of ij(m), jk(m) or ki(m) is zero (say ij(m), indicating the edge joining $p_{i(m)}$ and $p_{j(m)}$ is on the boundary of the convex hull) then we draw the ray emanating from V_m in the direction perpendicular to the appropriate edge (which is $p_{i(m)}p_{j(m)}$ if ij(m)=0, $p_{j(m)}p_{k(m)}$ if jk(m) = 0 and $p_{k(m)}p_{i(m)}$ if ki(m) = 0). If we go through the list of triangles and draw three edges for each V_m we will actually be drawing each edge (not each ray) twice. We can avoid this duplication by (for example) testing whether or not m > ij(m), m > jk(m), m > ki(m) before we draw the corresponding edge.

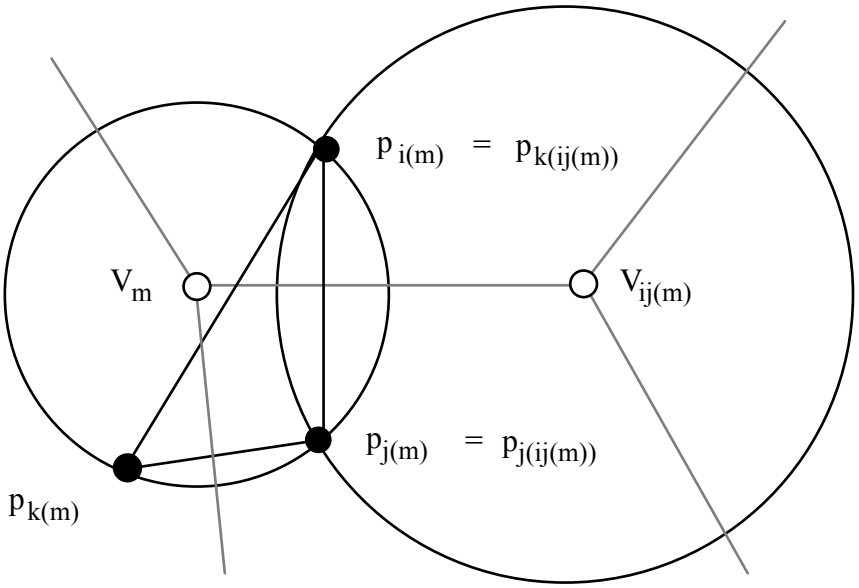


Figure 2.2.4. Drawing the Dirichlet tessellation from the triangular grid structure.

Because of this relationship between the Dirichlet tessellation and the optimal MaxMin triangulation, we can extend the idea of MaxMin or Delaunay triangulation to any domain where we can compute the distance between two points. The sphere provides an interesting and useful example. Here the distance between two points p and q is easily computed as $\cos^{-1}(p \bullet q)$ so the Dirichlet tessellation is also easy to compute. An example is shown in the right images of Figure 2.2.5. The left image depicts the triangulation which is dual to this tessellation.

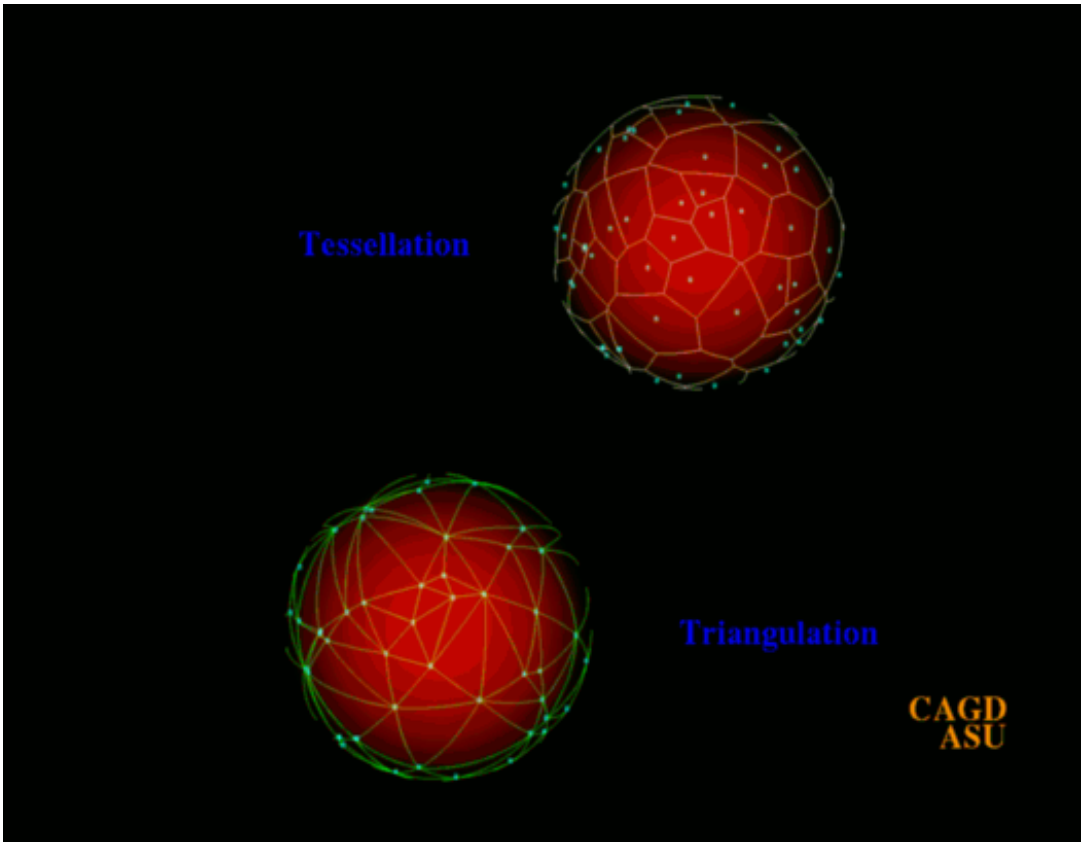


Figure 2.2.5. Spherical triangulation and tessellation

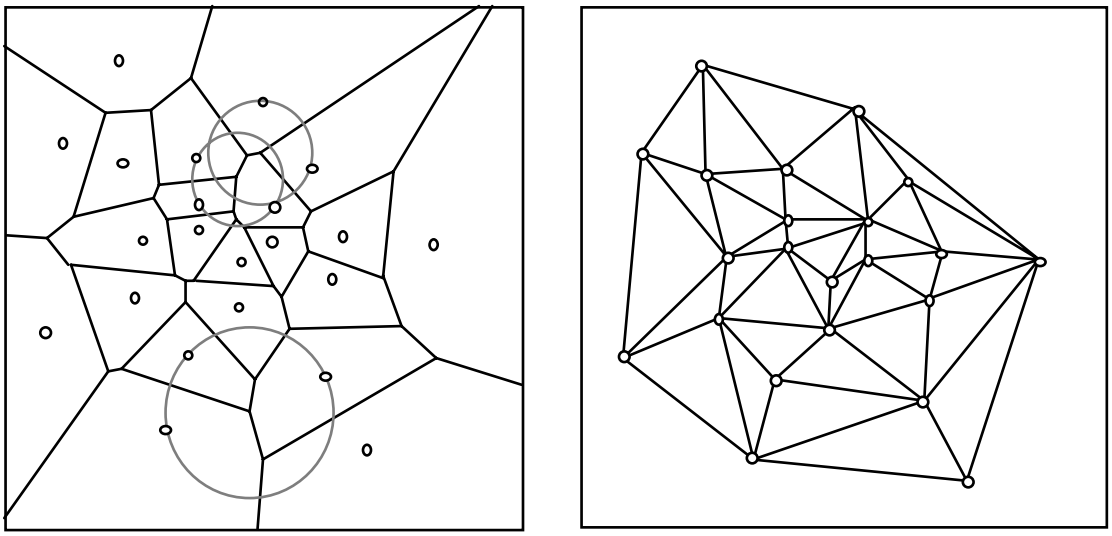


Figure 2.2.6. The Dirichlet tessellation and its dual triangulation.

There have been many other criteria for characterizing optimal triangulations that have been studied and discussed in the literature. Some turn out to be equivalent to those

we have mentioned here and some only appear to be similar and so one needs to be rather careful. Even though the terminology can be similar, the criterion of minimizing the maximum angle is not the same as the MinMax criterion we have described here. It is easily the case the two quite different triangulations with different vectors A_t (as defined above) could have the same maximum angle and could both be a triangulation which minimizes the maximum angle. The example of Figure 2.2.2 has this property. Each of the triangulations τ_6 , τ_7 , τ_9 , τ_8 and τ_4 have a maximum angle of 2.42 which turns out to be a minimum and so any one of these triangulations would satisfy the criterion of minimizing the maximum angle, while only τ_4 satisfies MinMax criterion described here. Overall, the topic of optimal triangulations can be rather technical and one has to be careful when comparing results found in the literature.

2.2.2 Algorithms for Delaunay Triangulations

In this section we discuss some ideas and techniques leading to algorithms for computing the Delaunay triangulation of a set of points in the plane. In general, this is a very rich and full area of research and here we can only provide a glimpse. The literature is very abundant with both practical and theoretical papers on this subject. There is not a single "best" algorithm. The choice depends upon the particular application and the tools and resources available. It is a good strategy to be armed with a collection of ideas, tools and techniques so that an effective algorithm can be custom designed for the application at hand. Our approach for the material for this section is based upon a discussion of the ideas behind a few number of selected algorithms. Our selection is based upon potential usefulness of the ideas and also what would be representative. In addition, we particularly interested in those ideas which extend most easily to three dimensions. But, just for the sake of interest, we have included the description of one 2D algorithm which does not extend at all to 3D!

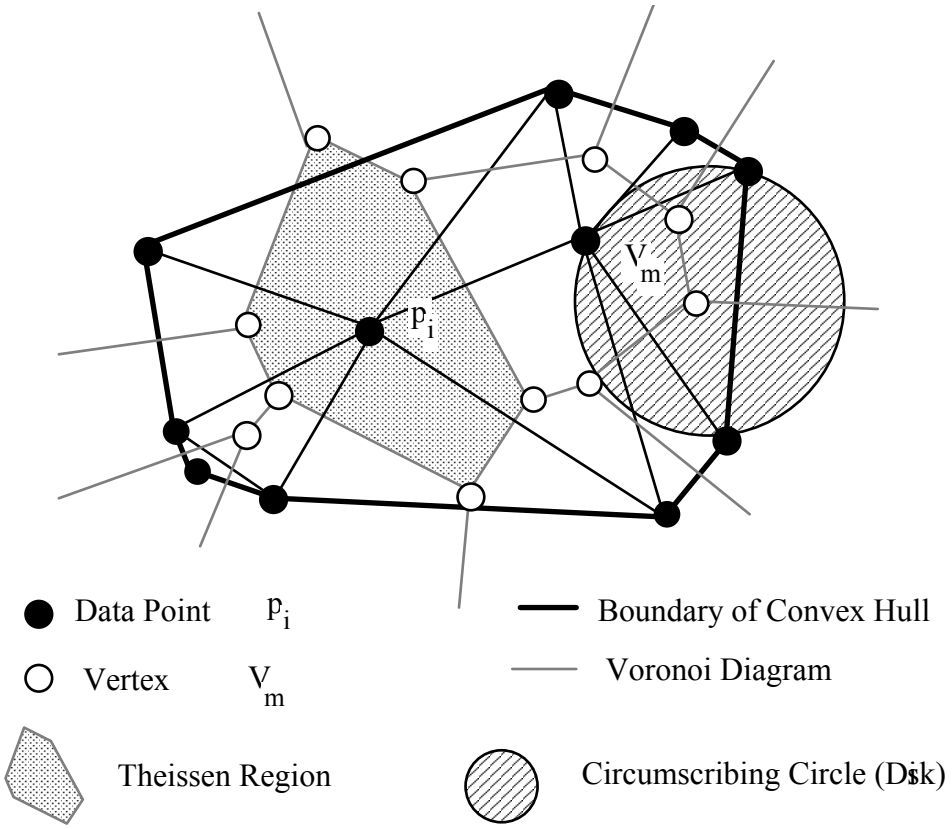


Figure 2.2.7. Notation and terminology for Delaunay triangulation and Dirichlet tessellation.

The Swapping Algorithm of Lawson [139]: The basic operation of this algorithm consists of swapping the diagonal of a convex quadrilateral. Lawson [138] showed that any triangulation of the convex hull can be obtained from any other triangulation by a sequence of these operations. (Later this property was established for nonconvex domains by Dyn and Goren [66].) Furthermore, Lawson proved that if the choice of the diagonal is made on the basis of the MaxMin criterion for the quadrilateral only, eventually the global optimal triangulation will be obtained. In other words, for this criterion, a local optimum is a global optimum. A typical implementation of this type of algorithm would insert new points (say in sorted x-order) in the interior of an existing triangulation or connect to all points on the boundary which are visible from the new point. This new triangulation is then optimized by testing and possibly swapping the diagonals of convex quadrilaterals. It is interesting to note that this type of algorithm will not necessarily produces the MinMax because for this criterion, a local extreme is not necessarily a global optimum. The example of Figure 2.2.2 of the previous section illustrates this. Based upon the MinMax criterion, τ_4 is optimal and τ_8 is a local minimum. Locally optimal swaps of diagonals from τ_8 would never lead to τ_4. The algorithm could easily get trapped in a local extreme at τ_8. The ideas of simulated annealing can be used to develop algorithms which can escape from these local extrema. See Schumaker [225] for example.

The Algorithm of Green & Sibson [107]: This algorithm depends heavily upon a particular data structure used to store the Delaunay triangulation (or Dirichlet tessellation). For each object (a Dirichlet tile or window boundary constraint) is recorded in a "contiguity list" consisting of all objects with which it is contiguous. This data structure is very similar to the contiguity list structure we described in Figure 2.1.9 but it also includes some window boundary constraints. New points are inserted sequentially. We quote directly from [107] as to how this done.

> The contiguity list for the new point is then built up in reverse (that is, clockwise) order and subsequently standardised. We begin by finding where the perpendicular bisector of the line joining the new point to its nearest neighbour meets the edge of the nearest neighbour's tile, clockwise round the new point. Identifying the edge where this happens gives the next object contiguous with the new point and this is in fact the first to go onto its contiguity list. The new perpendicular bisector is then constructed and its incidence on the edge of this new tile is examined to obtain the subsequent contiguous object: successive objects are added to the contiguity list in this way until the list is completed by the addition of the nearest neighbour. Whilst this being done old contiguity lists are being modified: the new point is inserted in each and any contiguities strictly between the entry and exit points of the perpendicular bisector are deleted, the anticlockwise-cyclic arrangement of the lists making both this and the determination (sic) of the exit very easy.

This insertion algorithm requires the computation of the nearest existing data point to the data point that is to be inserted. The authors discuss an algorithm which takes advantage of the tessellation computed so far. In the authors words: "Simply start at an arbitrary point and "walk" from neighbour to neighbour, always approaching the new point, until the point nearest to it is found."

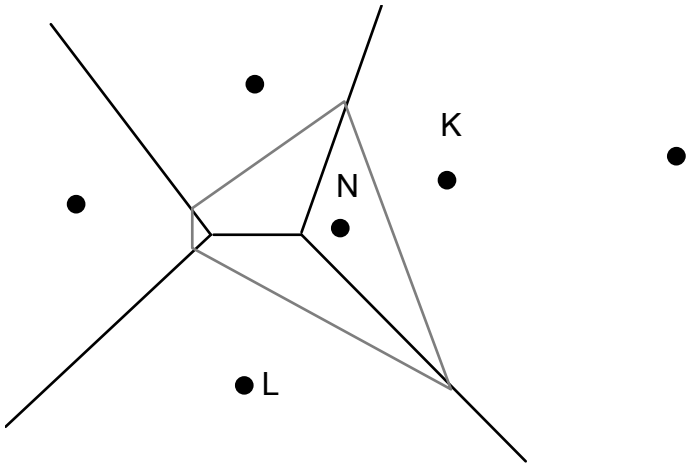


Figure 2.2.8. An aid to the Green and Sibson algorithm.

The algorithm of Bowyer [21]: Bowyer described an algorithm for inserting a new point (lying in the convex hull) into an existing Delaunay triangulation. An example

given by Bower and which we include in Figure 2.2.9 serves to define this data structure. (A careful examination of this data shows that it is the same as the triangular grid structure of Figure 2.1.8). In the terminology of Bowyer, the forming points for a vertex are simply the vertices of the triangle which has this particular vertex as the center of its circumcircle. Since each triangle gives rise to a vertex, giving a list of indices of the forming points for each vertex (as Bowyer does) is equivalent to giving a list of indices of the data points which comprise each triangle of triangulation. Except for change in ordering, the neighboring vertices is exactly the same as the indices of the triangle neighbors as given in the triangular grid data structure.

Vertex	Forming points			Neighboring vertices		
	1	2	3	1	2	3
V_1	P_6	P_4	P_5	V_4	ϕ	V_6
V_2	P_1	P_4	P_3	V_3	ϕ	V_7
V_3	P_2	P_3	P_4	V_2	V_4	V_5
V_4	P_2	P_5	P_4	V_1	V_3	ϕ
V_5	P_7	P_3	P_2	V_3	ϕ	ϕ
V_6	P_6	P_8	P_4	V_7	V_1	ϕ
V_7	P_1	P_8	P_4	V_6	V_2	ϕ

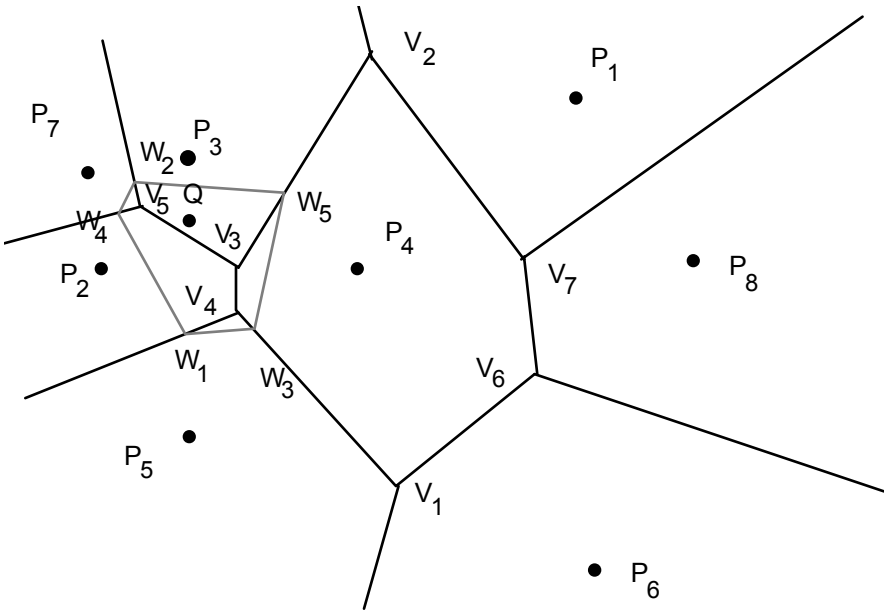


Figure 2.2.9. Illustrating the algorithm of Bower[21].

In order to insert a new point (Q in Figure 2.2.9) within the current convex hull of the data points, Bowyer [21] gives the following algorithm:

> 1. Identify a vertex currently in the structure that will be deleted by the new point (say V_4). Such a vertex is any that is *nearer to the new point than to its forming points*

2. Perform a tree search through the vertex structure starting at the deleted vertex looking for others that will be deleted. In this case the list will be: $\{V_4, V_3, V_5\}$

3. The points contiguous to Q are all the points forming the deleted vertices: $\{P_2, P_5, P_4, P_3, P_7\}$

4. An old contiguity between a pair of those point will be removed (P_2 - P_4 say) if all its vertices $\{V_4, V_3\}$ are in the list of deleted vertices.

5. In this case the new point has five new vertices associated with it: $\{W_1, W_2, W_3, W_4, W_5\}$. Compute their forming points and neighbouring vertices. The forming points for each will be the point Q and two of the points contiguous to Q. Each line in the tessellation has two points around it (the line V_3 - V_2, for example, is formed by P_3 and P_4). The forming points of the new vertices and their neighbouring vertices may be found by considering vertices pointed to by members of the deleted vertex list that are not themselves deleted, and finding the righs of points around them. Thus W_5 points outwards to V_2 from Q and is formed by $\{P_3, P_4, Q\}$.

6. The final step is to copy some of the new vertices, over writing the entries of those deleted to save space.

The Algorithm of Watson [254]: This algorithm relies on the property of a Delaunay triangulation that a triple of data point indices (i, j, k) will be in I_t provided the circumcircle of p_i, p_j, and p_k contains no other data points. As with the other algorithms, this algorithm is based upon inserting a new point into an already existing Delaunay triangulation. The general philosophy of Watson's approach is described by the following two steps:

1. Find all triangles whose circumcircle contains the point to be inserted.

2. For each of these triangles, form three new triangles from the point to be inserted and the three edges of this triangle and test to see if any of these three new triangles contain any other data points. If not, then add this new triangle to the triangulation.

More details for this general approach are given in the flow diagram of Figure 2.2.10 which is based upon the flow diagram of [254].

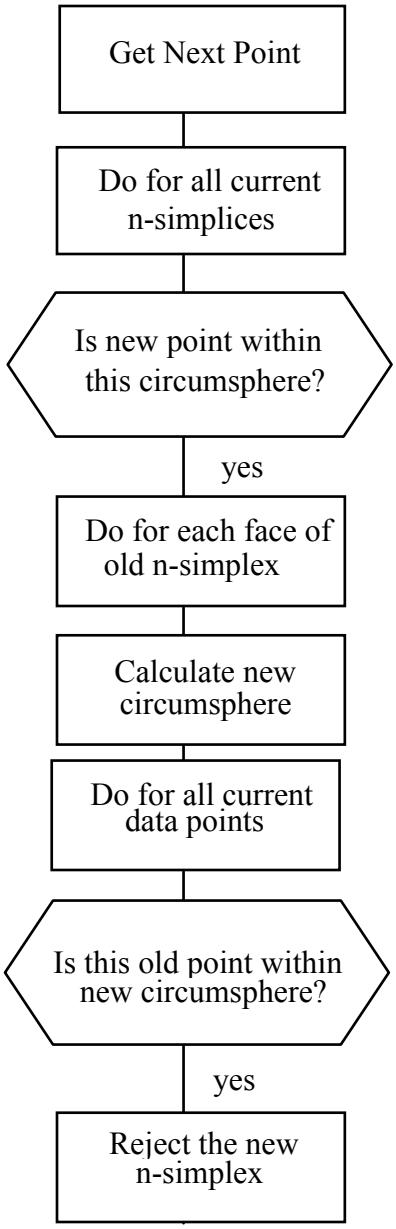


Figure 2.2.10. Flow Diagram for Watson's Algorithm.

Watson [254] describes a number of features and details to make the basic algorithm efficient and eventually discusses a particular implementation which he says has an expected running time which is observed to not increase more that $N^{3/2}$.

The embedding/lifting approach: Algorithms of this type are based upon a very interesting relationship that exists between the three dimensional convex hull of the lifted points (x_i, y_i, $x_i^2 + y_i^2$) and the Delaunay triangulation. Faces on the convex hull are designated as being either in the upper or lower part. The lower part consists of faces which are supported by a plane that separates the point set from (0, 0, -oo). The Delaunay triangulation is obtained directly form the projection onto the x-y plane of the lower part of the convex hull. See [27] and [68]. An algorithm for computing the convex

hull which is based on an initial sort followed by a recursive divide-and-conquer approach has been described by Preparata and Hong [202]. This algorithm is also covered in [68] and [203] Theoretically the algorithm is optimal time O(n log(n)), but Day [49] reports that empirical data implies a worst-case complexity of $O(N^2)$ The paper of Day [49] covers many of the details and special case issues of practical interest for implementation which are often brushed over in more theoretical papers.

Divide and Conquer Algorithms: The general structure of this type of algorithm is to divide the data set into subsets A and B, solve the problem for A and solve the problem for B and merge the results into a solution for $A \cup B$. See Figure 2.2.11. Divide and conquer algorithms can lead to theoretically optimal algorithms, but often fail to be competitive in practical usage. The merging portion is often the most troublesome in trying to maintain bounds on the running times and complexity of the algorithm.

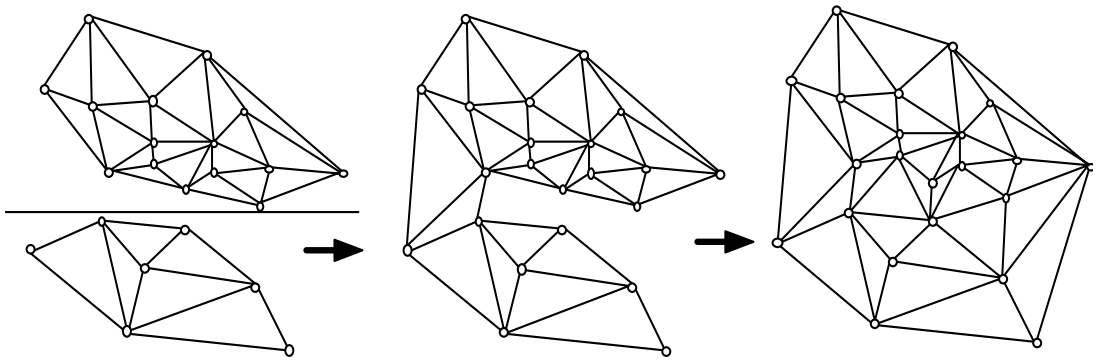


Figure 2.2.11. Divide and Conquer Algorithms

2.3 Visibility Sorting og Triangulations

This is an example of an area that is interesting in 3D but not in 2D. It is possible to make a definition of a visibility sort for a triangulation which is completely analogous to that of a tetrahedrization, but there does not appear to be any application or use for such a property. We defer further discussion on visibility sorting to Section 3.3.

2.4 Data Dependent Triangulations

The topic of data dependent triangulations arises within the context of determining a modeling function F(x, y) for the data $(F_i; x_i, y_i)$, i = 1, . . . , N. A relatively simple approach to defining a modeling function is to first form a triangulation of the convex hull of the independent data (x_i, y_i), i = 1, . . . , N and then define F to be piecewise linear over this triangulation. This will yield a C^0 (continuous) function which interpolates the data; that is, $F(x_i, y_i) = F_i$, i = 1, . . . , N. We denote this function by $F_T(x,y)$. Any triangulation of the independent data (x_i, y_i), i = 1, . . . , N will suffice for this approach. While we are well aware of the many desirable properties of the Delaunay triangulation, it might very well be the case that some other triangulation whose choice would depend

upon the values F_i, i = 1, ., . . , N would lead to some desirable properties for the modeling function F. This is the basic idea of data dependent triangulation. Of course, there are potentially many ways to accomplish this, but we choose for this discussion here to briefly describe the criteria called "nearly C^1" as proposed in [67]. An ordering is imposed on the collection of all possible triangulations of the convex hull in the following manner. First a local cost function for each edge e_i = 1, . . . , $N_{ie} = N_e - N_b$ is defined and denoted by $S(F_T, e_i)$. (We will shortly describe the four examples of local cost functions covered in [67]). If T and T' are two triangulations, then

$$T \leq T'$$

provided the vector

$$(s(F_T, e_1), s(F_T, e_2), . , . , . , \ s(F_T, e_{Nie}))$$

is lexicographically less than or equal to

$$(s(F_{T'}, e_1), s(F_{T'}, e_2), . , . , . , \ s(F_{T'}, e_{Nie})).$$

It is assumed that the components of these vectors are arranged in nonincreasing order. The goal is then to find the optimal data dependent triangulation which is defined by having the smallest associated vector under this lexicographical ordering. Since there are only a finite (albeit possibly very large) number of possible triangulations, we know that a global minimum exists even though it may not be unique and it may not be so easy to compute. The algorithm used in [67] is similar to the swapping algorithm of Lawson (which we have described above in Section 2.2) in that an initial triangulation is obtained and then an internal edge of a convex quadrilateral is considered. If T' < T, where T' is the same triangulation as T except the diagonal of the convex quadrilateral has been switched, then this switch is made and other edges are considered for potential swapping. Since each swap moves strictly lower in the lexicographic ordering, we are guaranteed that this algorithm will eventually converge after a finite number of steps. This means that swapping any edge would not move to a smaller triangulation. This limit triangulation may not be the global minimum, it is only guaranteed to be a local minimum and steps to find the global minimum must do more than swap diagonals which improve (with respect to the ordering) the triangulation.

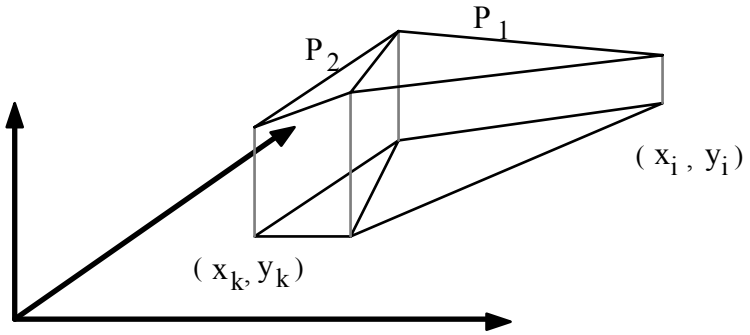


Figure 2.4.1. Notation for local cost function definitions.

We now describe the four local edge cost functions used in [67]. Let $P_1 = a_1x + b_1y + c_1$ and $P_2 = a_2x + b_2y + c_2$ be the two planes defined over the two triangles of a convex quadrilateral.

i) The angle between normals: The local cost function is taken as the acute angle between N_1 and N_2 which are the respective normals for P_1 and P_2.

$$s(F_T, e) = \cos^{-1}(A)$$

where

$$A = \frac{a_1a_2+b_1b_2+1}{\sqrt{(a_1^2+b_1^2+1)(a_2^2+b_2^2+1)}} \ .$$

ii) The jump in normal derivative: This cost function is the difference between the derivative of P_1 and P_2. This derivative is taken in the direction perpendicular to the edge dividing the two triangles.

$$s(F_T, e) = [n_x(a_1-a_2) - n_y(b_1-b_2)]$$

where (n_x, n_y) is a unit vector perpendicular to the edge e.

iii) The deviations from linear polynomials: The cost functions measures the error between P_1 and P_2, evaluated at the other point of the quadrilateral.

$$s(F_T, e) = \sqrt{(P_1(x_i, y_i) - F_i)^2 + (P_2(x_k, y_k) - F_k)^2}$$

iv) The distance from planes: This cost functions measures the distance between the planes P1 and P2 and the corresponding vertex of the quadrilateral.

$$s(F_T, e) = \sqrt{\frac{(P_1(x_i,y_i)-F_i)^2}{a_1^2+b_1^2+1} + \frac{(P_2(x_k,y_k)-F_k)^2}{a_2^2+b_2^2+1}}$$

Some typical results are given in [67] which confirm the expectation that using the optimal data dependent triangulation improves the overall fitting properties of F_T over that of the Delaunay triangulation, which, by the way, is used as the initial triangulation for the swapping algorithm. It is observed that long thin triangles tend to appear where the data seems to indicate a function that is increasing (or decreasing) relatively rapidly in a certain direction. The use of the data dependent triangulation generally gives an overall reduction in errors when certain test functions are used to generate the data.

As we have mentioned, the local swapping algorithm used in [67] can only find a local minimum. In order to move more closely to the globally optimal data dependent triangulation, Schumaker [225] and Quak and Schumaker [204], [205], [206] have involved the tools of simulated annealing. More details on this are contained in Section 3.6 on data dependent tetrahedrizations. We include here the results of one example described by Schumaker. The data consists of

$$(F_{ij}; x_i, y_j); \quad x_i , y_j = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0;$$

where

$$F(x,y) = (y - x^2)_+ .$$

Three triangulations are shown in Figure 2.4.2. The first is the Delaunay triangulation of the independent data. The next is the triangulation which results from the local swapping algorithm of [67] using the local cost function of "angle between normals". The last is the triangulation after simulated annealing has been applied. The associated vectors for each of these triangulations is given in Figure 2.4.4.

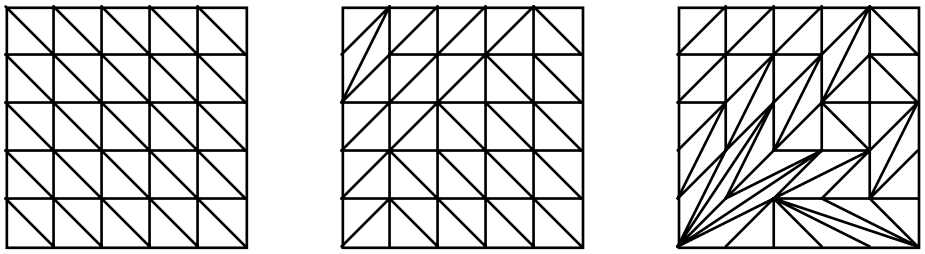


Figure 2.4.2. Examples of data dependent triangulations.

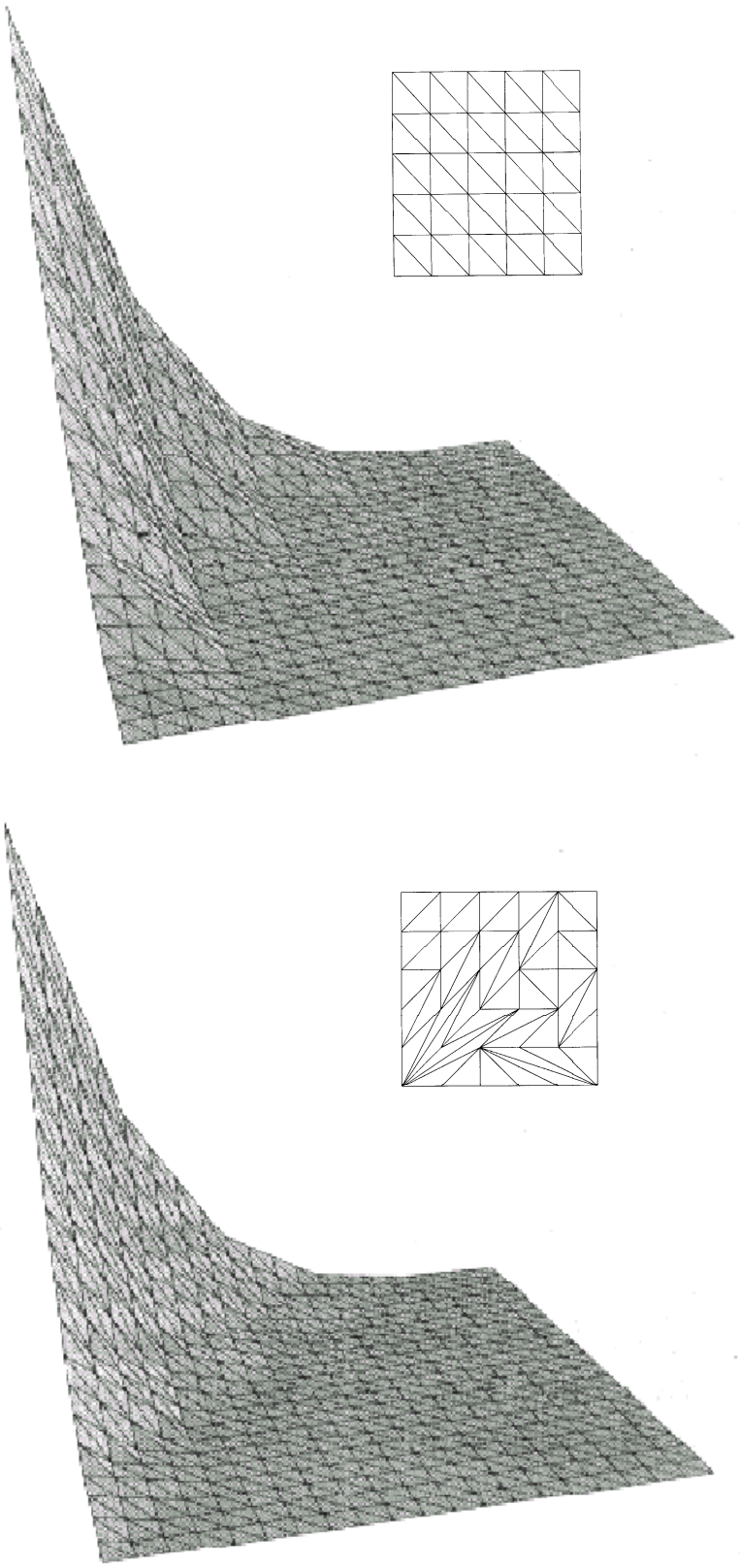


Figure 2.4.3. The graphs of Schumaker's example. See [225].

Angles between normals for Delaunay triangulation:

55.077	48.155	44.684	39.801	39.588	38.378	37.734	35.445	33.992
33.786	33.561	33.162	30.470	28.898	28.287	27.284	27.284	26.003
23.633	21.958	20.814	17.886	16.066	15.942	15.642	11.310	10.302
9.661	7.294	7.294	7.294	6.843	0.649	0.649	0.459	0.458
0.458	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000							

Angles between normals for locally optimal triangulation

35.993	30.590	26.070	23.610	21.813	21.558	16.563	16.521	15.793
12.810	11.929	11.310	10.646	10.261	9.622	8.844	8.707	8.321
8.076	8.047	5.794	5.563	3.777	0.649	0.649	0.459	0.459
0.458	0.458	0.458	0.448	0.020	0.020	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000							

Angles between normals for annealed triangulation

26.070	22.929	22.113	20.049	17.257	16.563	16.521	13.031	12.505
11.929	10.389	10.270	10.261	8.954	8.321	7.844	5.962	5.794
5.256	1.652	1.480	1.025	0.649	0.648	0.459	0.458	0.458
0.448	0.447	0.020	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000							

Figure 2.4.4. Angles for the data dependent triangulation.

2.5 Affine Invariant Triangulations

The desirable properties of the Delaunay triangulation have been previously discussed. Unfortunately, this optimal triangulation is not invariant under affine transformations and this means that methods for analyzing and visualizing data that use this particular triangulation can be affected by the choice of units used to measure the data. This could be considered an undesirable property. In this section we describe a relatively new method for characterizing and computing an optimal triangulation which is invariant under affine transformations. Before we proceed with the discussion of these techniques, we wish to motivate further the desirability of affine invariance.

As we have mentioned earlier, one of the main purposes for triangulations and tetrahedrizations is their use in defining functions in a piecewise manner over the domain of a data set. It would be undesirable if the happenstance of the choice of units used to measure the data were to affect the definition of a data modeling function. But this does happen with the Delaunay triangulation. The example of Figure 2.5.1 points this out. This data represents the independent data and the dependent data is not given as it not important in this context. The data is the same in both the left and right graphs of Figure

2.5.1; the only difference is that in the left graph we have used years and £ (pounds, British monetary unit, equal (approximately and assumed here to be exactly equal) to two US dollars) and in the right graph we have used months and dollars. If we use the units of years and £ then we can see that the three vertices (1yr, 1£), (0.5yr, 3£), (2yr, 2£) will mark out a triangle to be included in the list of triangles for the Delaunay triangulation. But on the other hand if we use months and $ we can see that the circumcircle defined by these same three vertices (12mon, 2$), (6mon, 6$), (24mon, 4$) contains the data point (6mon, 4$). Therefore, these three vertices will not comprise a triangle of the Delaunay triangulation if these units are used. This simple example points out the possible affects of the choice of the units of measurement. The choice of the units of measurement is the same as a change in scale, $x \leftarrow ax$ and $y \leftarrow by$. Uniform scale changes of the type $x \leftarrow \alpha x$, $y \leftarrow \alpha y$ will not affect the Delaunay triangulation.

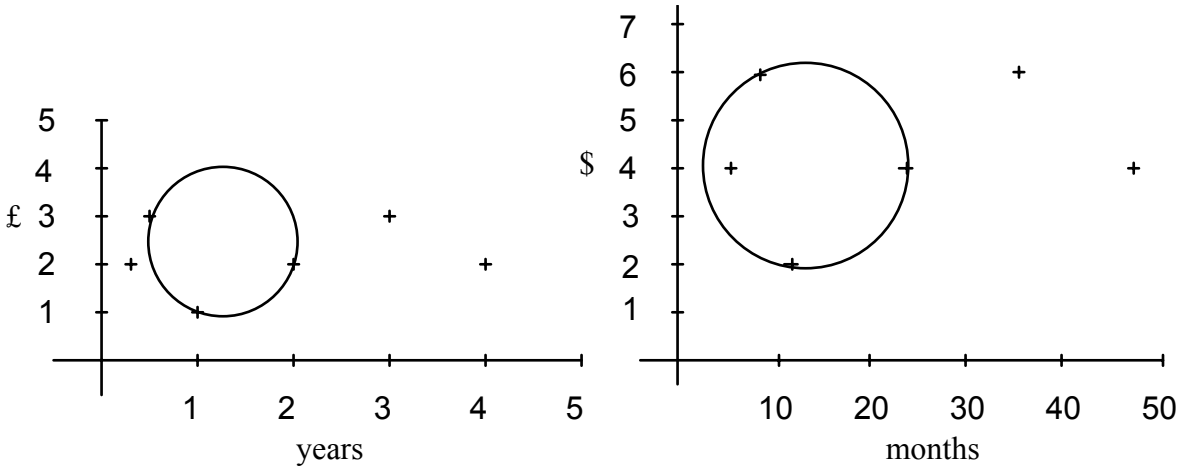


Figure 2.5.1. Two different units used to measure the same data lead to two different Delaunay triangulations.

We now discuss how to avoid this problem. It would be possible to simply normalize all data ranges to one unit by scaling by the range. But this approach would mean that rotations of the data could have an effect on the Delaunay triangulation meaning the final data model would be affected by rotations of the data. In other words, the placement and alignment of the axes for the measurement of the data would have an affect on the data modeling function and subsequently on our analysis of the data and this we would like to have the opportunity to avoid. It would, in general, be useful to have a characterization (and subsequent algorithms) for an optimal triangulation which is not affected by affine transformation. An affine transformation is a map of the form

$$(x,y) = A(x,y) + c$$

where A is a 2 x 2 matrix and c is two-dimensional point. Affine transformations include not only scale changes and rotations, but also, translations, reflections and shearing transformations. The approach to such an optimal triangulation covered here is through the duality that exists between the conventional Delaunay triangulation and the Dirichlet tessellation. As we previously described the characterization of the Delaunay

triangulation (as a MaxMin triangulation), it was heavily dependent upon angles and angles are affected by scaling transformations and so it should be no surprise that the Delaunay triangulation is also affected by scaling transformations. But the definition of the Dirichlet tessellation uses only distance and we know that the Delaunday triangulation is dual to (a direct result of) the Dirichlet tessellation. The approach here is to use a method of measuring distance which is invariant under affine transformations. The Dirichlet tessellation based upon this new method of measuring distance will have a dual which will serve as our optimal triangulation. Rather than use the standard Euclidean norm $\|(x,y)\|^2 = \sqrt{x^2 + y^2}$ we propose the use of the following norm

$$\|(x,y)\|_V^2 = (x,y)\begin{pmatrix} \dfrac{\sum_y^2}{\sum_x^2\sum_y^2 - \left(\sum_{xy}\right)^2} & \dfrac{-\sum_{xy}}{\sum_x^2\sum_y^2 - \left(\sum_{xy}\right)^2} \\ \dfrac{-\sum_{xy}}{\sum_x^2\sum_y^2 - \left(\sum_{xy}\right)^2} & \dfrac{\sum_x^2}{\sum_x^2\sum_y^2 - \left(\sum_{xy}\right)^2} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \qquad (2,5.1)$$

where

$$\Sigma_x^2 = \frac{\sum_{i=1}^{N}(x_i - \mu_x)^2}{N}, \qquad \mu_x = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\Sigma_y^2 = \frac{\sum_{i=1}^{N}(y_i - \mu_y)^2}{N}, \qquad \mu_y = \frac{\sum_{i=1}^{N} y_i}{N}$$

$$\Sigma_{xy} = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{N}$$

and

$$V = \begin{pmatrix} x_1 - \mu_x & x_2 - \mu_x & \cdots & x_N - \mu_x \\ y_1 - \mu_y & y_2 - \mu_y & \cdots & y_N - \mu_y \end{pmatrix}$$

We have used the subscript of V on the norm to explicitly indicate that this method of measuring distance is dependent upon the data set. Change the data set and you change how you measure distance but the distance between any two data points will remain constant.. This norm and its use within the context of scattered data modeling was first described in [181]. This norm has the property that it is invariant under affine transformations. More precisely,

$$\|P-Q\|_{\mathbf{V}} = \|T(P) - T(Q)\|_{T(\mathbf{V})} \tag{2.5.2}$$

for any two points P = (x, y) and Q = (u, v) and any affine transformation

$$T(P) = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

Here, T(**V**) (used as a subscript in (2.5.2)) is the transformed data

$$T(V) = \left(T\begin{pmatrix} x_1 - \mu_x \\ y_1 - \mu_y \end{pmatrix} \quad T\begin{pmatrix} x_2 - \mu_x \\ y_2 - \mu_y \end{pmatrix} \quad \cdots \quad T\begin{pmatrix} x_N - \mu_x \\ y_N - \mu_y \end{pmatrix} \right)$$

Figure 2.5.2 illustrates the properties of this new method of measuring distance. Each of the data sets shown in this figure are affine images of each other. Starting in the upper left and moving in a clockwise direction, the transformations are: counter clockwise rotation of 44 degrees; a scaling in x by a factor of 2; a scaling in y by a factor of 0.4. The four ellipses in each figure represent points which are 1/4, 1/2, 3/4 and 1 unit(s) from their center point as measured with the affine invariant norm. In Figure 2.5.3 we show the Dirichlet tessellation of these four affinely related data sets and in Figure 2.5.4 we show the corresponding dual triangulation and as one can see the triangulation is unchanged by these transformations.

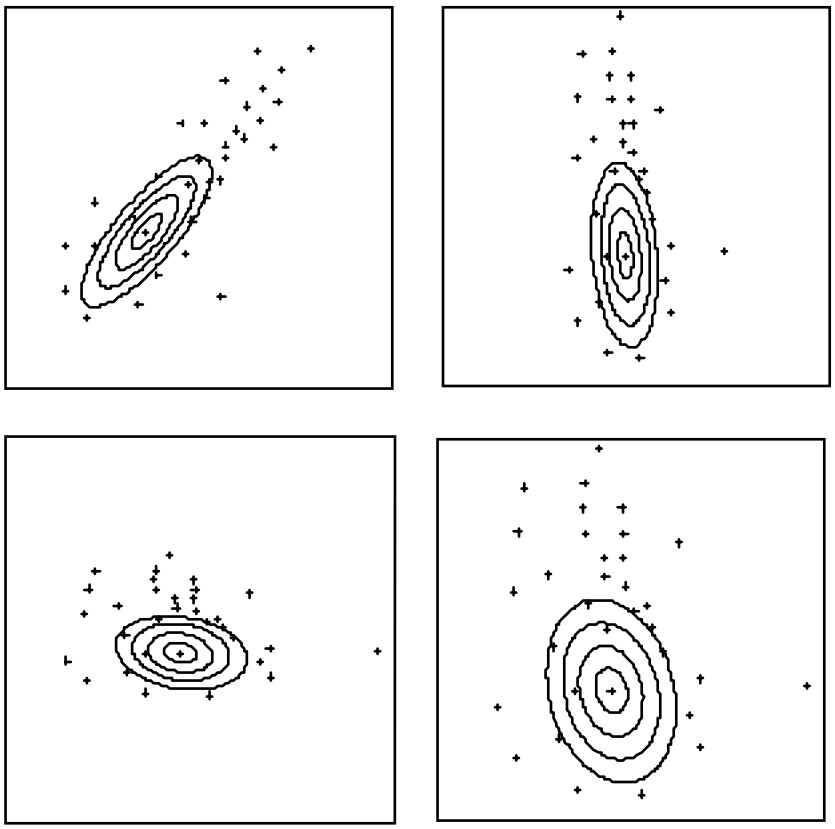


Figure 2.5.2. Affine transformations of a data set and points equally distant (*affine invariant norm*) from a point.

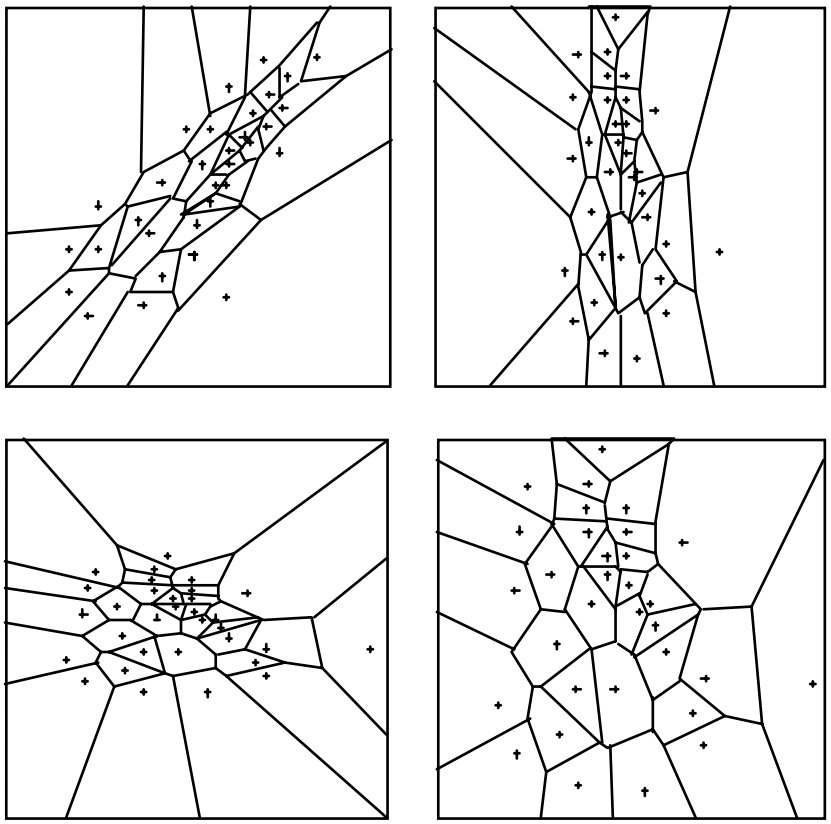


Figure 2.5.3. The Dirichlet tessellation (*affine invariant norm*) of affine transformations of a given data set.

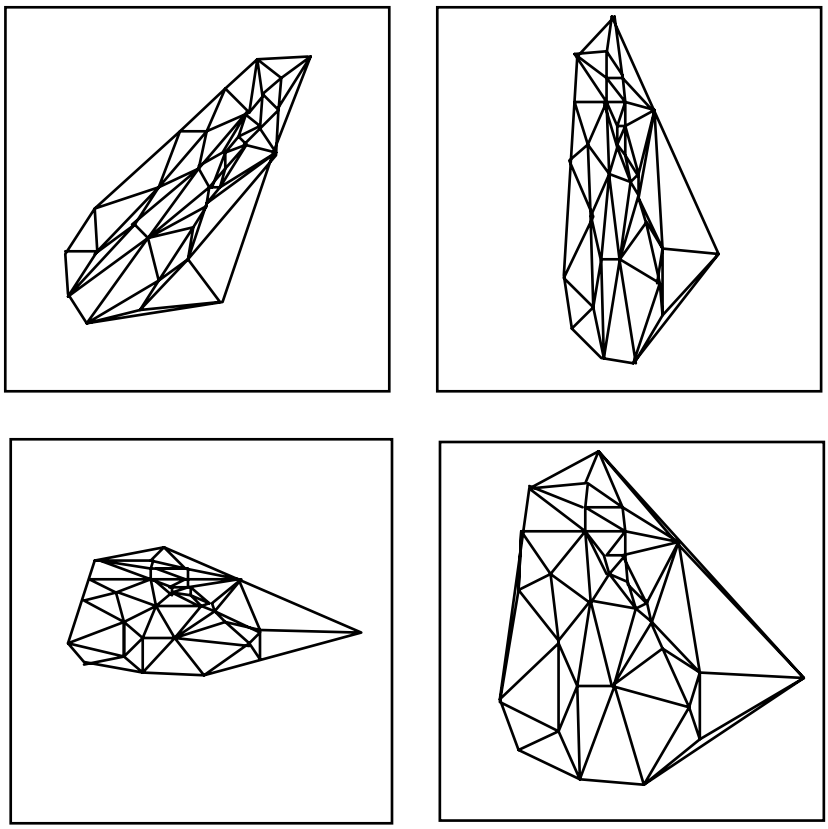


Figure 2.5.4. The triangulation dual to the Dirichlet tessellation (*affine invariant norm*) of a given data set and some affine transformations.

As a comparison, we have also included the Delaunay triangulation based upon the standard Euclidean norm in Figure 2.5.5. And as we indicated earlier, we can see that triangulation results are affected by the transformations. Not all triangles are changed, but some are.

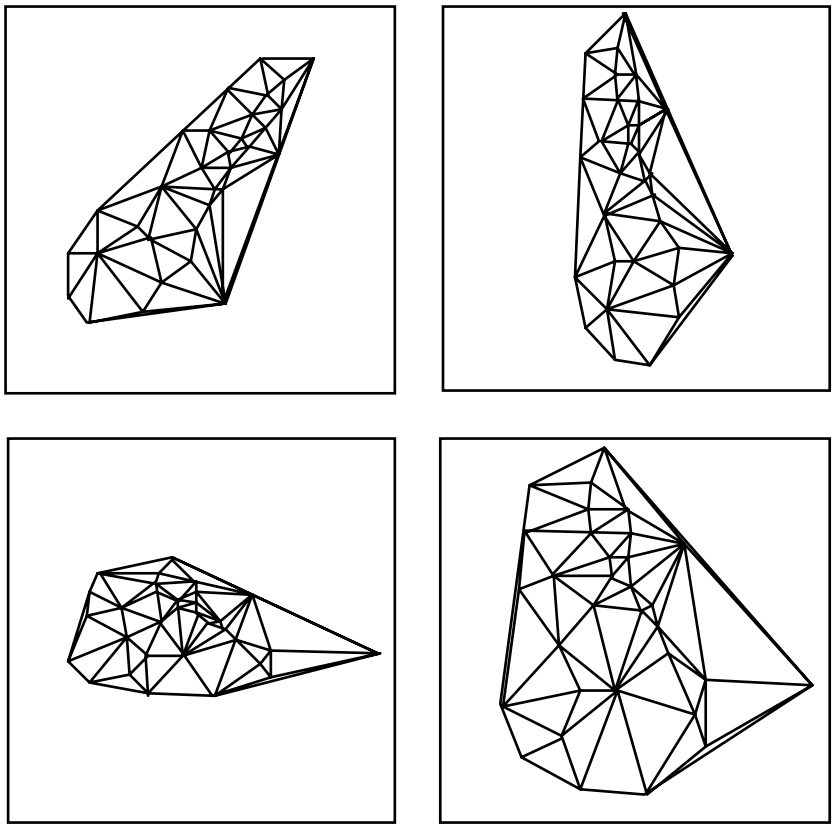


Figure 2.5.5. The Delaunay triangulation of a data set and some affine transformations.

And now some practical information on how to incorporate this feature in to an algorithm for computing triangulations. If you already have an procedure for computing an optimal triangulation, then it is possible to modify it slightly to achieve the results we have described in this section. Say for example that the procedure is based upon Lawson's algorithm and there is a subprocedure which decides whether or not to switch the diagonal of a quadrilateral formed from two triangles. It might be that this procedure is based solely on Euclidean distance. That is, the center and radius of the circumcircle of three points is determined and the distance to the center from the fourth point is computed so as to make this decision. In order to modify this subprocedure, we only need to replace the use of the Euclidean norm with the affine invariant norm described here. The equations for computing circumscribing circles (ellipses) for a quadratic norm in general are given in [182]. If, on the other hand, the procedure you are already using is known to be rotation invariant, then there is even an easier way to affect the results of the affine invariant triangulation. This is based upon the factorization of the matrix which defines the affine invariant norm. We denote this matrix by A(**V**) so that we have

$$\|(x,y)\|_V^2 = (x,y)A(V)\begin{pmatrix} x \\ y \end{pmatrix}$$

The matrix A(**V**) can be factored (Cholesky) into

$$A(V) = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{pmatrix} = L(V)L(V)^*$$

Here the notation L(V)* denotes the transpose of L(V).

Using this factorization, we have that

$$\|(x, y)\|_V^2 = (x, y)L(V)L(V)^* \begin{pmatrix} x \\ y \end{pmatrix} = \|(x, y)L(V)\|^2$$

which means measuring distances with the affine invariant norm is the same as measuring distance in the standard Euclidean but with the points transformed by multiplying by L(V). This means that we can achieve the result of the optimal affine invariant triangulation by computing the standard Delaunay triangulation on the transformed data

$$(X_i, Y_i) = (x_i, y_i)L(V)$$

In summary, we need only compute

$$l_{11} = \sqrt{a_{11}}\,, \qquad l_{21} = \frac{a_{21}}{\sqrt{a_{11}}}\,, \qquad l_{22} = \sqrt{\frac{a_{11}a_{22} - a_{21}^2}{a_{11}}}$$

where $a_{11} = \dfrac{\Sigma_y^2}{\Sigma_x^2\Sigma_y^2 - (\Sigma_{xy})^2}$, $a_{21} = \dfrac{-\Sigma_{xy}}{\Sigma_x^2\Sigma_y^2 - (\Sigma_{xy})^2}$ and $a_{22} = \dfrac{\Sigma_x^2}{\Sigma_x^2\Sigma_y^2 - (\Sigma_{xy})^2}$.

and apply any rotation invariant triangulation algorithm to the transformed data

$$X_i = l_{11}x_i + l_{21}y_i$$

$$Y_i = l_{22}y_i, \qquad i = 1, \ldots, N.$$

2.6 Interpolation in triangles

We now take up the topic of interpolating into (or over) a single triangular domain. The interpolants we describe here form the basic building blocks for constructing the global interpolants which have piecewise definitions over the individual triangles of a triangulation. The domain here is a single triangle, $T = T_{ijk}$ with vertices V_i, V_j and V_k, and the data consists of values given on the boundary of the triangular domain. We need to differentiate between two types of boundary data. If the data consists of function and certain derivative values specified only at the vertices (or possibly other points such as midpoints), then we call this *discrete data*. If, on the other hand, the data is provided on

the entire boundary of the triangle, we refer to this type of data as *transfinite data*. The importance of an interpolant which will match transfinite data is that it serves as a prototype for developing a large variety of discrete interpolants. This is accomplished through the process of discretization where the data required for a transfinite interpolant is provided by means of using some interpolation scheme only on the boundary, discrete data. For example, given only data values at the vertices, we can use linear interpolation along an edge to produce the transfinite data required by the transfinite interpolant.

There is a second concept which is rather important for interpolants defined over triangles and this has to do with the degree of continuity of the global interpolant. Often, we require that the global interpolant at least be continuous. We call such an interpolant a C^0 interpolant. If the global interpolant has continuous first order derivatives, we say it is a C^1 interpolant. A C^0 interpolant for a single triangle is one which interpolates to boundary data consisting of only position values, either at the vertices (and midpoints) only or on the entire boundary. A C^1 interpolant for a single triangle is one which will interpolate to first order derivative data specified on the boundary. But this must be done in a manner so as to guarantee C^1 continuity across the boundary edges. So, if the cross boundary derivative varies quadratically along an edge, then the data on this edge must be sufficient to uniquely determine this derivative so that on an adjoining triangle we will have exactly the same cross boundary derivative. For this reason, it is common for C^1 interpolants to have linearly varying cross boundary derivatives which are determined by their values at the two endpoint vertices.

Combining the two concepts of discrete and transfinite data and C^0 and C^1 data leads to four types of triangular interpolants as indicated in Figure 2.6.1 This general area of interpolation in triangles is fairly rich and well developed and we urge the really interested reader to follow the citations into the literature after taking a look of the sampling we have chosen to include here. Figure 2.6.1 serves as an outline for the remainder of this section. We first cover C^0, discrete interpolants, then a sampling of three C^0, transfinite interpolants. This is followed by the description of a C^1, discrete interpolants. We have chosen to include a discretized version of the minimum norm triangular interpolant (see [178]). Another rather popular C^1, discrete interpolant is the Clough/Tocher interpolant often mentioned in conjunction with the finite element method. Much has been written about this interpolant in the past and so we do not include it here. This section is concluded with a description of a C^1, transfinite interpolant called the side-vertex interpolant [177]. It is one of the easiest to describe and the most versatile to use. It also generalizes rather nicely to a tetrahedral domain.

	Discrete	Transfinite
C^0	**Section 2.6.1**	**Section 2.6.2**
C^1	**Section 2.6.3**	**Section 2.6.4**

Figure 2.6.1. Outline of Section 2.6

2.6.1 C^0, Discrete Interpolation in Triangles

The lowest degree polynomial, C^0 discrete interpolant is linear and it is unique. Given the data $F(V_i)$, $F(V_j)$ and $F(V_k)$, the coefficients of the linear function

$$F(x,y) = a + bx + cy$$

which interpolates this data can be found by solving the linear system of equations

$$a + bx_i + cy_i = F(V_i)$$

$$a + bx_j + cy_j = F(V_j)$$

$$a + bx_k + cy_k = F(V_k).$$

Another path to this basic linear interpolant is via barycentric coordinates. Given a point $V = (x, y)$, barycentric coordinates, b_i, b_j and b_k of this point relative to the triangle T_{ijk} are defined by the relationships

$$\begin{pmatrix} x \\ y \end{pmatrix} = b_iV_i + b_jV_j + b_kV_k$$

$$1 = b_i + b_j + b_k \ .$$

The linear interpolant now takes the form

$$F(x,y) = F(V) = b_iF(V_i) + b_jF(V_j) + b_kF(V_k) \ .$$

There are several alternative ways of defining or determining the barycentric coordinates of a point. For example,

$$b_i = \frac{A_i}{A} \qquad b_j = \frac{A_j}{A} \qquad b_k = \frac{A_k}{A}$$

where A_i, A_j and A_k represent the areas of the subtriangle shown in Figure 2.6.2 and A is the area of T_{ijk}. Also,

$$b_i = \frac{\begin{vmatrix} x-x_k & x_j-x_k \\ y-y_k & y_j-y_k \end{vmatrix}}{\begin{vmatrix} x_i-x_k & x_j-x_k \\ y_i-y_k & y_j-y_k \end{vmatrix}} \quad b_j = \frac{\begin{vmatrix} x-x_i & x_i-x_k \\ y-y_i & y_i-y_k \end{vmatrix}}{\begin{vmatrix} x_j-x_k & x_i-x_k \\ y_j-y_k & y_i-y_k \end{vmatrix}} \quad b_k = \frac{\begin{vmatrix} x-x_j & x_i-x_j \\ y-y_j & y_i-y_j \end{vmatrix}}{\begin{vmatrix} x_k-x_j & x_i-x_j \\ y_k-y_j & y_i-y_j \end{vmatrix}}$$

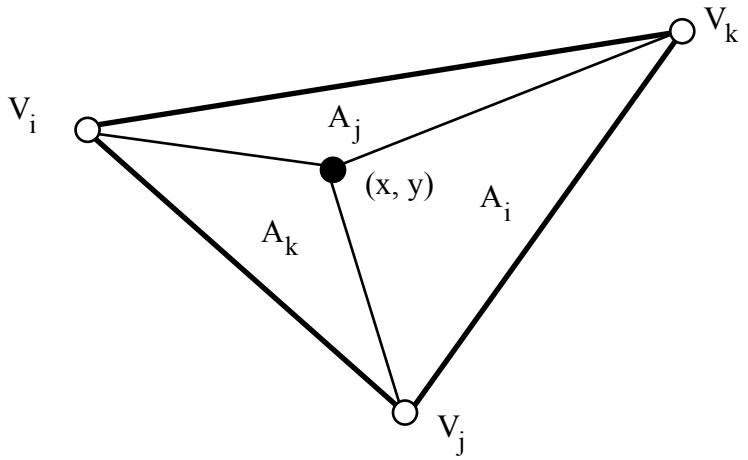


Figure 2.6.2. Areas leading to barycentric coordinates.

Given the values at the three vertices and the three midpoints of a triangle, there is a unique quadratic which interpolates this data,

$$\begin{aligned} Q(x,y) = \quad & F(V_i)b_i(b_i - b_j - b_k) + F(M_{jk})4b_jb_k \\ & + F(V_j)b_j(b_j - b_i - b_k) + F(M_{ik})4b_ib_k \\ & + F(V_k)b_k(b_k - b_i - b_j) + F(M_{ij})4b_ib_j \end{aligned}$$

where $M_{jk} = (V_j + V_k)/2$, $M_{ik} = (V_i + V_k)/2$ and $M_{ij} = (V_i + V_j)/2$.

A common way to specify a cubic along an edge is to use the Hermite form which involves the first order directional derivatives along the edges

$$F'_{ki}(V_i) = (x_k - x_i)F_x(V_i) + (y_k - y_i)F_y(V_i)$$

which are further illustrated in Figure 2.6.3.

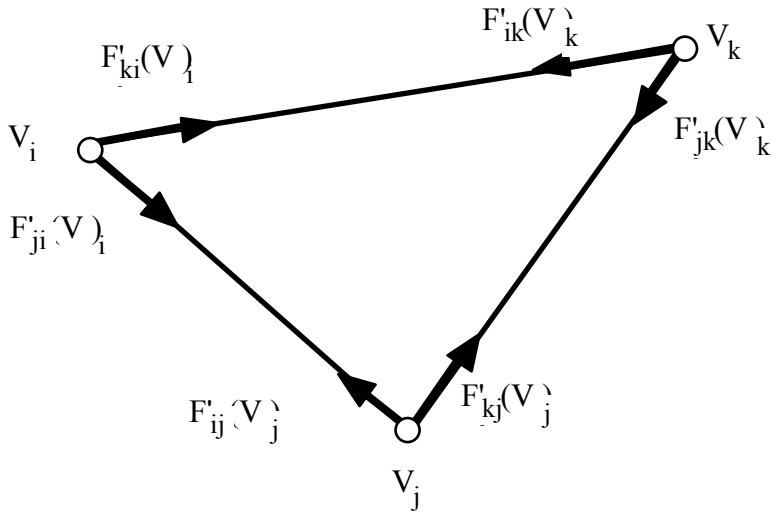


Figure 2.6.3. The notation for the six directional derivatives.

The six directional derivative at the three vertices along with $F(V_i)$, $F(V_j)$ and $F(V_k)$ do not uniquely determine a cubic since the bivariate cubics are of dimension 10. The interpolant

$$\begin{aligned} C(x, y) = \quad & F(V_i)b_i^2(3\text{-}2b_i) + F'_{ki}(V_i)b_i^2\, b_k + F'_{ji}(V_i)b_i^2\, b_j \\ & + F(V_j)b_j^2(3\text{-}2b_j) + F'_{ij}(V_j)b_j^2\, b_i + F'_{kj}(V_j)b_j^2\, b_k \\ & + F(V_k)b_k^2(3\text{-}2b_k) + F'_{ik}(V_k)b_k^2\, b_i + F'_{jk}(V_k)b_k^2\, b_j \\ & + wb_ib_jb_k \end{aligned}$$

will match this function and derivative data for any value of w. This remaining degree of freedom represented by w can be absorbed by a variety of conditions. For example, it can additionally be required that the interpolant match some predescribed value at the centroid. Another common choice is

$$\begin{aligned} w = \quad & 2[F(V_i) + F(V_j) + F(V_k)] \\ & + \tfrac{1}{2}[F'_{ki}(V_i) + F'_{ji}(V_i) + F'_{ij}(V_j) + F'_{kj}(V_j) + F'_{ik}(V_k) + F'_{jk}(V_k)] \end{aligned}$$

which guarantees quadratic precision and is a result of discretization of a number of transfinite interpolants (see [189]). Quadratic precision means that whenever the data comes from a bivariate quadratic function the interpolant will become this very same quadratic polynomial.

2.6.2 C^0, Transfinite Interpolation in Triangles

In this section, we only give a sampling of three interpolants which will interpolate to arbitrary function values on the boundary of a triangular domain, T_{ijk}. More information on this general topic can be found in [189].

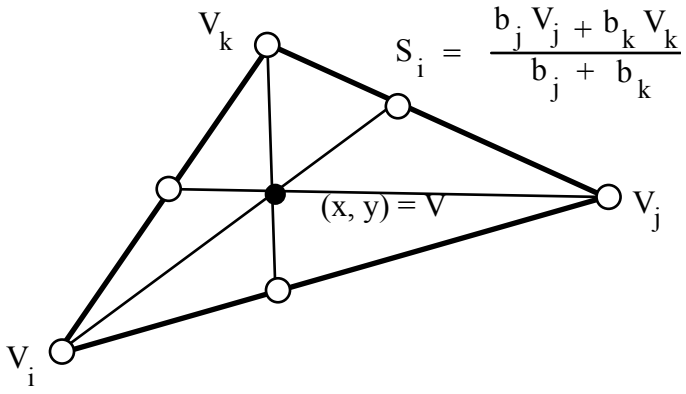


Figure 2.6.4. The side-vertex interpolant notation.

The Side-Vertex Interpolant: The side-vertex interpolant is built from three basic interpolants which are defined by linear interpolation along line segments joining a vertex and the opposing side. See Figure 2.6.4 In terms of barycentric coordinates, we have

$$A_i[F] = b_i F(V_i) + (1-b_i)F(S_i),$$

$$A_j[F] = b_j F(V_j) + (1-b_j)F(S_j),$$

$$A_k[F] = b_k F(V_k) + (1-b_k)F(S_k)$$

where $S_i = \frac{b_j V_j + b_k V_k}{b_j + b_k}$, $S_j = \frac{b_i V_i + b_k V_k}{b_i + b_k}$, $S_k = \frac{b_i V_i + b_j V_j}{b_i + b_j}$. Each of these interpolants will interpolate to arbitrary function values on one edge of the triangular domain. In order to obtain an interpolant which matches arbitrary values on the entire boundary of T_{ijk}, we form the Boolean sum of these three interpolants

$$\begin{aligned} A[F] &= A_i \oplus A_j \oplus A_k[F] = A_i[F] + A_j[F] + A_k[F] \\ &\quad - A_i[A_j[F]] - A_j[A_k[F]] - A_k[A_j[F]] + A_i[A_j[A_k[F]]] \\ &= (1-b_i)F(S_i) + (1-b_j)F(S_j) + (1-b_k)F(S_k) \\ &\quad - b_i F(V_i) - b_j F(V_j) - b_k F(V_k) \end{aligned}$$

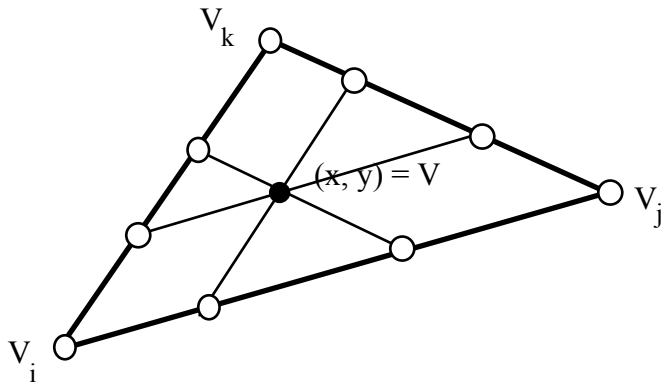


Figure 2.6.5. The evaluation points (ttencil) for side-side interpolant.

The Side-Side Interpolant: The side-side interpolant is based upon the basic operation of linear interpolation along edges which are parallel to the edges of T_{ijk}. There are three of this interpolants,

$$P_i[F] = \frac{b_k F(b_i V_i+(1-b_i)V_k) + b_j F(b_i V_i+(1-b_i)V_j)}{b_k + b_j}$$

$$P_j[F] = \frac{b_i F(b_j V_j+(1-b_j)V_i) + b_k F(b_j V_j+(1-b_j)V_k)}{b_i + b_k}$$

$$P_k[F] = \frac{b_i F(b_k V_k+(1-b_k)V_i) + b_j F(b_k V_k+(1-b_k)V_j)}{b_i + b_j}$$

Unlike the basic interpolants of the side-vertex interpolant, these interpolants do not commute and so their triple Boolean sum is not well defined. However, it is possible to form the average of all double Boolean sums (each of which interpolate to the entire boundary) to arrive at the following affine invariant interpolant

$$\begin{aligned} Q^*[F] = & \frac{b_k F(b_i V_i+(1-b_i)V_k) + b_j F(b_i V_i+(1-b_i)V_j)}{b_k + b_j} \\ & + \frac{b_i F(b_j V_j+(1-b_j)V_i) + b_k F(b_j V_j+(1-b_j)V_k)}{b_i + b_k} \\ & + \frac{b_i F(b_k V_k+(1-b_k)V_i) + b_j F(b_k V_k+(1-b_k)V_j)}{b_i + b_j} \\ & - b_i F(V_i) - b_j F(V_j) - b_k F(V_k). \end{aligned}$$

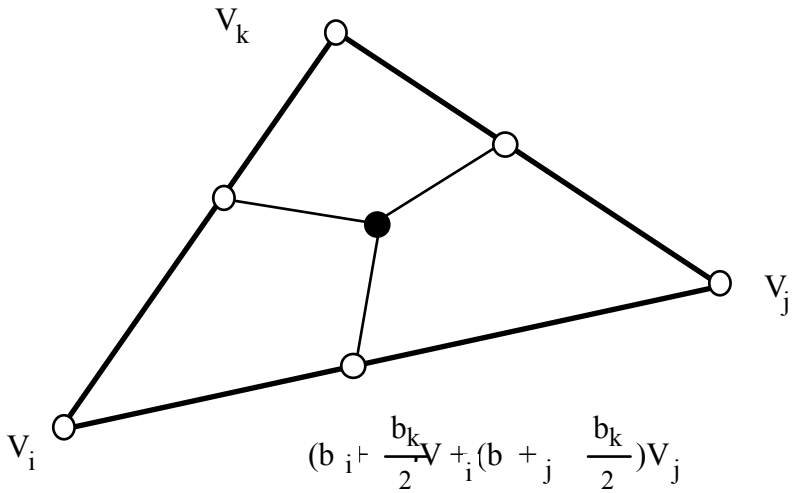


Figure 2.6.6. The stencil of the C* interpolant.

The C* Interpolant: The third transfinite, C^0, interpolant which we describe utilizes the stencil illustrated in Figure 2.6.6.

$$
\begin{aligned}
C^*[F](b_i,b_j,b_k\} = \quad & \frac{b_i b_j}{(b_i+\frac{b_k}{2})(b_j+\frac{b_k}{2})}\ F((b_i+\frac{b_k}{2})V_i + (b_j+\frac{b_k}{2})V_j) \\
& + \frac{b_i b_k}{(b_i+\frac{b_j}{2})(b_k+\frac{b_j}{2})}\ F((b_i+\frac{b_j}{2})V_i + (b_k+\frac{b_j}{2})V_k) \\
& + \frac{b_j b_k}{(b_j+\frac{b_i}{2})(b_k+\frac{b_i}{2})}\ F((b_j+\frac{b_i}{2})V_j + (b_k+\frac{b_i}{2})V_k) \\
& - \frac{3b_i b_j b_k}{(b_j+2b_k)(b_k+2b_j)}\ F(V_i) \\
& - \frac{3b_i b_j b_k}{(b_i+2b_k)(b_k+2b_i)}\ F(V_j) \\
& - \frac{3b_i b_j b_k}{(b_i+2b_j)(b_j+2b_i)}\ F(V_k)
\end{aligned}
$$

which can be written in the form

$$
\begin{aligned}
C^*[F](b_i,b_j,b_k\} = \quad & b_i F(V_i) + b_j F(V_j) + b_k F(V_k) \\
& + W_k \{ F(Q_k) - (b_i+\tfrac{b_k}{2})F(V_i) - (b_j+\tfrac{b_k}{2})F(V_j) \} \\
& + W_j \{ F(Q_j) - (b_i+\tfrac{b_j}{2})F(V_i) - (b_k+\tfrac{b_j}{2})F(V_k) \} \\
& + W_i \{ F(Q_i) - (b_j+\tfrac{b_i}{2})F(V_j) - (b_k+\tfrac{b_i}{2})F(V_k) \}
\end{aligned}
$$

where

$$W_i = \frac{4b_jb_k}{(2b_j+b_i)(2b_k+b_i)}\,,\ W_j = \frac{4b_ib_k}{(2b_i+b_j)(2b_k+b_j)}\,,\ W_k = \frac{4b_ib_j}{(2b_i+b_k)(2b_j+b_k)}\,,$$

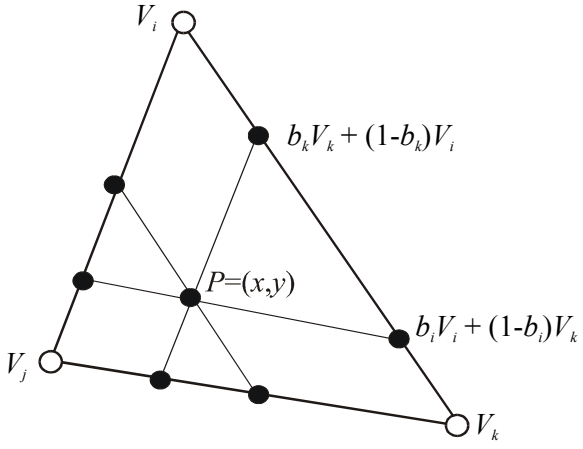
$$Q_i = (b_j + \frac{b_i}{2})V_j + (b_k + \frac{b_i}{2})V_k,$$

$$Q_j = (b_i + \frac{b_j}{2})V_i + (b_k + \frac{b_j}{2})V_k,$$

$$Q_k = (b_i + \frac{b_k}{2})V_i + (b_j + \frac{b_k}{2})V_j \ .$$

In this form of C* we can see that it consist of linear interpolation plus a correction term. It can easily be verified that C* is precise for all quadratic functions. That is, if f is a quadratic, bivariate polynomial, then C*[f] = f.

The NTW Interpolant: This may be the simplest of all triangular Coons patches. The weights are simple linear functions as are the stencil points.



$$NTW[F](b_i,b_j,b_k) = b_i[F(b_jV_j+(1\text{-}b_j)V_i)+F(b_kV_k+(1\text{-}b_k)V_i)\text{-}F(V_i)]$$
$$+ b_j[F(b_iV_i+(1\text{-}b_i)V_j)+F(b_kV_k+(1\text{-}b_k)V_j)\text{-}F(V_j)]$$
$$+ b_k[F(b_jV_j+(1\text{-}b_j)V_k)+F(b_iV_i+(1\text{-}b_i)V_k)\text{-}F(V_k)]$$

2.6.3 C^1, Discrete Interpolation in Triangles

A commonly used 9-parameter, C^1 interpolant, is

$$C_\Delta[F](x,y) = \sum_{(i,j,k) \in I} \{F(V_i)[b_i^2(3-2b_i)+6wb_i(b_k\alpha_{ij} + b_j\alpha_{ik})] + F'_{ki}(V_i)[b_i^2\, b_k+wb_i(3b_k\alpha_{ij} + b_j - b_k)] + F'_{ji}(V_i)[b_i^2\, b_j+wb_i(3b_j\alpha_{ik} + b_k - b_j)]\} ,$$

where

$$F'_{ki}(V_i) = (x_k-x_i)F_x(V_i) + (y_k-y_i)F_y(V_i) ,$$

$$F'_{ji}(V_i) = (x_j-x_i)F_x(V_i) + (y_j-y_i)F_y(V_i) ,$$

$$w = \frac{b_ib_jb_k}{b_ib_j+b_ib_k+b_jb_k} , \qquad I = \{ (i,j,k), (j,k,i), (k,i,j) \} ,$$

and

$$\alpha_{ij} = \frac{||e_{jk}||^2 + ||e_{ik}||^2 - ||e_{ij}||^2}{2||e_{ik}||^2} .$$

We use $||e_{ij}||$ to denote the length of edge e_{ij}. This 9-parameter, C^1 interpolant is a discretized version of a transfinite, C^1, triangular interpolant which is described in [178]. The derivatives which are in a direction perpendicular to an edge vary linear along an edge. This guarantees that when two of these interpolants share a common edge the two surface patches will join with continuous first order derivatives. It is possible to discrectize the same transfinite interpolant and use an additional three parameters consisting of cross boundary derivatives at the midpoints of the three edges. This leads to an interpolant that has all first order derivatives varying quadratically along the edges. For a comparison of the C_Δ interpolant to the Clough/Tocher interpolant within the context of triangle based scattered data models, see Franke and Nielson [97].

2.6.4 C^1, Transfinite Interpolation in Triangles

In this section, we extend the problem of interpolating to transfinite data on the boundary to include also the requirement that the interpolant match user specified transfinite derivative data on the boundary. These types of interpolants can be used to

construct surfaces over triangulated domains which are C^1; that is, functions which have continuous first order partial derivatives. One of the most versatile and easily described C^1, transfinite interpolants is the C^1, side-vertex interpolant [177].

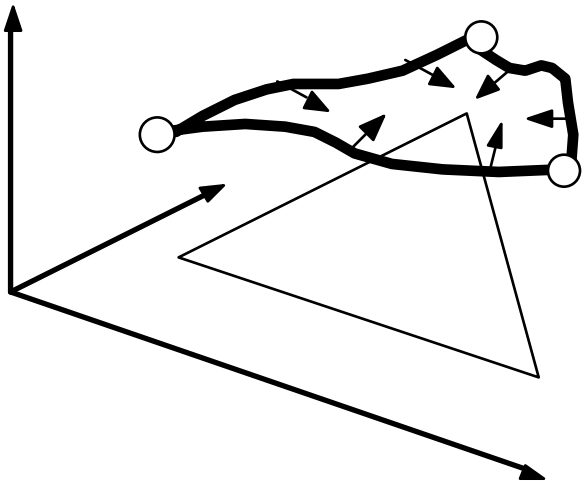


Figure 2.6.7. The data for C^1 interpolants position and derivative boundary values.

Earlier, we saw that the basic building blocks of the C^0, side-vertex interpolant consisted of linear interpolation along lines joining a vertex and its opposing side. In order to extend these ideas to C^1 data, we make use of the univariate cubic, Hermite interpolation applied along rays emanating from a vertex and joining to the opposing edge. See Figure 2.6.4. Cubic Hermite interpolation will match position and derivatives at the two ends of the interval. We assume that position and derivative information is available on the entire boundary of a triangle T_{ijk}.

$$S_i[F](p) = b_i^2(3-2b_i)F(V_i) + b_i^2(b_i-1)F'(V_i)$$

$$+ (1-b_i)^2(2b_i+1)F(S_i) + b_i(1-b_i)^2F'(S_i)$$

where $F'(V_i) = \dfrac{(x-x_i)F_x(V_i)+(y-y_i)F_y(V_i)}{1-b_i}$ and

$F'(S_i) = \dfrac{(x-x_i)F_x(S_i)+(y-y_i)F_y(S_i)}{1-b_i}$.

$S_i[F]$ has the property that it interpolates to the boundary data provided by F at V_i and on the entire opposing edge e_{kj}. It also matches first order derivatives on this edge and at V_i. It does not necessarily interpolate F or its derivatives on the other two edges. In order to have an interpolant for the entire boundary of the triangular domain, we could try to construct one using the ideas of Boolean sums as was done earlier for the C^0, side-vertex interpolant. Even though the interpolants S_i, S_j and S_k commute so that their Boolean sums are well defined, this approach does not work (see [177]) and so the use of convex combination techniques has been suggested. This leads to the interpolant

$$S[F] = \frac{b_j^2 b_k^2 S_i[F] + b_i^2 b_k^2 S_j[F] + b_j^2 b_i^2 S_k[F]}{b_i^2 b_j^2 + b_j^2 b_k^2 + b_i^2 b_k^2}$$

which has the property that it matches F and its first order derivatives on the entire boundary of the triangular domain. In the case where the boundary information has been discretized with cubically varying (Hermite) position values and linearly varying cross boundary derivatives, it is possible to obtain a final interpolant with simpler weights in the convex combination. Namely,

$$S[F] = \frac{b_j b_k S_i[F] + b_i b_k S_j[F] + b_j b_i S_k[F]}{b_i b_j + b_j b_k + b_i b_k}.$$

3 Tetrahedrizations

In this section we follow the outline of the previous section as best possible. Since the dimension is one less and since bivariate problems have been considered for a much longer period of time, the development in the 3D domain is not as rich as the 2D domain and so we can not exactly parallel the previous section, but most everything generalizes or leads to something interesting and often useful.

3.1 Basics

3.1.1 Definitions, Data Structures and Formulas for Tetrahedrizations

Our definition of a tetrahedrization follows very closely to that given for a triangulation at the beginning of Section 2.1. We start with a collection of points $p_i = (x_i, y_i, z_i)$, $i = 1, \ldots, N$ which we assume are not collectively coplanar. We denote this collection of point by P. A tetrahedrization consists of a list of 4-tuples which we denote by I_t. Each 4-tuple, $ijkl \in I_t$ denotes a single tetrahedron with the four vertices p_i, p_j, p_k and p_l. The following conditions must hold:

i) No tetrahedron T_{ijkl} , $ijkl \in I_t$ is degenerate. That is, if $ijkl \in I_t$ then p_i, p_j, p_k and p_l are not coplanar.
ii) The interior of any two tetrahedral do not intersect. That is if $ijkl \in I_t$ and $\alpha\beta\gamma\delta \in I_t$ then $\text{Int}(T_{ijkl}) \cap \text{Int}(T_{\alpha\beta\gamma\delta}) = \phi$.
iii) The boundary of two tetrahedra can only intersect at a common triangular face.
iv) The union of the all triangles is the domain $D = \cup_{ijkl \in It} T_{ijkl}$.

We should point out that condition iii) must hold in the strictest sense and so tetrahedra joining as shown in right side of Figure 3.1.1 are not allowed. The reason for this condition (and all the others) is that same as before with the conditions of a

triangulation and that is we eventually wish to be able to define C^0 functions in a piecewise manner over the domain consisting of the union of all tetrahedra.

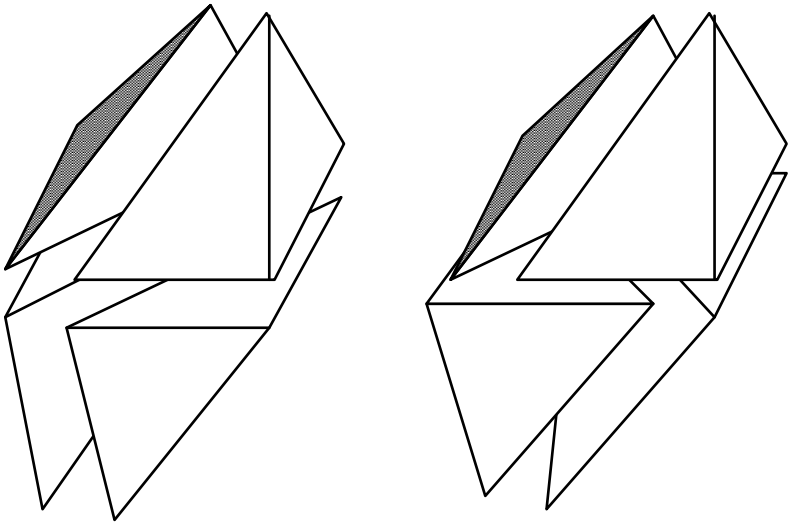
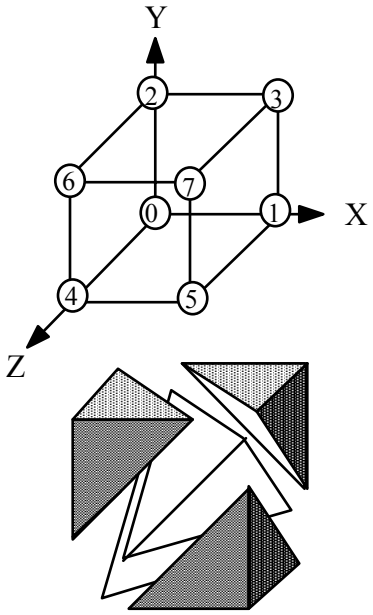


Figure 3.1.1. The configuration indicated by the diagram on the left is acceptable while that on the right is not acceptable for a tetrahedrization. It is eliminated by condition iii) above.

The triangular grid data structure for representing triangulations (illustrated in Figure 2.1.8) generalizes very nicely to a structure for representing tetrahedrizations. For example, in Figure 3.1.2, we show a tetrahedrization of the cube into 5 tetrahedra.



Tetrahedra				Neighbors			
p_i	p_j	p_k	p_l	N_{jkl}	N_{ikl}	N_{ijl}	N_{ijk}
7	4	5	1	ϕ	ϕ	5	ϕ
7	1	3	2	ϕ	ϕ	5	ϕ
2	4	0	1	ϕ	ϕ	5	ϕ
7	2	6	4	ϕ	ϕ	5	ϕ
2	4	7	1	1	2	3	4

Figure 3.1.2. An example which defines the tetrahedral grid data structure.

We saw earlier in the case of triangulations that once the boundary is specified, the number of triangles comprising the triangulation was fixed and more over we had a

simple approach to determining a formula for the number of triangles that existed in the triangulation. This property allowed for the definition of the vectors of angles which lead to the criterion for optimal triangulations and so was rather important. It would be nice if everything extended to 3D in a straightforward manner. That is, we would like to say that any polyhedron can be decomposed into tetrahedra and there is a fixed formula of the following form $N_t = aN_b + bN_i + c$ where as before, N_b and N_i are the number of vertices on the boundary and interior, respectively. Unfortunately, this is not the case and in fact the situation is much worse than that. We saw earlier that any polygon bounded region can be triangulated using only the vertices of the polygon. This is one of the first areas where matters differ significantly when going from 2D to 3D. It turns out that not every polyhedron can be tetrahedrized. The example illustrated in Figure 3.1.3 is originally due to Schoenhardt [221]. It can be visualized as a prism which has been twisted until each face (a quadrilateral comprised of two triangles) has "buckled" inward. Any tetrahedron we form from these vertices must include an edge which lies outside the domain of the "twisted prism" and so it is clear that the object can not be tetrahedrized.

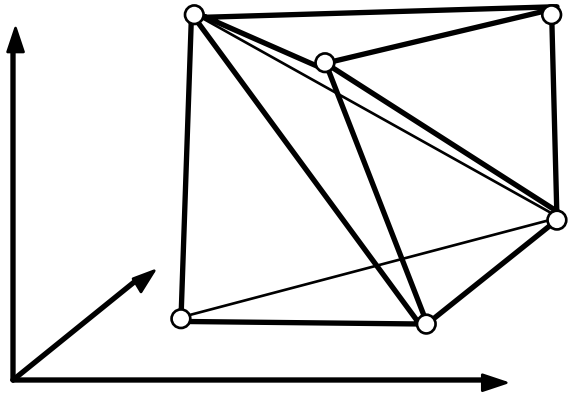


Figure 3.1.3. The twisted prism of Schoenhardt [221] which cannot be tetrahedrized.

One very basic operation does carry over in a straightforward manner from 2D to 3D and this the process of inserting an additional vertex into the interior of an existing tetrahedrization. If the new vertex p lies interior to an existing tetrahedron, say T_{abcd}, then this tetrahedron is simply replaced with the four tetrahedron, T_{abcp}, T_{abdp}, T_{bcdp}, T_{acdp} adding a net increase of three tetrahedra. If the new vertex p lies on the common triangular face of two tetrahedra, then these two tetrahedra are replaced with six new tetrahedra T_{abcp}, T_{bcdp}, T_{abdp}, T_{aecp}, T_{ecdp}, T_{aedp} resulting in a net increase of four new tetrahedra. This latter aspect of the number of tetrahedra increasing which is different here from the 2D case is that net increase in the number of tetrahedra depends on the actual location of the interior point to be inserted. This observation points out that not only can the number of ways that a data set is tetrahedrized vary, but even the number of tetrahedra can vary. We will illustrate this further with some examples even without interior points.

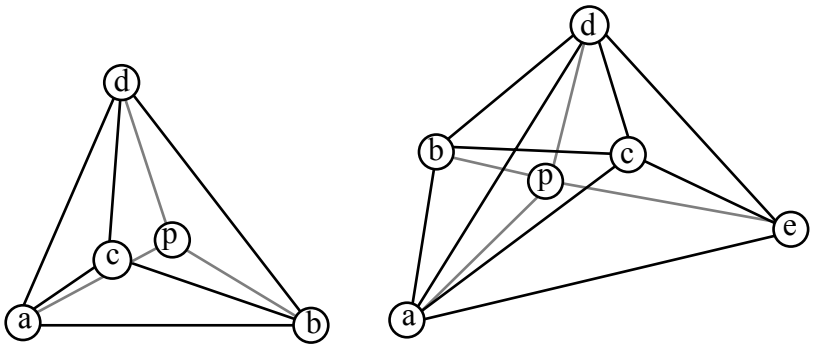
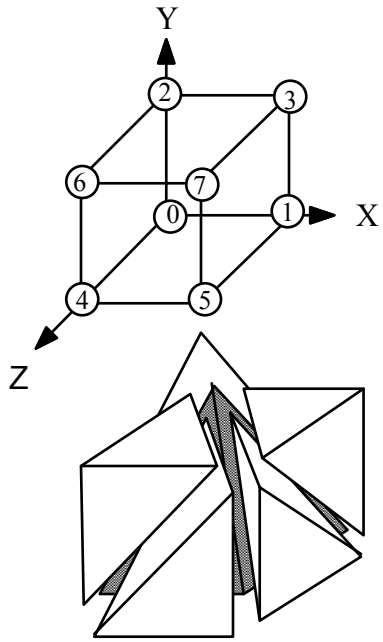


Figure 3.1.4. Inserting a point interior to an existing tetrahedrization. On the left, the new point is interior to a tetrahedron and on the right it is on a common face fo two tetrahedra.

We have already seen (see Figure 3.1.2) the decomposition of a cube into five tetrahedra. It is also possible to tetrahedrize the cube into six tetrahedra. This is illustrated in Figure 3.1.5.



Tetrahedra				Neighbors			
p_i	p_j	p_k	p_l	N_{jkl}	N_{ikl}	N_{ijl}	N_{ijk}
4	7	6	2	ϕ	ϕ	2	ϕ
4	5	7	2	4	1	3	ϕ
4	1	5	2	4	2	6	ϕ
5	1	7	2	5	2	3	ϕ
1	3	7	2	ϕ	4	ϕ	ϕ
4	1	2	0	ϕ	ϕ	ϕ	3

Figure 3.1.5. A tetrahedrization of the cube into six tetrahedra.

It is interesting to note that from the exterior, the tetrahedrization of Figure 3.1.5 looks exactly the same a that of Figure 3.1.2 as all external edges are the same. Another interesting connection between these two tetrahedrization of the cube is that one can be obtained from the other by "swapping" operations similar to those used in the Lawson algorithm for computing optimal triangulations. Previously, in the case of triangulations, there was the possibility of two triangulations of a convex quadrilateral. The analogous situation in 3D is the tetrahedrization of the region formed by five vertices when two tetrahedra meet at a common triangular face. If the line segment joining the two vertices

not on the common face intersect the interior of the common face then, analogous to the convex quadrilateral case in 2D, there is the possibility of an alternate tetrahedrization. But what is really different from the 2D case is that the number of tetrahedra changes from two to three!. This is illustrated in Figure 3.1.6. This basic operation was applied to the center and upper, back right tetrahedra of Figure 3.1.2 to arrive at the tetrahedrization of Figure 3.1.5.

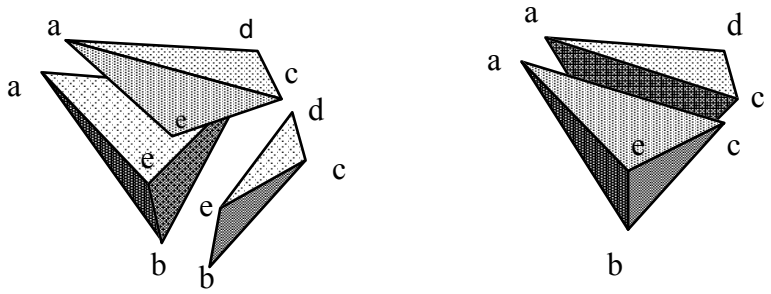


Figure 3.1.6. Two different tetrahedrizations of five points.

Another example worth noting in this context is the case where $p_i = (i, i^2, i^3)$, i = 1, . . . , N. The (Delaunay) tetrahedrization of the convex hull of this set of points consists of the tetrahedra with vertices p_i, p_{i+1}, p_j and p_{j+1} of which there are a total of ((N-2)(N-1))/2 tetrahedra. Bern and Eppstein [16] point out that this example provides an upper bound on the number of tetrahedra in a tetrahedrization of an N-vertex polyhedron and that a lower bound is provided by the fact that any tetrahedrization of a simple polyhedron has at least N-3 tetrahedra.

3.1.2 Some Special Tetrahedrizations

Following the pattern established in the earlier sections on triangulations, we first discuss tetrahedrizations related to Cartesian grids followed by tetrahedrizations associated with curvilinear grids. A 3D Cartesian grid involves three monotonically increasing sequences, x_i, i = 1, . . . , N_x , y_j, j = 1, . . . , N_y and z_k, k = 1, . . . , N_z. The grid points have coordinates (x_i, y_j, z_k) and these points mark out a cellular decomposition of the domain consisting of regular parallelepipeds. Each of these cells can be tetrahedrized in a manner similar to that given for the cube in the previous section. Probably the most popular, is the tetrahedrization involving five tetrahedra shown in Figure 3.1.2. So as to not end up with a non tetrahedrization with problems similar to those shown in the right side of Figure 3.1.1, it is necessary to "alternate" the tetrahedrization from cell to the next so that adjoining cells have the same diagonal on the common faces. This alternate tetrahedrization is not really different and is just a rotation of its companion. It is shown in Figure 3.1.8. Another popular choice is the tetrahedrization shown in the upper left corner of Figure 3.1.9. It has the advantage that all of the tetrahedra are the same shape (up to mirror images). Actually, it turns out that there are six different tetrahedrizations of a cube (parallelepiped). See Nielson [183]. We have previously shown pictures of two of them in Figure 3.1.2 and Figure 3.1.5. The other four are shown in Figure 3.1.9.

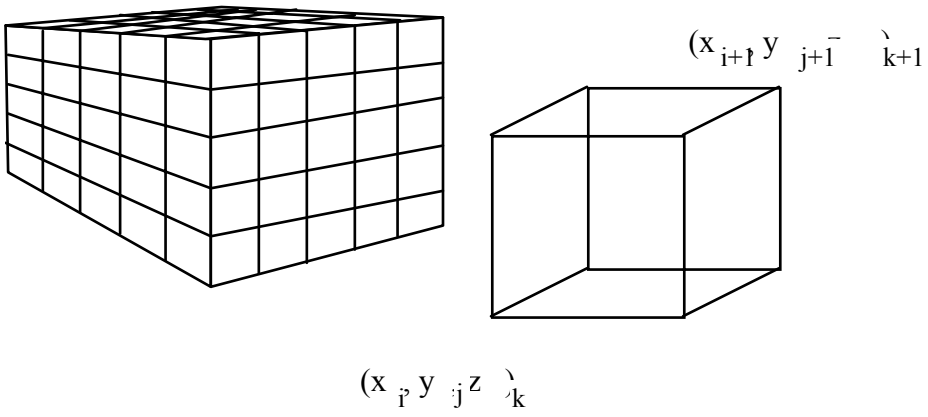


Figure 3.1.7. Three dimensional Cartesian grid.

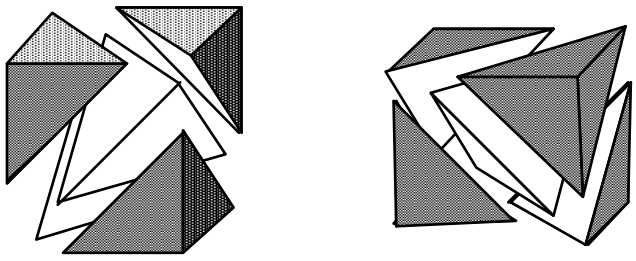


Figure 3.1.8. The two alternating tetrahedrizations with five tetrahedra of the cell of a 3D Cartesian grid. (One can be rotated to the other.)

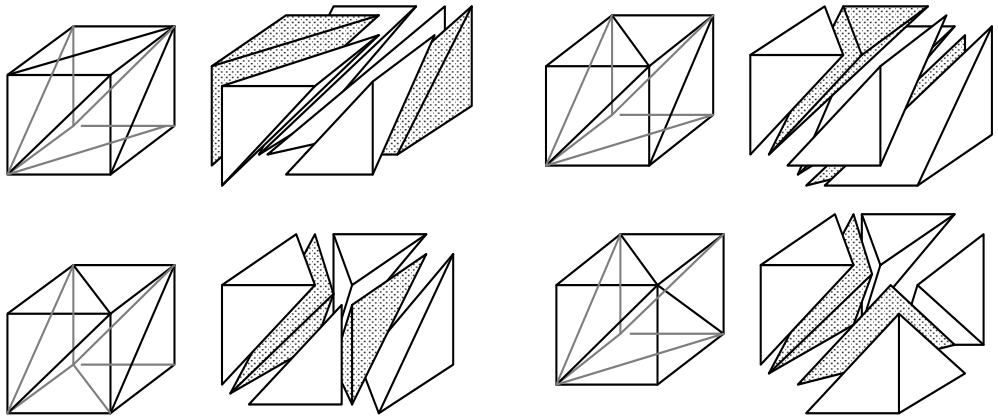


Figure 3.1.9. Four different tetrahedrizations of the cube each with six tetrahedra.

All six tetrahedrizations of the cube are comprised of five primitive tetrahedra which are shown in Figure 3.1.10. We use the names of 0F, 1F, 2Fr, 2Fl and 3F for these tetrahedra so as to indicate the number of exterior faces for each tetrahedra. There are two different primitive tetrahedra with two exterior faces; one is a mirror image version of the other and so it can not be rotated to the other. The tetrahedron 0F has volume 1/3 and all the others have volume 1/6. During informal discussion we most often use the

names 3F = "corner", 2Fr or 2Fl = "right wedge" or "left wedge", 1F = "kite" and 0F = "equi" or "fatboy".

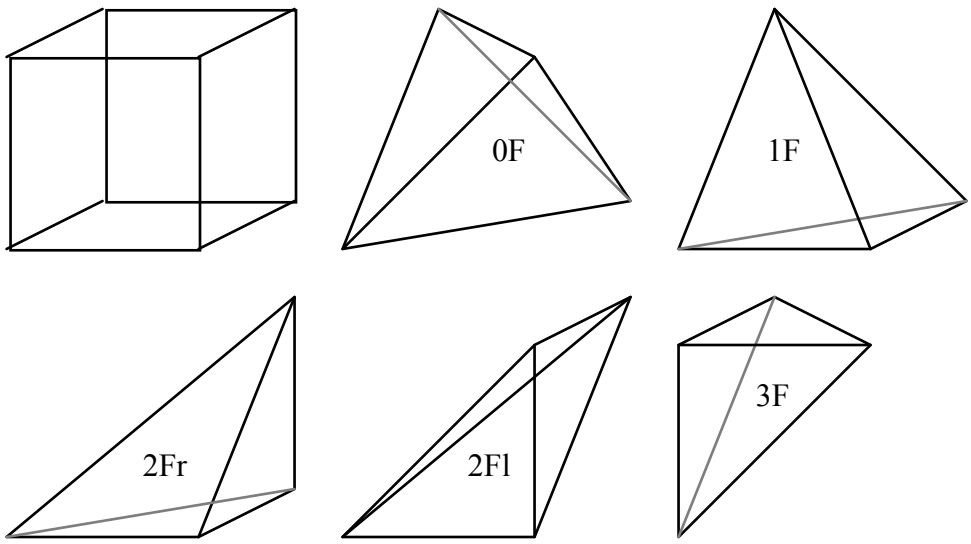


Figure 3.1.10. The five primitive tetrahedra comprising the tetrahedrizations of the cube.

In a joining similar to that shown in Figure 3.1.6, three 1F tetrahedra can come together to form the same exact shape formed by a 0F and a 3F together. Also a 2Fl and 2Fr together form the same shape as a 1F and a 3F, but two 2Fr's or two 2Fl's can not share a common face and remain inside a unit cube. There are four tetrahedrizations (each comprised of three primitive tetrahedra) of the prism making up half of the cube. They are 3F, 1F, 2Fl; 3F, 1F, 2Fr; 2Fr, 2Fl, 2Fr, and 2Fl, 2Fr, 2Fl. In Figure 3.1.11 we show the dual graphs of the six tetrahedrizations of the cube. A node is a primitive tetrahedron and an arc is a common triangular face. As expected, in each case the "names" add to twelve.

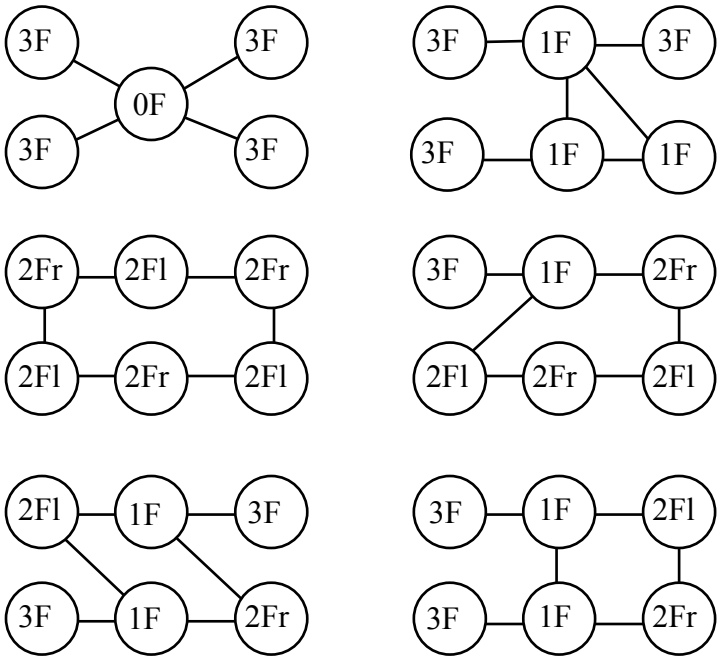


Figure 3.1.11. The six tetrahedrizations of the cube shown as dual graphs. (These are the only tetrahedrizations of the cube.)

Each of these six tetrahedrizations has its own unique and interesting properties. The tetrahedrization of 3.1.2 and Figure 3.1.5 both "swap" diagonals on all three pairs of opposing faces. The tetrahedrization shown in the lower right of Figure 3.1.9 swaps the diagonals of two pair of opposing faces and the of the upper right swaps one pair. The two tetrahedrizations on the left of Figure 3.1.9 do not swap any diagonals of any opposing faces. The tetrahedrization of the upper left of Figure 3.1.9 can be realized with three cuts of the entire cube. while the others cannot. This particular tetrahedrization also has the unique property of being comprised only of 2F primitives whose faces are all right triangles and they all (six) share the diagonal of the cube as a common edge. This tetrahedrization has been discussed and used widely. It is call the CFK-triangulation of the cube after Coxeter [47], Freudenthal [79] and Kuhn [137]. A replacement rule can be used to generate this tetrahedrization. Using the labeling scheme of Figure 3.1.2, we start with the four vertices P_{2i-1}, i = 0, 1, 2, 3 and replace each vertex Vj, other than V_0 and V_7, with $V_{j+1} + V_{j-1} - V_j$. Explicitly, this will successively generate the six tetrahedra: $p_0p_1p_3p_7$; $p_0p_2p_3p_7$; $p_0p_2p_6p_7$; $p_0p_4p_6p_7$; $p_0p_4p_5p_7$; $p_0p_1p_5p_7$. The CFK triangulation generalizes to n-dimensions as does the "replacement" algorithm for generating the simplicial decomposition.

It is interesting to note that not all possible face triangulations are realized by the six possible tetrahedrizations of the cube. In addition to the five different face triangulations (note that two tetrahedrizations have the same face triangulations) which are realizable there are three others which can not be realized. They are shown in Figure 3.1.12. In order to determine these eight unique face triangulations, we start with the $64 = 2^6$ face triangulations and then grouped them into these eight equivalence classes by rotations.

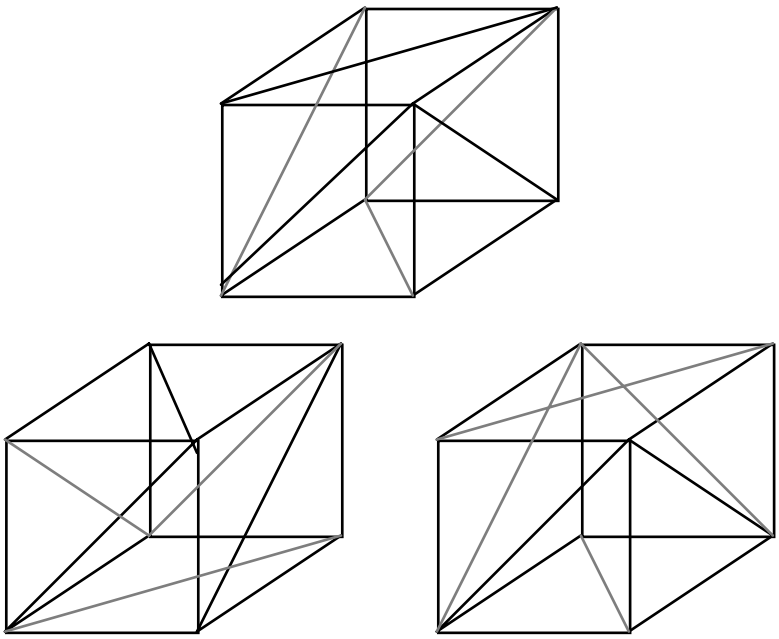


Figure 3.1.12. Face triangulations which are not consistent with any tetrahedrization of the cube.

Theorem: It is impossible to tetrahedrize a cube and yield face triangulations as shown in Figure 3.1.12.

Proof: We give only the proof for the case in the top, center as the others are similar. We use the same labeling as shown in Figure 3.1.5. We start with the face 457. Only vertex 0 can be attached to the face 457 which gives the tetrahedron 0457. The internal face 047 must be shared by some other tetrahedron. Any vertex, however, cannot be joined to the face of 457 without violating the conditions of the face triangulations and so this completes the argument.

Earlier we discussed triangulations related to curvilinear grids. We now take up the topic of tetrahedrization of 3D curvilinear grids. Analogous to the 2D situation, a 3D curvilinear grid is specified by three geometry arrays x_{ijk}, y_{ijk}, z_{ijk}, i = 1, . . ., Nx; j = 1, . . . , Ny; k = 1, . . . , Nz. In the 2D case a cell C_{ij} consisted of the quadrilateral with vertices (x_{ij}, y_{ij}), $(x_{i+1,j}, y_{i+1,j})$, $(x_{i,j+1}, y_{i,j+1})$, $(x_{i+1,j+1}, y_{i+1,j+1})$, and the cells serve as a decomposition of the domain.

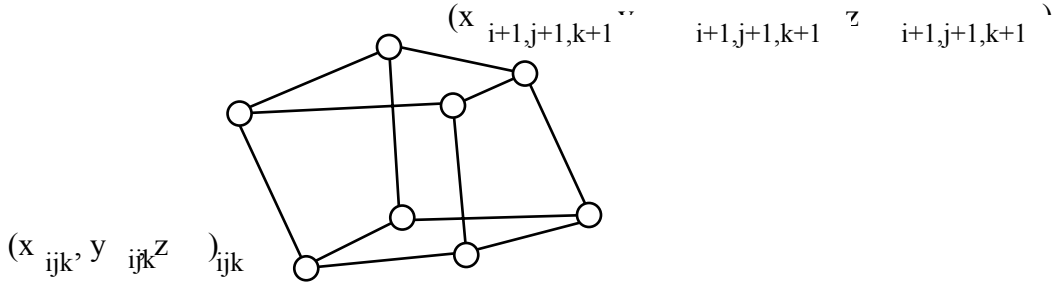


Figure 3.1.13. Single cell of a 3D curvilinear grid.

In the 3D case, matters are not as straightforward as we might expect and there are some areas where we need to be concerned. These have mainly to do with just exactly what comprises a cell. In 3D the cell C_{ijk} has the eight vertices $(x_{abc}, y_{abc}, z_{abc})$, a = i, i+1, b = j, j+1, c = k, k+1 but there is not always a consistent definition for the cell boundaries. We mention briefly some possible choices. If the geometry arrays are constrained so that each collection of four vertices of the six "faces" of the cells are coplanar, then an obvious choice for the cell boundaries is this common planar quadrilateral. In this case the cells are hexahedron and it is relatively easy to determine whether or not an arbitrary point, (x, y, z) is in a particular cell or not. Often this planarity condition does not hold and cell boundaries are taken to be the parametrically defined (hyperbolic) surface obtained by substituting 0 or 1 for any of the parameter value s, t, u in the following trilinear mapping:

$$\begin{aligned} C_{i,j,k}(s,t,u) \quad &= (1-s)(1-t)(1-u)P_{i,j,k} + (1-s)(1-t)uP_{i,j,k+1} \\ &+ (1-s)t(1-u)P_{i,j+1,k} + (1-s)tuP_{i,j+1,k+1} \\ &+ s(1-t)(1-u)P_{i+1,j,k} + s(1-t)uP_{i+1,j,k+1} \\ &+ st(1-u)P_{i+1,j+1,k} + stuP_{i+1,j+1,k+1} \end{aligned}$$

where

$$P_{i,j,k} = (x_{i,j,k}, y_{i,j,k}, z_{i,j,k})$$

Given a point (x,y,z) in the cell C_{ijk}, the value (s, t, u) which associates with it via the trilinear mapping is called the corresponding *computational coordinate*. In fact, in order to determine whether or not an arbitrary point is in this type of cell or not requires that we solve the three nonlinear equations which represent this association. This can be a considerable problem from a computational point of view. Most methods use some heuristics to obtain an initial approximation for some type of Newton's method. Another choice for the cell boundaries in the event the four vertices of a face are not coplanar is choose them to be piecewise planar. That is, a diagonal edge is selected and boundary between the two cells consists of the two triangles which result. Often the cell would be further decomposed into tetrahedra thus leading to a an overall tetrahedrization of the curvilinear grid. We should point out that not all choices for the diagonals can lead to a tetrahedrization of the cell. In order to be specific about this, consider the cell illustrated in Figure 3.1.14. This cell was created from a unit cube by cutting notches in the faces so as to force the diagonal edges p_2p_7, p_4p_1, p_3p_5, p_3p_0, p_0p_6, p_6p_5 to be exterior to the cell. If the depth of the notches is ε then this results in the points $p_o = (0, \varepsilon, 0)$, $p_1 = (1-\varepsilon, 0, \varepsilon)$, $p_2 = (\varepsilon, 1, \varepsilon)$, $p_3 = (1, 1-\varepsilon, 0)$, $p_4 = (\varepsilon, 0, 1-\varepsilon)$, $p_5 = (1, \varepsilon, 1)$, $p_6 = (0, 1-\varepsilon, 1)$ $p_7 = (1-\varepsilon, 1, 1-\varepsilon)$. Note that p_6, p_3, p_4 and p_1 all lie in the plane $x + z - 1 = 0$ and p_2, p_7, p_0 and p_5 are in the plane $x - z = 0$.

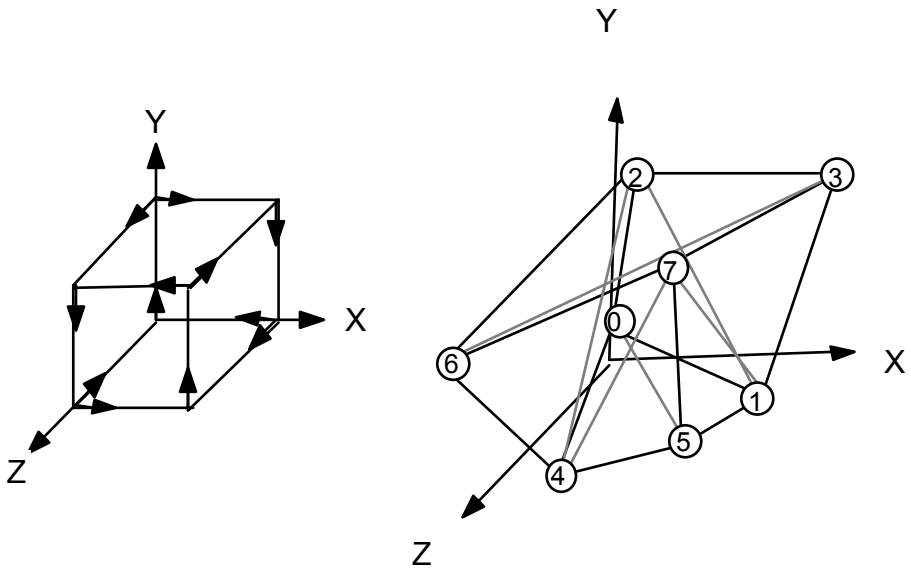


Figure 3.1.14. A curvilinear grid cell (polyhedron) that can't be tetrahedrized.

Theorem: The polyhedron of Figure 3.1.14 can not be tetrahedrized.

Proof: Consider the triangle face with vertices p_6, p_4 and p_7. In any tetrahedrization, this face must be joined to some vertex to form a tetrahedron. By considering the remaining five vertices p_5, p_0, p_2, p_1 and p_3 we find that the only p_3 would not lead to a tetrahedron with an edge which is outside the cell. If the tetrahedron p_6, p_4, p_7 and p_3 is included in the list of tetrahedra, then the interior triangle face $p_3p_4p_7$ must connect to another vertex (besides p_6) to form a tetrahedron. But a consideration of each of the possible vertices p_5, p_1, p_2 and p_0 each lead to an edge which is exterior to the cell and this concludes the argument.

We conclude this discussion on the tetrahedrization of the cells of a curvilinear grid by pointing out that some hexahedra will decompose into seven tetrahedra. Consider the cell of Figure 3.1.13 and let the six faces be planar, but assume that the four diagonal points p_{ijk}, $p_{i+1,j+1,k+1}$, $p_{i,j,k+1}$ and $p_{i+1,j+1,k}$ are not coplanar so that they will form a tetrahedron. Remove this tetrahedron leaving two prisms with two planar quadrilateral faces which can each be decomposed into three tetrahedra. We should point out that we have observed cases where this decomposition was the Delaunay tetrahedrization.

In Section 2.1.3 we described two different approaches leading to nested subdivision triangulations and pointed out their potential value in multiresolution approximations. These both have analogs in 3D and these are shown in Figures 3.1.15 and 3.1.16 respectively. The first one is based upon recursive subdivision and the second one is called "symmetric" subdivision and is related to the CFK-tetrahedrization of the cube [170]. It is comprised of six 2Fr's and two 2Fl's and is the same shape and twice the size of one 2Fr.

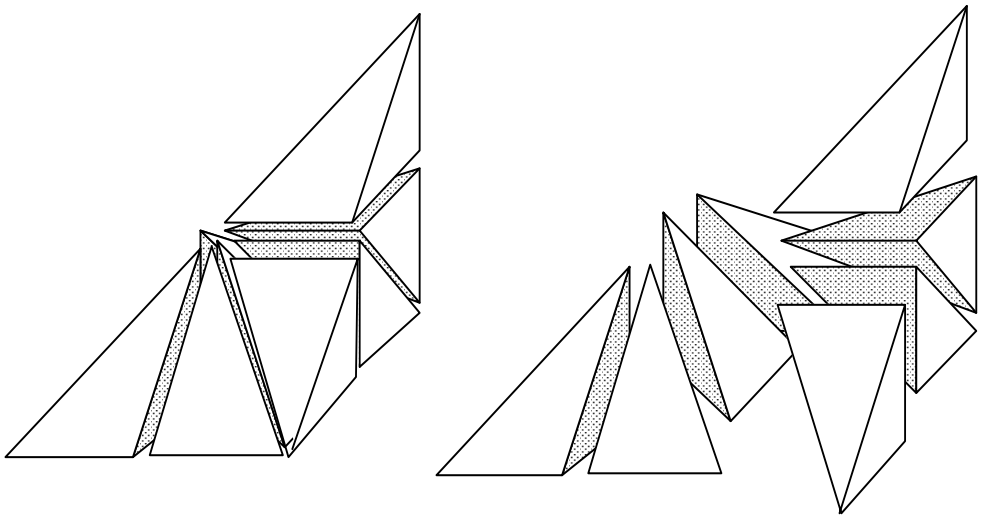


Figure 3.1.15. Nested tetrahedral subdivision analogous to that of Figure 2.1.16.

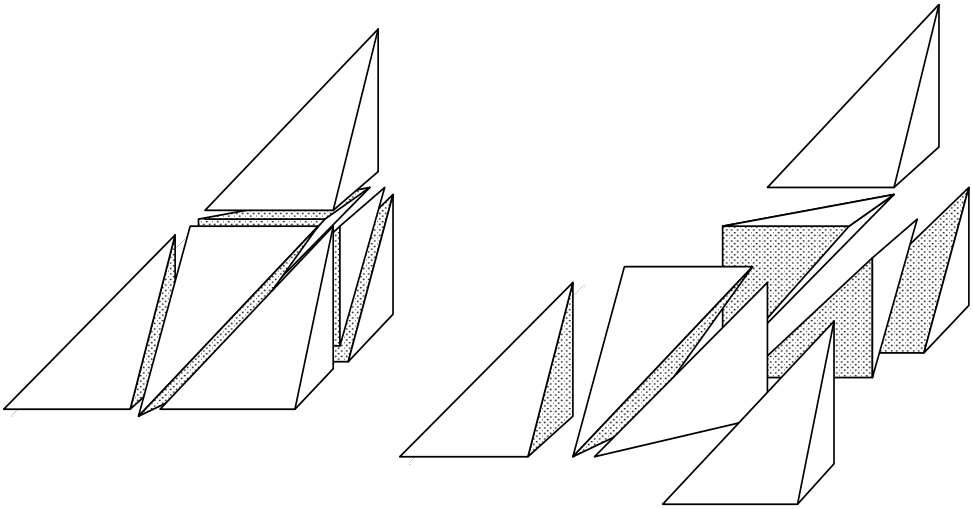


Figure 3.1.16. Symmetric nested tetrahedral subdivision.

It should be noted that if primitive tetrahedra of the shape shown in Figure 3.1.17 are assembled as in Figure 3.1.16, then we obtain a composite tetrahedron which is twice the size and exactly the same shape as the primitive tetrahedron. This particular tetrahedrization of tetrahedra is related to the Delaunay tetrahedrization of the BCC lattice which is the union of the lattices { (i,j,k) : i, j and k are integers} and {(i+1/2, j+1/2, k+1/2): i, j and k are integers}. See also Senechal [229] for a discussion of tetrahedra that can be decomposed into similar tetrahedra.

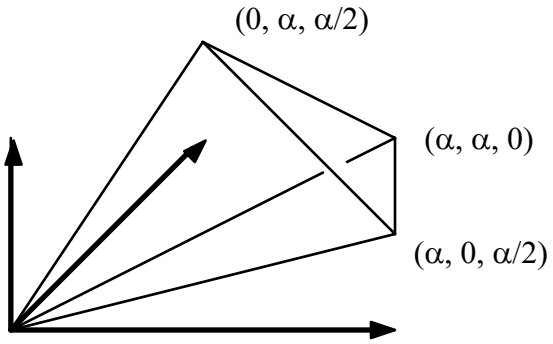


Figure 3.1.17. A tetrahedron that can be tetrahedrized into eight tetrahedra each of which are the same shape as the original yet half size.

3.2 Algorithms for Delaunay Tetrahedrizations

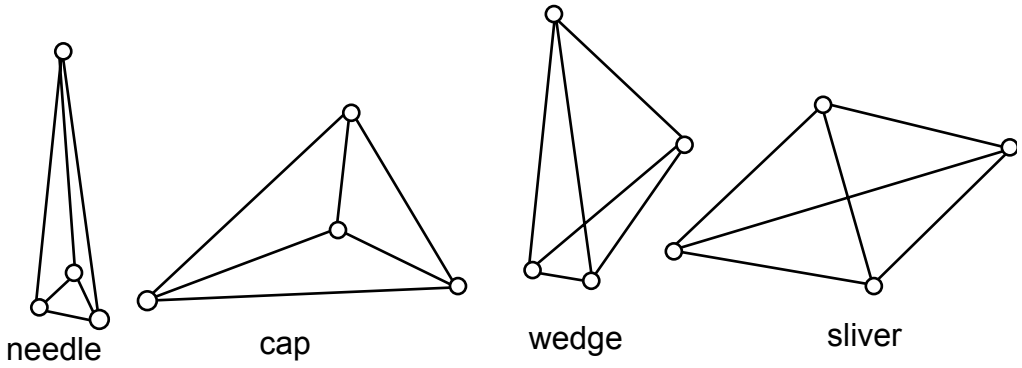


Figure 3.2.1. Examples of poorly shaped tetrahedra.

Analogous to the examples of Figure 2.2.1, examples of poorly shaped tetrahedra are shown in Figure 3.2.1. The sliver has small dihedral angles, but need not have any small planar angles. Several measures of the quality of tetrahedrizations have been proposed. See Baler [12] and Field [86]. For example the ratio of the inradius (radius of inscribed sphere) and the circumradius. The problem here is there is no apparent way to order the collection of all tetrahedrizations of a point set. The approach of lexicographically ordering the associated vectors of angles as we described in Section 2.2 does not extend to 3D because the number of tetrahedra in a tetrahedrization is not necessarily fixed. Nevertheless, the Delaunay tetrahedrization of the convex hull which is dual to the Dirichlet tessellation is well defined (in the absence of neutral cases where points lie on a common sphere) and so the remainder of this section is devoted to a discussion of the extension of the previously discussed 2D algorithms for computing the Delaunay triangulations to the case of 3D tetrahedrizations.

Extension of Lawson's Algorithm (Incremental Flipping): It is possible to extend this algorithm to 3D, but the extension is not as simple as one might expect. The first major difference that one encounters is the character of the basic swapping step. In 2D we take an edge and consider the quadrilateral formed by the two triangles which share this edge. If the quadrilateral is convex we can swap the diagonal if this step moves us closer to the optimal solution which can easily be determined by applying the circle

inclusion test. Two triangles are replaced by two other triangles. But the analogous steps in 3D can lead to a situation where the two tetrahedra sharing a face can be replaced with 3 tetrahedra. See Figure 3.2.2 for an example.

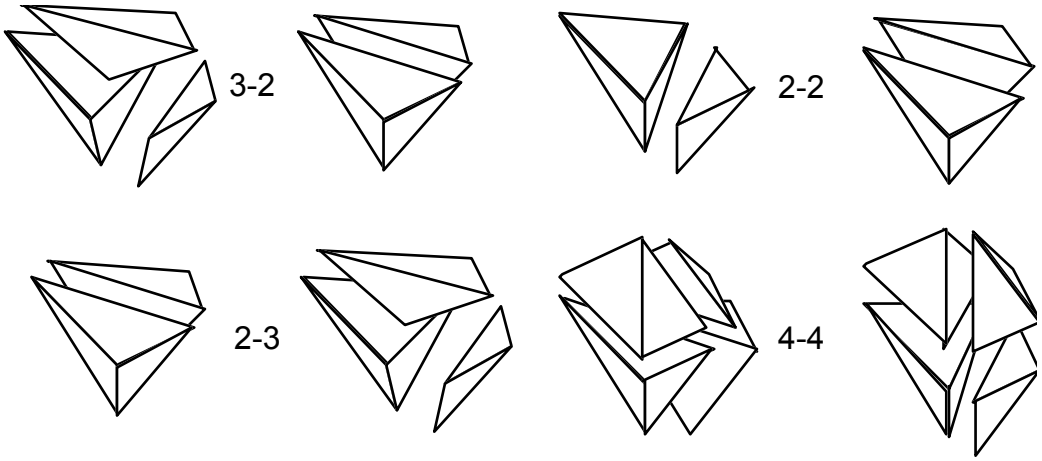


Figure 3.2.2. Different cases of swapping for 3D version of Lawson's algorithm.

Joe [122] showed that if the points are inserted in a particular manner, then incremental flipping will lead to the optimal Delaunay tetrahedrization. Edelsbrunner and Shah have generalized these results [72]. Software based upon these ideas is provided by the Software Development Group at the National Center for Supercomputing Application is available at the WWW site:
http://www.ncsa.uiuc.edu/SDG/Brochure/Overview/ALVIS.overview.html.

Extensions of the algorithm of Green & Sibson: There does not seem to be an apparent method of extending this type of algorithm to 3D. The algorithm is dependent upon the "contiguity list" and here lies the difficulty to extend to 3D. We included this algorithm in our selection of 2D algorithms so that this very point could be made. Some concepts extend easily to 3D and others do not.

Bowyer's Algorithm for 3D: It is a straight forward exercise to extend Bowyer's 2D algorithm to 3D. In fact, the original paper of Bowyer [21] describes the algorithm for arbitrary dimensions. Bowyer also mentions that with some care, the algorithm can be extended to other domains. In [164] there is a brief discussion of Bowyer's algorithm along with some code.

Watson's Algorithm for 3D: The original description of Watson's algorithm applies to arbitrary dimension. In the paper [254] results for 2, 3 and 4 dimension are reported. Information on implementing this algorithms in 3D is given by Field in [86] and [87]. It is also the basis for the 3D algorithms discussed in [29].

Embedding/Lifting Algorithms for 3D: Software for computing general dimension convex hulls and Delaunay tetrahedrizations based on the relationship mentioned earlier in Section 2.2 are provided by the Geometry Center, University of Minnesota at the WWW site: http://freeabel.geom.umn.edu/software/download/qhull.html.

3.3 Visibility Sorting of Tetrahedra

We first give a motivation for the definition and the need of a visibility sort. We use the example of volume rendering which is a means of graphing (visualizing) a density function (cloud) d(x,y,z) defined over a 3D domain (which is often a cube). A view point V is selected along with a projection plane. A rectangular portion of the projection plane is subdivided into a rectangular array of subrectangles which associate directly with the pixels of an image to be generated. The RGB value for each pixel is defined by

$$F(i,j) = \int_0^D \delta(s)C(s)\, e^{-\int_s^D \delta(u)du} ds \; + F_0 e^{-\int_0^D \delta(u)du} \tag{3.3.1}$$

where the integral is taken along the ray emanating from the viewpoint and passing through the center of the subrectangle associated with the pixel at location (i,j), F_0 is the background intensity and D is a distance along the ray sufficiently large so that the ray completely passes through the domain of interest. The function C, also defined over the same domain as δ, is called the color function and governs the color of light emanating (by reflection say) from a point within the density cloud. In actual application the integrals are approximated by numerical schemes based upon sampled values of the integrand. The sample values are often obtained by some simple interpolation into the cells covering the domain. And these cells are often a result of the positions where δ has been measured. If we let $0 = x_0 < x_1 < x_2 < \bullet\bullet\bullet < x_{n-1} < x_n = D$ be the distances from the viewpoint to each sampled value along the ray then the upper Riemann sum approximation to this integral is

$$F_n = \sum_{i=o}^{n} \Delta x_i \delta(x_i) C_i \prod_{j=i+1}^{n} t_j \;, \tag{3.3.2}$$

where $C_i = C(x_i)$, $t_j = e^{-\Delta x_i \delta(x_i)}$ and $\Delta x_i = x_i - x_{i-1}$. This discrete approximation can be computed by the compositing process

$$F_i = t_i F_{i-1} + I_i \;, \tag{3.3.3}$$

where $I_i = \Delta x_i \delta(x_i) C_i$.

Another way to view this compositing process is as a simple model of transparency where an object of thickness Δx_i attenuates the incoming light intensity F_{i-1} by the factor t_i and this object emits light of intensity I_i. Algorithms which accumulate these values

into a frame buffer (with each location holding the value for a pixel) can either be image space oriented or object space oriented. Image space algorithms proceed along the lines of our development here and accumulate all contributions for a pixel along a particular ray. Object space algorithms compute exactly the same values but the calculations are done in a different order. These algorithms sequential process each cell by accumulating into the proper location of the frame buffer all contributions of a particular cell. Due to the nature of the compositing process, it is mandatory that these accumulations be done in the proper order. It is this latter approach which motivates the definition and need for visibility sorting in this context.

Definition of Visibility Order: Let T and T' be tetrahedra of a tetrahedrization and let V be the enter of perspective projection. If there is a ray emanating from V which intersects T' before T, then T is said to precede T' and we write $T < T'$.

The purpose of a visibility sort is find a linear ordering of all of the tetrahedra of a tetrahedrization so that the ordering relation is never violated.

Definition of Visibility Ordering: A visibility ordering of a tetrahedrization is a sequence, $n_1, n_2, \ldots, n_T$ which has the property that whenever $T_{n_i} < T_{n_j}$ then $i < j$.

The implication of the definition of visibility ordering for splatting or object space traversal algorithms for volume rendering is that a tetrahedron T must be processed (sampled and composited into the frame buffer) before T' whenever $T < T'$.

A couple of items should be noted at this point. The relation of visibility order is in the strict mathematical sense not a partial ordering. A partial ordering is required to be i) transitive: $x < y$, $y < z$ implies $x < z$; ii) antisymmetric: $x < y$ and $y < x$ implies $x = y$; and iii) reflexive: $x < x$. It is entirely possible that a visibility order could not exist at all due the presence of cycles as shown in Figure 3.3.1.

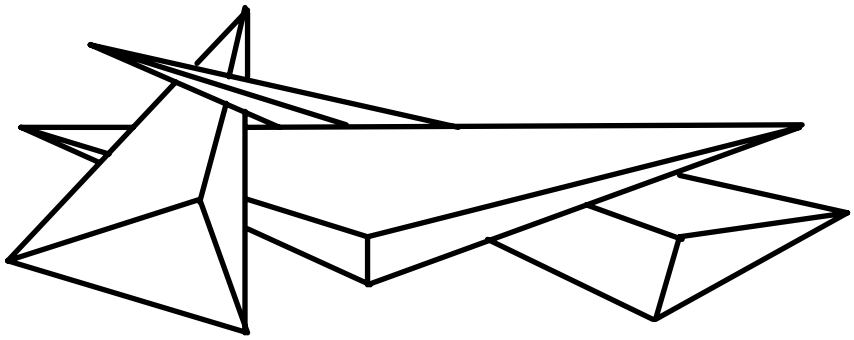


Figure 3.3.1. An example of three tetrahedra that can not be visibility ordered.

Knuth [136] has discussed in some detail (including MIX programs) the topological sort algorithm as a means of "embedding a partial order in a linear order." A linear ordering is a partial ordering where either $x < y$ or $y < x$ for all x, y. Even though this does not strictly apply in the context of a general tetrahedrization, the basic ideas (mainly due to the manner in which it is described) are very useful for developing visibility

sorting algorithms for specific applications and so we include a description of the topological sort algorithm here.

Topological Sort Algorithm: The topological sort algorithm as described by Knuth [136] starts with a directed, acyclic graph (DAG). The DAG can be represented with a diagram using nodes and arrows. See Figure 3.3.2. The nodes represent the elements of the set to be ordered and an arrow from node x to node y represents the relation of the partial ordering, x < y. The algorithm is simple. Any node that has no incoming arrow is removed from the DAG (with all of it attached arrows) and placed in the linear ordering. This process is repeated until the DAG is empty. It is easy to prove (left to the reader) that if the DAG represents a partial ordering, a linear ordering will always be produced by this algorithm.

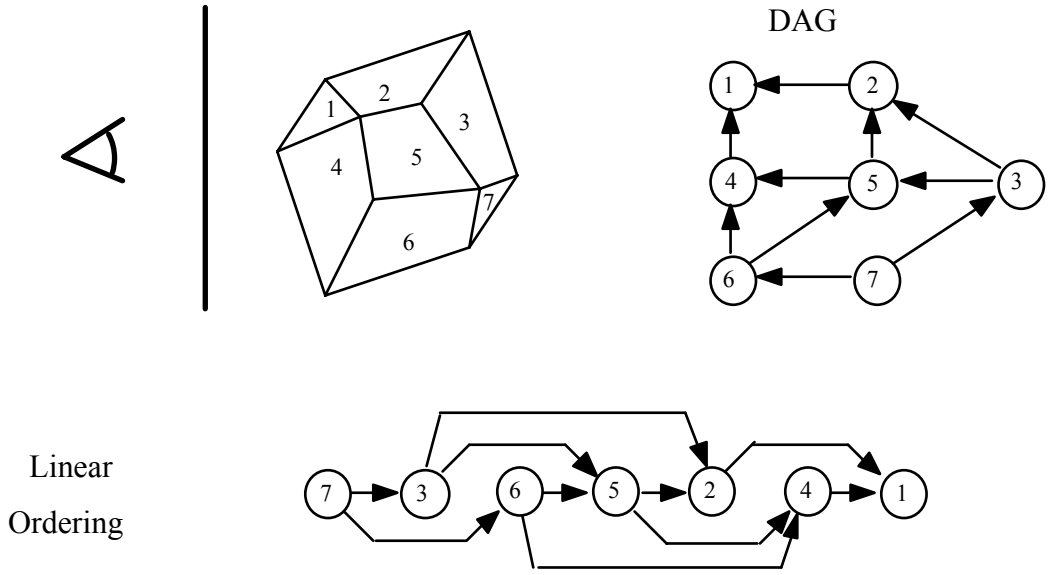


Figure 3.3.2. An example of the topological sort algorithm.

Max [166] has discussed the application of the ideas of the topological sort algorithm to the problem of producing a visibility sort for a cellular decomposition of a domain. Max defines the order relation in the following way. The DAG contains an arrow for each face common to two cells x and y. The arrow is directed from x to y if the viewpoint is on the same side of the face as x meaning that y must be process before x. Max mentions that the topological sorting algorithm will be successful "if every ray through the data volume intersects it in a single sequence of adjacent cells." Of course, if the cell complex contains cycles (see Figure 3.3.1), then a visibility sort is not possible. Williams [257] discusses similar algorithms applied to a very general cellular decomposition which may contain empty cavities.

We conclude this section with some rather interesting properties about the special case of the Delaunay tetrahedrization of the convex hull of a collection of 3D points. The *power* of a tetrahedra is defined as $D^2 - R^2$ where D is the distance to the viewpoint from the center of the circumsphere of the tetrahedron and R is the radius of the circumscribing sphere. A visibility sort can be accomplished by a simple sort based upon

the power. This property is covered [69] and used by Max, Hanrahan, and Crawfis [167]. We caution the reader that this approach breaks down in the presence of neutral cases where possibly several tetrahedra have the same power (as in the case of decomposing the cube). One additional; interesting observation in this context is that a sort based upon the power of the tetrahedra does not require the neighborhood information as is required for the algorithms using the ideas of topological sorting. Another method which does not use adjacency information is described by Stein, Becker and Max [240].

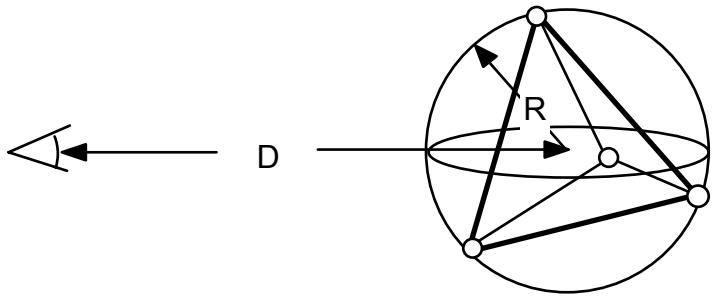


Figure 3.3.3. Elements of the definition of the *power* of a tetrahedron.

3.4 Data Dependent Tetrahedrizations

Lee [148] has investigated the topic of data dependent tetrahedrizations. This work generalizes from 2D to 3D the ideas and techniques of [67] and [225]. Similar to the algorithms of [225], simulated annealing is used. The initial tetrahedrization is the Delaunay tetrahedrization of the convex hull of the independent data site locations. Local swapping of tetrahedra is performed based upon random values compared to an annealing schedule and a cost function. This "randomness" of the simulated annealing approach allows the algorithm to escape local extrema of the cost function. Local swapping for 2D simply involves the choice of one or the other of the diagonals of a quadrilateral. In 3D the situation is more complex. There are four cases which are shown in Figure 3.2.2 which are the same as those used in the 3D version of Lawson's algorithm. In the first case, three triangles are swapped for two. The second case is the reverse of the first case and two tetrahedra are replaced by three tetrahedra. The third case is where two triangles are on the boundary of the convex hull and the two tetrahedra can be swapped for two other tetrahedra. In the last case four tetrahedra are swapped for four other tetrahedra.

In Section 2.4, we described the cost function used by Dyn, Levin & Rippa [67]. Analogous to these cost functions for 2D, Lee [148] uses the following criterion for 3D:

Gradient Difference: Let T_1 and T_2 be two tetrahedra with a common triangular face. Let G_1 be the gradient of the linear function which interpolates the data at the four vertices of T_1 and let G_2 be the similar gradient for the linear interpolant of T_2. The *gradient difference* is defined as $||G_1 - G_2||$.

Jump in Normal Direction Derivatives: Let $L_1(x,y,z) = a_1x + b_1y + c_1z + d_1$ be the linear function which interpolates to the data at the four vertices of T_1 and let $L_2(x,y,z) =$

$a_2x + b_2y + c_2z + d_2$ be the similar function for T_2. Let $N = (n_x, n_y, n_z)$ be the normal (normalized) of the common triangular face of T_1 and T_2. The $D_1 = a_1n_x + b_1n_y + c_1n_z$ is the directional derivative of L1 in the direction of N. $D2 = a_2n_x + b_2n_y + c_2n_z$ is the analogous value for T_2. The *jump in normal direction* criterion is $|D_1 - D_2| = |(a_1-a_2)n_x + (b_1-b_2)n_y + (c_1-c_2)n_z)|$.

Some example results reported by Lee [148] are repeated here in Figure 3.4.1. This example involves a test function, F(x,y) = (Tanh(9y-9x-9z) + 1)/9, which provides the dependent data. The piecewise linear interpolant over the tetrahedrization is compared to the test function. The RMS errors based upon evaluations of the functions and this approximation over a 20×20×20 Cartesian grid. The dependent data sit locations are taken to be 1000 random points in the unit cube.

Method	RMS Error
Delaunay	.007475
Difference in Gradient	.005445
Jump in Normal Derivative	.004361

Figure 3.4.1. Errors for the piecewise linear interpolant using different tetrahedrizations.

In Figure 3.4.2 are shown some graphs which can be considered as 3D analogs of the graphs shown in Figure 2.4.2 of Section 2.4. Similar to the 2D case the data dependent tetrahedrization involves some badly shaped tetrahedra. This is the cost of having an optimal (or nearly optimal) piecewise linear approximation.

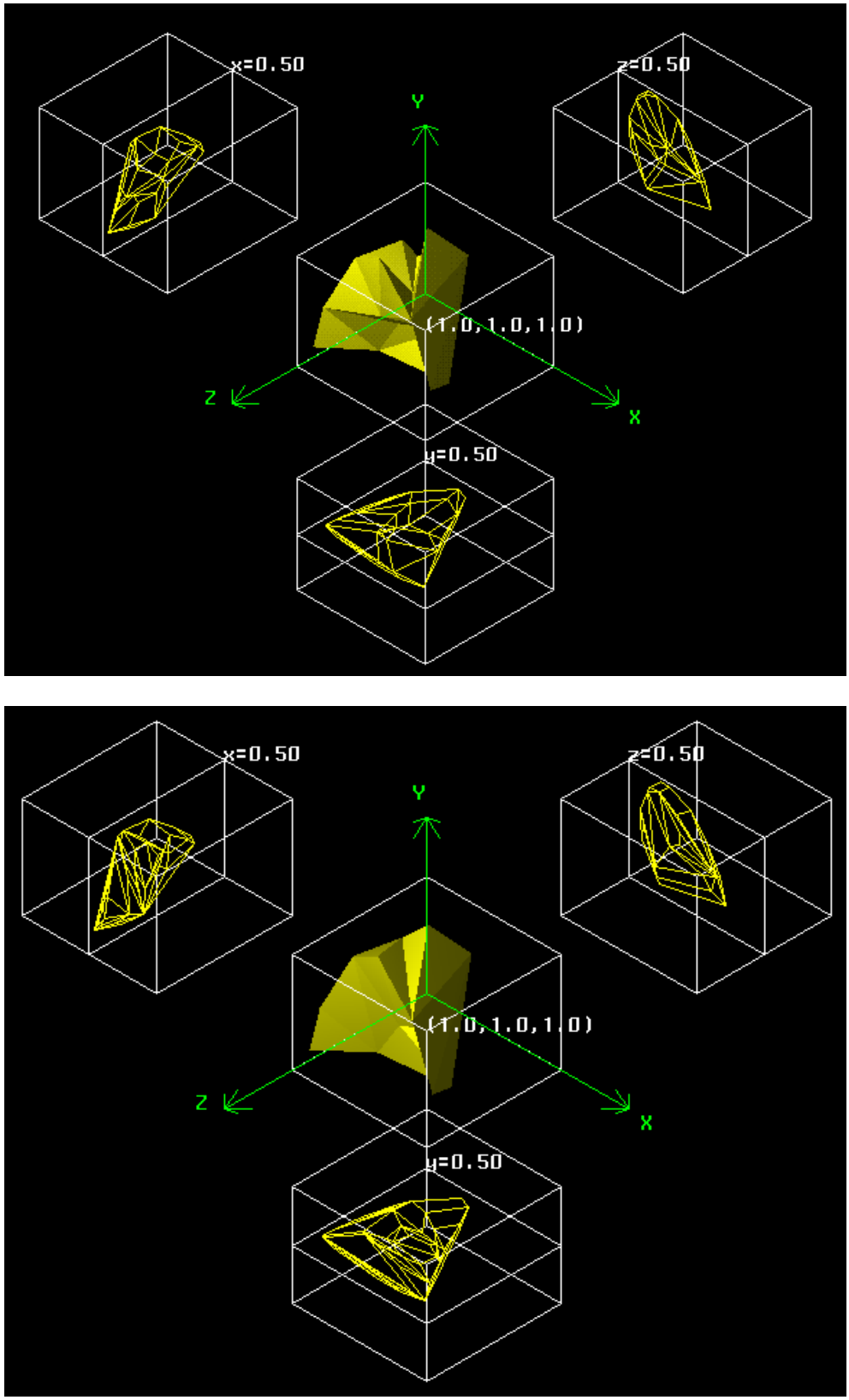
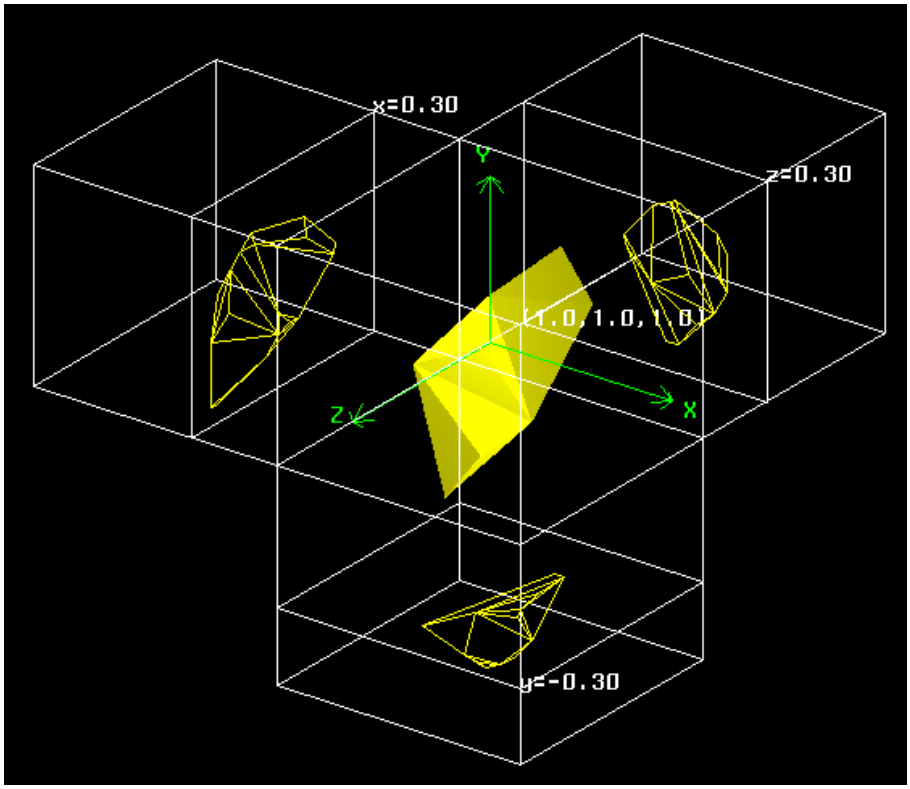


Figure 3.4.2. Data dependent tetrahedrization compared to the Delaunay tetrahedrization.

3.5 Affine Invariant Tetrahedrizations

In this section we extend the results of Section 2.5 on affine invariant triangulations to that of affine invariant tetrahedrizations. Prior to discussing the characterization and computation of this type of tetrahedrization, we make some comments about the need for such a tetrahedrization over and above those reasons for the 2D case. It appears that as the dimension of the independent data increases, our need to be concerned about lack of affine invariance also increases.

One source of 3D independent data is the case of time varying 2D data. In some cases the data measurement locations might stay fixed over time and some cases they may vary over time. For example, if we have a vector field which is known (say by means of a numerical simulation) at the locations of a 2D curvilinear grid (x_{ij}, y_{ij}), i = 1, . . . , Nx; j = 1, . . . , Ny. As time proceeds, the vector field varies, but the dependent data site locations stay fixed. So in this case, we have data which can be represented as $(V_{ijk}; x_{ij}, y_{ij}, t_k)$, i = 1, . . . , Nx , j = 1, . . . , Ny, k = 1, . . . , N_t. If the definition of a modeling function F(x, y,t), designed to interpolate the data, $F(x_{ij}, y_{ij}, t_k) = F_{ijk}$, is based upon a tetrahedrization of the 3D independent data (x_{ij}, y_{ij}, t_k), then this model will not necessarily be affine invariant and the units used to measure and represent the physical coordinates and time could have an affect on the modeling function F(x, y, t) and subsequently an affect on the visualization and analysis. The same problem could also occur for time varying vector field over a curvilinear grid which also varies over time. That is, data of the type $(F_{ijk}; x_{ijk}, y_{ijk}, t_k)$, i = 1, . . . , Nx, j = 1, . . . , Ny, k = 1, . . . , N_t. In general, any tetrahedrization of the independent data of (F_{ijk}, x_i, y_j, z_k) where the choice of the units of measurement used for the independent data could lead to a non-uniform scaling could have the problem of being dependent on the choice of the units used. If each of the variables use the same units then there will be no problems of this type because a scale transformation of the form $x \leftarrow ax$, $y \leftarrow ay$, $z \leftarrow az$ where the scale change is uniform for each variable will not affect the tetrahedrization. It is only the non-uniform scaling $x \leftarrow ax$, $y \leftarrow by$, $z \leftarrow cz$ which creates the problem. An example of a scale change affecting the tetrahedrization is shown in Figure 3.5.1. Here there are 10 data points. In the right image, the data has been scaled in the y-variable by a factor of 2. Not only does the tetrahedrization change, but even the number of tetrahedra changes. The Delaunay tetrahedrization of the original 10 data points has eighteen tetrahedra and the scaled data has thirteen tetrahedra.



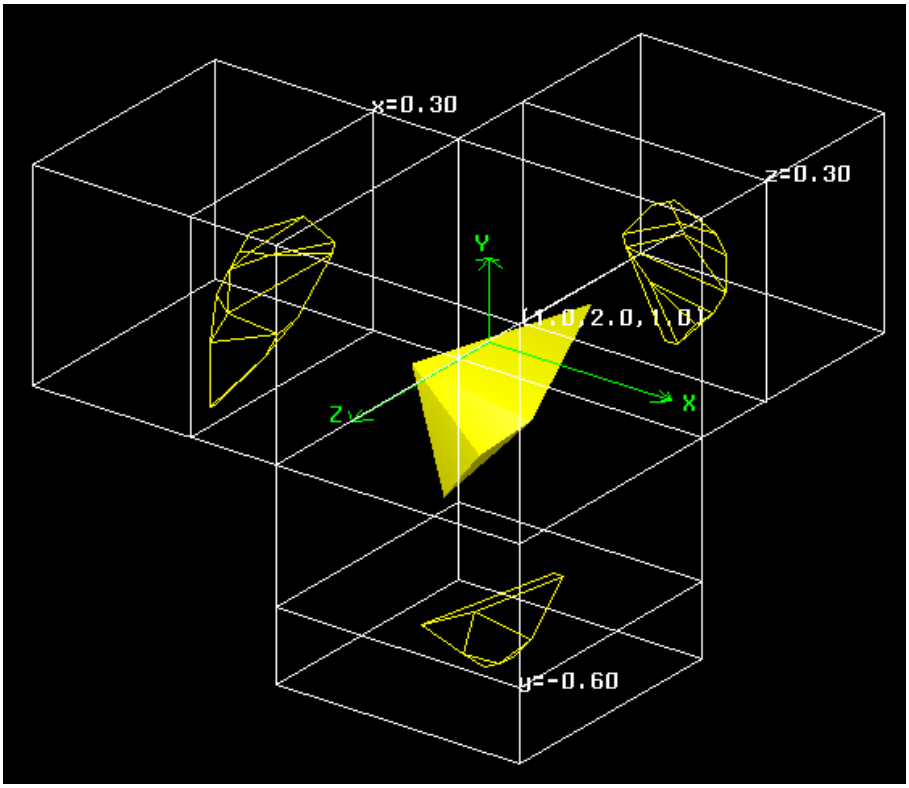


Figure 3.5.1 Delaunay tetrahedrization of 10 data points and a scaled version of the same data points.

We now describe the 3D version of the affine invariant norm which leads (by way of the Dirichlet tessellation) to an affine invariant tetrahedrization. Actually, we can define it so that it is clear what the generalization is for any dimension. Let

$$\|(x,y,z)\|_V^2 = (x,y,z)\left(VV^*\right)^{-1}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

where V is the 3×N matrix of translated data values

$$V = \begin{pmatrix} x_1-\mu_x & x_2-\mu_x & \cdots & x_N-\mu_x \\ y_1-\mu_y & y_2-\mu_y & \cdots & y_N-\mu_y \\ z_1-\mu_z & z_2-\mu_z & \cdots & z_N-\mu_z \end{pmatrix}$$

As with the 2D case, there are some different approaches to modifying an existing tetrahedrization procedures. Probably the simplist is to preprocess the data with the transformation given by the lower triangular matrix, L(V) which results form the Cholesky decomposition of $(VV^*)^{-1}$

$$L(V)\, L(V)^* = (VV^*)^{-1} \quad .$$

Explicitly in the 3D case, we use the transformed data

$$X_i = l_{11}x_i + l_{21}y_i + l_{31}z_i$$

$$Y_i = l_{22}y_i + l_{32}z_i$$

$$Z_i = l_{33}z_i$$

where

$$l_{11} = \sqrt{a_{11}} \ , \qquad l_{21} = \frac{a_{11}}{l_{11}} \ , \qquad l_{31} = \frac{a_{13}}{l_{11}}$$

$$l_{22} = \sqrt{a_{22}-(l_{21})^2} \ , \qquad l_{32} = \frac{a_{33}-l_{21}l_{31}}{l_{22}} \ ,$$

$$l_{33} = \sqrt{a_{33}-(l_{31})^2-(l_{32})^2}$$

$$A = (a_{ij}) = (VV^*)^{-1} = L(V)L(V)^*$$

$$= \frac{1}{\det}\begin{pmatrix} \sum_y^2\sum_z^2 - \sum_{yz}^2 & -\left(\sum_{xy}\sum_z^2 - \sum_{xz}\sum_{yz}\right) & \sum_{xy}\sum_{yz} - \sum_{xz}\sum_y^2 \\ -\left(\sum_{xy}\sum_z^2 - \sum_{xz}\sum_{yz}\right) & \sum_x^2\sum_z^2 - \sum_{xz}^2 & -\left(\sum_{yz}\sum_x^2 - \sum_{xz}\sum_{xy}\right) \\ \sum_{xy}\sum_{yz} - \sum_{xz}\sum_y^2 & -\left(\sum_{yz}\sum_x^2 - \sum_{xz}\sum_{xy}\right) & \sum_x^2\sum_y^2 - \sum_{xy}^2 \end{pmatrix}$$

where

$$\Sigma_x^2 = \frac{\sum_{i=1}^{N}(x_i - \mu_x)^2}{N}, \qquad \mu_x = \frac{\sum_{i=1}^{N} x_i}{N},$$

$$\Sigma_y^2 = \frac{\sum_{i=1}^{N}(y_i - \mu_y)^2}{N}, \qquad \mu_y = \frac{\sum_{i=1}^{N} y_i}{N}$$

$$\Sigma_z^2 = \frac{\sum_{i=1}^{N}(z_i - \mu_z)^2}{N}, \qquad \mu_z = \frac{\sum_{i=1}^{N} z_i}{N}$$

$$\Sigma_{xy} = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{N},$$

$$\Sigma_{yz} = \frac{\sum_{i=1}^{N}(y_i - \mu_y)(z_i - \mu_z)}{N}$$

$$\Sigma_{xz} = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(z_i - \mu_z)}{N},$$

and

$$\det = \Sigma_x^2\,\Sigma_y^2\,\Sigma_z^2 + 2(\Sigma_{xy}\,\Sigma_{yz}\,\Sigma_{xz}) - \Sigma_x^2(\Sigma_{yz})^2 - \Sigma_y^2(\Sigma_{xz})^2\,\Sigma_z^2(\Sigma_{xy})^2 .$$

We conclude this section with some examples illustrating this affine invariant norm and its use in characterizing affine invariant tetrahedrizations. In Figure 3.5.2 there are shown four graphs of 13 data points. The transparent ellipsoids represent all the points that are 0.25, 0.50 and 1.0 units from the center point using the affine invariant norm. The different graphs show the data after it has undergone an affine transformation. The original data is displayed in the upper left. The upper right show the data after it has been rotated by 44 degrees about the z-axis. The lower right is after it has subsequently been scaled in the x-variable by a factor of 1.5. The lower left is after it has been scaled in y by a factor of 0.6. A close examination of these graphs will show that the relative

distances (as measured by the affine invariant norm) between points is unchanged by this transformations. In Figure 3.5.3 an affine invariant tetrahedrization is shown. In comparison the conventional Delaunay tetrahedrization is shown in Figure 3.5.4.

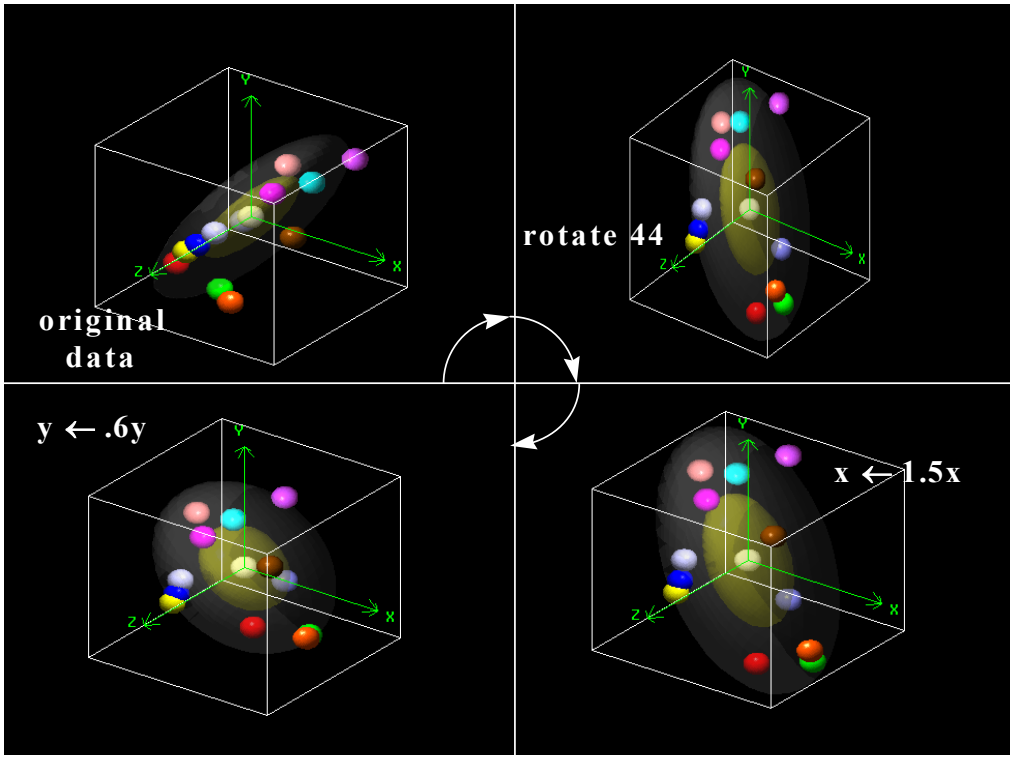


Figure 3.5.2. Examples illustrating the affine invariant norm. The ellipsoids are 0.25, 0.50 and 1.0 units from the center point.

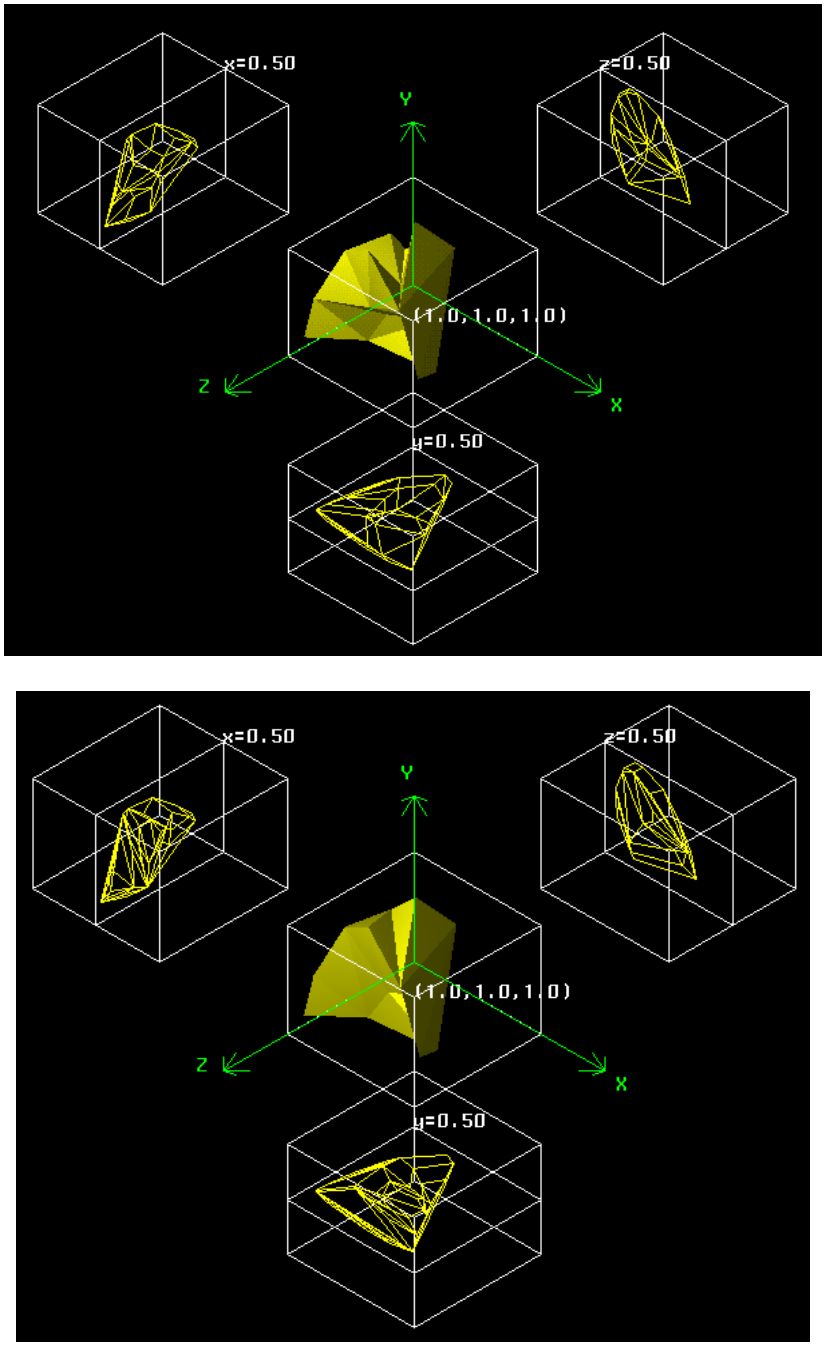


Figure 3.5.3. Examples of affine invariant tetrahedrization.

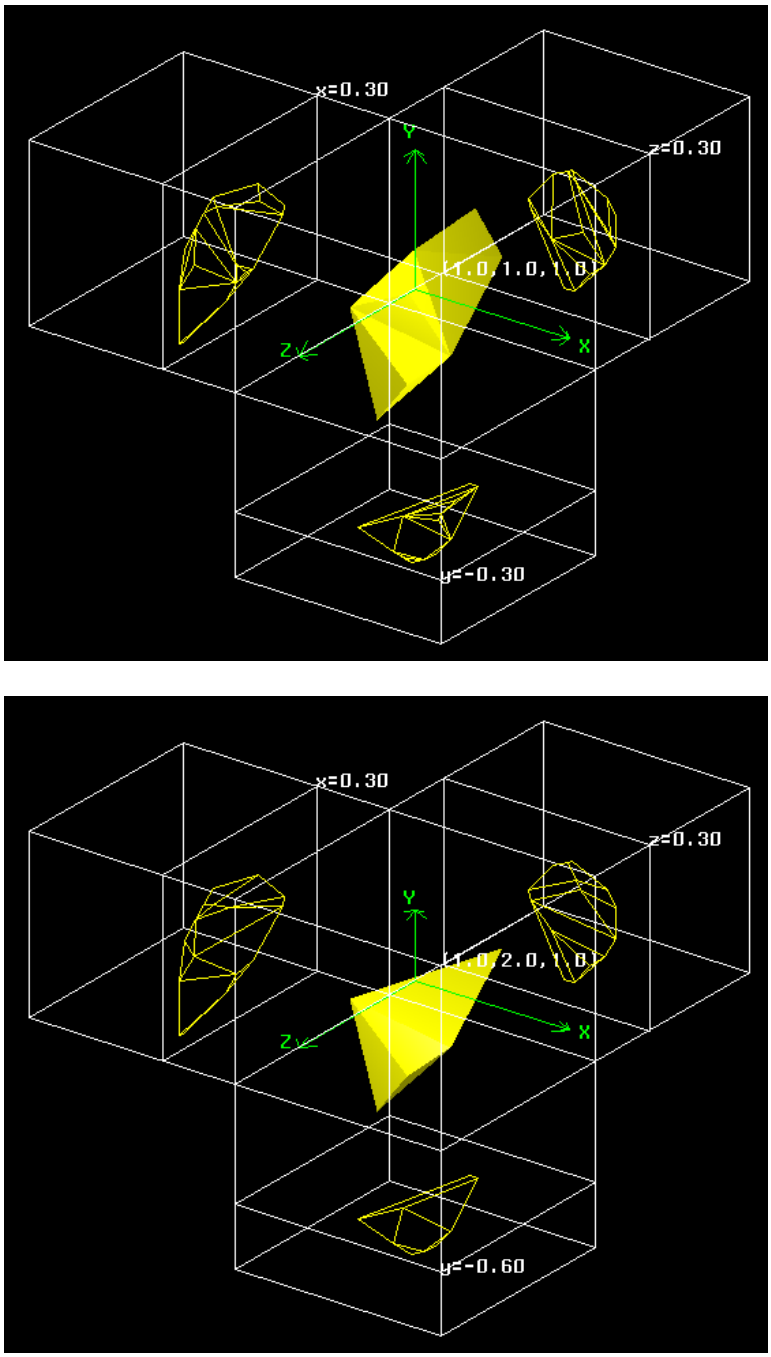


Figure 3.5.4. Delaunay tetrahedrization of the same data as in Figure 3.5.3.

3.6 Interpolation in Tetrahedra

As with the bivariate case covered in Section 2.6, there are two concepts of interest for interpolation in tetrahedra. The first is concerned with the amount of boundary data that is proved or assumed to be available. This can be discrete data provided at a finite number of locations (usually the vertices or midpoints) or transfinite data where boundary data values are assumed to be available at all locations on the boundary. The

second concept relates to the degree of continuity of a piecewise defined interpolant using the local interpolants described here. C^0 interpolants only use boundary position data and lead to overall interpolants which are continuous. C^1 interpolants utilize first order derivative information and lead to global interpolants which have all first order derivative continuous. These two concepts lead to four possibilities which comprise the outline of this section.

	Discrete	Transfinite
C^0	**Section 3.6.1**	**Section 3.6.2**
C^1	**Section 3.6.4**	**Section 3.6.3**

Figure 3.6.1. Outline of Section 3.6.

Section 3.6.1 C^0, Discrete Interpolation in Tetrahedra

Analogous to the bivariate linear interpolant which will match predescribed values at the three vertices of a triangle, there is a unique trivariate linear interpolant which will match data at the four vertices of a tetrahedra, T_{ijkl}. Given $F(V_i)$, $F(V_j)$, $F(V_k)$ and $F(V_l)$ the coefficients of this linear function

$$F(x, y, z) = a + bx + cy + dz$$

which interpolates this data can be found by solving the linear system of equations

$$a + bx_i + cy_i + dz_i = F(V_i)$$

$$a + bx_j + cy_j + dz_j = F(V_j)$$

$$a + bx_k + cy_k + dz_k = F(V_k)$$

$$a + bx_l + cy_l + dz_l = F(V_l)$$

As before, it is also possible to use barycentric coordinates. The barycentric coordinates of a point V= (x, y, z) are defined by the relationships

$$V = b_iV_i + b_jV_j + b_kV_k + b_lV_l$$

$$1 = b_i + b_j + b_k + b_l$$

and the linear interpolant has the form

$$F(x, y, z) = F(V) = b_iF(V_i) + b_jF(V_j) + b_kF(V_k) + b_lF(V_l) \quad (3.6.1)$$

As before, there are several ways of defining or computing barycentric coordinates. The analog of the ratios of areas before is the ratio of volumes of subtetrahedra,

$$b_i = \frac{\mathrm{Vol}(T_{Vjkl})}{\mathrm{Vol}(T_{ijkl})}, \qquad b_j = \frac{\mathrm{Vol}(T_{iVkl})}{\mathrm{Vol}(T_{ijkl})}$$

$$b_k = \frac{\mathrm{Vol}(T_{ijVl})}{\mathrm{Vol}(T_{ijkl})}, \qquad b_l = \frac{\mathrm{Vol}(T_{ijkV})}{\mathrm{Vol}(T_{ijkl})}$$

where T_{Vjkl} is the tetrahedron with vertices V, V_j, V_k, and V_l and similar definitions for the other subtetrahedra. The volume of a tetrahedron, T_{abcd}, with vertices a, b, c and d is

$$\mathrm{Vol}(T_{abcd}) = \frac{1}{6}\,[(d\text{-}a)\bullet((b\text{-}a)\times(c\text{-}a)))]$$

Also determinants can be used,

$$b_i = \frac{\begin{vmatrix} x-x_j & x-x_k & x-x_l \\ y-y_j & y-y_k & y-y_l \\ z-z_j & z-z_k & z-z_l \end{vmatrix}}{\begin{vmatrix} x_i-x_j & x_i-x_k & x_i-x_l \\ y_i-y_j & y_i-y_k & y_i-y_l \\ z_i-z_j & z_i-z_k & z_i-z_l \end{vmatrix}}, \quad b_j = \frac{\begin{vmatrix} x-x_i & x-x_k & x-x_l \\ y-y_i & y-y_k & y-y_l \\ z-z_i & z-z_k & z-z_l \end{vmatrix}}{\begin{vmatrix} x_j-x_i & x_j-x_k & x_j-x_l \\ y_j-y_i & y_j-y_k & y_j-y_l \\ z_j-z_i & z_j-z_k & z_j-z_l \end{vmatrix}},$$

$$b_k = \frac{\begin{vmatrix} x-x_i & x-x_j & x-x_l \\ y-y_i & y-y_j & y-y_l \\ z-z_i & z-z_j & z-z_l \end{vmatrix}}{\begin{vmatrix} x_k-x_i & x_k-x_j & x_k-x_l \\ y_k-y_i & y_k-y_j & y_k-y_l \\ z_k-z_i & z_k-z_j & z_k-z_l \end{vmatrix}}, \quad b_l = \frac{\begin{vmatrix} x-x_i & x-x_j & x-x_k \\ y-y_i & y-y_j & y-y_k \\ z-z_i & z-z_j & z-z_k \end{vmatrix}}{\begin{vmatrix} x_l-x_i & x_l-x_j & x_l-x_k \\ y_l-y_i & y_l-y_j & y_l-y_k \\ z_l-z_i & z_l-z_j & z_l-z_k \end{vmatrix}}.$$

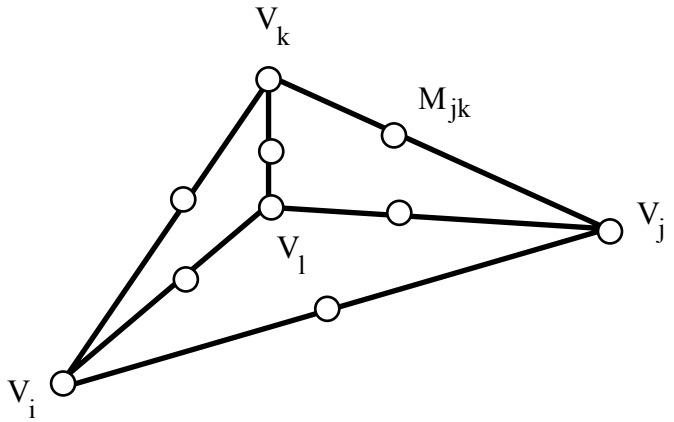


Figure 3.6.2. Data site locations for trivariate quadratic interpolation.

Given the values at the four vertices and the six midpoints of a tetrahedron, there is a unique trivariate quadratic which interpolates this data,

$$\begin{aligned} Q(x, y, z) = \quad & F(V_i)b_i(b_i - b_j - b_k - b_l) + F(V_j)b_j(b_j - b_i - b_k - b_l) \\ + \quad & F(V_k)b_k(b_k - b_i - b_j - b_l) + F(V_l)b_l(b_l - b_i - b_j - b_k) \\ + \quad & F(M_{ik})4b_ib_k + F(M_{jl})4b_jb_l + F(M_{ij})4b_ib_j \\ + \quad & F(M_{jk})4b_jb_k + F(M_{il})4b_ib_l + F(M_{kl})4b_kb_l \end{aligned} \tag{3.6.7}$$

where $M_{ij} = (V_i + V_j)/2$ and the other midpoints are defined similarly.

Section 3.6.2 C^0, Transfinite Interpolation in Tetrahedra

As before in Section 2.6.2, we give a sampling of interpolants. One is a generalization of the side-vertex interpolant and the other is a generalization of the C* interpolant. Both of these bivariate interpolants were discussed previously in Section 2.6.2.

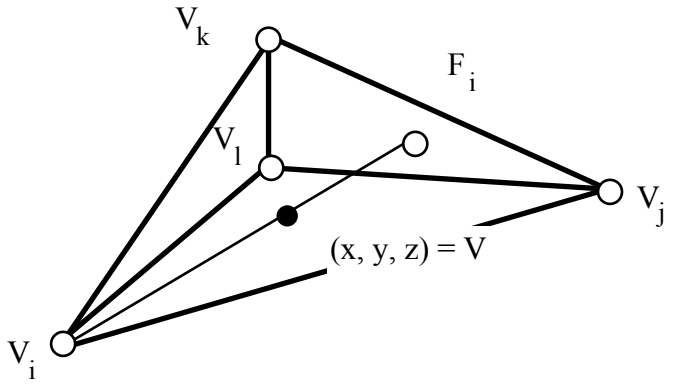


Figure 3.6.3. Notation for the Face-Vertex interpolant.

The C^0, Face-Vertex Interpolant: Analogous to the basic interpolants used to construct the side-vertex interpolant, we have the interpolants which consist of linear interpolation along edges joining a vertex and the opposing face

$$\begin{aligned} A_i[F] &= b_iF(V_i) + (1-b_i)F(F_i) \\ A_j[F] &= b_jF(V_j) + (1-b_j)F(F_j) \\ A_k[F] &= b_kF(V_k) + (1-b_k)F(F_k) \\ A_l[F] &= b_lF(V_l) + (1-b_l)F(F_l) \end{aligned} \tag{3.6.3}$$

where $F_i = \frac{b_jV_j+b_kV_k+b_lV_l}{b_j+b_k+b_l}$, $F_j = \frac{b_iV_i+b_kV_k+b_lV_l}{b_i+b_k+b_l}$, $F_k = \frac{b_iV_i+b_jV_j+b_lV_l}{b_i+b_j+b_l}$ and $F_l = \frac{b_iV_i+b_jV_j+b_kV_k}{b_i+b_j+b_k}$.

Computing the Boolean sum of these four interpolants leads to

$$\begin{aligned} A[F] = \quad & (1-b_i)F(F_i) + (1-b_j)F(F_j) + (1-b_k)F(F_k) + (1-b_l)F(F_l) \\ & - (b_k+b_l)F(S_{kl}) - (b_i+b_l)F(S_{il}) - (b_j+b_l)F(S_{jl}) \\ & - (b_j+b_k)F(S_{jk}) - (b_i+b_k)F(S_{ik}) - (b_i+b_j)F(S_{ij}) \\ & + b_iF(V_i) + b_jF(V_j) + b_kF(V_k) + b_lF(V_l) \end{aligned} \tag{3.6.4}$$

where $S_{mn} = \frac{b_mV_m+b_nV_n}{b_m+b_n}$, mn = kl, il, jl, jk, ik, ij

The C* Interpolant (for a tetrahedron): The analog of the bivariate C* interpolant described in Section 2.6.2 is

$$\begin{aligned} C^*[F] = \quad & b_iF(V_i) + b_jF(V_j) + b_kF(V_k) + b_lF(V_l) \\ & + W_l\{F(Q_l) - (b_i+\tfrac{b_l}{3})F(V_i) - (b_j+\tfrac{b_l}{3})F(V_j) - (b_k+\tfrac{b_l}{3})F(V_k)\} \\ & + W_k\{F(Q_k) - (b_i+\tfrac{b_k}{3})F(V_i) - (b_j+\tfrac{b_k}{3})F(V_j) - (b_l+\tfrac{b_k}{3})F(V_l)\} \\ & + W_j\{F(Q_j) - (b_i+\tfrac{b_j}{3})F(V_i) - (b_k+\tfrac{b_j}{3})F(V_k) - (b_l+\tfrac{b_j}{3})F(V_l)\} \\ & + W_i\{F(Q_i) - (b_j+\tfrac{b_i}{3})F(V_j) - (b_k+\tfrac{b_i}{3})F(V_k) - (b_l+\tfrac{b_i}{3})F(V_l)\} \end{aligned} \tag{3.6.5}$$

where $Q_l = (b_i+\frac{b_l}{3})V_i + (b_j+\frac{b_l}{3})V_j + (b_k+\frac{b_l}{3})V_k$,
$W_l = \frac{27b_ib_jb_k}{(3b_i+b_l)(3b_j+b_l)(3b_k+b_l)}$ and the other Q's and W's are defined in a similar manner.

The NTW Interpolant (for a tetrahedron): The analog of the bivariate NTW interpolant described in Section 2.6.2 is

$$NTW[F] = b_iS_i + b_jS_j + b_kS_k + b_lS_l$$

where

$$
\begin{aligned}
S_i = S_i(b_j, b_k, b_l) = {} & F(b_j V_j + b_k V_k + (1 - b_j - b_k) V_i) \\
& + F(b_k V_k + b_l V_l + (1 - b_k - b_l) V_i) \\
& + F(b_l V_l + b_j V_j + (1 - b_l - b_j) V_i) \\
& - F(b_j V_j + (1 - b_j) V_i) \\
& - F(b_k V_k + (1 - b_k) V_i) \\
& - F(b_l V_l + (1 - b_l) V_i) \\
& + F(V_i)
\end{aligned}
$$

Section 3.6.3 C^1, Transfinite Interpolation in Tetrahedra

The C^1, Face-Vertex Interpolant: It is a straightforward process to extend the C^1, transfinite side-vertex interpolant to a tetrahedral domain, T_{ijkl}. It is called the C^1, face-vertex interpolant and we assume that position and derivative information is available at all locations on the four faces which make up the boundary of the tetrahedron T_{ijkl}. The basic face-vertex operator is defined as

$$
\begin{aligned}
S_i[F](p) = {} & b_i^2(3-2b_i)F(V_i) + b_i^2(b_i-1)F'(V_i) \\
& + (1-b_i)^2(2b_i+1)F(F_i) + b_i(1-b_i)^2F'(F_i)
\end{aligned}
\tag{3.6.6}
$$

where $F'(V_i) = \dfrac{(x-x_i)F_x(V_i)+(y-y_i)F_y(V_i)+(z-z_i)F_z(V_i)}{1-b_i}$ and

$F'(F_i) = \dfrac{(x-x_i)F_x(S_i)+(y-y_i)F_y(S_i)+(z-z_i)F_z(V_i)}{1-b_i}$. The point F_i is the intersection point of the ray from V_i through V and the face opposite V_i and the derivatives are taken in the direction of this same ray. If we form the convex combination

$$
S[F] = \frac{b_j^2 b_k^2 b_l^2 S_i[F] + b_i^2 b_k^2 b_l^2 S_j[F] + b_j^2 b_l^2 b_i^2 S_k[F] + b_j^2 b_k^2 b_i^2 S_l[F]}{b_j^2 b_k^2 b_l^2 + b_i^2 b_k^2 b_l^2 + b_j^2 b_k^2 b_i^2 + b_i^2 b_j^2 b_k^2}
$$

then S[F] will match position and derivative values on the entire boundary of T_{ijkl}.

Section 3.6.4 C^1, Discrete Interpolation in Tetrahedra

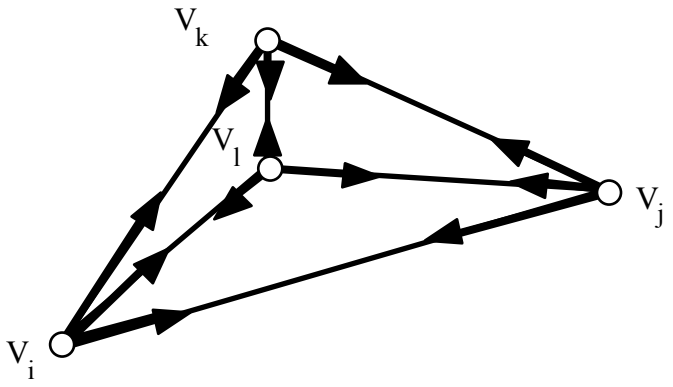


Figure 3.6.4. The data for a 16 parameter, C^1, interpolant over a tetrahedron.

For a C^1, discrete interpolant, we assume that position and first order derivative information is given at all four vertices of the tetrahedron T_{ijkl}. Since there are three (linearly independent) directional derivatives at each vertex, this amounts to a total of sixteen data values. The method for describing an interpolant that will match these sixteen pieces of data and which also has the property that all first order derivatives across a face with common data will be continuous is somewhat different that the previous interpolants we have described so far. Our description (and subsequent implementation) is based upon a two step procedural discretization process. We use the transfinite interpolant of the previous section. In order to apply this transfinite interpolant, we need to define position and derivative values on the entire boundary of T_{ijkl}. First we assume that information is known on all the edges of the tetrahedra and we describe how to extend it to the entire boundary. Secondly, we describe how to provide this transfinite edge data from only the discrete data at the vertices. If we know both position and derivative information on the edges, then we can use any C^1 transfinite planar triangular interpolant to define position values on the interior points of the face triangles. For example, the side-vertex method itself could be used. Specifying position information on a face also implies some information about the derivatives on the interior of a triangle. Namely, all directional derivatives in a direction parallel to the face triangle are determined and so, in order to completely specify all derivatives, we need only provide a definition for the derivative perpendicular to the face. For this we use the C^0 version of the side-vertex interpolant which interpolates position data only and not derivatives, but we apply it to the edge data consisting of derivatives normal to a face.

We now describe the second step of the discretization which is how to compute edge information when only the point and derivative values are known at the four vertices. For position only on an edge, we simply use univariate cubic Hermite interpolation. This will also specify one directional derivative on the edge; namely $\frac{\partial F}{\partial e_{ij}}$ which will vary as a quadratic polynomial. In order to get a C^1 join from one tetrahedron to the next, the other two directional derivatives must vary linearly along this edge. This is accomplished by specifying the gradient, ∇F, by the relationship

$$\nabla F_{ij}(p) = (1-t)\nabla F_i + t\nabla Fj$$

$$+ [\frac{\partial F}{\partial e_{ij}}(p) - ((1-t)\nabla F_i + t\nabla F_i, e_{ij})]e_{ij} \qquad (3.6.8)$$

where $\nabla Fi = (F_x(p_i), F_y(p_i), F_z(p_i))$ and $t = \frac{\|p-p_i\|}{\|p_j-p_i\|}$. This interpolation of the gradient is consistent with the value $\frac{\partial F}{\partial e_{ij}}$ already specified because $(\nabla F_{ij}(p),e_{ij}) = \frac{\partial F}{\partial e_{ij}}$ and it also has the property that for $(n,e_{ij}) = 0$,

$$(\nabla F_{ij}(p), n) = (1-t)(\nabla F_i,n) + t(\nabla F_j,n),$$

and so we have linear interpolation for any derivative in a direction perpendicular to e_{ij}. This completes the definition of the 16-parameter, C^1, tetrahedral interpolant which is based upon the face-vertex interpolant. Examples and more discussion on this interpolant can be found in [187]. The Clough-Tocher interpolant has been generalized to n-dimensional by Worsey and Farin [261]. Other C^1, discrete interpolants for a tetrahedral domain are discussed in [2], [3], and [Worsey and Piper in CAGD, 1988], but each have some problem or drawback. The method of [2] is based upon the side-side, transfinite method of interpolation and apparently it has a problem with the linear independence of the discretized data. The method of [3] requires C^2 data for a C^1 interpolant and the method of [260] has a problem similar to its bivariate precursor [199] and [198]. This problem lies in the constraint that the center of the circumcircle of each triangle must lie interior to the triangular domain.

Acknowledgments

This work was supported by the North Atlantic Treaty Organization under grant RG 0097/88. We wish to thank Herbert Edelsbrunner for the idea of the dual graphs of Section 3.1 and other insightful discussions about tetrahedrizations. We wish to thank Kun Lee for his help in generationg the images of Sections 3.4 and 3.5.

References

1. A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor, A linear time algorithm for computing the Voronoi diagram of a convex polygon, Disc. and Comp. Geometry 4, 1989, pp. 591-604.

2. P. Alfeld, A discrete C^1 interpolant for tetrahedral data, The Rocky Mountain Journal of Mathematics, Vol. 14, No. 1, Winter 1984, pp. 5-16.

3. P. Alfeld, A trivariate Clough-Tocher scheme for tetrahedral data, Computer Aided Geometric Design, Volume 1, Number 2, 1984, pp. 169-181.

4. T. Asano and R. Pinter, Polygon triangulation: efficiency and minimality, J. Algorithms, Vol. 7, 1986, pp. 221-231.

5. D. Avis, and B. K. Bhattacharya, Algorithms for computing d-dimensional Voronoi diagrams and their duals, Advance in Computing Research, Vol. 1, pp. 159-180.

6. D. Avis and K. Fukuda, A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra, Proceedings 7th Annual ACM Symposium Computational Geometry, 1991, pp. 98-104.

7. D. Avis and G. T. Toussaint, An efficient algorithm for decomposing a polygon into star-shaped polygons, Pattern Recogn. Vol. 13, No. 6, 1981, pp. 395-398.

8. F. Aurenhammer, Voronoi diagrams - A survey of a fundamental geometric data structure, ACM Computing Surveys, Vol. 23, 1991, pp. 345-405

9. I. Babuska, and A. Aziz, On the angle condition in the finite element method, SIAM J. Numer. Analysis, Vol. 13, 1976, pp. 214-227.

10. P. I. Baegmann, M. S. Shepard, and J. E. Flaherty, A posterori error estimation for triangular and tetrahedral quadratic elements using interior residuals, Internat. J. Numer. Meth. Eng., Vol. 34, 1992, pp. 979-996.

11. F. Bagemihl, On indecomposable polyhedra, American Mathematical Monthly, September 1948, pp. 411-413.

12. T. J. Baker, Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation, Eng. with Computers, Vol. 5, 1989, pp. 161-175.

13. B. S. Baker, E. Grosse, and C. S. Rafferty, Nonobtuse triangulation of polygons, Disc. and Comp. Geom., Vol. 3, 1988, pp. 147-168.

14. G. Baszenski and L. L. Schumaker, Use of simulated annealing to construct triangular facet surfaces, in: Curves and Surfaces, P.-J. Laurent, A. Le Mehaute and L. L. Schumaker (eds.), Academic Press, Boston, 1991, pp. 27-32.

15. M. Bern and D. Eppstein, Polynomial-size nonobtuse triangulation of polygons, Proc. 7th ACM Symp. Comp. Geometry, 1991, pp. 342-350.

16. M. Bern, and D. Eppstein, Mesh generation and optimal triangulation. In F. K. Hwang and D.-Z. Du, Editors, *Computing in Euclidean Geometry*, pp. 23-90. World Scientific, Singapore, 1992.

17. M. Bern, D. Dobkin, and D. Eppstein, Triangulating polygons with large angles, Proc. 8th ACM Sym. Comp. Geometry, 1992.

18. M. Bern, D. Eppstein, and F. Yao, The expected extremes in a Delaunay triangulation, Int. J. Comp. Geometry and Applications, Vol. 1, 1991, pp. 79-92.

19. J. Bloomenthal, Polygonization of implicit surfaces, CAGD, Vol. 5, 1988, pp. 341-355.

20. C. Borgers, Generalized Delaunay triangulations of nonconvex domains, Computers & Mathematics with Applications, Vol. 20, No. 7, 1990, pp. 45-49.

21. A. Bowyer, Computing Dirichlet tessellations, Computer J., Vol. 24, 1981, pp. 162-166.

22. C. Bradford, D. Barber, D. P. Dobkin, and H. Huhdanpaa, The quickhull algorithm for convex hull, Tech. Rep. GCG53-93, Geometry Center, University of Minnesota, July 1993.

23. J. Bramble and M. Zlamal, Triangular elements in the finite element method, Math. Comp., Vol. 24, 1970, pp. 809-820.

24. K. E. Brassel and D. Reif, A procedure to generate Thiessen polygons, Geograph. Anal., Vol. 11, 1979, pp. 289-303.

25. W. Brostow, J. P. Dussault, and B. L. Fox, Construction of Voronoi polyhedra, J. Comp. Physics, Vol. 29, 1978, pp. 81-92.

26. J. L. Brown, Vertex based data dependent triangulations, Computer Aided Geometric Design, Vol. 8, 1991, pp. 239-251.

27. K. Q. Brown, Voronoi diagrams from convex hulls, Inform. Process. Lett., Vol. 9, 1979, pp. 223-228.

28. J. C.. Cavendish, Automatic triangulation of arbitrary planar domains for the finite element method, Int. J. for Numer. Methods in Engr., Vol. 8, 1974, pp. 679-696.

29. J. C. Cavendish, D. A. Field, and W. H. Frey, An approach to automatic three-dimensional finite element mesh generation, Int. J. Numer. Meth. Eng., Vol. 21, 1985, pp. 329-347.

30. M. S. Chang, N.-F. Huang, and C. Y. Tang, Optimal algorithm for constructing oriented Voronoi diagrams and geographic neighborhood graphs, Information Processing Letters, Vol. 35, No. 5, August 1990, pp. 255-260.

31. R. C. Chang and R. C. T. Lee, On the average length of Delaunay triangulations, BIT, Vol. 24, 1984, pp. 269-273.

32. S. Chattopodhyay and P. P. Das, Counting thin and bushy triangulations, Pattern Recognition Letters, Vol. 12, No. 3, 1991, pp. 139-144.

33. B. Chazelle, Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm, SIAM J. Comput., Vol. 13, 1984, pp. 488-507.

34. B. Chazelle, Triangulating a simple polygon in linear time, Disc. and Comp. Geometry, Vol. 6, 1991, pp. 485-524.

35. B. Chazelle and D. Dobkin, Decomposing a polygon into its convex parts, in: ACM Proceedings of the 11th Symposium on Theory of Computing, 1979, pp. 38-48.

36. B. Chazelle, H. Edelsbrunner, L. J. Guibas, J. E. Hershberger, R. Reidel, and M. Sharir, Selecting multiply covered points and reducing the size of Delaunay triangulations, In Proc. 6th ACM Symp. Comp. Geometry, 1990, pp. 116-127.

37. B. Chazell and J. Incerpi, Triangulating a polygon by divide and conquer, Proceedings of the 21st Allerton Conference on Communications, Control and Computing, 1983, pp. 447-456.

38. B. Chazelle and J. Incerpi, Triangulation and shape complexity, ACM Trans on Graphics, Vol. 3, 1984, pp. 135-152.

39. B. Chazelle and L. Palios, Triangulating a nonconvex polytope, Disc. and Comp. Geometry, Vol. 5, 1990, pp. 505-526.

40. L. P. Chew, Constrained Delaunay triangulations, Algorithmica, Vol. 4, 1989, pp. 97-108.

41. B. K. Choi, H. Y. Shin, Y. I. Yoon, and J. W. Lee, Triangulation of scattered data in 3D space, CAD, Vol. 20, 1988, pp. 239-248.

42. K. L. Clarkson, R. E. Tarjan, and C. J. Van Wyk, A fast Las Vegas algorithm for triangulating a simple polygon, Discrete and Computational Geometry, Vol. 4, 1989, pp. 423-432.

43. A. K. Cline and R. J. Renka, A constrained two-dimensional triangulation and the solution of closest node problems in the presence of barriers, SIAM Journal on Numerical Analysis, Vol. 27, No. 5, 1990, pp. 1305-1321.

44. A. K. Cline and R. L. Renka, A storage-efficient method for construction of a Thiessen triangulation, Rocky Mountain Journal of Mathematics, Vol. 14, No. 1, Winter 1984, pp. 119-140.

45. H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford, and B. C. Teeter, Two algorithms for the reconstruction of surfaces from tomograms, Medical Physics, June 1988.

46. Y. Correc and E. Chapuis, Fast computation of Delaunay triangulations, Advances in Engineering Software, Vol. 9, No. 2, 1987, pp. 77-83.

47. H. S. M. Coxeter, Discrete groups generated by reflections, Ann. Math. Vol. 35, 1934, pp. 588-621.

48. J. R. Davy and P. M. Dew, A note on improving the performance of Delaunay triangulation. in: *New Advances in Computer Graphics: Proceedings of Computer Graphics International 89*, R. A. Earnshaw and B. Wyvill (eds.), Springer, Tokyo, 1989, pp. 209-226.

49. A. M. Day, The implementation of an algorithm to find the convex hull of a set of three-dimensional points, ACM Transactions on Graphics, Vol. 9, No. 1, January 1990, pp. 105-132.

50. L. De Floriani, A pyramidal data structure for triangle-based surface representation, IEEE Computer Graphics and Applications, Vol. 9, March 1989, pp. 67-78.

51. L. De Floriani, B. Falcidieno, and C. Pienovi, Delaunay-based representation of surfaces defined over arbitrarily shaped domains, Computer Vision, Graphics, and Image Processing, Vol. 32, 1985, pp. 127-140.

52. L. De Floriani and E. Puppo, An on-line algorithms for constrained Delaunay triangulation, CVGIP: Graphical Models and Image Processing, Vol. 54, No. 3, 1992, pp. 290-300.

53. L. De Floriani, B. Falcidieno, G. Nagy, and C. Pienovi, On sorting triangles in a Delaunay tessellation, Algorithmica, Vol. 6, 1991, pp. 522-532.

54. B. Delaunay, Sur la sphère vide, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7, (Bull. Acad. Sci. U.S.S.R.(VII), Classe Sci. Mat. Nat), 1934, pp. 793-800.

55. P. A. Devijver and M. Dekesel, Insert and delete algorithms for maintaining dynamic Delaunay triangulations, Pattern Recogn. Lett., Vol. 1, 1982, pp. 73-77.

56. T. Dey, Triangulation and CSG representation of polyhedra with arbitrary genus. In Proc. 7th ACM Symp. Comp. Geometry, 1991, pp. 793-800.

57. T. Dey, K. Sugihara and C. L. Bajaj, Delaunay triangulations in three dimensions with finite precision arithmetic, Computer Aided Geometric Design, Volume 9, Number 6, pp. 457-470.

58. M. B. Dillencourt, Realizability of Delaunay triangulations, Information Processing Letters, Vol. 33, No. 6, 1990, pp. 283-287.

59. G. L. Dirichlet, Über die reduktion der positiven quadratischen formen mit drei unbestimmten ganzen zahlen, J. Reine u. Angew. Math., Vol. 40, 1850, pp. 209-227.

60. H. Djidjev and A. Lingas, On computing the Voronoi diagram for restricted planar figures, Proc. 2nd Workshop Algorithms Data Struct. Volume 519 of Lecture Notes in Computer Science, Springer, 1991, pp. 54-64.

61. D. P. Dobkin, Computational geometry and computer graphics, Proc. IEEE, Vol. 80, No. 9, September 1992, pp. 1400-1411.

62. D. P. Dobkin and M. J. Laszlo, Primitives for the manipulation of three-dimensional subdivisions, Algorithmica, Vol. 4, 1989, pp. 3-32.

63. D. Dobkin, S. Friedman, and K. Supowit, Delaunay graphs are almost as good as complete graphs, Disc. and Comp. Geometry, Vol. 5, 1990, pp. 389-423.

64. D. Dobkin, S. Levy, W. Thurston, and A. Wilks, Contour tracing by piecewise linear approximations, ACM Trans. on Graphics, Vol. 9, 1990, 389-423.

65. R. A. Dwyer, A faster divide and conquer algorithm for construction Delaunay triangulation, Algorithmica, Vol. 2, 1987, pp. 137-151.

66. N. Dyn and I. Goren, Transforming triangulations in polygon domains, Computer Aided Geometric Design, Volume 10, Number 6, December 1993, pp. 531-536.

67. N. Dyn, D. Levin, and S. Rippa, Data dependent triangulations for piecewise linear interpolation, IMA Journal of Numerical Analysis, Vol. 10, 1990, pp. 137-154.

68. H. Edelsbrunner, Algorithms in Combinatorial Geometry, Springer Verlag, 1987.

69. H. Edelsbrunner, An acyclicity theorem for cell complexes in d dimensions, Combinotorica, Vol. 18, 1990, pp. 251-260. Also: Proceedings of the 5th Annual ACM Symposium on Computation Geometry, 1989, pp. 145-151.

70. H. Edelsbrunner and E. P. Mücke, Simulation of simplicity, a technique to cope with the degenerate cases in geometric computations, ACM Trans. Graphics, Vol. 9, 1990, pp. 66-104.

71. H. Edelsbrunner and E. P. Mücke, Three dimensional alpha shapes, ACM Transactions on Graphics, Vol. 13, 1994, pp. 43-72.

72. H. Edelsbrunner and N. R. Shah, Incremental topological flipping works for regular triangulations, in: *Proceedings of the 8th Annual ACM Symposium on Computational Geometry*, pages 43-52, June 1992.

73. H. Edelsbrunner and T. S. Tan, A quadratic time algorithm for the minmax length triangulation, in: *Proc. 32nd IEEE Symp. foundations of Comp. Science*, 1991, pp. 414-423.

74. H. Edelsbrunner and T. S. Tan, A cubic bound for conform Delaunay triangulations, In Proc. 8th Symp Comp. Geometry, 1992.

75. H Edelsbrunner, T. S. Tan, and R. Waupotitsch, A polynomial time algorithm for the minmax angle triangulation, In Proc. 5th Symp Comp. Geometry, 1990.

76. H. Edelsbrunner, T. S. Tan, and R. Waupotitsch, $O(N^2 \log N)$ time algorithm for the minmax angle triangulation, SIAM Journal on Scientific and Statistical Computing, Vol. 13, No. 4, July 1992, pp. 994-1008.

77. H. Edelsbrunner, F. P. Preparata, and D. B. West, Tetrahedrizing point sets in three dimensions, J. Symbolic Comp., Vol. 10, 1990, pp. 335-347.

78. M. Elbaz and J.-C. Spehner, Construction of Voronoi diagrams in the plane by using maps, Theoretical Computer Science, Vol. 77, No. 3, 1990, pp. 331-343.

79. H. ElGindy and G. T. Toussaint, On geodesic properties of polygons relevant to linear time triangulation, Visual Computer, Vol. 5, No. 1, 1989, pp. 68-74.

80. D. Eppstein, The farthest point Delaunay triangulation minimizes angles, Comput. Geom. Theory Appl., Vol. 1, 1992, pp. 143-148.

81. D. Eppstein, Approximating the minimum weight triangulation, In Proc. 3rd ACM-SIAM Symp. Disc. Algorithms, 1992.

82. G. Erlebacher and P. R. Eiseman, Adaptive triangular mesh generation, AIAA Journal, Vol. 25, 1987, pp. 1356-1364.

83. T. P. Fang and L. A. Piegl, Delaunay triangulation using a uniform grid, IEEE Computer Graphics and Application, Vol. 13, No. 3, pp. 36-47, May 1993.

84. G. Farin, A modified Clough-Tocher Interpolant, Computer Aided Geometric Design, Volume 2, Numbers 1-3, pp. 19-27.

85. G. Fekete, Rendering and managing spherical data with sphere quadtrees, Proceedings of Visualization '90, IEEE Computer Society Press, 1990, pp. 176-186.

86. D. Field, Implementing Watson's algorithm in three dimension. In Pro. 2nd ACM Symp. Comp. Geometry, 1986, pp. 246-259.

87. D. Field, A generic Delaunay triangulation algorithm for finite element meshing, Adv. Eng. Software, Vol. 13, 1991, pp. 263-272.

88. D. Field, Laplacian smoothing and Delaunay triangulations, Comm. in Applied Numer. Analysis, Vol. 4, 1988, pp. 709-712.

89. D. Field and W. D. Smith, Graded tetrahedral finite element meshes, Int. J. Numer. Meth. Eng. Vol. 31, pp. 1991, pp. 413-425.

90. R. Forrest, Computational Geometry, Proc. Royal Society London, Vol. 321, Series 4, 1971, pp. 187-195.

91. S. Fortune, Numerical stability of algorithms for 2-d Delaunay triangulations and Voronoi diagrams, Proc. 8th Annual. ACM Symposium. Comput. Geom., 1992, pp. 83-92.

92. S. Fortune, Voronoi diagrams and Delaunay triangulations. In F. K. Hwang and D.-Z. Du (eds.), Computing in Euclidean Geometry, pp. 193-233. World Scientific, Singapore, 1992.

93. S. Fortune, Sweepline algorithm for Voronoi diagrams, Algorithmica, Vol. 2, No. 2, 1987, pp. 153-174.

94. A. Fournier and D. Y. Montuno, Triangulating simple polygons and equivalent problems, ACM Transaction on Graphics, Vol. 3, No. 2, April 1984, pp. 153-174.

95. R. J. Fowler and J. J. Little, Automatic extraction of irregular network digital terrain models, Computer Graphics, Vol. 13, No. 2, August 1979, pp. 199-207.

96. R. Franke, Scattered data interpolation: Tests of some methods, Math. Comp., Vol. 38, 1982, pp. 181-200.

97. R. Franke and G. Nielson, Surface construction based upon triangulations, in Surfaces in Computer Aided Geometric Design, Springer, 1983, pp. 163-179.

98. R. Franke and G. Nielson, Scattered Data Interpolation and Applications: A Tutorial and Survey, in: *Geometric Modelling: Methods and Their Application*, H. Hagen and D. Roller (eds.), Springer, 1990.

99. H. Freudenthal, Simplizialzerlegungen von beschränkter Flachheit, Ann. Math. Vol. 43, 1942, pp. 580-582.

100. W. H. Frey and D. A. Field, Mesh relaxation: a new technique for improving meshes, Int. J. Numer. Neth. Eng. Vol. 31, 1991, pp. 1121-1133.

101. M. R. Garey, D. S. Johnson, F. P. Preparata, and R. E. Tarjan, Triangulating a simple polygon, Inform. Process. Lett., Vol. 7, 1978, pp. 175-179.

102. P. L. George and F. Hermeline, Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra, International Journal for Numerical Methods in Engineering, Vol. 33, No. 5, April 1992, pp. 975-995.

103. J. Gleue, Triangulierung und Interpolation von im R^2 unregelmäßig verteilten Daten, HMI B 357, 1981.

104. M. T. Goodrich, Efficient piecewise-linear function approximation using the uniform metric, Proc. 10th Annu. ACM Sympos. Comput. Geom., 1994, pp. 322-331.

105. S. Goldman, A space efficient greedy triangulation algorithm, Information Processing Letters, Vol. 31, No. 4, 1989, pp. 191-196.

106. T. Gonzalez and M. Razzazi, Properties and algorithms for constrained Delaunay triangulations, Proc. 3rd Canad. Conf. Comput. Geom., 1991, pp. 114-117.

107. P. J. Green and R. Sibson, Computing Dirichlet tessellations in the plane, The Computer Journal, Vol. 21, 1978, pp. 168-173.

108. P. J. Green and B. W. Silverman, Constructing the convex hull of a set of points in the plane, The Computer Journal, Vol. 22, No. 3, 1979, pp. 262.

109. J. A. Gregory, Error bounds for linear interpolation on triangles, in: *The Mathematics of Finite Elements and Application II*, J. R. Whiteman, ed., Academic Press, London, 1975, pp. 163-170.

110. J. A. Gregory, A blending function interpolant for triangles, in D. G. Handscomb. ed., Multivariate Approximation, Academic Press, London.

111. J. A. Gregory, Interpolation to boundary data on the simplex, Computer Aided Geometric Design, Volume 2, Numbers 1-3, pp. 43-52.

112. J. A. Gregory, Error bounds for linear interpolation on triangles, in: *The Mathematics of Finite Elements and Applications II*, J. Whiteman (ed.), Academic Press, London, 1975, pp. 163-170.

113. L. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, ACM Trans. Graphics, Vol. 4, 1985, pp. 74-123.

114. L. J. Guibas, D. E. Knuth, and M. Sharir, Randomized incremental construction of Delaunay and Voronoi diagrams, in: *Automata, Languages and Programming*, LNCS N.443, pages 414-431, Springer-Verlag, 1990.

115. A. J. Hansen and P. L. Levin, On conforming Delaunay mesh generation, Adv. Engineering Software, Vol. 14, No. 2, 1992, pp. 129-135.

116. D. Hansford, The neutral case for the min-max triangulation, CAGD, Vol. 7, 1990, pp. 431-438.

117. F. Hermeline, Triangulation automatique d'un polyedre in dimension n, RAIRO Anal. Numer., Vol. 76, 1982, pp. 211-242.

118. C. Hazelwood, Approximating constrained tetrahedrizations, Computer Aided Geometric Design, Volume 10, Number 1, pp. 67-87.

119. S. Hertel and K. Mehlhorn, Fast triangulation of simple polygons, 4th Conf. Foundations of Computation Theory, Springer LNCS 158, 1983, pp. 207-218.

120. H. Jin and R. I. Tannel, Generation of unstructured tetrahedral meshes by advancing front technique, Internat. J. Numer. Meth. Eng. Vol. 36, 1993, pp. 1805-1823.

121. B. Joe, Three-dimensional triangulations from local transformations, SIAM Journal Sci. Stat. Comput., Volume 10, pp. 718-741, 1989.

122. B. Joe, Construction of three dimensional Delaunay triangulations using local transformations, Computer Aided Geometric Design, Volume 8, Number 2, pp. 123-142, 1991

123. B. Joe and C. A. Wang, Duality of constrained Voronoi diagrams and Delaunay triangulations, Algorithmica, Vol. 9, No. 2, 1993, pp. 149-155.

124. D.-M. Jung, An optimal algorithm for constrained Delaunay triangulation, Proceedings Twenty-Sixth Annual Allerton Conference on Communication, Control and Computing, Urbana, Il, 1988, pp. 85-86.

125. Y. H. Jung and K. Lee, Tetrahedron-based octree encoding for automatic mesh generation, Computer Aided Design, Vol. 25, 1993, pp. 141-153.

126. T. C. Kao and D. M. Mount, An algorithm for computing compacted Voronoi diagrams defined by convex distance functions, Proc. 3rd Canad. Conf. Comput. Geom., 1991, pp. 104-109.

127. T. C. Kao and D. M. Mount, Incremental construction and dynamic maintenance of constrained Delaunay triangulations, Proc. 4th Canad. Conf. Comput. Geom., 1992, pp. 170-175.

128. M. D. Karasick, D. Lieber, and L. R. Nackman, Efficient Delaunay triangulation using rational arithmetic, ACM Transactions on Graphics, Vol. 10, No. 1, January 1991, pp. 71-91.

129. J. Katajainen and M. Koppinen, Constructing Delaunay triangulations by merging buckets in quadtree order, Annales Societatis Mathematicae Polonae, Series IV, Fundamenta Informaticae, Vol. 11, No. 3, 1988, pp. 275-288.

130. D. G. Kirkpatrick, A note on Delaunay and optimal triangulations, Inform. Process. Lett., Vol. 10, 1990, pp. 127-128.

131. D. G. Kirkpatrick, M. M. Klawe, and R. E. Tarjan, Polygon triangulation in O(n log log n) time with simple data structures, Proc. 6th Annual ACM Symposium. Comput. Geom. 1990, pp. 34-43.

132 V. Klee, On the complexity of d-dimensional Voronoi diagrams, Arch. Math., Vol. 34, 1980, pp. 75-80.

133. R. Klein, Concrete and abstract Voronoi Diagrams, Volume 400 of Lecture Notes in Computer Science, Springer, 1989.

134. R. Klein and A. Lingas, A note on generalizations of Chew's algorithm for the Voronoi diagram of a simple polygon, Proc. 9th Annu. ACM Sympos. Comput. Geom., 1993, pp. 124-132.

135. G. T. Klincsek, Minimal triangulations of polygonal domains, Ann. Disc. Math., Vol. 9, 1980, pp. 121-123.

136. D. Knuth, *The Art of Computer Programming, Volume 1; Fundamental Algorithms*, Addison Wesley, Reading MA, 1973.

137. H. W. Kuhn, Simplicial approximation of fixed points, Proc. Nat. Acad. Sci. USA, Vol. 61, 1968, pp. 1238-1242.

138. C. Lawson, Transforming triangulations, Discrete Mathematics, Vol. 3, 1972, pp. 365-372.

139. C. Lawson, Software for C^1 surface interpolation, in Mathematical Software III, J. R. Rice (ed.), Academic Press, New York, 1977, pp. 161-194.

140. C. Lawson, Properties of n-dimensional triangulations, Computer Aided Geometric Design, Volume 3, Number 4, pp. 231-246.

141. C. Lawson, C^1 surface interpolation for scattered data on a sphere, Rocky Mountain Journal of Mathematics, Vol. 14, No. 1, Winter 1984, pp. 177-202.

142. C. Lee, Regular triangulations of convex polytopes, in: Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift, edited by Gritzmann and B. Strumfels, Amer. Math. Soc., Providence, RI, 1991, pp. 443-456.

143. D. T. Lee, Two dimensional Voronoi diagram in the L_p-metric, J. ACM , Vol. 27, 1980, pp. 604-618.

144. D. T. Lee and A. Lin, Generalized Delaunay triangulation for planar graphs, Disc. and Comp. Geometry, Vol. 1, 1986, pp. 201-217.

145. D. T. Lee and C. K. Wong, Voronoi diagrams in L_1 (L_∞) metrics with 2-dimensional storage applications, SIAM J. Comput., Vol. 9, 1980, pp. 200-211.

146. D. T. Lee, and B. J. Schacter, Two algorithms for constructing a Delaunay triangulation, Int. J. of Computer and Information Science, Vol. 9, No. 3, 1980, pp. 219-242.

147. J. Lee, Comparison for existing methods for building triangular irregular network models of terrain from grid digital elevation models, Int. J. of Geographical Information Systems, Vol. 5, No. 2, July-September 1991, pp. 267-285.

148. K. Lee, Data dependent tetrahedrizations, Ph. D. Thesis, Arizona State University, 1995.

149. N. J. Lennes, Theorems on the simple finite polygon and polyhedron, American Journal of Mathematics, Vol. 33, 1911, pp. 37-62.

150. C. Levcopoulos and A. Lingas, On approximation behavior of the greedy triangulation for convex polygons, Algorithmica, Vol. 2, 1987, pp. 175-193.

151. B. A. Lewis and J. S. Robinson, Triangulation of planar regions with applications, Computer J., Vol. 21, 1978, pp. 324-332.

152. A. Lingas, Advances in minimum weight triangulation, Ph. D. Thesis, Linköping Univ., 1983.

153. A. Lingas, Voronoi diagrams with barriers and the shortest diagonal problem, Inform. Process. Lett., Vol. 32, 1989, pp. 191-198.

154. D. Lischinski, Incremental Delaunay triangulation, In Graphic Gems IV, Paul S. Heckbert (editor), Academic Press, 1994, pp. 47-59.

155. E. L. Lloyd, On triangulations of a set of points in the plane, In Proc. 18th IEEE Symp. Found. Comp. Sci., 1977, pp. 228-240.

156. S. Lo, Delaunay triangulations of nonconvex planar domains, Int. J. Numer. Meth. Eng., Vol. 28, 1989, pp. 2695-2707.

157. S. Lo, Volume discretizations into tetrahedra. I. verification and orientation of boundary surfaces, Computers and Structures, Vol. 39, 1991, pp. 493-500.

158. R. Loehner and P. Parikh, Generation of three-dimensional unstructured grids by the advancing front method, Internat. J. Numer. Meht. Fluids, Vol. 8, 1988, pp. 1135-1149.

159. M. K. Loze and R. Saunders, Two simple algorithms for constructing a two-dimensional constrained Delaunay triangulation, Applied Numerical Mathematics, Vol. 11, 1993, pp. 403-418.

160. W. Lorensen and Cline H.E., Marching cubes: A high-resolution 3D surface construction algorithm, SIGGRAPH 87 Conference Proceedings, *Computer Graphics*, Vol. 21, No. 4, July 1987, pp. 163-169.

161. G. Macedonio and M. T. Pareschi, An algorithm for the triangulation of arbitrarily distributed points: Applications to volume estimate and terrain fitting, Computers & Geosciences, Vol. 17, No. 7, 1991, pp. 859-874.

162. G. K. Manacher, and A. L. Zobrist, Neither the greedy nor the Delaunay triangulation approximates the optimum, Inform. Process. Lett., Vol. 9, 19790, pp. 31-34.

163. L. Mansfield, Interpolation to boundary data in tetrahedra with applications to compatible finite elements, J. Mant. Anal. Appl., Volume 56, pp. 137-164.

164. G. Marton, Acceleration of ray tracing via Voronoi diagrams, in: Graphic Gems V, Alan Paeth, editor, Academic Press, 1995, pp. 268-284

165. A. Maus, Delaunay triangulation and the convex hull of n points in expected linear time. BIT, Vol. 24, 1984, pp. 151-163.

166. N. Max, Sorting for polyhedron compositing, in: *Focus on Scientific Visualization*, H. Hagen, H. Müller, G. M. Nielson (eds.), Springer, 1993, pp. 259-268.

167. N. Max, P. Hanrahan, and R. Crawfis, Area and volume coherence for efficient visualization of 3D scalar functions, Computer Graphics, Vol. 24, November 1990, pp. 27-33.

168. A. Mirante and N. Weingarten, The radial sweep algorithm for constructing triangulated irregular networks, IEEE Computer Graphics and Applications, May 1982, pp. 11-21.

169. G. H. Meisters, Polygons have ears, Amer. Math. Monthly, Vol. 82, 1975, pp. 648-651.

170. D. Moore, Subdividing simplices, in *Graphics Gems III*, D. Kirk (ed.), Academic Press, 1992, pp. 244-249.

171. D. Moore, Understanding simploids, in *Graphics Gems III*, D. Kirk (ed.), Academic Press, 1992, pp. 250-255.

172. J.-M. Moreau and P. Volino, Constrained Delaunay triangulation revisited, Proc. 5th Canad. Conf. Comput. Geom., 1993, pp. 340-345.

173. D. E. Muller and F. P. Preparata, Finding the intersection of two convex polyhedra, Theoretical Computer Science Vol. 7, 1978, pp. 217-236.

174. E. J. Nadler, Piecewise linear approximation on triangulations of a planar region, Ph.D. Thesis, 1985, Division of Applied Mathematics, Brown University.

175. A. Narkhede and D. Manocha, Fast polygon triangulation based on Seidel's algorithm, in: Graphic Gems V, Academic Press, 1995, pp. 394-397.

176. J. M. Nelson, A triangulation algorithm for arbitrary planar domains, Appl. Math. Modelling, Vol. 2, 1978, pp. 151-159.

177. G. M. Nielson, The side-vertex method for interpolation in triangles, Journal of Approx. Theory, Vol. 25,1979, pp. 318-336.

178. G. M. Nielson, Minimum norm interpolation in triangles, SIAM Journal Numer. Analysis, Vol. 17,1980, pp. 46-62.

179. G. M. Nielson, A method for interpolating scattered data based upon a minimum norm network, Mathematics of Computation, Vol. 40, 1983, pp. 253-271.

180. G. M. Nielson, An example with a local minimum for the MinMax ordering of triangulations, Arizona State University Computer Science Technical Report TR-87-014, 1987.

181. G. M. Nielson, Coordinate free scattered data interpolation, in: *Topics in Multivariate Approximation*, C. Chui, F. Utreras, L. Schumaker (eds.), Academic Press, NY, 1987, pp. 175-184.

182 G. M. Nielson, A characterization of an affine invariant triangulation, in: *Geometric Modelling*, Computing Supplementum 8, G. Farin, H. Hagen, H. Noltemeier, W. Knoedel (eds), Springer, 1993, pp. 191-210.

183. G. M. Nielson, How many ways can a cube be subdivided into tetrahedra?, Arizona State University Computer Science Department Technical Report TR-95-13, 1995.

184. G. M. Nielson and T. Foley, A Survey of Applications of an Affine Invariant Metric, in: *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker (eds.), Academic Press, New York, 1989, 445-467.

185. G. M. Nielson and R. Ramaraj, Interpolation over a sphere, Computer Aided Geometric Design, Vol. 4, 1987, pp. 41-57.

186. G. M. Nielson and B. Hamann, The asymptotic decider: Resolving the ambiguity in marching cubes, in: *Proceedings of Visualization '91*, IEEE Computer Society Press, Los Alamitos, California, 1990, pp. 83-91.

187. G. M. Nielson and K. Opitz, The face-vertex method for interpolating in tetrahedra, in: *Workshop on Computational Geometry*, A. Conte, V. Demichelis, F. Fontanella and I. Galligani (eds.), World Scientific, 1993, pp. 231-244.

188 G. M. Nielson and J. Tvedt, Comparing methods of interpolation for scattered volumetric data, in: *State of the Art in Computer Graphics - Aspects of Visualization,* D. Rogers and R. A. Earnshaw eds., Springer-Verlag, 1994, pp. 67-86.

189. G. M. Nielson, D. H. Thomas, and J. A. Wixom, Interpolation in triangles, Bull. Austral. Math. Soc. Vol. 20, 1979, pp. 115-130.

190. T. Ohya, M. Iri, and K. Murota, Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms, Journal of the Operations Research Society of Japan, Vol. 27, No. 4, 1984, pp. 306-336.

191 A. Okabe, B. Boots, and K. Sugihara, *Spatial tessellations: Concepts and applications of Voronoi diagrams*, Wiley & Sons, 1992.

192. A. A. Oloufa, Triangulation applications in volume calculation, Journal of Computing in Civil Engineering, Vol. 5, No. 1, January 1991, pp. 103-121.

193. T. K. Peucker, R. J. Fowler, and J. J. Little, The triangulated irregular network, Proceedings ASP-ACSM Symposium on Digital Terrain Models, 1978.

194. C. S. Peterson, Adaptive contouring of three-dimensional surfaces, CAGD, Vol. 1, 1984, pp. 61-74.

195. L. A. Piegl and A. M. Richard, Algorithm and data structure for triangulating multiply connected polygonal domains, Computers & Graphics, Vol. 17, No. 5, 1993, pp. 563-574.

196. D. A. Plaisted and J. Hong, A heuristic triangulation algorithm, J. Algorithms, Vol. 8, 1987, pp. 405-437,

197. Pourazady, M. and M. Radhakrishnan, Optimization of a triangular mesh, Computers and Structures, Vol. 40, No. 3, 1991, pp. 795-804.

198. M. J. D. Powell, Piecewise quadratic approximation on triangles, in; Software for Numerical Mathematics, D. J. Evans (ed.), Academic Press, NY, 1974

199. M. J. D. Powell and M. A. Sabin, Piecewise quadratic approximation on triangles, ACM Trans. on Mathematical Software, Vol. 3, 1977, pp. 316-325.

200. P. L. Power, Minimal roughness property of the Delaunay triangulation: a shorter approach, Computer Aided Geometric Design, Vol. 9, 1992, pp. 491-494.

201. P. L. Power, The neutral case for the min-max angle criterion: a generalized approach, Computer Aided Geometric Design, Vol. 9, 1992, pp. 413-418.

202. F. P. Preparata and S. J. Hong, Convex hull of a finite set of points in two and three dimension, Commun. ACM, Vol. 20, No. 2, Feb. 1977, pp. 87-93.

203. F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*, Springer-Verlag, New York, 1985.

204. E. Quak and L. L. Schumaker, Cubic spline fitting using data dependent triangulations, Computer Aided Geometric Design, Volume 7, Numbers 1-4, 1990, pp. 293-301.

205. E. Quak and L. L. Schumaker, C^1 surface fitting using data dependent triangulations, in: *Approximation Theory VI*, C. Chui, L. L. Schumaker, and J. Ward, editors, Academic Press, 1989, pp. 545-548.

206. E. Quak And L. L. Schumaker, Least squares fitting by linear splines on data dependent triangulations, in: *Curves and Surfaces*, P.-J. Laurent, A. Le Mehaute and L. L. Schumaker, editors, Academic Press, 1991, pp. 387-390.

207. R. J. Renka, Algorithm 624: Triangulation and interpolation of arbitrarily distributed points in the plane, ACM TOMS, Vol. 10, 1984, pp. 440-442.

208. V. T. Rajan, Optimality of the Delaunay triangulation in R^d, In Proc. 7th ACM Symp. Comp. Geometry, 1991, pp. 357-363.

209. P. N. Rathie, A census of simple planar triangulations, J. Comb. Theory B, Vol. 16, 1974, pp. 134-138.

210. D. Rhynsburger, Analytic delineation of Thiessen polygons. Geograph. Anal., Vol. 5, 1973, pp. 133-144.

211. S. Rippa, Minimal roughness property of the Delaunay triangulation, Computer Aided Geometric Design, Vol. 7, 1990, pp. 489-497.

212. S. Rippa, Long and thin triangles can be good for linear interpolation, SIAM Journal on Numerical Analysis, Vol. 29, No. 1, February 1992, pp. 257-270.

213. S. Rippa, Piecewise linear interpolation and approximation schemes over data dependent triangulations, Ph. D. Thesis, 1989, Tel Aviv.

214. S. Rippa and B. Schiff, Minimum energy triangulations for elliptic problems, Comp. Meth. in applied Mech. and Eng., Vol. 84, 1990, pp. 257-274.

215. C. A. Rogers, *Packing and covering*, Cambridge University Press, 1964.

216. J. Ruppert and R. Seidel, On the difficulty of tetrahedralizing 3-dimensional nonconvex polyhedra, In Proc. 5th ACM Symp. Comp. Geometry, 1989, pp. 380-393.

217. N. Sapidis and R. Perucchio, Delaunay triangulation of arbitrarily shaped planar domains, Computer Aided Geometric Design, Vol. 8, 1991, pp. 421-438.

218. V. Sarin and S. Kapoor, Algorithms for relative neighbourhood graphs and Voronoi diagrams in simple polygons, Proc. 4th Canad. Conf. Comput. Geom. 1992, pp. 292-298.

219. L. Scarlatos and T. Pavlidis, Optimizing triangulation by curvature equalization, Proceedings of Visualization '92, IEEE CS Press, October 1992, pp. 333-339.

220. B. Schachter, Decomposition of polygons into convex sets., IEEE Transactions on Computing C-27, Volume 11, November 1978, pp. 1078-1082.

221. E. Schönhardt, Über die zerlegung von dreieckspolyedern in tetraeder, Math. Annalen, Vol. 98, 1928, 309-312.

222. W. J. Schroeder and M. S. Shephard, Geometry-based fully automatic mesh generation and the Delaunay triangulation, Int. J. Numer. Meth. Eng. Vol. 26, 1988, pp. 2503-2515.

223. W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, Decimation of triangle meshes, In SIGGRAPH '92, Volume 26, July 1992, pp. 65-70.

224. L. L. Schumaker, Fitting surfaces to scattered data, in: Approximation Theory II, G. G. Lorentz, C. K. Chui, and L. L. Schumaker (eds.), Academic Press, 1976, pp. 203-268.

225. L. L. Schumaker, Computing optimal triangulations using simulated annealing, Computer Aided Geometric Design, Volume 10, Numbers 3-4, pp. 329-345.

226. L. L. Schumaker, Triangulation methods, in: *Topics in Multivariate Approximation*, L. L. Schumaker, C. Chui and F. Utreras (eds.), Academic Press, New York, 1987, pp. 219-232.

227. L. L. Schumaker, Triangulations methods in CAGD, IEEE Computer Graphics and Applications, Vol. 13, January 1993, pp. 47-52.

228. A. Seidel, Constrained Delaunay triangulations and Voronoi diagrams with obstacles, in: 1*978-1988 Ten Years IIG*, H. S. Poingratz and W. Schinnerl (eds.), 1988, pp. 178-191.

229. M. Senechal, Which tetrahedra fill space?, Math. Magazine, Vol. 54, 1981, pp. 227-243.

230. M. I. Shamos, Computational Geometry, Ph. D. Dissertation, Yale University, 1978.

231. M. Shapiro, A note on Lee and Schachter's algorithm for Delaunay triangulation, International Journal of Computer and Information Sciences, Vol. 10, N. 6, 1981, pp. 413-418.

232. D. N. Shenton and Z. J. Cendes, Three-dimensional finite element mesh generation using Delaunay Tessellation, IEEE Trans. on Magetics, MAG-21, 1985, pp. 2535-2538.

233. D. Shirley and A. Tuchman, A polygonal approximation to direct scalar volume rendering, Computer Graphics, Vol. 24, November 1990, pp. 63-70.

234. G. M. Shute, L. L. Deneen, and C. D. Thomborson, An O(N log N) plane-sweep algorithm for L_1 and L_∞ Delaunay triangulations, Algorithmica, Vol. 6, 1978, pp. 207-221

235. R. Sibson, Locally equiangular triangulations, Computer J., Vol. 21, 1978, pp. 243-245.

236. R. Sibson, A brief description of natural neighbour interpolation, Chapter 2 of Interpreting Multivariate Data, Wiley, New York, 1981.

237. C. T. Silva, J. S. B. Mitchell, and A. E. Kaufman, Automatic generation of triangular irregular networks using greedy cuts, Proceedings of Visualization '95, IEEE CS Press, October 1995, pp. 201-208.

238. S. W. Sloan, A fast algorithm for constructing Delaunay triangulations in the plane, Advances in Engineering Software, Vol. 9, January 1987, pp. 34-55.

239. S. W. Sloan and G. T. Houlsby, An implementation of Watson's algorithm for computing 2-dimensional Delaunay triangulations, Advances in Engineering Software, Vol. 6, 1984, pp. 192-197.

240. C. Stein, B. Becker, and N. Max, Sorting and hardware assisted rendering for volume visualization, in: 1994 Symposium on Volume Visualization, Washington DC, October 1994, pp. 83-89.

241. K. Sugihara and M. Iri, Construction of the Voronoi diagram for "one million" generators in single-precision arithmetic, Proceedings of the IEEE, Vol. 80, 1992, pp. 1471-1484.

242. M. Tanemura, T. Ogawa, and W. Ogita, A new algorithm for three-dimensional Voronoi tessellation, Journal of Computational Physics, Vol. 51, 1983, pp. 191-207.

243. R. E. Tarjan and C. J. Van Wyk, An O(n log log n)-time algorithm for triangulating a simple polygon, SIAM J. Comput., Vol. 17, 1988, pp. 143-178.

244. A. H. Thiessen, Precipitation averages for large areas, Monthly Weather Review, Vol. 39, 1911, pp. 1032-1034.

245. J. F. Thompson, *Numerical Grid Generation*, North-Holland, 1982.

246. J. C. Tipper, Straightforward iterative algorithm for the planar Voronoi diagram, Information Processing Letters, Vol. 34, No. 3, April 1990, pp. 155-160.

247. J. C. Tipper, FORTRAN programs to construct the planar Voronoi diagram, Computers & Geosciences, Vol. 17, 1991, pp. 597-632.

248. G. Toussaint, Efficient triangulation of simple polygons, Visual Comput., Vol. 7, 1991, pp. 280-295.

249. G. T. Toussaint, C. Verbrugge, C. Wang, and B. Zhu, Tetrahedrization of simple and not simple polyhedra, CCCG Proc. of the fifth Canadian Conference on Computational Geometry, 1994.

250. V. J. D. Tsai, Delaunay triangulation in TIN creation: An overview and a linear-time algorithm, Int. J. Geographical Information Systems, Vol. 7, 1993, pp. 501-524.

251. W. T. Tutte, A census of planar triangulations, Canadian J. Math., Vol. 14, 1962, pp. 21-38.

252. G. Voronoi, Nouvelles applications des paramatres continusala theorie des formes quadratiques, Deuxieme Memoire, Recherches sur les parallelloedres primitifs, J. reine angew. Mathe. , Vol. 134, 1908, pp. 198-287.

253. C. Wang and L. Schubert, An optimal algorithm for constructing the Delaunay triangulation of a set of line segments, In Proc. 3rd ACM Symp. Comp. Geometry, 1987, pp. 223-232.

254. D. F. Watson, Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes, Comp. J., Vol. 24, 1981, pp. 167-172.

255. D. F. Watson and G. M. Philip, Systematic triangulations, Computer Vision, Graphics, and Image Processing, Vol. 26, 1984, pp. 217-223.

256. N. D. Weatherhill and O. Hassan, Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints, Internat. J. Numer. Meth. Eng. Vol. 37, 1994, pp. 2005-3039.

257. P. Williams, Visibility ordering meshed polyhedra, ACM Transactions on Graphics, Vol. 11, No. 2, 1992, pp. 103-126.

258. B. Woerdenweber, Automatic mesh generation of 2- and 3-dimensional curvilinear manifolds, Ph. D. Dissertation, University of Cambridge, 1981.

259. B. Woerdenweber, Finite-element analysis for the naive user, in *Solid Modeling by Computers from Theory to Applications*, M. S. Pickett and J. Boyse (eds), Plenum, Ny, 1984, pp. 81-100.

260. A. J. Worsey and B. Piper, A trivariate Powell-Sabin interpolant, Computer Aided Geometric Design, Volume 5, Number 3, 1988 pp. 177-186.

261. A. J. Worsey and G. Farin, An n-dimensional Clough-Tocher interpolant, Constructive Approximation, Vol. 3, 1987, pp. 99-110.

262. F. F. Yao, Computational geometry, in: *Handbook of Theoretical Computer Science*, Vol. A, Chapter 7, J. van Leeuwen (ed.), Elsevier and MIT Press, 1990, pp. 343-389.

263. A. Zenisek, Polynomical approximation on tetrahedrons in the finite element method, J. Approximation Theory, Vol. 7, 1973, pp. 334-351.

Volume Modelling

Gregory M. Nielson

Reference:
Nielson, GM, Volume Modelling. In: M. Chen et al. (eds.). *Volume Graphics,* Springer, 2000; 29-48.

2.1 Introduction

This chapter will present an overview of the emerging research area of *volume modelling*. To date, there has been considerable research on the development of techniques for visualising volume data, but very little on modelling volume data. This is somewhat surprising since the potential benefits of volume models are tremendous. This situation is somewhat explained by the fact that volume data is relatively new and researchers have spent their efforts in figuring out ways to "look" at the data and have not been able to afford the resources needed to develop methods for modelling volume data. In addition to providing a means for visualising volume data, some of the benefits of a volume model are the generation of hierarchical and multi-resolution models which are extremely useful for the efficient analysis, visualisation, transmission, and archiving of volume data. In addition, the volume model can serve as the mathematical foundation for subsequent engineering simulations and analysis required for design and fabrication.

While interest is steadily growing, the area of volume modelling is still in its infant stages and currently there are few techniques and little expertise available. In the next section, we give some precise definitions and describe the scope of our vision of volume modelling and generally make an appeal for its development. It is important to realise that practically all visualisation tools require some type of volume model for their application. Sometimes the model is so obvious that we may fail to notice it. (For example, the linear interpolation into voxels used by the standard marching cubes algorithm.) Many of the relatively simple modelling techniques used for the more popular visualisation tools of today do not apply or scale up to the data sets currently of interest. These data sets require much more sophisticated modelling techniques. Another barrier to analysing volume data sets is the fact that they are often large, and because of this, they are normally associated with complex and complicated phenomena. Multi-resolution models can be helpful in this regard. Wavelet models (and the concepts related to wavelet models) have traditionally been targeted at compression, but they can also form the basis for analysis tools that allow for removal of clutter and detail and assist in efficient browsing and zooming. In the third section of this chapter, we will discuss some research issues in representing volume models.

We think that it would benefit our readers if we were to be somewhat clear about some very commonly used terminology in this area:

- **Volume Visualisation.** We use this as the umbrella term. It encompasses all aspects of analysing and visualising volume data and models.
- **Volume Graphics.** This topic deals with the issues of producing the images associated with volume visualisation. It is analogous to the traditional polygon graphics. It includes viewing models, illumination models, and scan conversion algorithms and related issues required for the creation of images. A more comprehensive definition can be found in [1]
- **Volume Rendering.** While this term appears somewhat generic, over the years it has become associated with that particular method of rendering a volume model which is based upon a certain model of transparency (called the volume rendering integral). We use it in this context.
- **Volume Modelling.** This is the topic of this chapter. Our purpose is to more precisely define this topic and to make a general appeal for its development and growth.

2.2 Definition and Scope of Volume Modelling

In this section, we take three possible approaches to a definition of volume modelling: (1) A volume model can be viewed as the process of modelling volume data. (2) It can be thought of as a generalisation in dimension to surface modelling. (3) It can be viewed as the means to provide the input to the volume rendering integral. In the following subsections, we expand on each of these approaches.

2.2.1 Definition by Modelling of Volume Data

Volume scanning devices produce a value of a dependent quantity at various locations in space. Examples are widespread, and include:

1. the results of MRI and CAT scanners in the medical field,
2. measurements of mineral concentration from core samples scattered over some typography,
3. results of a 3D, CFD simulation and
4. free-hand ultrasound where a 3D position/orientation sensor is attached to an ultrasound probe.

What is common here is that each sample of the data consists of a position in space and the measurement or computation of an associated dependent variable. Invoking mathematical means of modelling and representing this type of data is one definition of volume modelling. Volume data does not need to have just a single scalar dependent variable, there may be several. In fact, some volume data has a dependent variable that is a vector. This is the case for the data of CFD simulations. Here the dependent data consists of a single scalar (pressure) and a vector (velocity of the flow).

In this subsection, we describe four examples of volume data sets. Each requires some type of volume model before a visualisation tool can be applied. For some of the data sets, adequate volume models are not currently available.

Rectilinear, Cartesian Grids from Medical Scanners

This is an example of the most conventional type of data we see in volume visualisation. It represents the results of some scanning device (such as MRI or CAT) and can be viewed as measurements on a Cartesian grid. Because of this, the domain is implied and so a simple three dimensional array of dependent values, $d_{ijk}, i = 1, \cdots, N_x, j = 1, \cdots, N_y, k = 1, \cdots, N_z$ can be used to represent the data. The images of Figure 2.1 show isosurfaces which have been extracted from a type of wavelet model applied to this type of data.

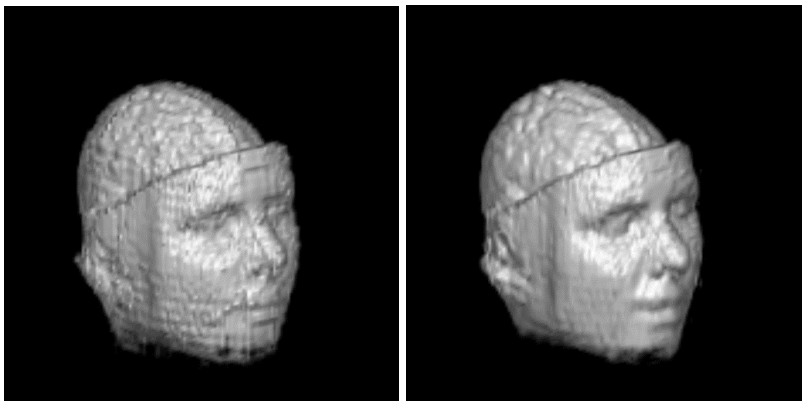
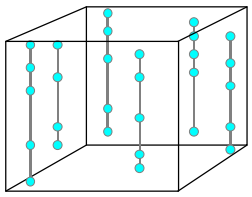


Figure 2.1. Examples of isosurfaces extracted from a volume model called Blend of Linear and Constant (BLaC) wavelets. The left image is based upon $\Delta = 0.0$ and the right, $\Delta = 0.43$.

Seismic Data Samples in Geophysical Studies

This data is typical of measured data extracted from core samples which are taken at scattered locations, as shown in Figure 2.2. The measurements within core samples are not necessarily at uniform depths and can vary from one core sample to the next. Mathematically, we can represent this data as

$$(X_i, Y_i, Z_{i_j}; M_{i_j}), i = 1, \cdots, N; i_j = 1, \cdots, N_i .$$



Location (X, Y, Z)			**Mineral**
5.50	1.00	0.00	11.0
5.50	1.00	10.00	10.0
…	…	…	…
…	…	…	…
…	…	…	…

Figure 2.2. Diagram depicting core samples.

In Figure 2.3, we show a snapshot of an interactive tool for interrogating this type of data. Colour contours are shown at a user specified height. Any number of these can exist and they can be moved up and down. It is clear that this type of visualisation or any other would not be possible without a volume model.

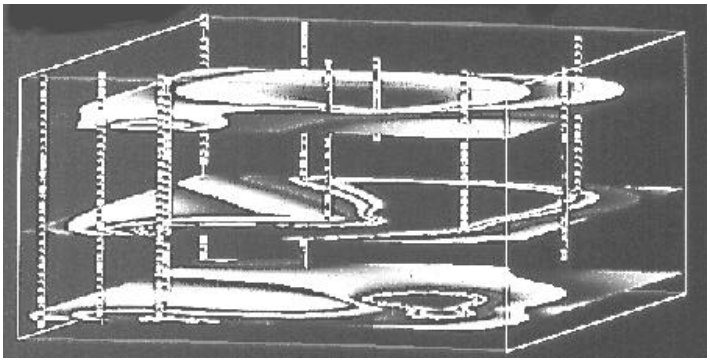


Figure 2.3. Screen snapshot of an interactive tool for visualising model of core sample data. (Courtesy of David Kao).

Curvilinear Grids from Computational Fluid Dynamics Simulations

A time-dependent, 3D curvilinear grid is described by three, four-dimensional geometry arrays, $(X_{ijk\ell}, Y_{ijk\ell}, Z_{ijk\ell})$ which provide the vertices for a cellular decomposition of the domain of interest for each time step, t_ℓ (Figures 2.4 and 2.5). The numerical simulation provides the solution to the Navier/Stokes equations at these vertices. This information is provided by four additional arrays $(P_{ijk\ell}, U_{ijk\ell}, V_{ijk\ell}, W_{ijk\ell})$, where $P_{ijk\ell}$ is a scalar representing pressure and $(U_{ijk\ell}, V_{ijk\ell}, W_{ijk\ell})$ is the velocity at vertex $(X_{ijk\ell}, Y_{ijk\ell}, Z_{ijk\ell})$ at time step t_ℓ. Typical spatial resolutions of interest and value today range from 10^2 to 10^3. For efficiency, time-dependent grids are often partitioned into blocks with vertices of some blocks moving over time and others being stationary. For example, the V-22 Tiltrotor data set [2] consists of 26 blocks, of which 9 are time dependent and the remaining ones are steady. For this data set there are 1,400 time steps each consisting of about 100 MB of data.

Free-hand Ultrasound Data

A typical ultrasound probe produces a "slice" of data through an object. These are called B-scans and are viewed and manipulated as images (Figure 2.6). The use of the phrase "free-hand" means the addition of a 3D POSE (Position and Eularian angles) sensor attached to the conventional ultrasound probe. This allows one to associate a position in 3D space for each of the pixels of a B-scan image. Mathematically, we can then view each pixel as a sample of the volume model and represent it as $(x_i, y_i, z_i; d_i), i = 1, \cdots, N$.

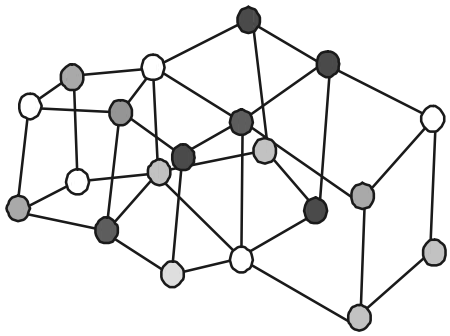


Figure 2.4. A diagram illustrating a 3D curvilinear grid.

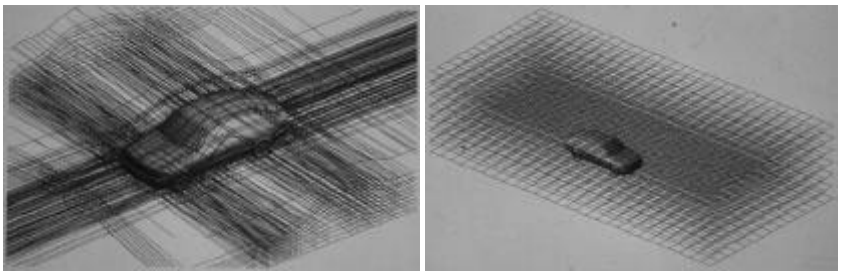


Figure 2.5. Curvilinear grid on left and resampled rectilinear grid on right.

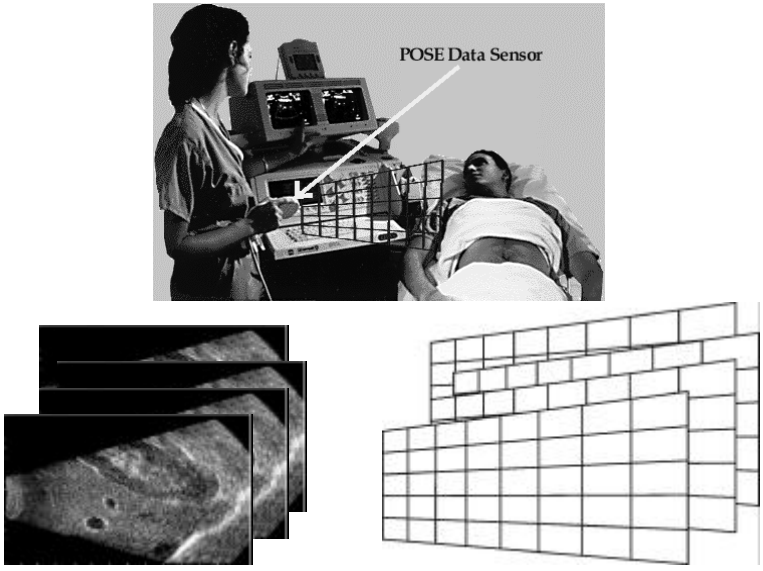


Figure 2.6. The process of collecting free-hand ultrasound data

The idea of free-hand ultrasound data is over ten years old [3-4], but effective and efficient modelling of this type of data is a very difficult problem which is only recently receiving some attention [5]. We will cover some exciting new work in this area in the next section of this chapter. A volume rendering (MIP) based upon these new progressive models is shown in Figure 2.7.

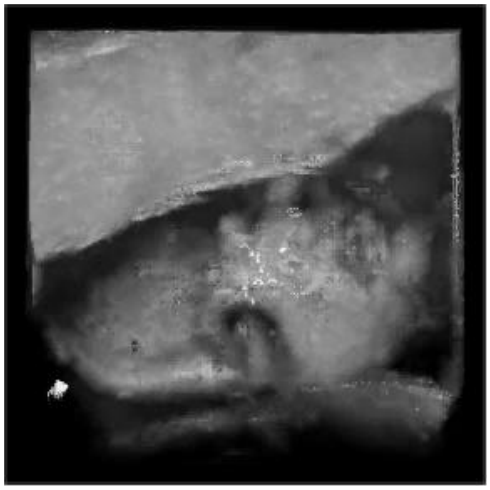


Figure 2.7. A snapshot from an interactive viewing of a progressive volume model (discussed in the next section) of ultrasound data. (Data is courtesy of Bill Lorensen.)

2.2.2 Definition by Analogy to Surface Modelling

In Figure 2.8, we see that the flow of information from top to bottom is “surface” to “volume” and left to right is “modelling” to “graphics”. The traditional computer graphics pipeline, which is illustrated in the top half of Figure 2.8, consists of a parametric surface model that is evaluated at a set of parameter values in order to obtain a polygon tessellation or approximation. The polygons are mapped by the viewing transformation to device coordinates and then scan converted. In a similar manner we can envision a “volume graphics” system that takes cells (the 3D analogues of polygons) that have an associated intensity at each vertex and scan converts them to a 3D frame buffer which subsequently is used to produce a volume rendering (either by hardware or software). In the diagram of Figure 2.8, volume modelling is represented by the oval, which is providing the information for the tessellation process. That is, volume modelling from this point of view is whatever is evaluated and used to produce the 3D tessellation with density values at the vertices.

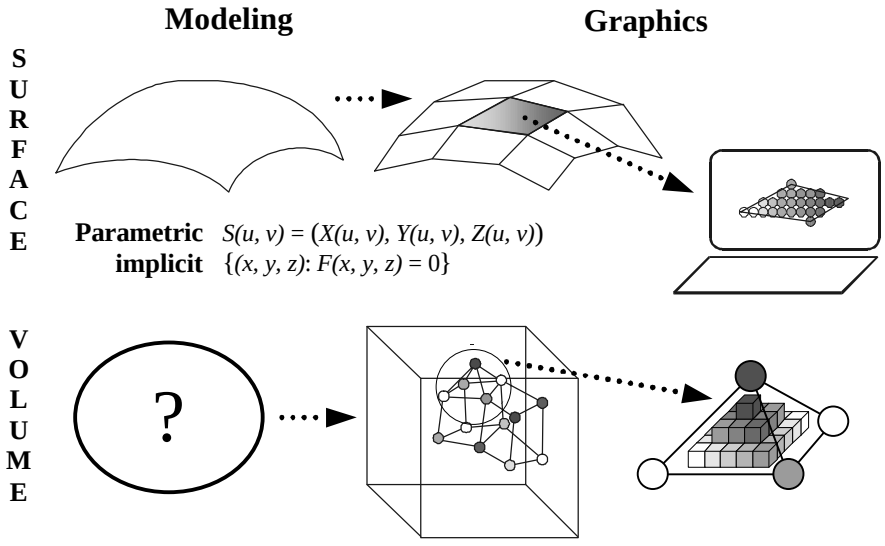


Figure 2.8. Diagram depicting the analogy between surface modelling and volume modelling.

2.2.3 Definition by Input to the Volume Rendering Equation

Ray cast volume rendering images are based upon a compositing process. Given a sorted collection of objects, which emit C_i and have transparency t_i, we compute the observed intensity by applying a very simple model of transparency and successively computing $F_i = (1-t_i)C_i + t_i F_{i-1}$ (Figure 2.9). A standard limiting process on this discrete compositing leads to the volume rendering integral

$$F(x) = \int_{x_0}^{x} d(s)C(s)e^{-\int_{s}^{x} d(t)dt} ds$$

where the density and the transparency are related as $t(x_0, x) = e^{-\int_{x_0}^{x} d(u)de}$. From this point of view, in order to produce a volume rendering we need a trivariate density function, d, and a trivariate colour function C. The mathematical model from which these two trivariate functions are obtained is called a volume model. In many applications, one data function D leads to both of these. A transfer operator (function) is applied to D to yield d. One can use the choice of this transfer function to make certain values opaque and visible and other ranges transparent. If additional attributes are known, or if information is known about the location of objects, then it may be possible to also define the colour function C. This is related to the very difficult problem of segmenting the data into different classes of materials from

which the colour function can be determined (possibly in a piece-wise constant fashion). Often C is taken as a direct relation from D, or possibly it is computed from D and the gradient of D in order to flush out isosurfaces.

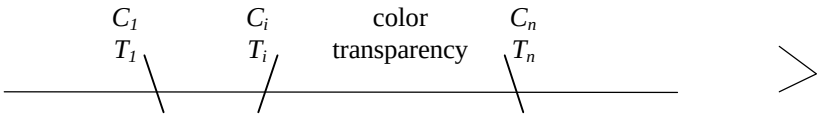


Figure 2.9. Compositing.

2.2.4 Summary of Definition of Volume Modelling

As we can see, all three of these approaches (volume data, volume rendering integral and analogue to surface modelling) lead to the same definition of volume modelling. A volume model is a trivariate relationship whose independent argument is a position in 3D space and whose dependent argument is a scalar or tuple of scalars or even a vector. The volume model might also have the aspect of varying over time.

Before we leave this section, we should mention some concepts with similar terminology, which are not volume models. Even though it is an important part of certain algorithms in volume graphics the problem of scan converting lines, curves, surfaces or solids into discrete voxels [6-7] is not the process of volume modelling. Nor is a model of a volume (Figure 2.10), as for example described by the methods of CSG (constructive solid geometry). It is a region of space. For the same reasons

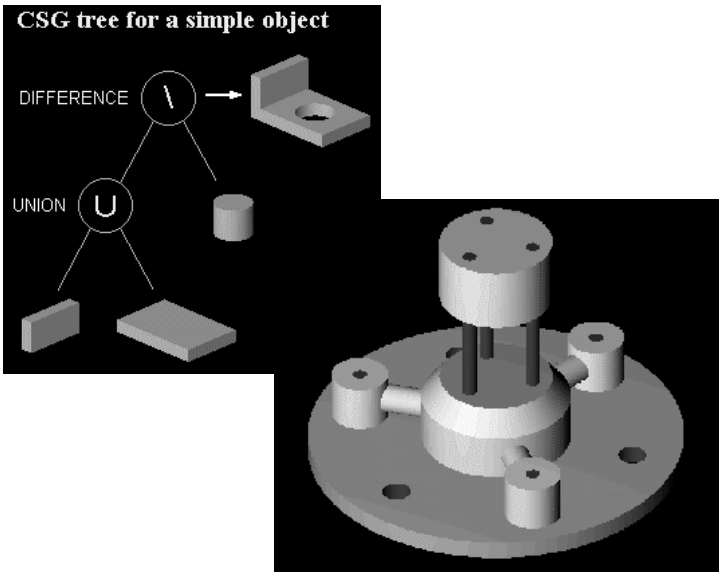
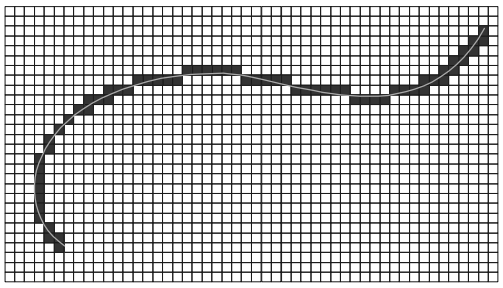


Figure 2.10 A model of a volume is not a volume model

that a collection of pixels is not a curve (Figure 2.11) and a cloud of points is not a surface (Figure 2.12), a collection of voxels or tetrahedra (see Figure 2.13) is not a volume model. It is a spatial enumeration and is missing the important component of a relationship possessed by a volume model. But on the other hand, we can make the following observation. Just as it is possible (though not necessarily easy) to parameterise and fit a collection of points to a curve and just as it possible (but even more difficult) to parameterise a cloud of unorganised points to a surface, it is possible to construct a volume model based upon discrete voxels and points.



pixels:

(6,16),
(33,19),
(13,20),
. . .

Figure 2.11. A collection of pixels is not a curve, but it may lead to one when an ordering and other implications are added.

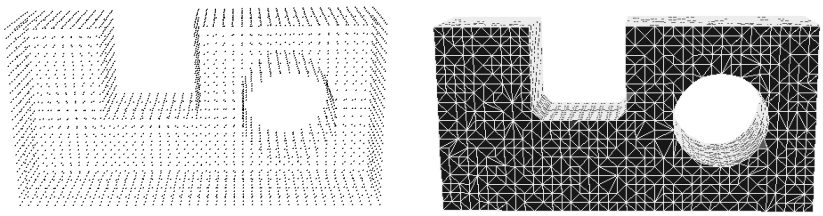


Figure 2.12. A collection of points is not a surface, but it may lead to one when the topology of a triangulation is added. (Images courtesy of UNI-KL.)

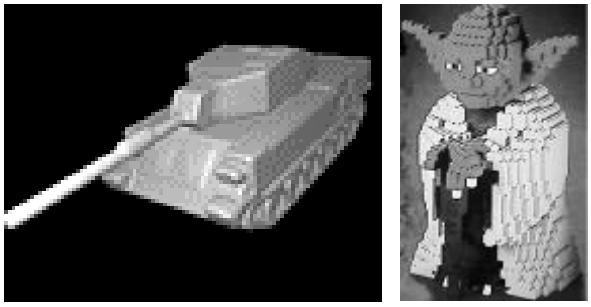


Figure 2.13. A collection of voxels is not a volume model. It is a voxelised volume that serves as a means to describe a region of 3D space. (The left image is courtesy of Arie Kaufman and the right image is courtesy of Lego.)

2.3 Research Issues in Representing Volume Models

In this section, we cover a sampling of methods for representing volume models along with some research issues for each of these methods.

Basis Functions

This is the most straightforward approach to representing a volume model. In this approach, we assume a general form of the volume model. It involves coefficients and basis functions. The volume data or other considerations are then used to select the coefficients in the generic form of the model. In mathematical terms, the volume model takes the form:

$$VM(x,y,z) = \sum_{i=1}^{N} a_i b_i(x,y,z) \tag{2.1}$$

where $b_i(x,y,z), i = 1,\cdots,N$ denote the basis functions and the unknown coefficients are $a_i, i = 1,\cdots,N$. This type of volume model is often used in a visualisation tool even though it may not be completely apparent what the form of volume model really is. For example, with the marching cubes algorithm piece-wise linear interpolation into voxels is used. This is equivalent to using a volume model of the form given in Equation 2.1 where the basis functions are the 3D versions of the "hat" functions based upon a Cartesian grid. Viewing the modelling process this way opens up the possibility of using many other, possibly more efficient, basis functions.

Research Issues: The research question for a particular application then becomes how to select the basis functions and then how to select the method of computing the coefficients of the volume model. Ideas about choices for basis functions come from generalising useful and successful basis functions for lower dimension problems. For example, splines have served the surface modelling community very well. So then the question arises as to what are the proper basis functions for a spline volume model. Some suggestions and comparisons are discussed in [8]. Issues as to whether interpolation or approximation is most appropriate must be addressed. Also, should the basis functions have local or global support? Are there numerical condition problems in the computation of the coefficients? What type of basis functions will allow interactive speeds? □

Mathematical Modelling

Prior to describing this method of representing volume models, let us first establish a context by mentioning a few simple things about mathematical/physical modelling. One of the first uses of mathematical modelling, that we all see, are the equations for a pendulum introduced in a beginning physics course. If $\Theta(t)$ represents the angle

of deflection at time *t*, then Newton's second law takes the form $\frac{d^2\Theta}{dt^2} = \frac{-g}{L}\sin(\Theta)$ where *g* is the gravitational constant and *L* is the length of the pendulum. To completely determine the solution, the initial conditions $\Theta(0) = \Theta_0$ and $\frac{d\Theta}{dt}(0) = \Theta_0'$ must be provided. Even though these equations uniquely determine the solution, there is not a simple formula for $\Theta(t)$. A solution usually requires infinite expansions or numerical techniques.

For CFD (computational fluid dynamics) studies, the Navier/Stokes equations characterise the volume model as the solution of a second-order PDE:

$$r\left(\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} * \nabla)\mathbf{V}\right) = \nabla p + m\nabla^2\mathbf{V} + \mathbf{F} \tag{2.2}$$

where $\boldsymbol{V} = (u, v, w) = (u(x,y,z), v(x,y,z), w(x,y,z))$ represents the velocity vector and $p = p(x,y,z)$ is a scalar valued function representing pressure. The scalar constant *r* is fluid density, and *m* is the dynamic viscosity. The external forces are $\boldsymbol{F} = (X, Y, Z)$. As with the pendulum problem, a solution of Equation 2.2 for $\boldsymbol{V}$ and *p* requires some type of approximation or numerical technique. This is where curvilinear grids come into play. They are often used for the numerical solutions of the PDE's that characterise a volume model. Either they serve as a cellular decomposition for a finite element approach to a solution or they are used in the finite difference approach where partial derivatives are replaced with discrete approximations. In either case, a solution to the volume model is computed at each of the nodes of the curvilinear grid. Later, this data is passed along for post visualisation and analysis. What is often not passed along is the method of solution. Most data visualisation/analysis tools require that the discrete data be modelled or interpolated in some manner. Quite often, the simplest or most readily available method is used for this purpose whether or not it has anything to do with the underlying volume model. This is an unfortunate situation which is likely to change as the scientists themselves are getting more and more involved in the analysis and visualisation of their data and as the general level of knowledge and mathematical sophistication is increasing in the area of volume visualisation.

Research Issues: Can the mathematical model be "attached" to the simulated data? Can the mathematical model be applied locally? In a multi-resolution manner? How much error is associated with each approach? □

Deformations

In a nutshell, the basic idea here is described as follows: We start with a generic model which has an associated classification function and morph this generic model to a particular model inferred by the collected data. This is done with the use of function norms and a minimisation strategy. The classification function for the particular data is now obtained by composing the morphing function and the generic

classification function.

A 3D morph can be accomplished with a trivariate map:

$$\begin{pmatrix} X_p \\ Y_p \\ Z_p \end{pmatrix} = \begin{pmatrix} F_x(X_g, Y_g, Z_g) \\ F_y(X_g, Y_g, Z_g) \\ F_z(X_g, Y_g, Z_g) \end{pmatrix} = \sum \begin{pmatrix} a_i \\ b_i \\ c_i \end{pmatrix} M_i(X_g, Y_g, Z_g)$$

which maps a portion of 3D space onto itself. It deforms the space. These types of maps have been used for designing objects [9] and animations [10]. The basis functions $M_i(X_g, Y_g, Z_g)$ would usually be polynomial or piece-wise polynomial. The coefficients of the morph $(a_i, b_i, c_i)^t$ may be thought of as control points and the idea is to manipulate these values so as to accomplish the desired end.

And now more details: Suppose the generic model has been segmented so that we have a trivariate function $C(x, y, z)$ which represents the colour or classification function. This function tells us what material is located at position (x, y, z). It might be that $C(x, y, z)$ is piece-wise defined (say over voxels) but the precise type of function it is, is not important in this context. Also associated with this generic model is a data function $d(x, y, z)$. This is to represent, for example, a model obtained from applying our scanning device to the generic model and then fitting this data with a volume model. This function may possibly be obtained by a simulation of a mathematical model of the generic model using $C(x, y, z)$ and the physical properties of the materials that are classified or even scanning a physical phantom model. Both C and d are based upon some type of basis functions and therefore we can represent them as follows:

$$C(x, y, z) = \sum a_i C_i(x, y, z), \quad d(x, y, z) = \sum a_i d_i(x, y, z).$$

Next we obtain the scanned data which we represent as $d_p(x, y, z)$ where p is for "particular". What we want is the colour or classification function for this particular data. Let us call it $C_p(x, y, z)$. We first find a morphing function M which maps the generic model into the particular model. This is done on the basis of the scanned data. We choose M so that the function $d(M(x, y, z))$ is close to function $d_p(x, y, z)$. This will require a representation of M in terms of some unknown coefficients and a norm or method of discretely measuring the error between these two functions. This leads to a minimisation problem where the parameters of M are manipulated until the optimal or best fitting morphing function is obtained. We then take as the classification function for the particular model to be:

$$C_p(x, y, z) = C(M(x, y, z))$$

Research Issues: What is the form of the morphing map? Trivariate Bezier? Catmull/Clark solid? Piece-wise linear over tetrahedra? How to incorporate particular data into the computation of the morphing map coefficients? Least squares with cost function? Simulated annealing? □

Wavelet-Type Multi-resolution Models

The ideas and concepts of wavelets have their origins in the univariate world of time varying signals [11-12]. Many of the more useful techniques have been extended to certain types of surface models in the past several years [13]. The first use of wavelet techniques for volume data was by Muraki who used tensor product techniques to obtain wavelet models for MRI data. In [14] he discussed the application of Batelle-Lamarie wavelets and later [15] he compared these results to those of the DOG wavelets (difference of Gaussian). While tensor product methods afford a relatively easy way to extend the original univariate wavelet models to 3D data, they are often not suitable for certain applications and types of volume data. This includes the volumetric curvilinear grids of CFD data, as we will explain later in this section.

Wavelet expansions often are based upon basis functions with different resolutions and within each of these resolutions there are basis functions with different regions of support. This yields two views of the wavelet expansion and allows for two very useful types of reconstructions. We can pick out the resolution of interest and approximate with only these basis functions. This would allow, for example, the elimination of noise or clutter in order to visualise an overview or trend in some data. We can pick a region of interest and only use the basis functions that have support (non-zero values) in this region. This allows for efficient means to zooming in and out for browsing.

$$F(x) = \sum a_i L_i(x) + \sum b_i M_i(x) + \sum c_i H_i(x) = \sum_{regn\,1} a_{1i} R_{1i}(x) + \cdots + \sum_{regn\,N} a_{Ni} R_{Ni}(x)$$

low medium high

Both of the properties of compact support and orthogonality are important to the successful application of wavelets. Unfortunately if we also impose symmetry then we are frustrated in our attempts to define piece-wise linear (polynomial in general) wavelets. A recently developed wavelet [16] overcomes this obstacle with a technique for blending the piece-wise constant and piece-wise linear wavelets. There is a parameter, Δ, which allows the user to emphasise the compact support properties of the Haar wavelet or to emphasise the higher order approximations possible with a piece-wise linear wavelet.

We now turn our attention to wavelets for curvilinear grids. Recently, we published some results on the development of wavelets for 2D curvilinear grids in [17]. We are currently working on extending these techniques to 3D. In this work, the nested wavelet spaces are defined in a piece-wise manner over nested cellular decompositions. One important constraint that we imposed on this cellular decomposition was that the original inner boundary must persist at all levels. This constraint added considerable complexity to the models and subsequent algorithms, but without it, we felt that the application of the wavelets would suffer. One of the reasons for this is the fact that much of the activity of the flow takes place near the inner boundary and a degradation of this representation at low resolutions would deter the possibility to analyse the flow. We opted for a knot removal approach for

building the nested cellular decomposition. We will report on this work in the near future.

Research Issues: How to define wavelet volume models for the types of grids and data sets of interest in volume visualisation today? For example, 3D, time dependent curvilinear grids, tetrahedral decompositions, spherical curvilinear grids, free-hand ultrasound data and, in general, scattered volume data. How important is the trade-off between orthogonality and local support for this general application? Are nested spaces critical? Is it better to build multi-resolution models for isosurfaces or the volumes from which they are extracted. Can both be done at the same time? □

Progressive Volume Models

The basic idea of progressive models can be quickly gleaned from the univariate data example illustrated in Figure 2.14. On the left, the raw data is modelled by a piece-wise linear function in the bottom left graph. Successive local, piece-wise approximations are replaced with more global models leading to the final model in the top left graph. (See [18] for a model of this type applicable to Cartesian grids and [19] for a more general algorithm which was applied to curvilinear grids.) On the right, we start with a global model (linear least squares for example) and examine if it is acceptable or not. If not, then the domain is split, a new piece-wise model is computed and the same acceptability criterion is repeated for the sub-models. These are simple, yet powerful ideas for obtaining models whose complexity and ability to fit conform to the complexity and variability of the data. We feel there is a great promise for these ideas in volume models, but it is not a trivial matter to extend these ideas to 3D.

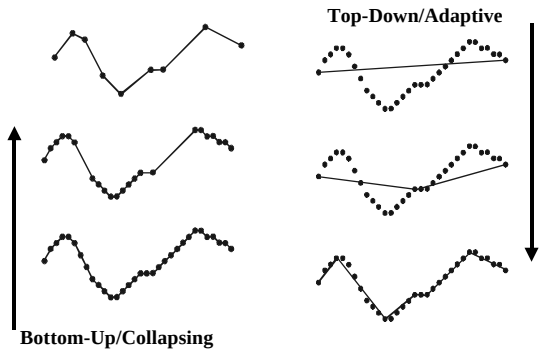


Figure 2.14. The univariate example illustrates the basic ideas of two approaches to progressive models. Volume modelling is interested in these concepts extended to 3D.

In addition to the oracle (the general collapsing or subdivision decision making process) there is the requirement of an effective and useful means of actually doing the subdivision and collapsing. A general approach to solving the problem is to think up something for 2D and then try to generalise it to 3D. The simplest and most robust cells are triangles in 2D and tetrahedra in 3D. (See [20] for basic algorithms

and data structures for triangles and tetrahedra.) The basic problem with building meshes that are coarse in one region and fine in another is the avoidance of the so-called “cracking problem”. We mention three approaches. See Figure 2.15. The method of Maubach [21] performs a local subdivision and repairs the crack by propagating this split out through the mesh. The method of Bey [22] has been used in FEM [23] and a variation has been discussed and used for volume models by Grosso et al. [24]. It uses a combination of two types of subdivisions to avoid cracks and avoid poorly shaped tetrahedra. A new approach is based upon not worrying about the crack, but rather using a Coons patch local model that covers it over [25-26]. Each of these approaches has it own set of research issues that must be worked out before the methods become viable, but each shows promise.

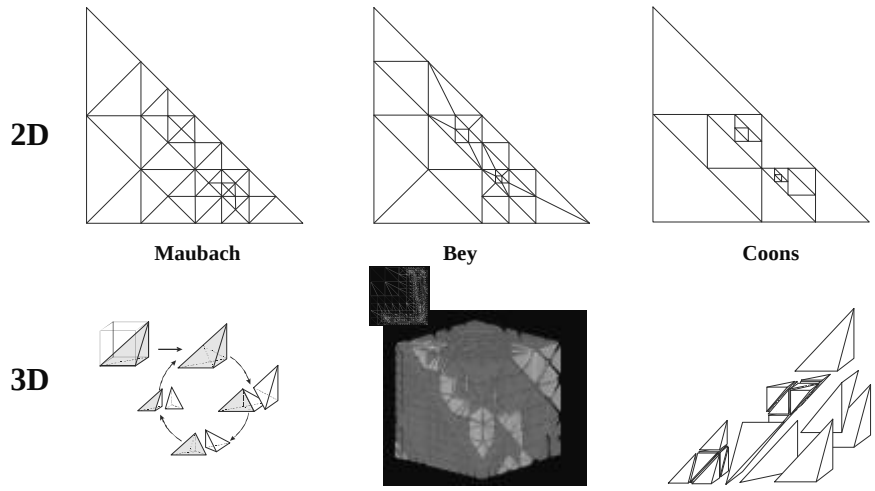


Figure 2.15. Three different approaches to the cellular decomposition for progressive models which avoid the cracking problem. (The 2D is shown only for illustration purposes. Volume modelling is concerned with the 3D case.)

We now describe some current research results in this area. They are rather exciting. In Figure 2.16, samples of some free-hand ultrasound data are shown. This data was collected in the neck region and includes portions of the carotid artery and the thyroid gland. The complete set of data consists of approximately a million data points. It is noisy (due to the ultrasound sensors) and it is redundant due to the overlaps caused by the free-hand method of collection. A typical tetrahedral mesh resulting from the adaptive method of fitting is shown in the right image of Figure 2.16. Note that the tetrahedra are much smaller and denser in the regions where the data is more dense and exhibit greater variation. This shows the ability of the model to conform to the complexity of the data. In Figure 2.17, we show the results of a very low resolution model. The left image of Figure 2.17 shows 5 B-scans of the original data and a “floating window” reconstructed from the volume model. On the right the same five data B-scans are shown alongside their corresponding approximations. Also the reconstruction of the “floating window” is shown on the

right. Figure 2.18 shows a higher resolution model. The results are impressive in light of the number of vertices and the detail that is present in the approximations. A different and quite interesting way to compare the approximation is shown in Figure 2.19. And due to the existence of the volume model, tools such as that shown in Figure 2.20 are possible. Figure 2.7 is also based upon the volume modelling techniques we have just described.

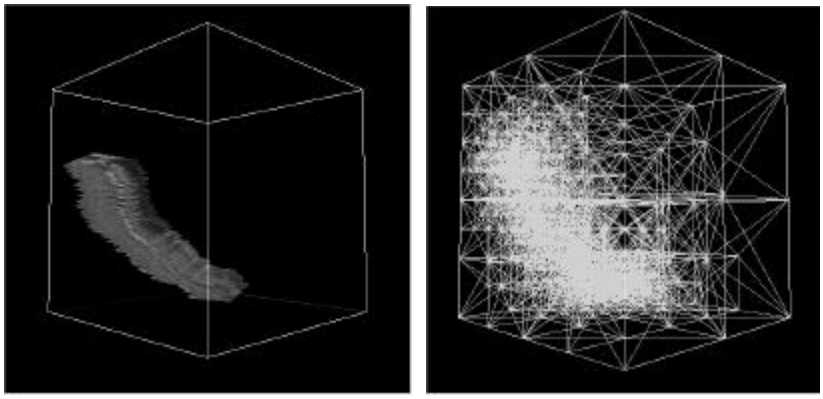


Figure 2.16. Free-hand ultrasound data collection and typical tetrahedral mesh for progressive model. (Data courtesy of Cambridge University.)

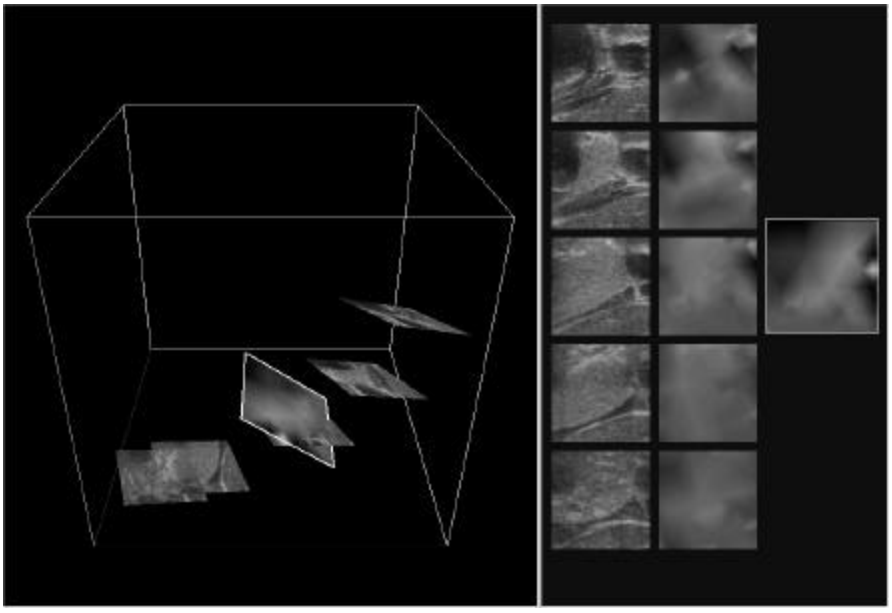


Figure 2.17. Results from progressive fit with a fairly large RMS error of 17.3 with only 909 vertices (approximately 1000 to 1 reduction).

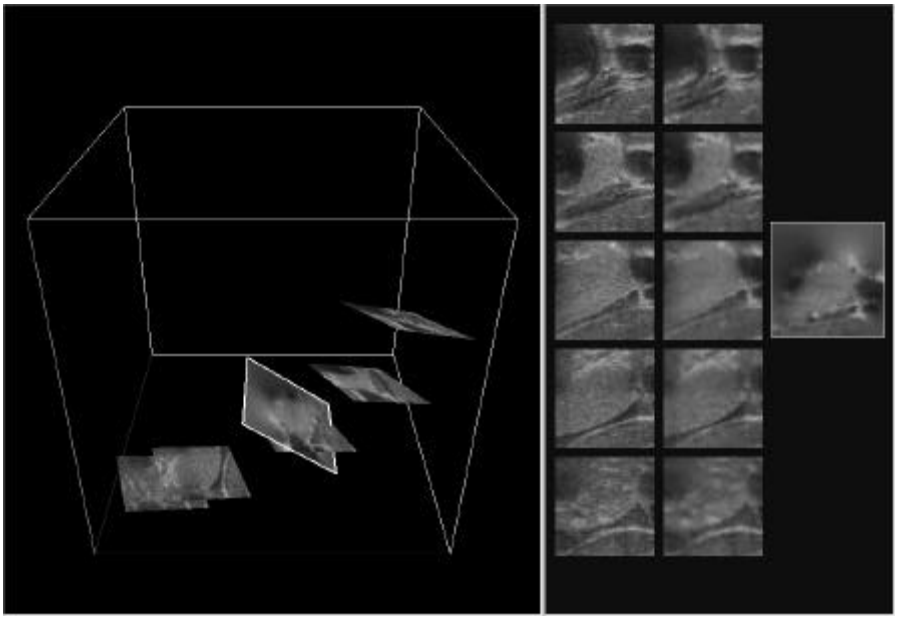


Figure 2.18. The same information as in the previous figure, but the RMS is 9.94 and the number of vertices is 53,995 (approximately a 20 to 1 reduction). The quality of the reconstructed images is excellent!

Research Issues: These results show the promise and potential of progressive models for this type of data and for this reason they are exciting. Is it possible to develop very fast and efficient algorithms that will operate in real time? Imagine an environment where a user sees the results of the volume model as the data is being collected. If a region is of special concern, the probe can be positioned so as to collect more and more data in this region resulting in better and better fitting models. This type of performance will require efficient data structures for the tetrahedralisation and efficient means to compute the coefficients of the volume model. What is the best subdivision strategy? The results we just described (Figure 2.7 and Figures 2.17-2.21) are based upon a particular 3D version of the Maubach algorithm [21]. We previously mentioned two other possibilities: (1) The red/green strategy of Bey [22] and Banks [23] and (2) the idea of using triangular Coons patches. Are there others, and what special properties do they have? What is the best oracle or fitting strategy? Top-Down/Adaptive or Bottom-Up/Collapsing? Within either general strategy there are choices to be made. For example, how do you decide which cells to split or collapse? For some splitting strategies and applications, it may be advantageous to go very deep and for others there are reasons for keeping the overall meshing more uniform and shallow. The results reported here use a piece-wise linear model. Is it worth it to use higher order functions? Second order, for example? We suspect that the savings in the total number of tetrahedra will indicate that this is a wise decision for some applications. □

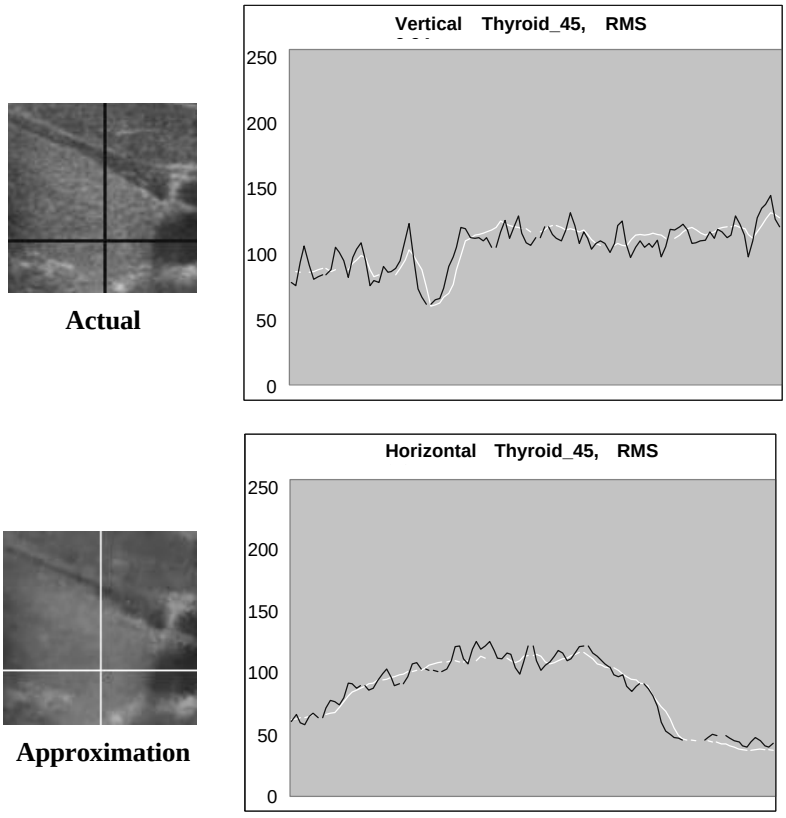


Figure 2.19. Comparison of actual B-scan and approximation.

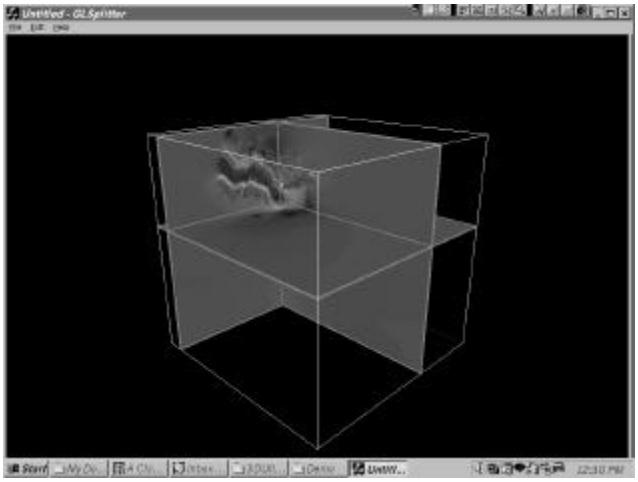


Figure 2.20. Using a slice tool to interactively view a volume model of ultrasound data.

2.4 Conclusions

In this chapter, we have presented a definition of volume modelling, made an appeal for its general development and covered some basic methods of representing volume models. The methods covered are only a sampling. Many techniques have not been covered. For example there have been a number of procedural techniques developed where the primary goal is to generate an image or animation which is acceptable to the viewer. In these applications the picture is the main goal and the volume model is not so important. Fire, gases, clouds, fluids and many other phenomena have been considered. Discussion of these procedural techniques can be found in [27] and the references therein. Also, we did not cover the very interesting and potentially very useful topic of layered manufacturing (see Chapter 5). Volume models are needed to drive these new and interesting methods of fabrication, for instance, the use of transfinite deformation maps for describing volume models [28].

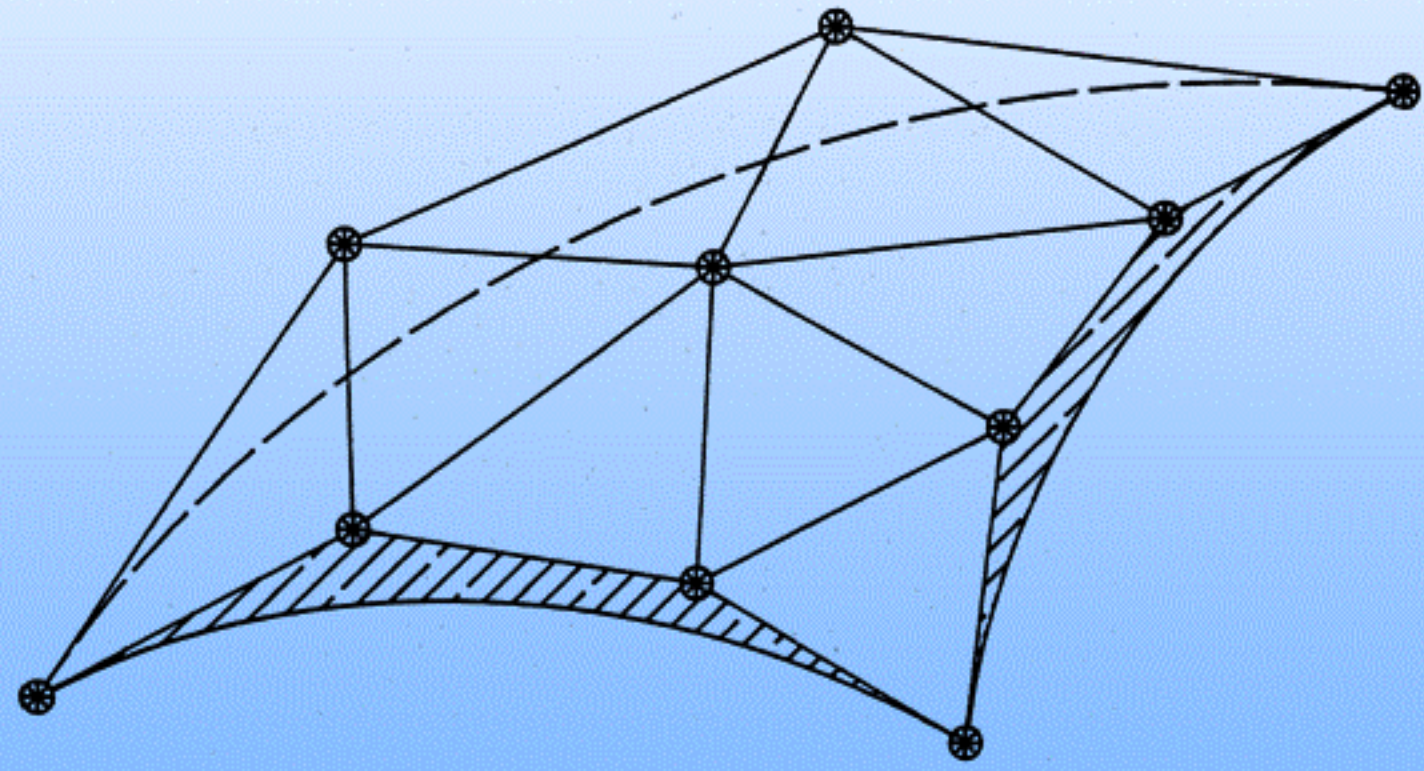
References

1. Kaufman K, Cohen D, Yagel R. Volume graphics. IEEE Computer 1993; 26(7):51-64.
2. Kenwright D, Kao D. Optimization of time-dependent particle tracing using tetrahedral decomposition. In: Proc. IEEE Visualization'95, Atlanta, GA, October 1995; 321-328.
3. Nelson TR. Ultrasound visualization. Advances in Computers 1998; 47:185-253.
4. Fenster A, Downey DB. 3-D ultrasound imaging — a review. IEEE Engineering in Medicine and Biology Magazine 1996; 15(6):41-51.
5. Rohling RN, Gee AH, Berman L. Radial Basis Function Interpolation for 3-D Ultrasound. TR 327, Engineering Department, Cambridge University, UK, 1998.
6. Wang SM, Kaufman A. Volume sampled voxelization of geometric primitives. In: Proc. IEEE Symposium on Volume Visualization, Los Alamos, CA, October 1993; 78-84.
7. Wang S, Kaufman A. Volume sculpting. In: Proc. Symposium on Interactive 3D Graphics, April 1995; 151-156.
8. Nielson GM. Scattered data modeling. IEEE Computer Graphics and Applications 1993; 13(1):60-70.
9. Sederberg T, Parry S. Free-form deformation of solid geometric models. ACM/SIGGRAPH Computer Graphics 1986, 20(4): 151-160.
10. MacCracken R, Joy K. Free-form deformations with lattices of arbitrary topology. ACM/SIGGRAPH Computer Graphics 1996; 30(4):181-188.
11. Chui CK. An Introduction to Wavelets. Academic Press, San Diego, CA 1992.
12. Daubechies I. Ten Lectures on Wavelets. CBMS-NSF Regional Conference Series in Appl. Math. 1992; vol. 61, SIAM, Philadelphia, PA, 1992.
13. Stollnitz E, DeRose A, Salesin D. Wavelets for Computer Graphics, Morgan Kaufman, San Francisco, 1996.
14. Muraki S. Volume data and wavelet transforms. IEEE Computer Graphics and

Applications 1993; 13(4): 50-56.
15. Muraki S. Multiscale volume representation by a DoG wavelet. Transactions on Visualization and Computer Graphics 1995; 1(2):109-116.
16. Bonneau GP, Hahmann S, Nielson GM. BlaC wavelets: a multiresolution analysis with non-nested spaces. In: Proc. IEEE Visualization'96, San Francisco, CA, October 1996; 43-48.
17. Nielson GM, Jung I, Sung J. Wavelets over curvilinear grids. In: Proc. of IEEE Visualization'98, Research Triangle Park, NC, October 1998; 313-317.
18. Zhou Y, Chen B, Kaufman A. Multiresolution Tetrahedral Framework for Visualising Regular Volume Data. In: Proc. IEEE Visualization'97, Phoenix, AZ, October 1997; 135-142.
19. Trotts I, Hamann B, Joy K, Wiley D. Simplification of tetrahedral meshes with error bounds. To appear in TVCG, 1999.
20. Nielson GM. Tools for triangulations and tetrahedrizations, In: Nielson, Hagen, Mueller (eds). Scientific Visualization: Surveys, Techniques and Methodologies. IEEE CS Press, 1997; 429-525.
21. Maubach JM. Local bisection refinement for N-simplicial grids generated by reflection. SIAM J. Sci. Compt.; 16(1):210-227.
22. Bey J. Tetrahedral mesh refinement. Computing 1995; 55(13):355-378.
23. Bank RE, Sherman AH, Weiser A. Refinement algorithms and data structures for regular local mesh refinement. In: Stepleman R (ed), Scientific Computing, North Holland, Amsterdam, 1983; 3-17.
24. Grosso R, Luerig C, Ertl T. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. In: Proc. IEEE Visualization'97, Phoenix, AZ, October 1997; 387-394.
25. Coon SA. Surfaces for Computer-Aided Design of Space Forms. MIT, MAC TR-41, June 1967.
26. Nielson GM, Holliday D, Rox Roxborough T. Cracking the cracking problem with Coons patches. To appear in: Proc. IEEE Visualization'99, San Fancisco, CA, 1999.
27. Ebert D, Musgrave K, Peachy D, Worley S, Perlin K. Texturing and Modeling: A Procedural Approach. Academic Press, San Diego, CA, 1998.
28. Qian X, Dutta D. Features in layered manufacturing of heterogeneous objects. In: Proc. SFFS 98, Austin, Texas, 1998.

Surface Reconstruction

a. Triangular Patches

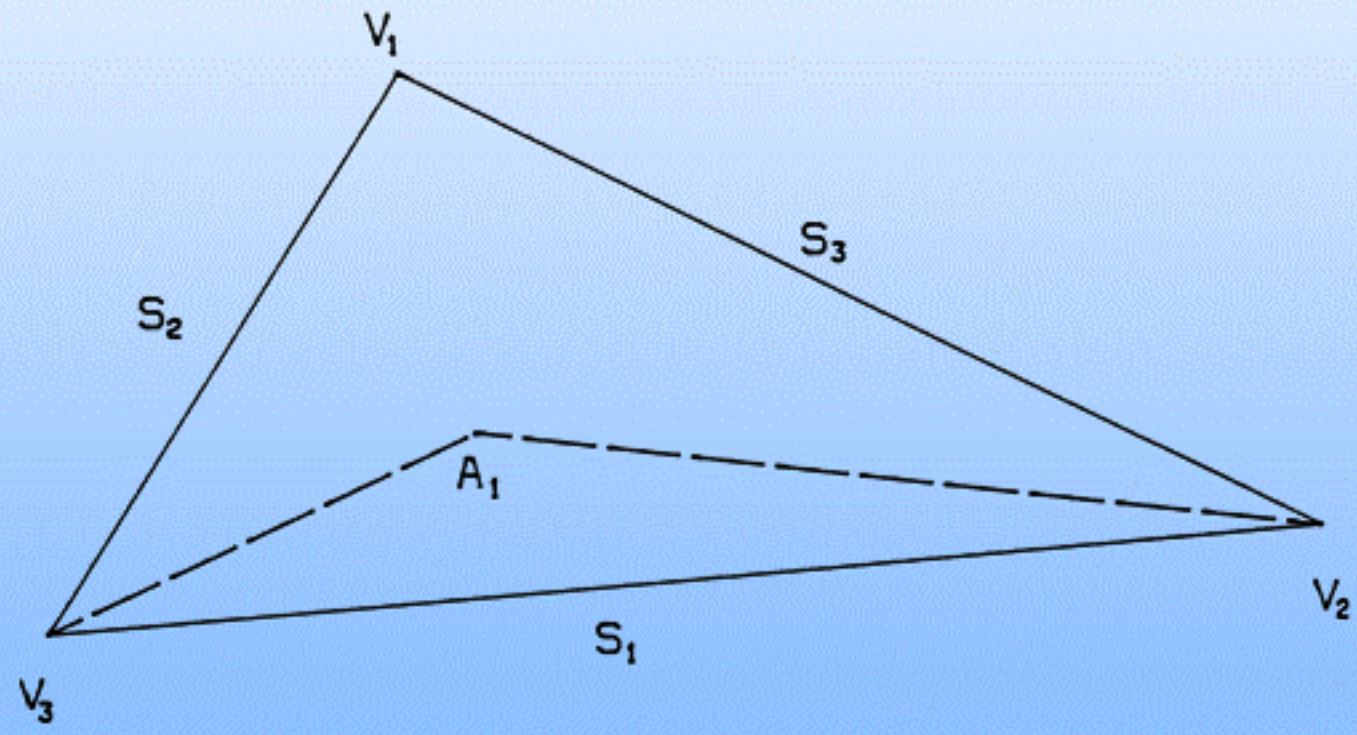


UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

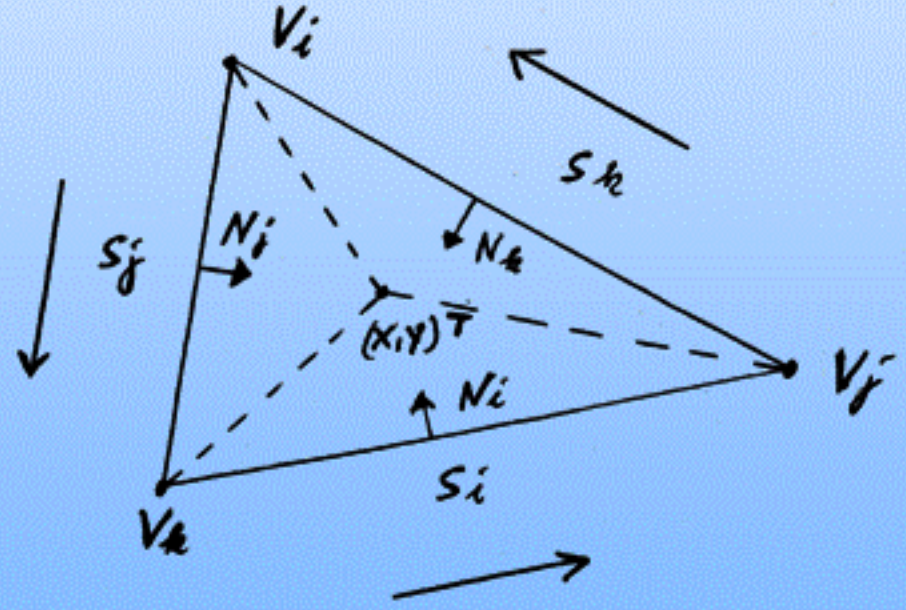
barycentric coordinates

$$b_i := \frac{A_i}{A} \qquad i = 1, 2, 3$$



barycentric coordinates

$$b_i(x,y) := \frac{\begin{vmatrix} x - x_j & x - x_k \\ y - y_j & y - y_k \end{vmatrix}}{\begin{vmatrix} x_i - x_j & x_i - x_k \\ y_i - y_j & y_i - y_k \end{vmatrix}}, \qquad i = 1,2,3 \qquad b_i = \frac{A(P,V_j,V_k)}{A(V_1,V_2,V_3)}$$



3

$$\sum_{i=1} b_i = 1$$

$$\frac{\partial\, b_k}{\partial\, s_k} = 0\,; \quad k = 1,2,3$$

$$\frac{\partial\, b_j}{\partial\, s_k} = \pm 1\,; \quad j,k = 1,2,3; \quad j \neq k$$

$$\frac{\partial\, b_i}{\partial\, N_i} = 1\,; \quad i = 1,2,3$$

$$\frac{\partial\, b_j}{\partial\, N_i} = \frac{<s_j, s_i>}{||s_i||^2}\,; \quad k = 1,2,3$$

b. Beziér Techniques

Bézier - curves

$$X_l(u) := \sum_{i=0}^{m} b_{lm+i} B_i^m\left(\frac{u-u_l}{u_{l+1}-u_l}\right)$$

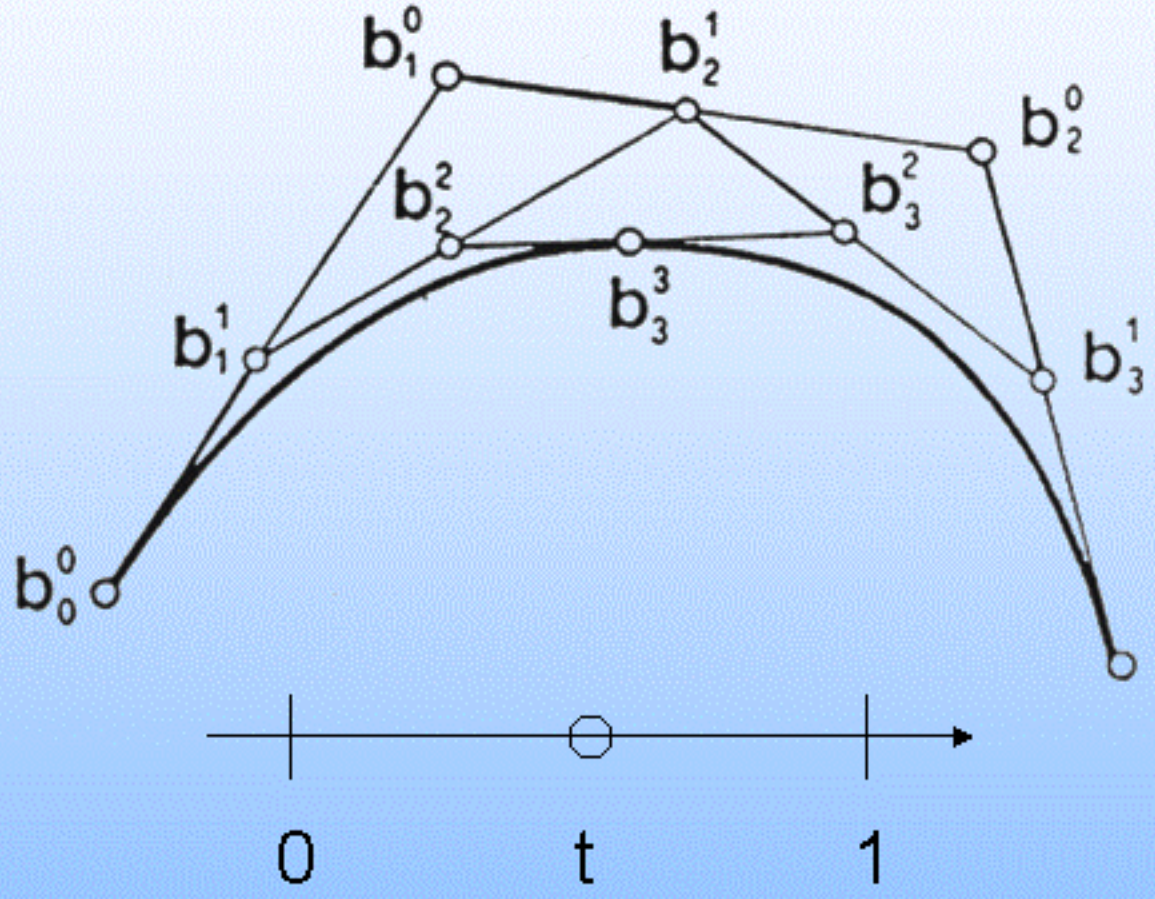
Bernstein - polynomials

$$B_i^m(t) := \binom{m}{i}(1-t)^{m-i}t^i$$

$$0 \le t \le 1$$

as blending functions

Rational Bézier - curves $\quad X(t) = \sum b_i \frac{\lambda_i B_i^m(t)}{\sum \lambda_i B_i^m(t)}$



b_0^0

b_1^1

b_1^0 b_2^2

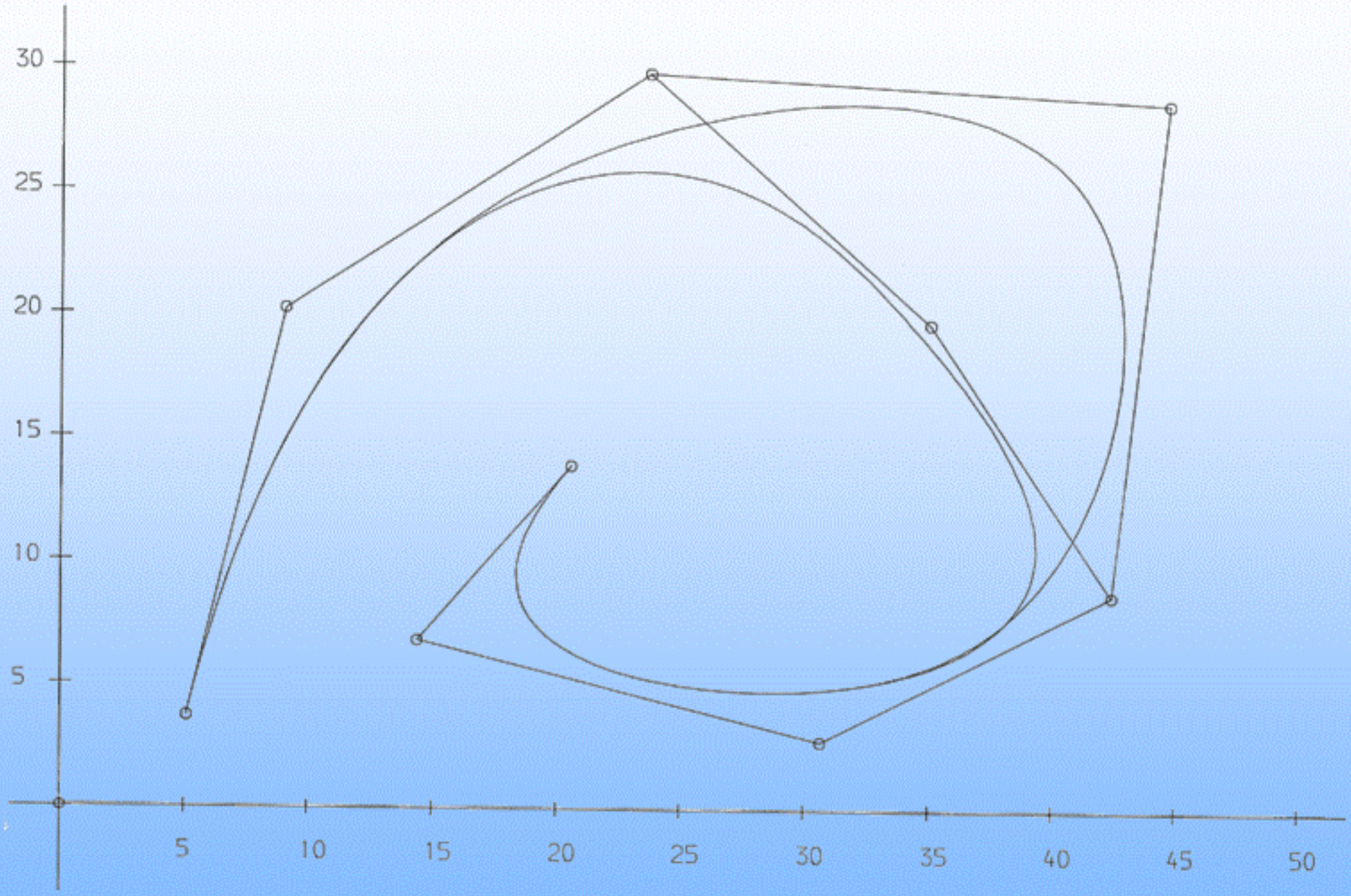
b_2^1 $b_3^3 = x(u)$

b_2^0 b_3^2

b_3^1

b_3^0

De Casteljau algorithm and scheme, $n = 3$



UNIVERSITÄT
KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

Bézier – surfaces

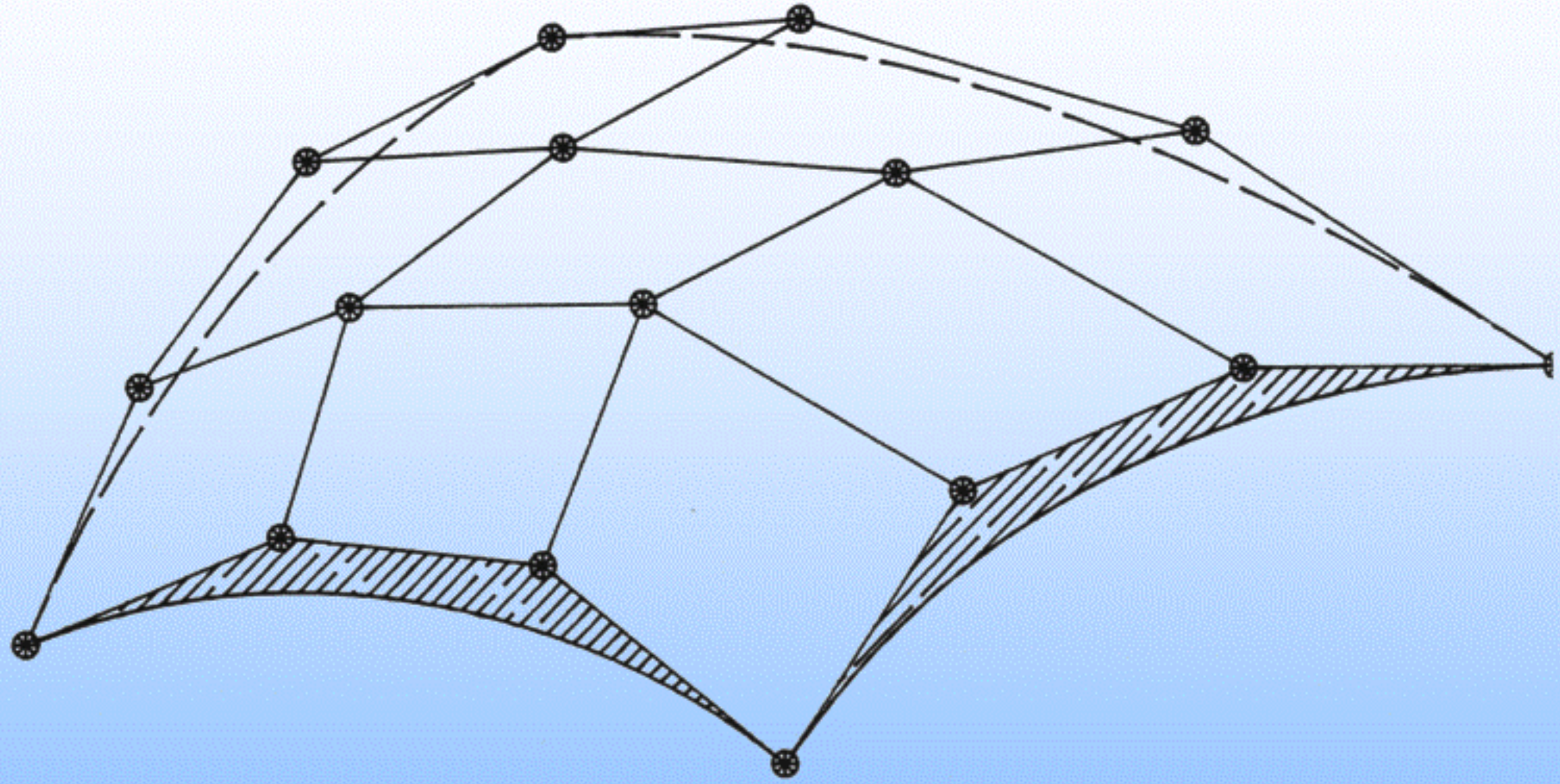
$$X_{pq}(u,w) := \sum_{i=0}^{m}\sum_{j=0}^{n} b_{pm+i\,/\,qn+j} B_i^m\left(\frac{u-u_p}{u_{p+1}-u_p}\right) B_j^n\left(\frac{w-w_q}{w_{q+1}-w_q}\right)$$

Rational Bézier - surfaces

$$X(u,w) := \sum\sum b_{ij} \frac{\lambda_{ij} B_i^n(u) B_j^m(w)}{\sum\sum \lambda_{ij} B_i^n(u) B_j^m(w)}$$

UNIVERSITÄT KAISERSLAUTERN

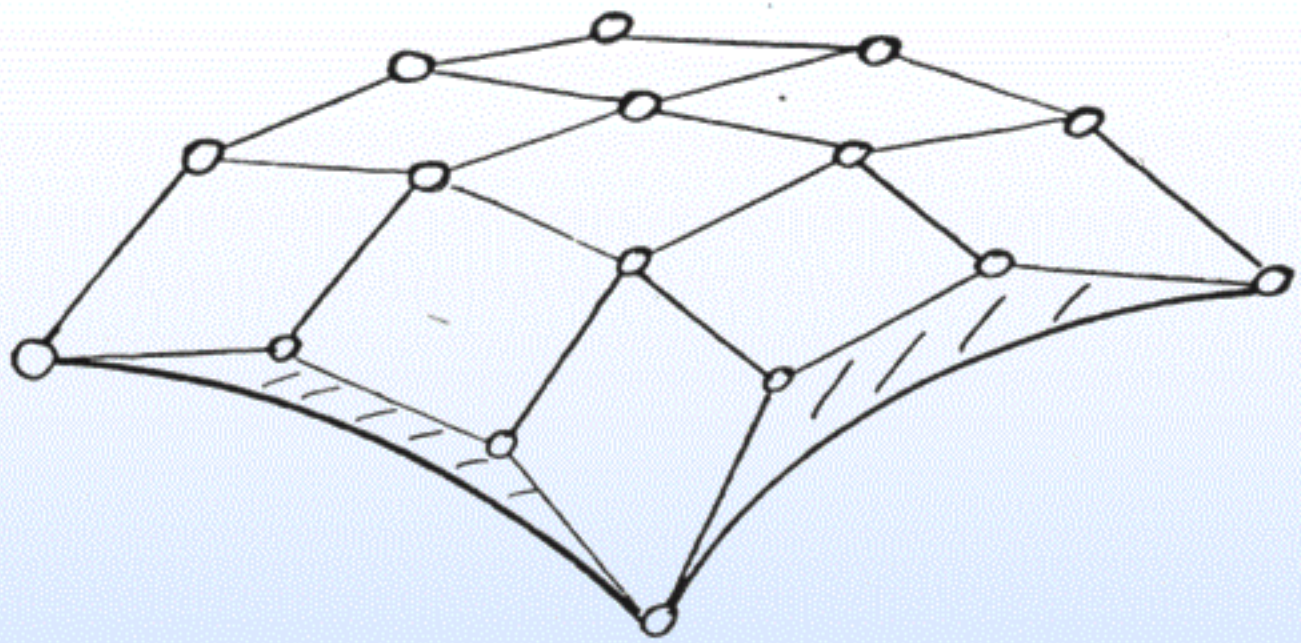
Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

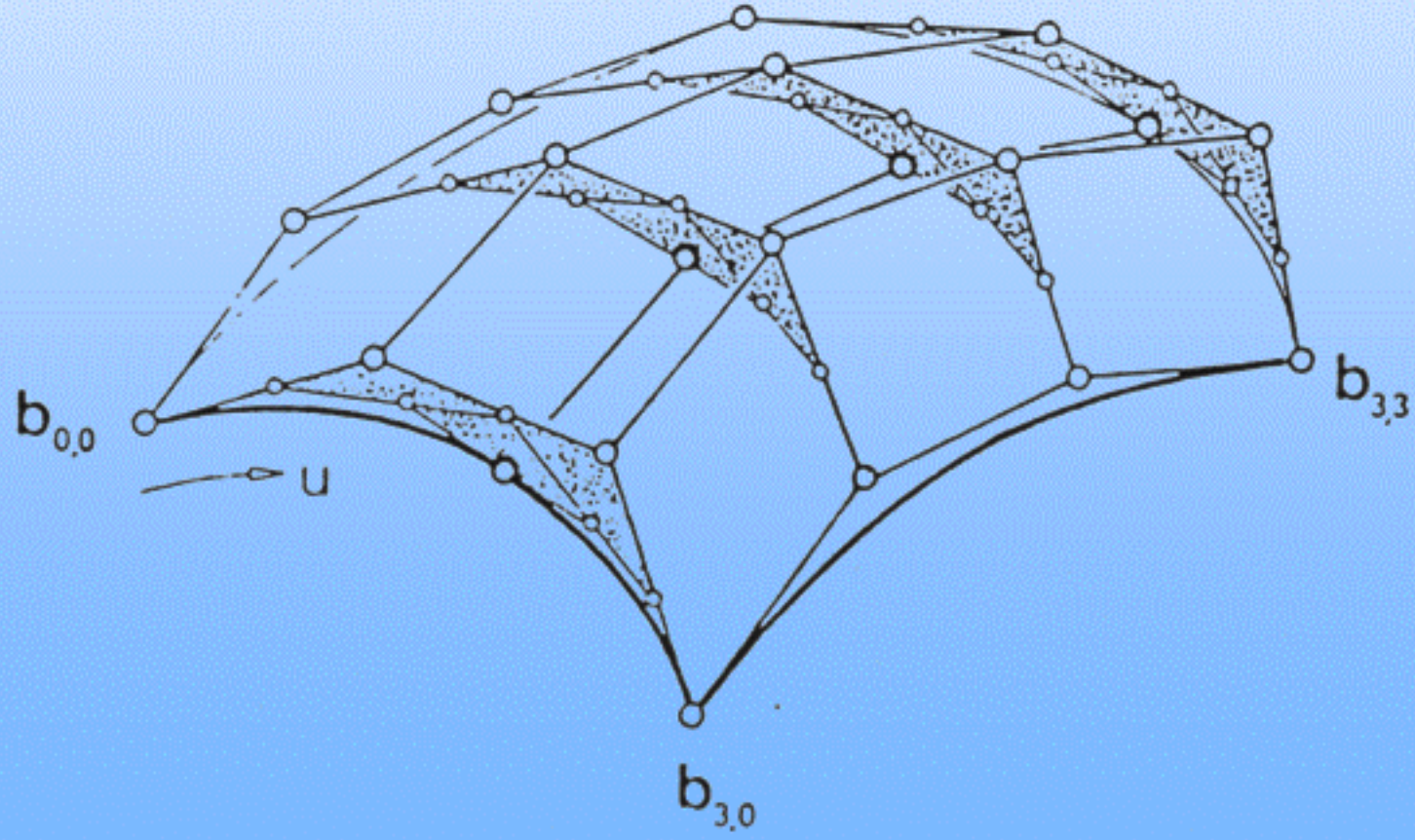


Bézier-patch

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie



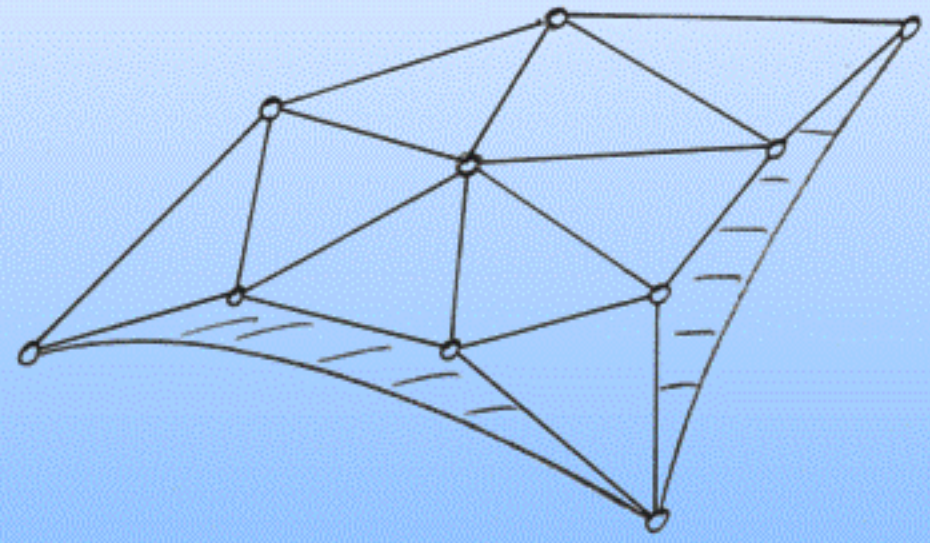


UNIVERSITÄT KAISERSLAUTERN
Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

triangular Bézier - patches

$$X(u,v,w) := \sum_{i,j,k=0}^{i+j+k=n} b_{i,j,k} B^n_{i,j,k}(u,v,w)$$

$$0 \le u,v,w \le 1 \quad \text{and} \quad u+v+w = 1$$

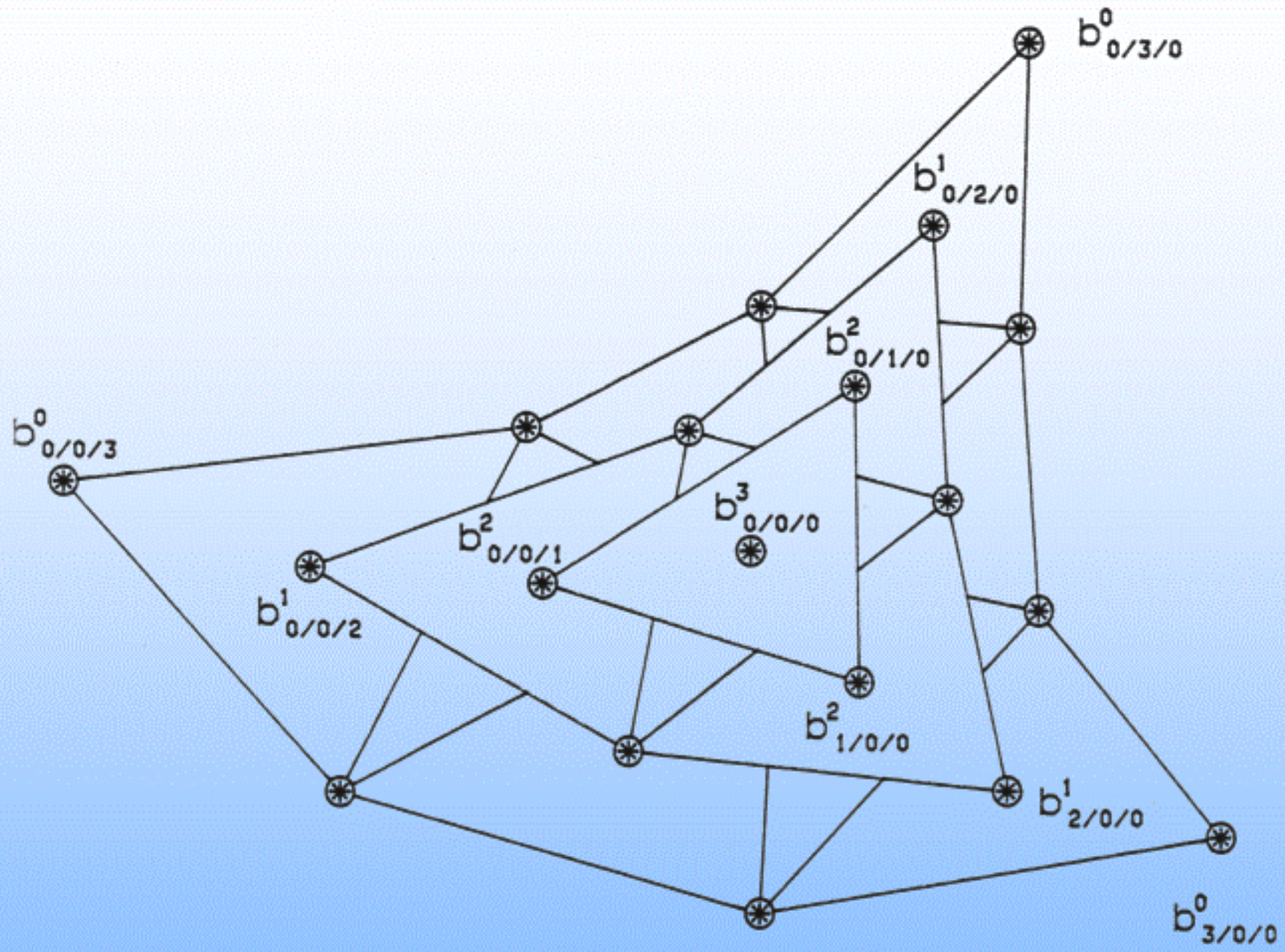


$$B^n_{i,j,k}(u,v,w) = \frac{n!}{i!\,j!\,k!} u^i v^j w^k$$

Bernsteinpolynomials

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

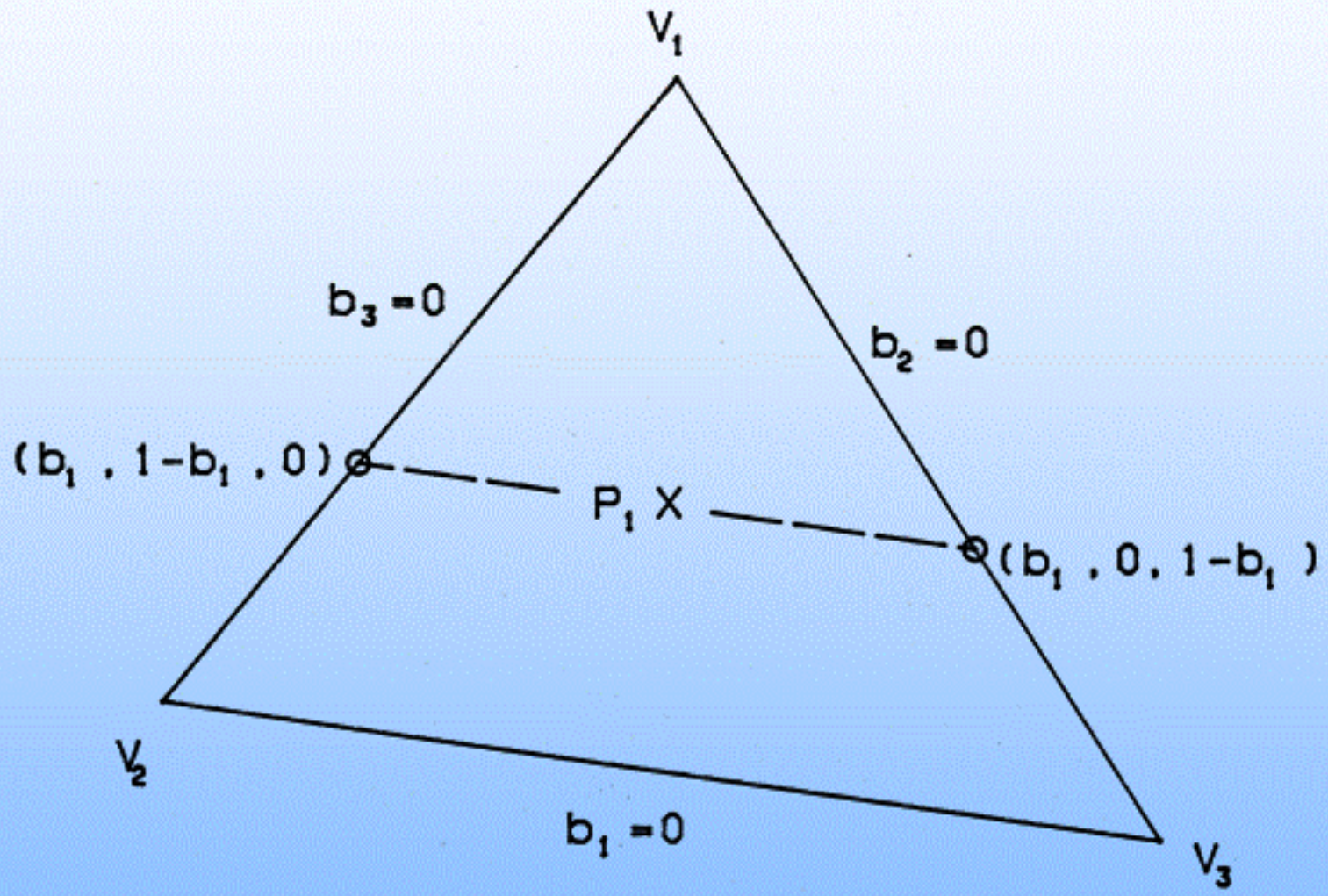


Triangular Bézier-patch-algorithm

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

transfinite methods



parallel projectors

Triangular Interpolation schemes with parallel projectors

Barnhill – Birkhoff – Gordon and Barnhill – Gregory

Compatibly corrected BBG – scheme :

$$\mathbf{X}(\mathbf{u},\mathbf{w}) := \left[\mathbf{P_3} \oplus (\mathbf{P_1} \oplus \mathbf{P_2})\right] \mathbf{X}$$

where

$$P_1X := H_0\left(\frac{b_3}{1-b_1}\right) X(P_0) + H_1\left(\frac{b_3}{1-b_1}\right) X(P_1)$$

$$+\bar{H}_0\left(\frac{b_3}{1-b_1}\right)(1-b_1)\,\frac{\partial X}{\partial e_1}(P_0) + \bar{H}_1\left(\frac{b_3}{1-b_1}\right)(1-b_1)\,\frac{\partial X}{\partial e_1}(P_1)$$

$$P_0 := b_1V_1 + (1-b_1)V_2; \quad P_1 := b_1V_1 + (1-b_1)V_3$$

P_2X and P_3X analogously defined

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

Convex combination schemes (Gregory)

$$
\begin{aligned}
P_i[F] \quad := \quad & F(S_j^k) + b_j F_{n_j}(S_j^k) + F(S_k^j) + b_k F_{n_k}(S_k^j) \\
& -F(V_i) - b_j F_{n_j}(V_i) - b_k F_{n_k}(V_i) - b_j b_k F_{n_j n_k}(V_i)
\end{aligned}
$$

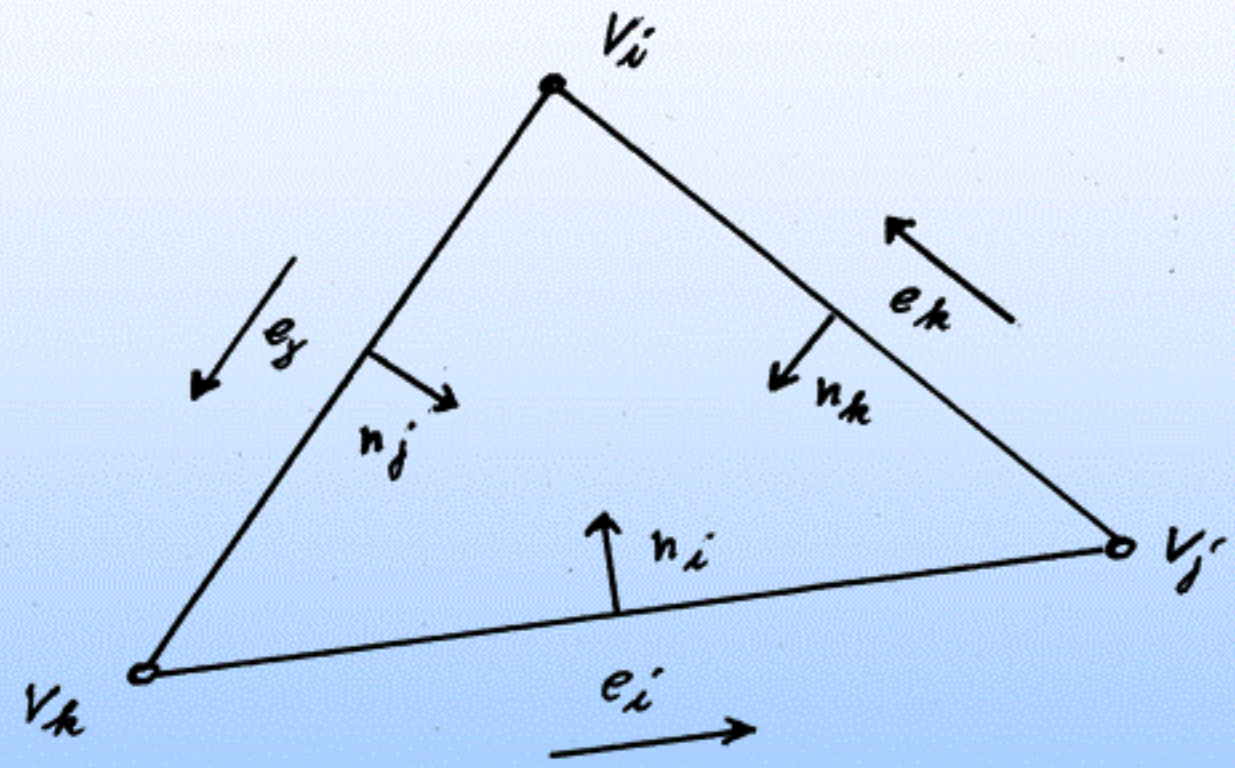
where

$$S_j^k := (1 - b_k)V_i + b_k V_k$$

$$\mathbf{P(V)} := \sum_{\mathbf{i=1}}^{\mathbf{3}} \alpha_{\mathbf{i}}(\mathbf{V})\mathbf{P_i(V)}$$

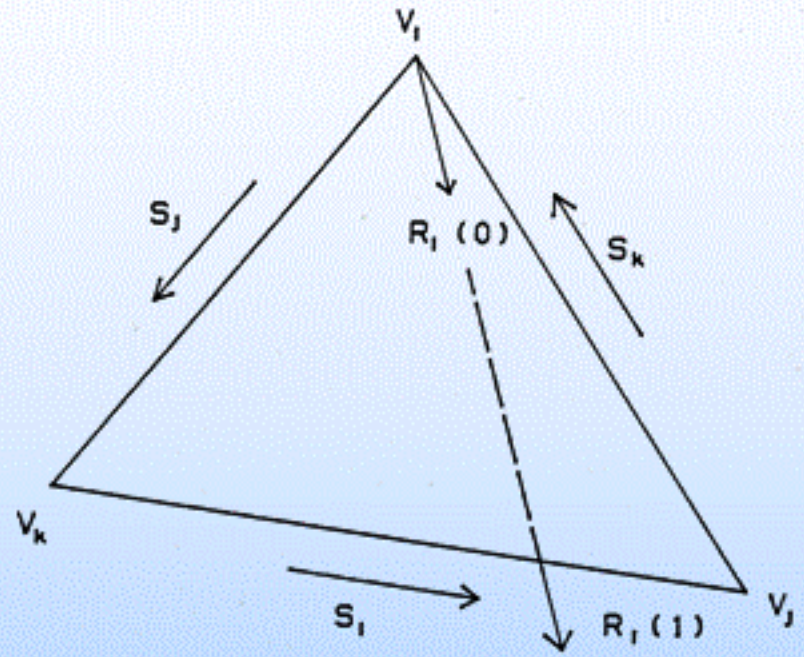
where

$$\alpha_i(V) := b_i^2(3 - 2b_i + 6b_j b_k)$$



$$S_i^j := b_j V_j + (1 - b_j) V_k \qquad S_i^k := (1 - b_k) V_i + b_k V_k$$

side vertex methods (Nielson)



point opposite the vertex V_i :

$$S_i(x,y) = \left(\frac{x - x_i b_i}{1 - b_i}; \frac{y - y_i b_i}{1 - b_i}\right); \quad i = 1,2,3$$

C^0 – side vertex method :

$$\mathbf{A(F)} := \sum_{\mathbf{i=1}}^{\mathbf{3}} [(\mathbf{1 - b_i})\mathbf{F(S_i)} - \mathbf{b_i F(V_i)}]$$

applying the Hermite operator

$$H[g](t) := H(t)g(1) + \bar{H}(t)g'(1) + H(1-t)g(0) - \bar{H}(1-t)g'(0)$$

to the radial operator

$$R_i(t) := F(tS_i + (1-t)V_i); \qquad i = 1, 2, 3$$

we can define

$$D_i[F] = H(1-b_i)F(S_i) + \bar{H}(1-b_i)R_i'(1) + H(b_i)F(V_i) - \bar{H}(b_i)R_i'(0)$$

using convex combinations we get

$$D(F) := \frac{b_2^2 b_3^2 D_1[F] + b_1^2 b_3^2 D_2[F] + b_1^2 b_2^2 D_3[F]}{b_2^2 b_3^2 + b_1^2 b_3^2 + b_1^2 b_2^2}$$

curvature continuous triangular patches (Hagen)

geometric Hermite-operator :

$$GH(y) := \sum_{i=0,1} H_i(t)y(i) + \bar{H}_i(t)y'(i) + G_i(t)\left[[y'(i), y''(i)], y'(i)\right]$$

$$\left(G_i(t) := \frac{\bar{H}_i(t)}{2\|y'(i)\|} \quad i = 1, 2\right)$$

$$[[y'(i), y''(i)], y'(i)] = \|y'\|^4 \left(k_N \dot{N} + k_g [N, y']\right)$$

k(t) curvature of the space curve y(t)

k_N(t) normal section curvature, k_g geodesic curvature

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

combing the geometric Hermite-operator with tha radial operator

$$R_i(t) = F(tS_i + (1-t)V_i)\ ; \quad i = 1,2,3$$

we can define

$$\begin{aligned} P_i[F] \ := \ & H_0(1-b_i)F(V_i) + H_1(1-b_i)F(S_i) \\ + \ & \bar{H}_0(1-b_i)R'(0) + \bar{H}_1(1-b_i)R_i'(1) \\ + \ & G_0(1-b_i)\left[[R_i'(0), R_i''(0)], R_i'(0)\right] \\ + \ & G_1(1-b_i)\left[[R_i'(1), R_i''(1)], R_i'(1)\right] \end{aligned}$$

$$\left[[R_i'(0), R_i''(0)], R_i'(0)\right] = \|R_i'(0)\|^4 \left(k_N(V_i)N(V_i) + k_g(V_i)[N(V_i), R_i'(0)]\right)$$

$$\left[[R_i'(1), R_i''(1)], R_i'(1)\right] = \|R_i'(1)\|^4 \left(k_N(S_i)N(S_i) + k_g(S_i)[N(S_i), R_i'(1)]\right)$$

using convex combinations we get

$$P(F) := \frac{b_2^2 b_3^2 P_1[F] + b_1^2 b_3^2 P_2[F] + b_1^2 b_2^2 P_3[F]}{b_2^2 b_3^2 + b_1^2 b_3^2 + b_1^2 b_2^2}$$

The user can supply N, k_N and k_g along each side and these data are reproduced exactly

Triangular Patches with first and second order geometric continuity

Nielson – Hagen – Pottmann

Nielson's G^1 – patch interpolates surface normals and is based upon

$$g_1[V_0, V_1, N_0, N_1](t) := H_0(t)V_0 + H_1(t)V_1$$

$$+\alpha\bar{H}_0(t)\frac{[N_0, V_1 - V_0, N_1]}{\|[N_0, V_1 - V_0, N_1]\|} + \beta\bar{H}_1(t)\frac{[N_0, V_1 - V_0, N_1]}{\|[N_0, V_1 - V_0, N_1]\|}$$

with

$$g(i) = V_i \ \langle g'(i), N_i \rangle = 0; \ i = 1, 2$$

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

combining this operator with the radial operator we can define

$$G_i[F](b_1,b_2,b_3) = g[F(V_i), F\left(\frac{b_jV_j+b_kV_k}{1-b_i}\right), N[F](V_i), N[F]\frac{b_jV_j+b_kV_k}{1-b_i}](1-b_i)$$

using convex combinations we get:

$$G[F] = W_1G_1[F] + W_2G_2[F] + W_3G_3[F]$$

where

$$W_i = \frac{b_j^2b_k^2}{b_1^2b_2^2 + b_2^2b_3^2 + b_3^2b_1^2}$$

G^2 – patch (Hagen – Pottmann)

geometric extension of the quintic Hermite-operator

$$H_5[V_0, V_1, V_0', V_1', V_0'', V_1''](t) =$$
$$H_0^5(t)V_0 + H_1^5(t)V_0' + H_2^5(t)V_0'' + H_3^5(t)V_1'' + H_4^5(t)V_1' + H_5^5(t)V_1$$

$$V_0' = \lambda_0 T_0 \qquad V_0'' = \lambda_0^2 k_0 + \mu_0 T_0 \qquad \lambda_0\lambda_1 > 0$$
$$V_1' = \lambda_1 T_1 \qquad V_1'' = \lambda_1^2 k_1 + \mu_1 T_1 \qquad \lambda_0\lambda_1, \mu_0\mu_1 \in I\!R$$

T_0, T_1 unit tangent vectors
$k_i = \kappa_i e_2(V_i)\ i = 1, 2$ curvature vectors

UNIVERSITÄT KAISERSLAUTERN
Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

$$g[V_0, V_1, C_0, C_1](t) := H_5[V_0, V_1, \lambda_0 T_0, \lambda_1 T_1, \lambda_0^2 k_0 + \mu_0 T_0, \lambda_1^2 k_1 + \mu_1 T_1](t)$$

applying this operator to the radial operator:

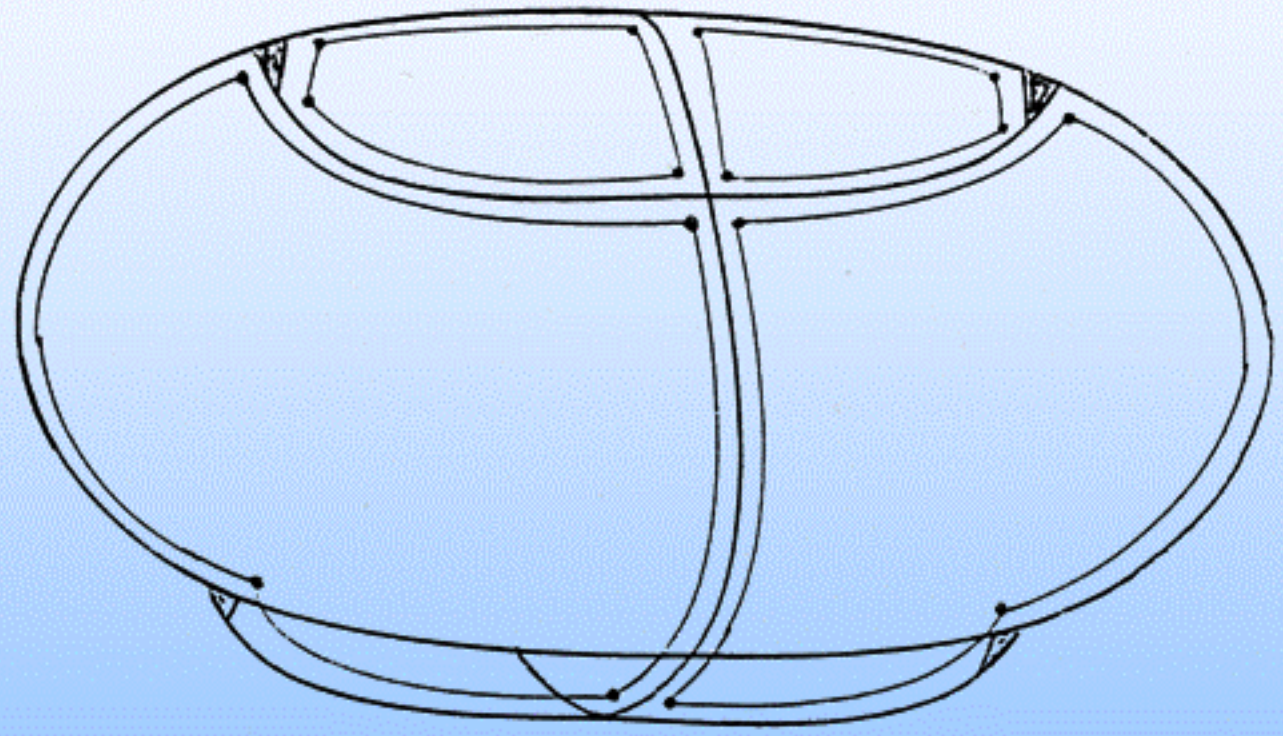
$$\begin{aligned} P_i[F](b_1, b_2, b_3) &= g[F(V_i), F\left(\frac{b_j V_j + b_k V_k}{1 - b_i}\right), C[F](V_i), \\ & \qquad C[F]\frac{b_j V_j + b_k V_k}{1 - b_i}](1 - b_i) \end{aligned}$$

C_0, C_1 "curvature elements" at V_0, V_1

$$P[F] = W_1 P_1[F] + W_2 P_2[F] + W_3 P_3[F]$$

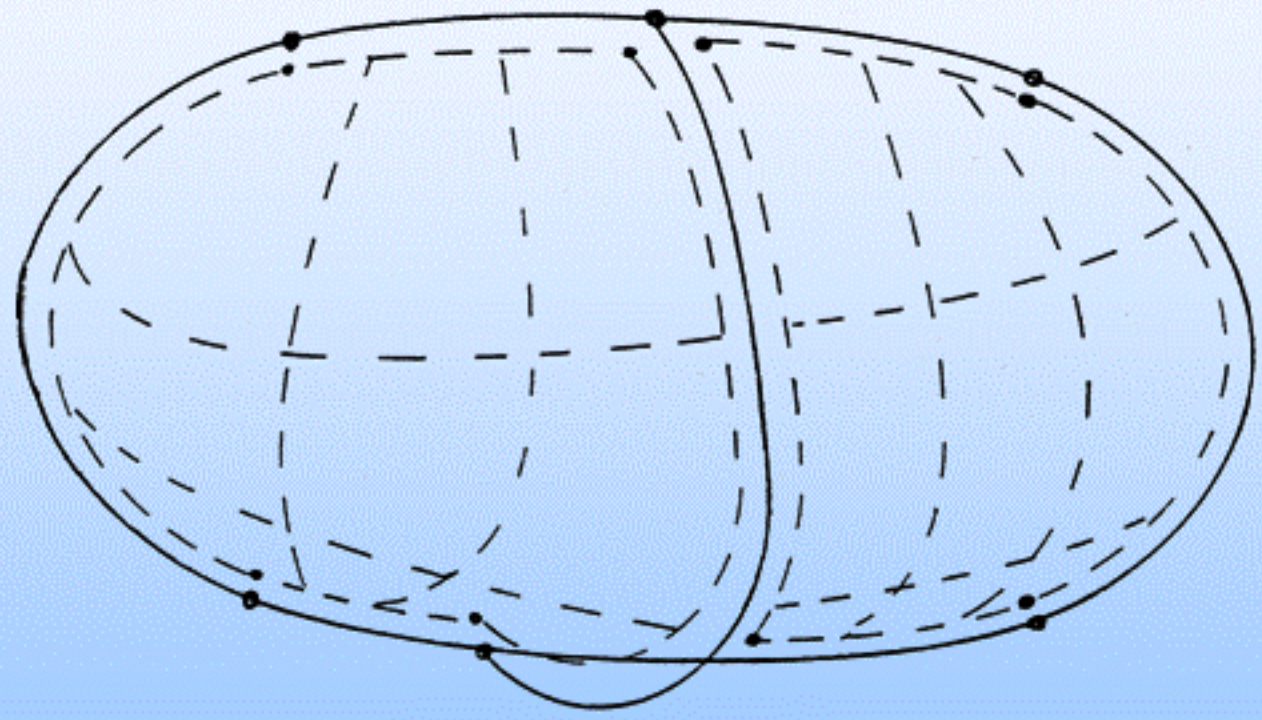
where

$$W_i = \frac{b_j^2 b_k^2}{b_1^2 b_2^2 + b_2^2 b_3^2 + b_3^2 b_1^2}$$



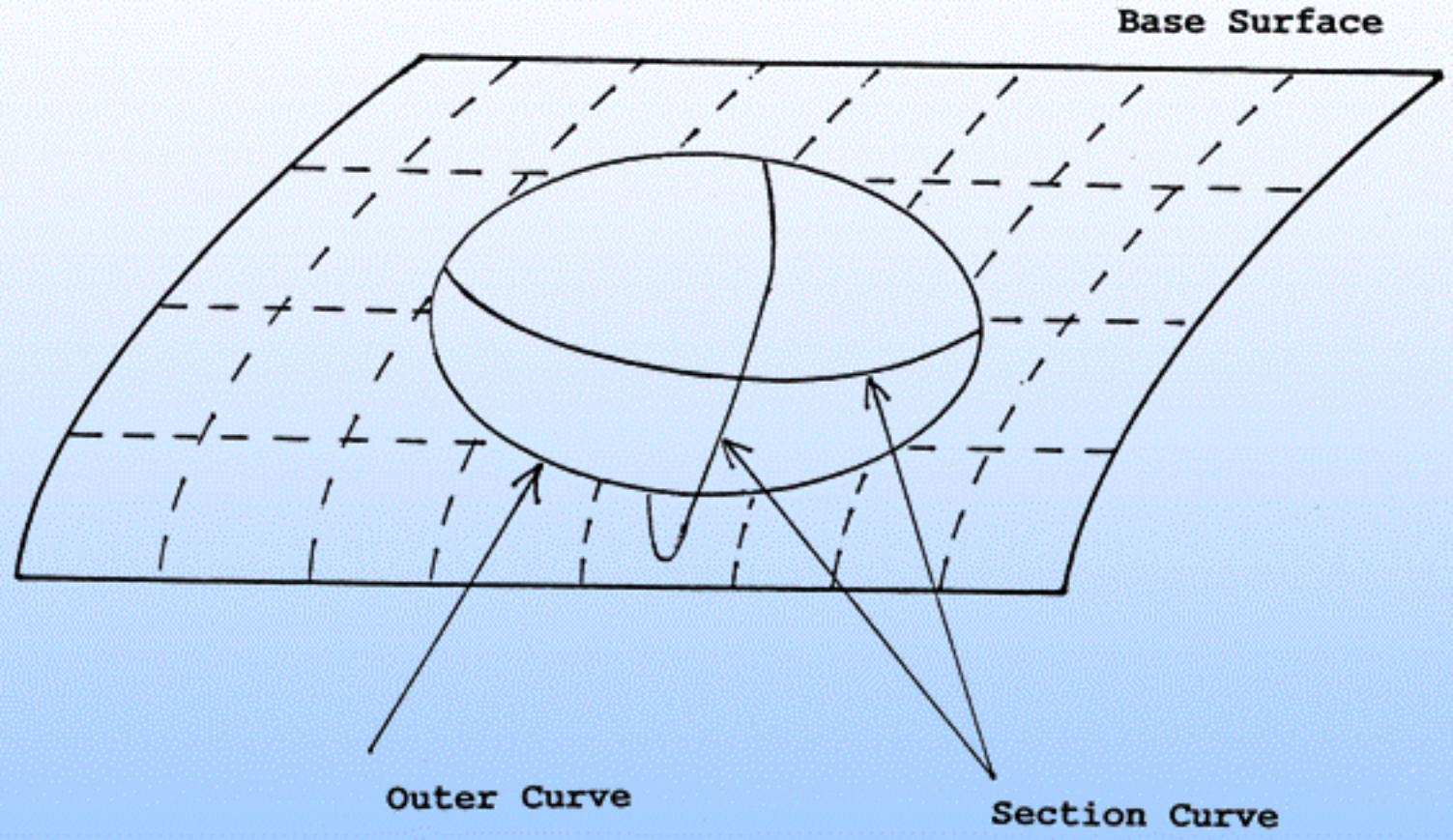
Hollow surface construction

(6 surfaces)

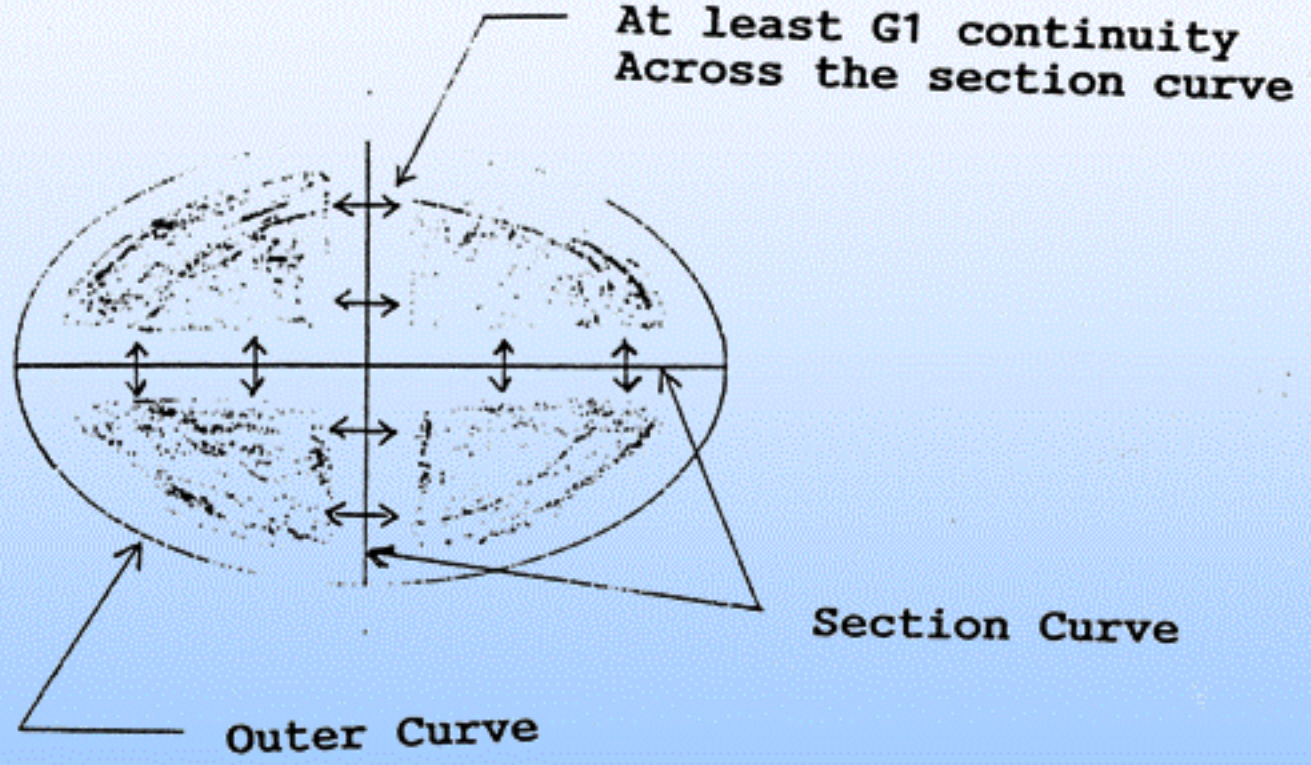


Hollow surface construction

(2 surfaces)



Trditional construction of hollow surface



Four triangular surface construction

Simulation Based Modelling

data generation:	measurements	➔ **large unstructured**
data reception:	numerical simulations	data sets
data enrichment:	filtering	
and improvement:	clustering	
data analysis:	structure recognition	
data reduction:	testing of features representation sets	
modelling:	variational design physically based modelling	
interrogation:	reflection lines, GFS,	
quality analysis:	isophotes, …	
manufacturing process		

UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

Clustering for structuring and data reduction

Clustering means grouping of similar objects by optimizing a certain function or other object dependent properties.

input: a set P of arbitrary distributed data points
$p_1, p_2, \ldots, p_n$ with weights $w_1, w_2, \ldots, w_n$

output: a smaller set Q of cluster centre points
$q_1, q_2, \ldots, q_m$ $(m \leq n)$, which minimizes the least squared errors

$$\sum\sum w_i \left\| p_i - q_j \right\|^2 \rightarrow \min$$

with the index sets:

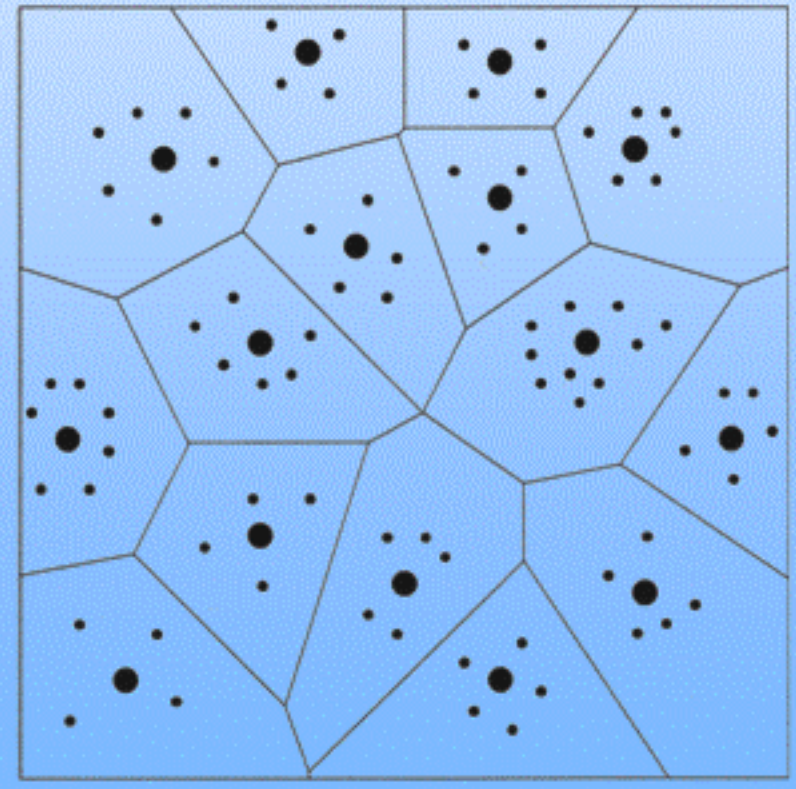
$I_j := \{i \mid p_i$ closer to q_j than to any other $q_k, \forall i \neq k$

Voronoi Diagrams

A multidimensional Voronoi diagram is a partitioning of the space E^d in regions R_j with the following properties: each point q_j lies in exactly one region R_j, which consists of all points x of E^d that are closer to q_j than to any other point q_k ($j \neq k$).

$$R_j = \{x \in E^d : \|x - q_j\| < \|x - q_k\|, j \neq k\}$$

$I_j = \{i : p_i$ lies in region $R_j\}$



UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

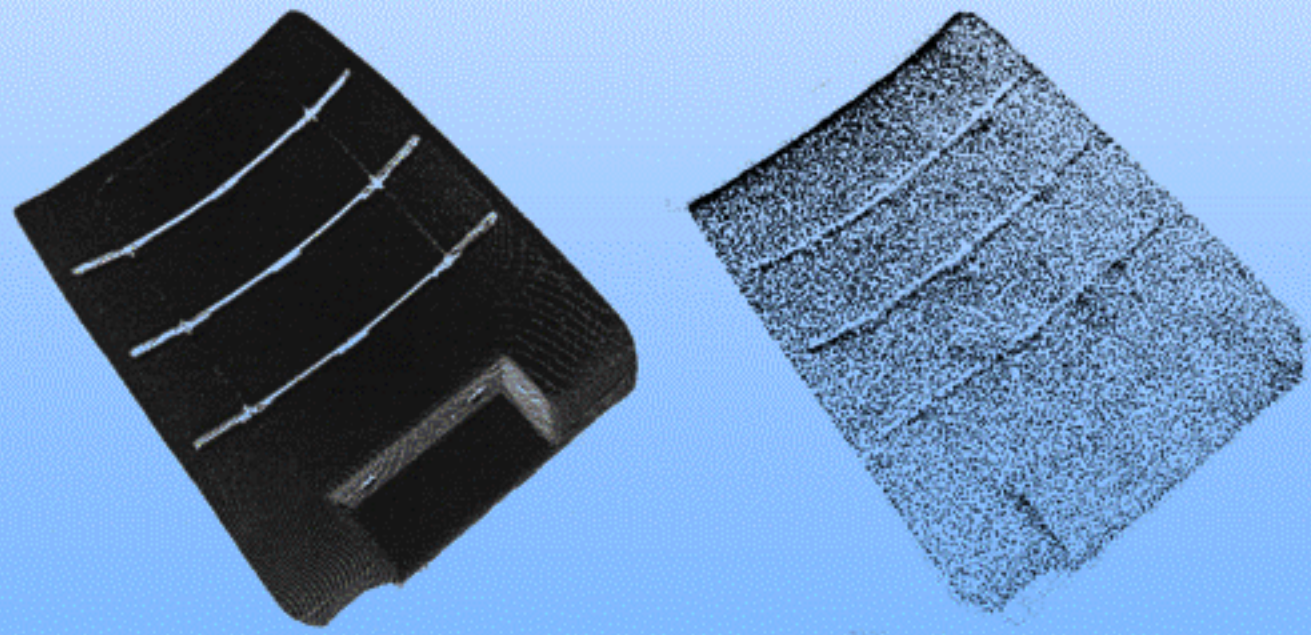
Algorithm

– initialize a starting cluster configuration $q_1, q_2, \ldots, q_m$ with at least one data point for each cluster

– for each cluster ($j = 1, 2, \ldots, m$) do:
move the cluster point q_j in the weighted mean of data points $p_i\ (i \in I_j)$ and update the index sets

– iterate until the clusters don't "change" anymore

Keiper Recaro GmbH & Co KG



300.000 measured points reduced to 10.000 points

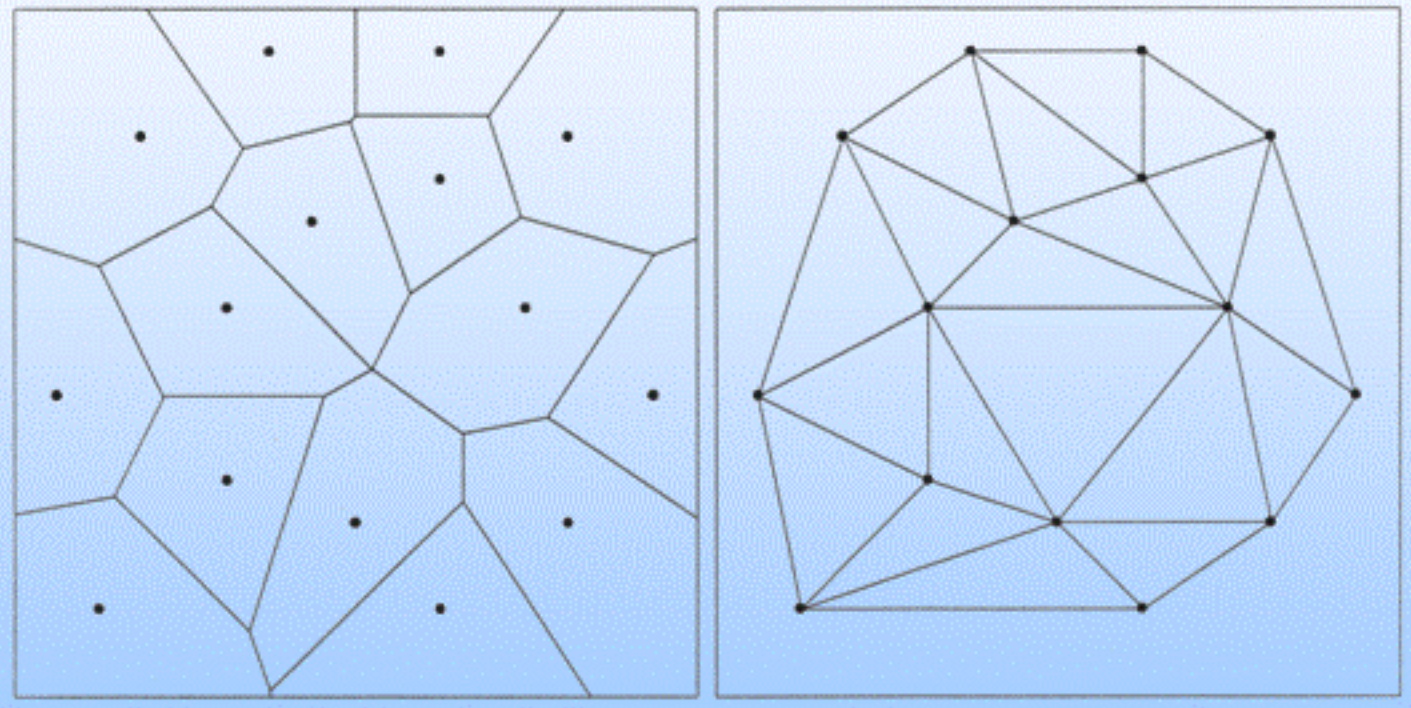
UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

– project the 3D points onto the best fitting plane

– triangulation of this 2D point set

Delaunay-Triangulation



Voronoi-Diagram Delaunay-Triangulation

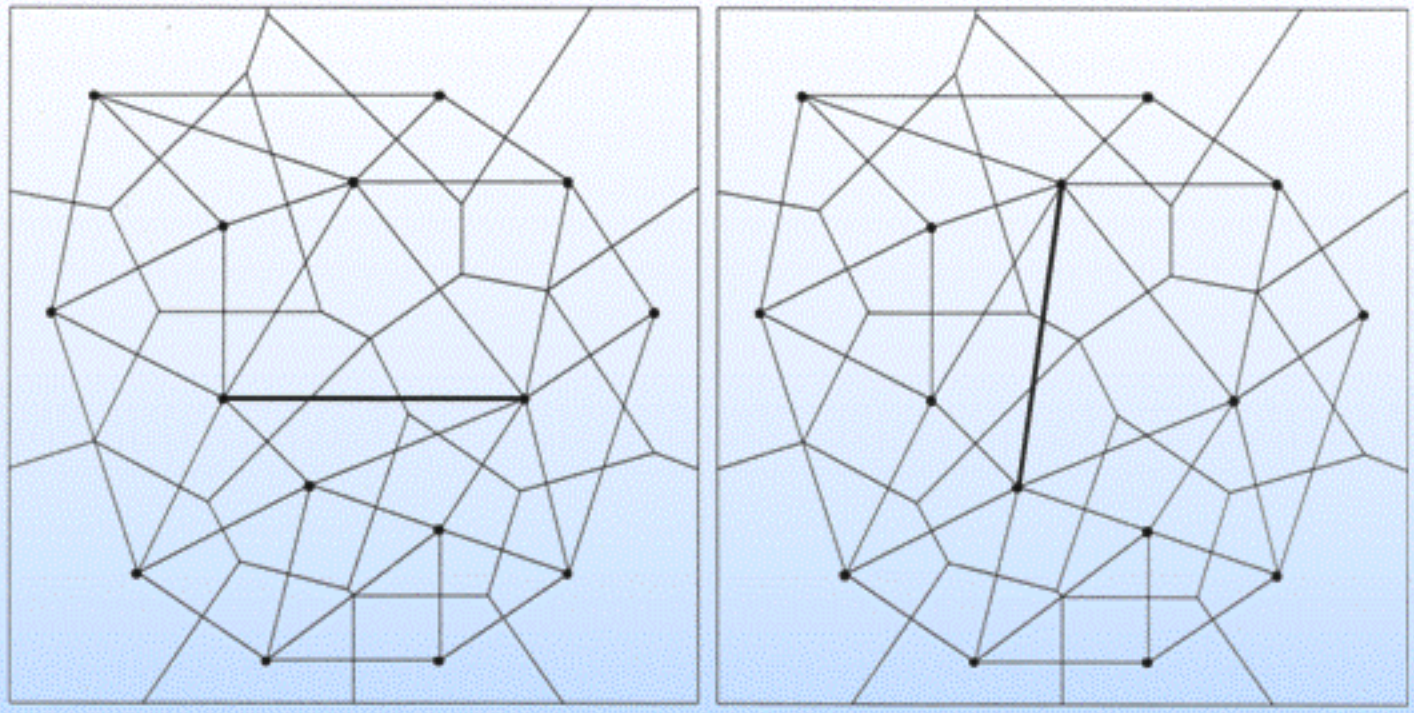
divide-and-conquer algorithm

$O(n \log n)$

iterative (dynamic) algorithm

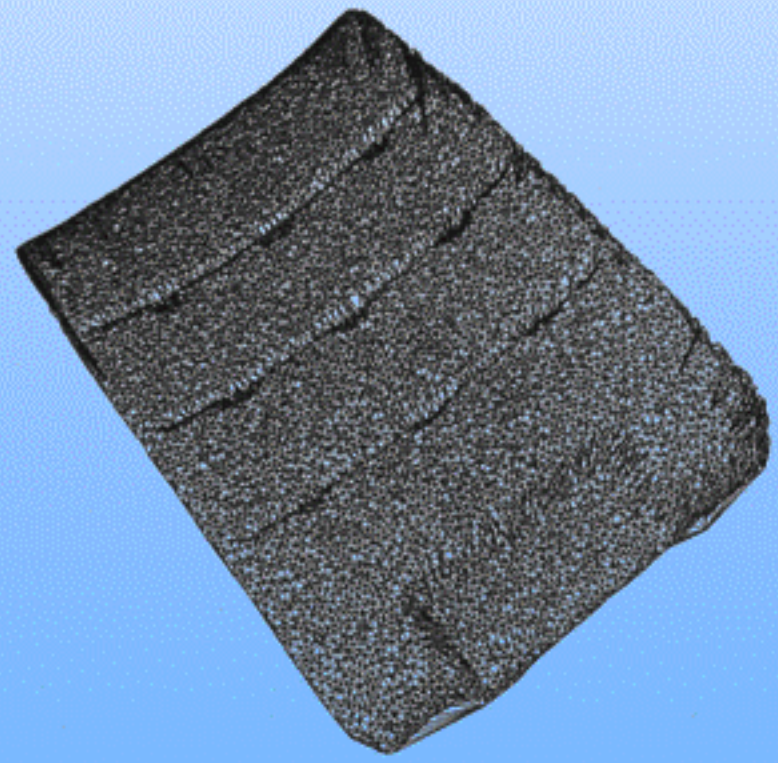
$O(n^2)$

neighbourhood-criterion



angle-criterion

circle-criterion



UNIVERSITÄT KAISERSLAUTERN

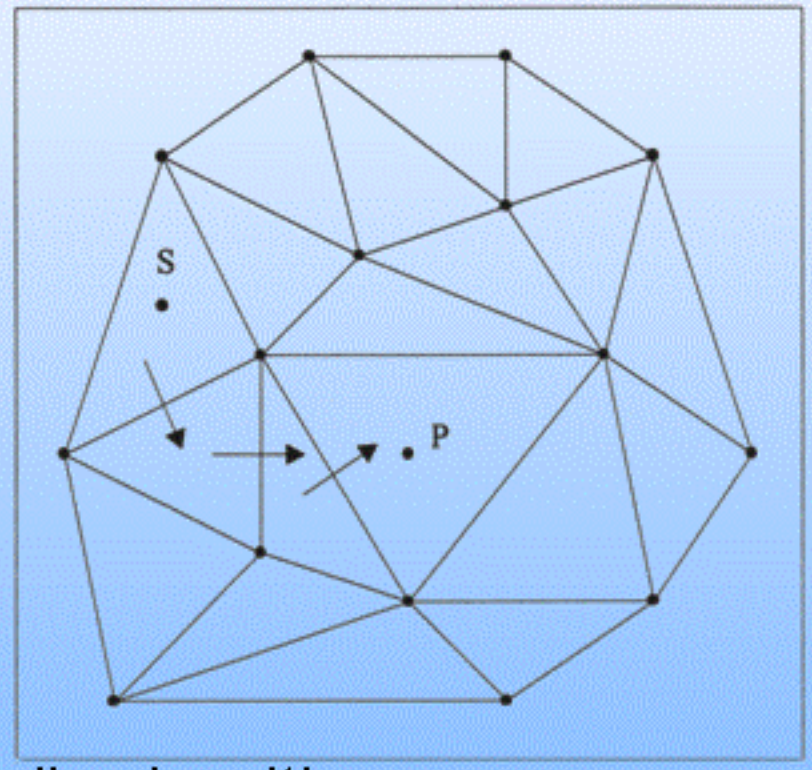
Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

– create initial triangulation using circumscribed rectangle of set P

– for each point p of set P do:

- determine the triangle, that contains p:

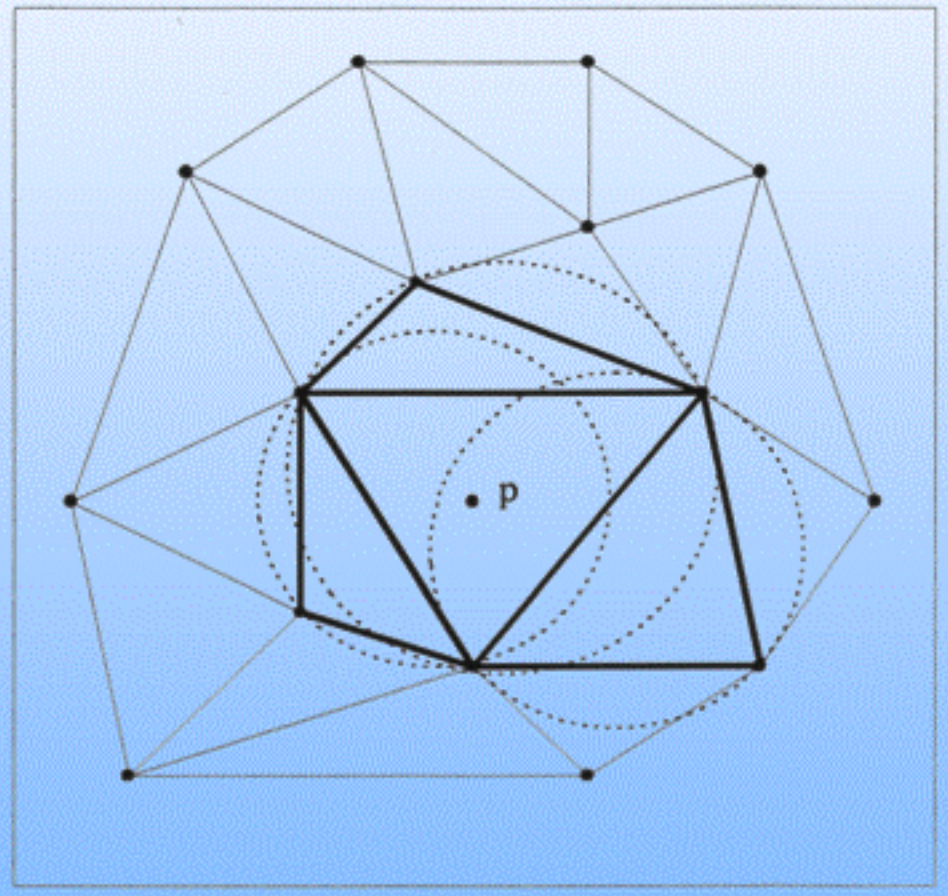
p is „visible“ from edge $E\,(e_0, e_1)$:$\Leftrightarrow$ $\det\begin{vmatrix} p & e_0 & e_1 \\ 1 & 1 & 1 \end{vmatrix} > 0$



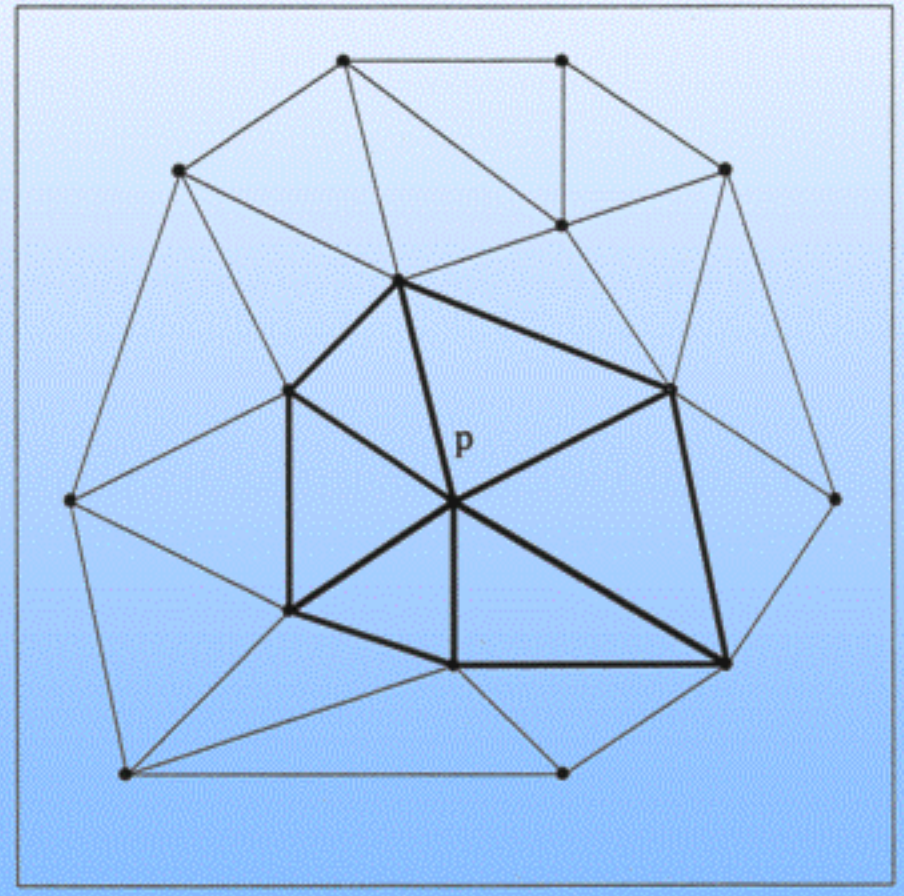
oriented-walk-algorithm:

starting with a first triangle, follow the edges that are not „visible“ until all edges of the current triangle are „visible“

- delete all triangles with a circumcircle containing point p



- create triangles, that replace deleted triangles and embed point p into the triangulation



Segmentation based on curvature distribution

- curvature estimation with osculating paraboloids (B. Hamann)
- least square fitting with

$$f(u,w)=\sum\sum c_{pq}u^p w^q$$

based on our neighbourhood structure and a first „best plane fitting parametrization“

- grouping of the points to this curvature measure

  two adjacent points belong to the same group :⇔

  - deviation of the normals is „small“
  - the difference of the minimal curvatures is less than a given tolerance

Variational modelling

$$(1-ws)\left\{\sum_{k=1}^{n_p} w_{pk}[X(u_k,v_k)-P_k]^2\right\}$$

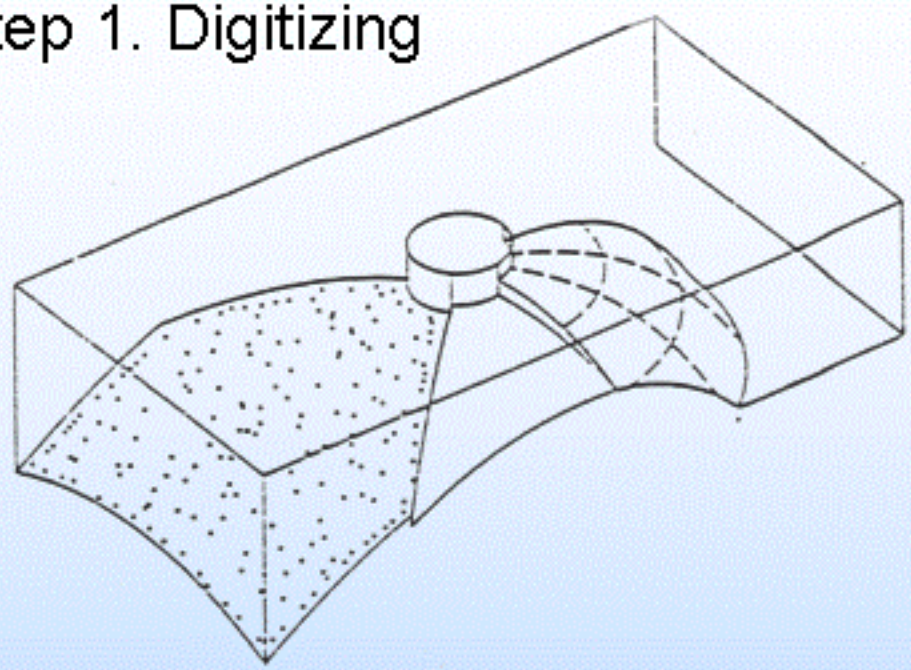
$$+ws\left\{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(w3ug\int_{v_j}^{v_{j+1}}\int_{u_i}^{u_{i+1}} w3u_{ij}\left\|\frac{\partial^3 X(u,v)}{\partial u^3}\right\|^2 dudv\right.\right.$$

$$\left.\left.+w3vg\int_{v_j}^{v_{j+1}}\int_{u_i}^{u_{i+1}} w3v_{ij}\left\|\frac{\partial^3 X(u,v)}{\partial v^3}\right\|^2 dudv\right)\right\}\rightarrow \min$$

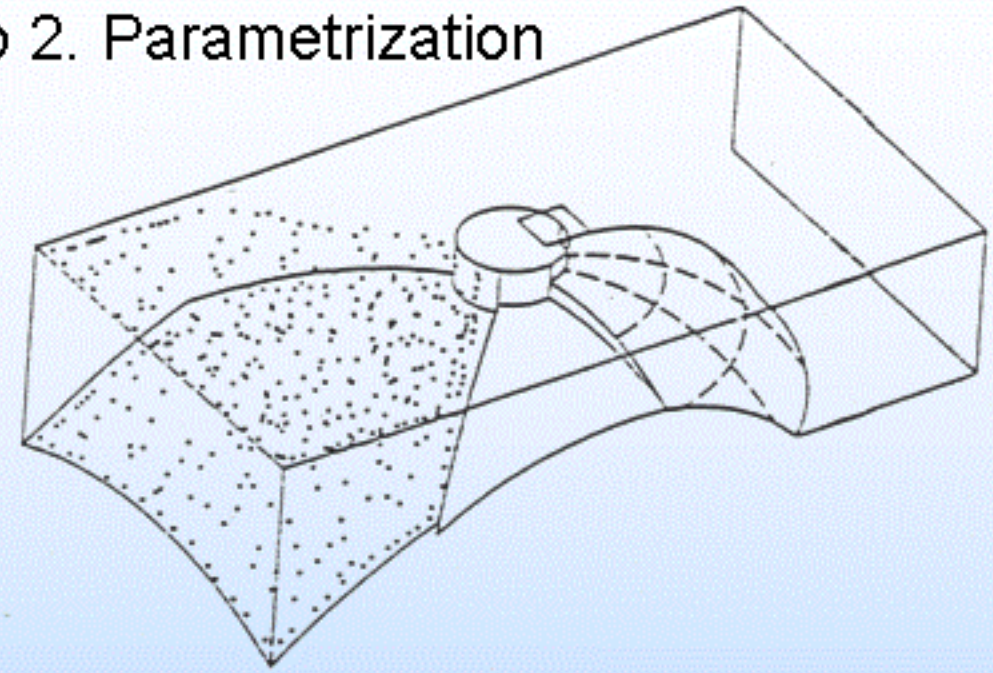
minimal strain energy in a thin elastic plate:

$$\int(\chi_1^2+\chi_2^2)ds\rightarrow \min$$

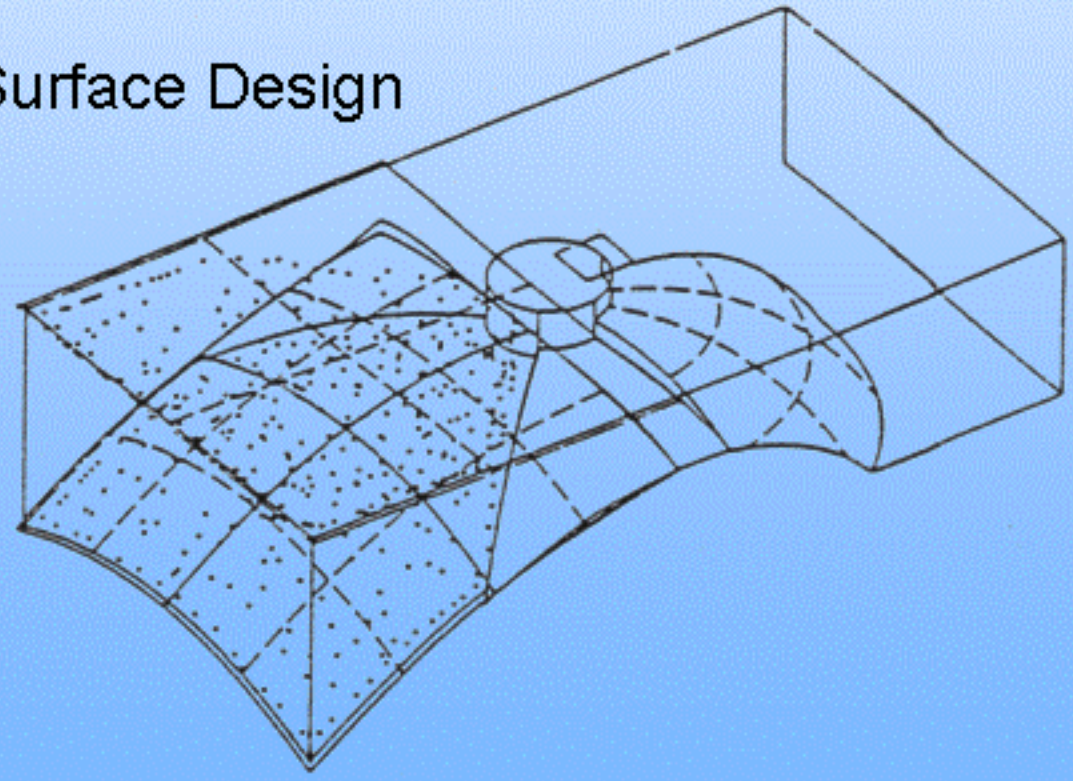
Step 1. Digitizing



Step 2. Parametrization



Step 3. Variational Surface Design



UNIVERSITÄT KAISERSLAUTERN

Prof. Dr. Hans Hagen
AG Graphische Datenverarbeitung und Computergeometrie

Stiffness degree concept for curves:

$$\alpha \int \|x''(s)\|^2 ds + \beta \int \|x'''(s)\|^2 ds \to \min$$

$$\alpha, \beta \geq 0 \quad \alpha + \beta = 1$$

general concept

curves:

$$\sum_{k=1}^{n} \varphi_k \int_{t_k}^{t_{k+1}} \left\| X^{(L)}(t) \right\|^2 dt + \sum_{k=0}^{n} \sum_{1=1}^{L} \nu_{k,1} \left\| X^{(1)(t_k)} \right\|^2 \to \min$$

$\varphi_k = \varphi$ geometric spline curves (Hagen)

$L = 3$ weighted τ - splines (Foley - Neuser)

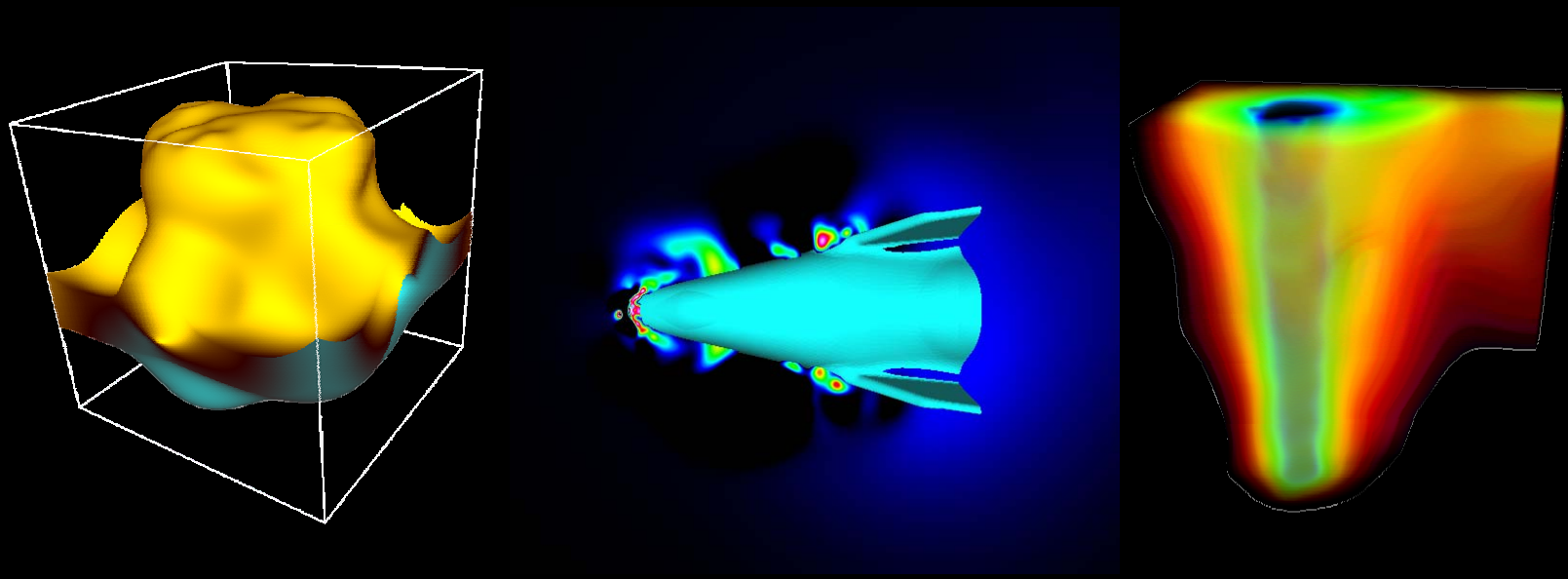
τ - splines (Hagen)

$L = 2$ ν - splines (Nielson)

interval tension splines (Saulkaskas)

$L = 2,3$ $\nu_{k,1} = 0$ (Nowacki - Meier)

Volume Encoding



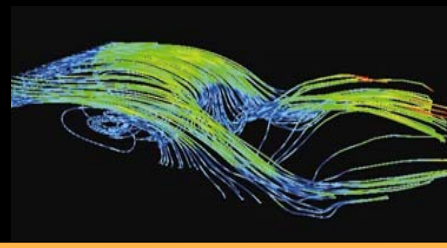
David S. Ebert & Yun Jang

School of Electrical and Computer Engineering

Purdue University

Collaborator Material from Kelly Gaither, Thomas Ertl, Manfred Weiler, Matthias Hopf, Jingshu Huang

Introduction



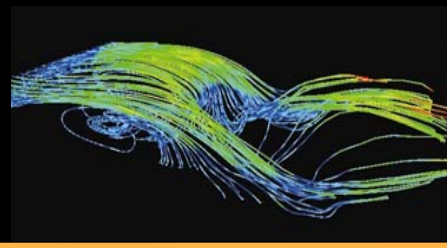
Enormous datasets from scientific simulation

Goal: interactive visualization on desktop PC

Volume rendering

- Regular or rectilinear gridded volume datasets solved
  - *Texture mapping on commodity PCs*
- Large scattered or unstructured volume datasets
  - *Still a challenging problem*

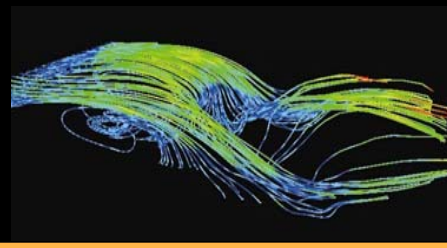
Introduction



Purpose of presentation

- Procedurally encoding arbitrary scalar, vector, and multifield datasets using Radial Basis Functions (RBFs)
- Improvement on spatial distribution of RBFs for interactive rendering
- Interactively render RBF encoded data

Previous Work



Interpolation of surface data

- Hardy(1971, 1990), Franke(1982), Franke and Nielson(1991), Franke and Hagen(1999), Carr et al.(2001)

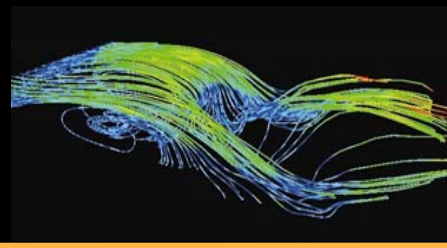
Interpolation of volume data

- Nielson et al. (1991), Nielson(1993)

Knot selection

- McMathon and Franke(1992)

Previous Work



Compactly supported RBFs

- Morse et al.(2001), Ohtake et al.(2004)

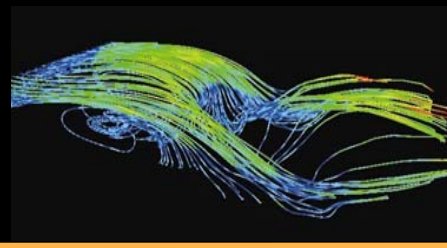
Hierarchical representation of volumetric data

- Co et al.(2003)

Meshless isosurface generation

- Co et al.(2004)

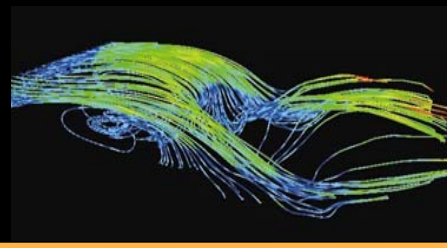
Previous Work



Reconstruction and Representation of 3D Objects with Radial Basis Functions by Carr et al. (2001)

- Zero level set implicit surface
- Fast fitting and evaluation
- Greedy algorithm for RBF fitting
- Energy minimization for the smoothest interpolant
- RBFs have global support
  - *Problem for interactive rendering*

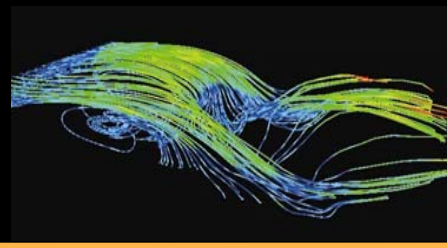
Previous Work



3D Scattered Data Approximation with Adaptive Compactly supported Radial Basis Functions by Ohtake et al. (2004)

- Implicit surface fitting
- Compactly supported RBFs
  - *Randomly chosen centers according to point density and surface geometry*
  - *Noise-robust approximation*
- Link between RBF fitting and partition of unity approximation

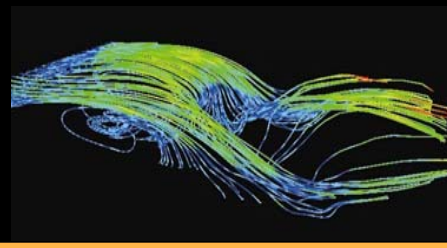
Previous Work



Hierarchical Clustering for Unstructured Volumetric Scalar Fields by Co et al. (2003)

- Multi-resolution representation of volumetric scalar data
- Cluster generation using Principal Component Analysis (PCA)
- Level-of-detail extraction by level-based and error-based traversal

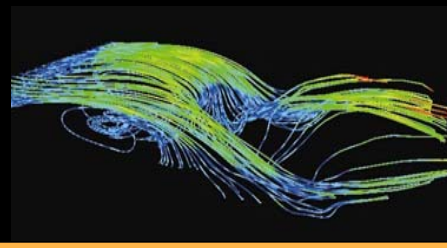
Previous Work



Meshless Isosurface Generation from Multiblock Data by Co et al.(2004)

- Extraction of continuous isosurface from volumetric data
- Continuous interpolant by locally defined RBFs using partition of unity method
- Sample points on Marching Cube triangle
- Project points onto isosurface defined by interpolant
- Surface splatting for visualization

Radial Basis Functions: Basic RBFs



RBF is a circularly-symmetric function centered at a single point μ

Examples

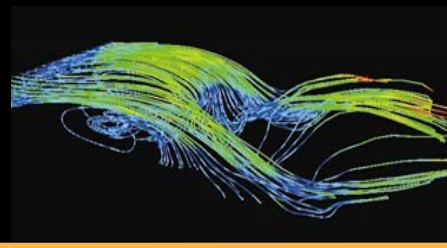
$$r = \|x - \mu\|$$

$$\phi(r) = r^2 \log(r)$$ Thin plate spline

$$\phi(r) = \exp(-cr^2)$$ Gaussian

$$\phi(r) = \sqrt{r^2 + c^2}$$ Multiquadric

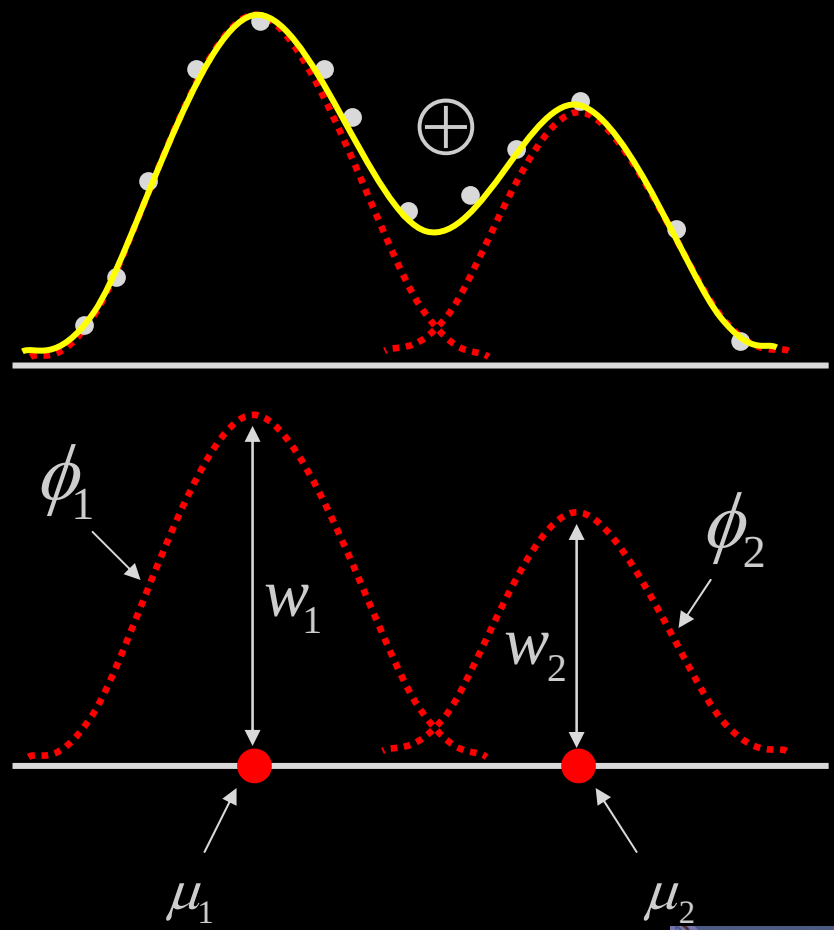
Radial Basis Functions: RBF Interpolation



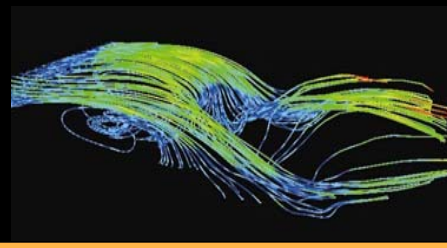
$$f(x) = \sum_{i=1}^{N} w_i \phi_i \left(\| x - \mu_i \| \right)$$

N	Number of Inputs
x	D-dimensional input vector
w_i	RBF weight
ϕ_i	Basis function
μ_i	RBF center
$\|x - \mu_i\|$	Vector norm

Example



RBF Encoding of Volume Data: Basis Function



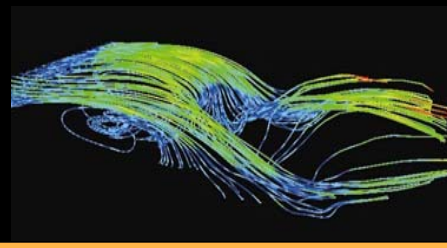
Gaussian function

- Functional value converges to zero exponentially
- Can adjust the width for accurate representation of local features
- Can easily compute derivatives

$$f(x) = w_0 + \sum_{j=1}^{M} w_j \exp\left(-\frac{\left\| x - \mu_j \right\|^2}{2\sigma_j^{\,2}} \right)$$

M	Number of Basis Functions
w_j	RBF Weight
$\left\| x - \mu_j \right\|$	3D-space distance
$\sigma_j^{\,2}$	RBF Width
w_0	Bias

RBF Encoding of Volume Data: Encoding System



Input Point Vectors

$(x_i, y_i), \{i = 1, \ldots, N\}$

x_i 3D position vector

y_i Scalar value

Find Center

Value weighted average point

Maximum error point

Calculate Width

Using nonlinear optimization

Calculate Weight

Minimizing sum squared error

Evaluate Errors

$|y - f(x)|$

Max Error > Target Error

?

Max Error < Target Error

Split Space

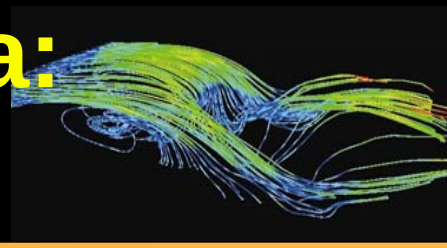
Principle component analysis

Final Encoding Output

(μ_j, σ_j^2, w_j)

$\{j = 1, \ldots, M\}$

RBF Encoding of Volume Data: Clustering methods



k-means

- One of the simplest clustering methods
- Iterative search the nearest cluster for each point

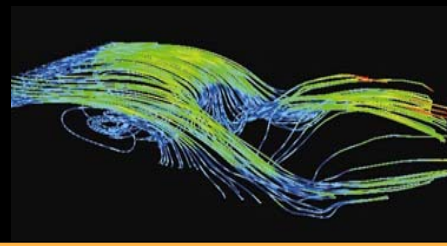
Principal component analysis (PCA)

- Dimension reduction method
- Splits data distribution into sphere-like shapes

Gaussian mixture model

- Density estimation method according to neighborhood
- Expectation and Maximization algorithm for training

RBF Encoding of Volume Data: Center Determination



Value weighted cluster average point

$$\mu_j = \frac{1}{n_j} \sum_{i=1}^{n_i} y_i \cdot x_i$$

$$position\_x_i = \{x \mid x \in j_{th} cluster\}$$

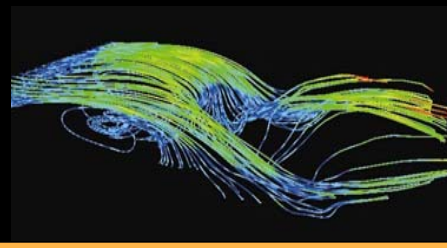
$$value\_y_i = value\_at\_x_i$$

Maximum error point

$$\mu_j = \max\{\varepsilon_i, i = 1, \ldots, n_j\}$$

$$\varepsilon_i = |y_i - f(x_i)|$$

RBF Encoding of Volume Data: Calculate Width



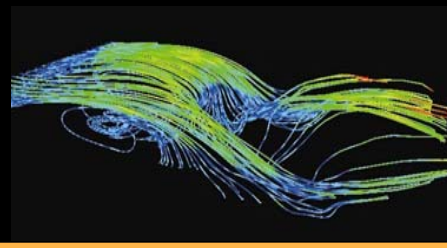
Levenberg-Marquardt method

- Gradient descent nonlinear optimization
- Find σ^* , a local minimizer for

$$F(\sigma) = \frac{1}{2}\sum_{i=1}^{N}[f_i(\sigma)]^2$$

$$f_i(\sigma) = |y_i - f(x_i, \sigma)|$$

RBF Encoding of Volume Data: Calculate Weight



Minimize the sum squared error for all data points

$$F(w) = \frac{1}{2}\sum_{i=1}^{N}\left[\left\|y_i - f(x_i, w)\right\|\right]^2$$

Matrix representation

$$A = \begin{bmatrix} f_1(x_1) & \cdots & f_M(x_1) \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ f_1(x_N) & \cdots & f_M(x_N) \end{bmatrix}$$

$$f_j(x_i) = \exp\left(-\frac{\left\|x_i - \mu_j\right\|}{2\sigma_j^2}\right)$$

Weight solving system

$$W = [w_1, \cdots, w_M]$$

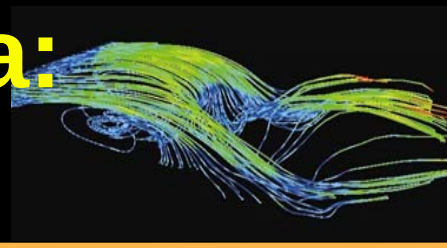
$$Y = [y_1, \cdots, y_N]$$

$$AW^T = Y^T$$

$$A^T AW^T = A^T Y^T$$

$$W^T = \left(A^T A\right)^{-1} \cdot A^T Y^T$$

RBF Encoding of Volume Data: Optimization methods



Nonlinear optimization methods

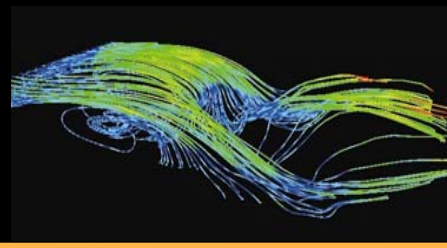
- Gauss-Newton method
- Levenberg-Marquardt method
- Trust region algorithm

Combination for solution optimization

- Width
- Width + Center
- Width + Weight
- Width + Center + Weight

High computational cost but nearly optimal solution

RBF Encoding of Volume Data: Vector Data Encoding



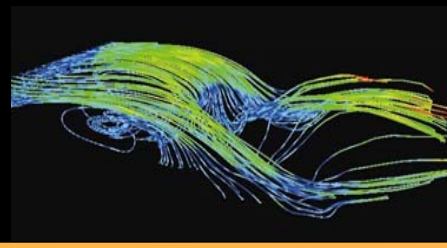
Extension of scalar field encoding

- Solve each RBF system for each vector component
- For 3D:
  - *3 center sets, 3 width sets, 3 weight sets*

Encode all components in one RBF system

- 1 center set, 1 width set, 3 weight sets
  - *Might not capture variances between vector components*
- 1 center set, 3 width set, 3 weight sets
  - *Can capture variances*
  - *Fast evaluation of RBFs in rendering process*

RBF Encoding of Volume Data: Error Measurement



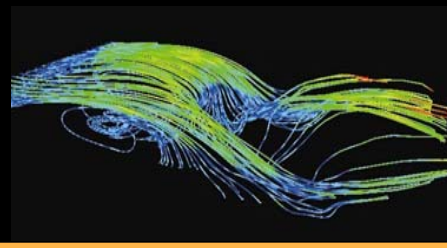
Scalar fields

- Absolute error
  - *Difference between original value and evaluated value*
- Percentage error

Vector fields

- Absolute error
- Percentage error
- Angular and magnitude error
  - *Separate error measurement of these two*
  - *Combination of both*
    - Balance between angular error and magnitude error is important
  - *Importance determined by features to be analyzed*

RBF Encoding of Volume Data: Large Scale Datasets



Domain decomposition by fast multipole method

- Weight solving method for the large RBF system

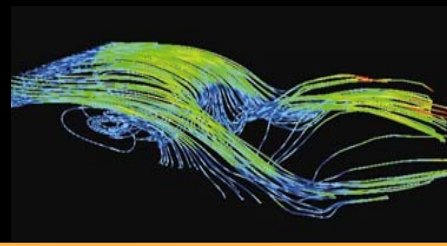
Domain localization

- Generate several independent RBF systems for large domain

Partition of unity

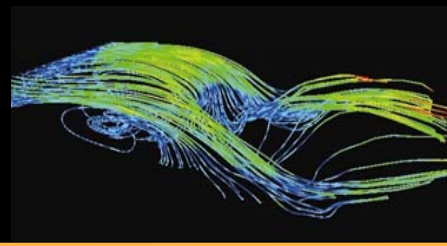
- Solve decomposed domains independently
- Need summation of decomposed domain solutions in rendering process

Encoding Statistics



	X38 Shock	Natural Convection	Black Oil Reservoir	Neghip	Blunt Fin
# of Cells	1,943,483	48,000	156,642	32,768	40,960
# of Cells Encoded	89,140	48,000	156,642	32,768	40,960
# of RBFs	2,932	435	458	812	695
Data Range	0.00 – 1.65	0.00 – 1.00	0.00 – 1.00	0.00 – 1.00	0.19 – 4.98
Avg. Error	0.05	0.04	0.007	0.012	0.11

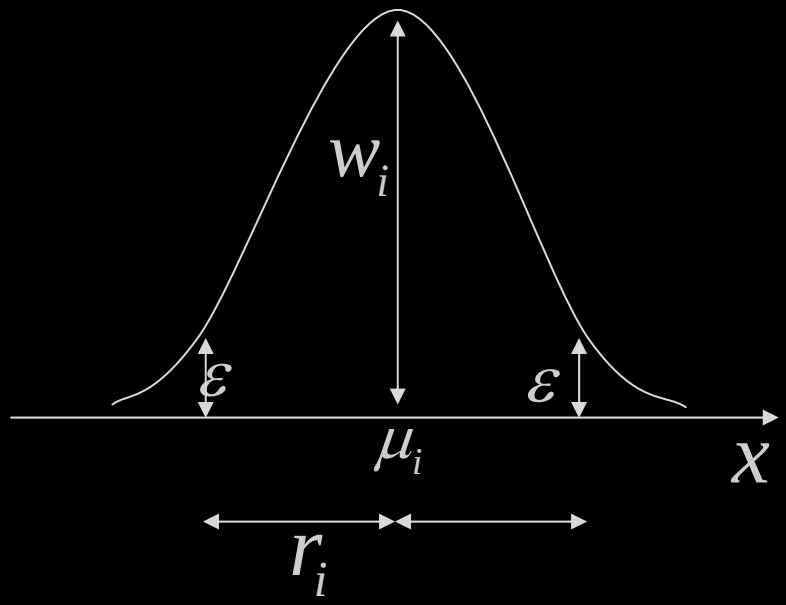
Spatial Data Structure: RBF Influence Calculation



For improved rendering performance

$$f(x) = w_i \exp\left(-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}\right)$$

$$\varepsilon = w_i \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

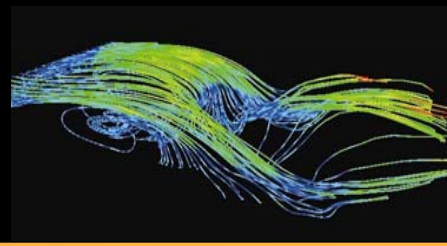


$$r_i = \sigma_i \cdot \sqrt{2 \cdot \ln\left(\frac{|w_i|}{\varepsilon}\right)}$$

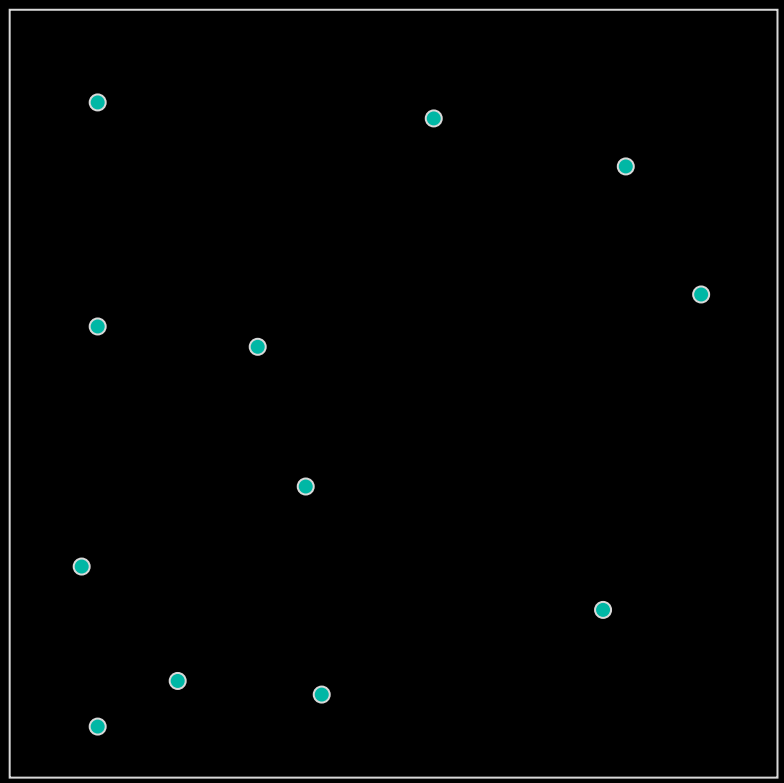
r_i Radius of influence

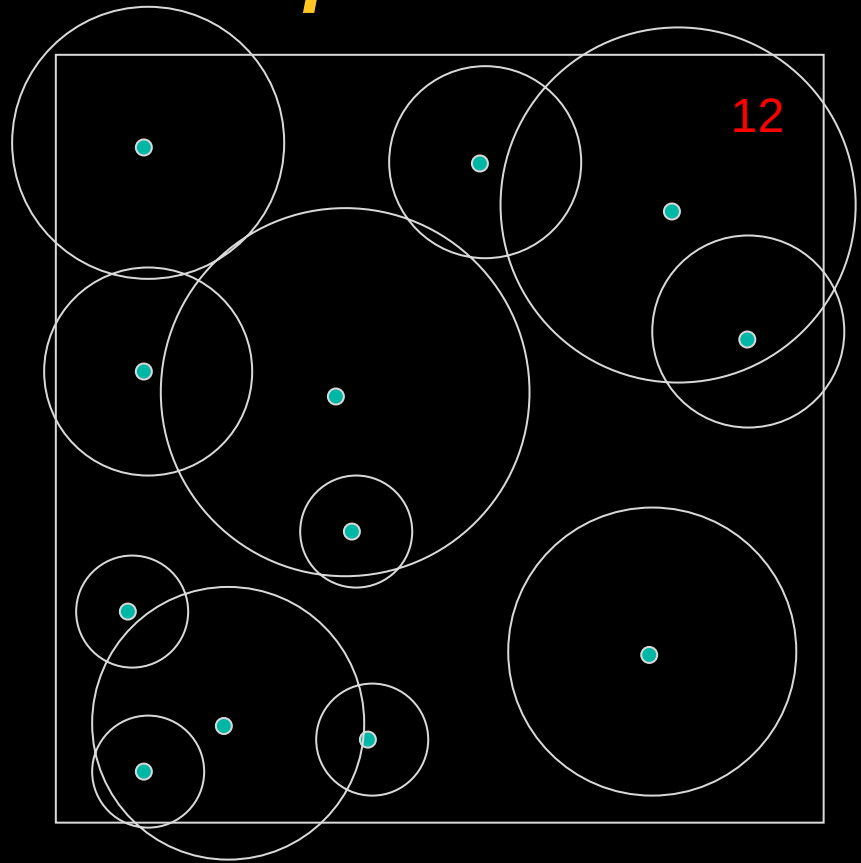
ε Error tolerance

Spatial Data Structure:
Spatial RBF Distribution and Octree Generation

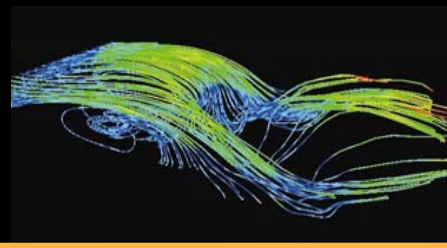


Example, Max number of RBF per cell = 4

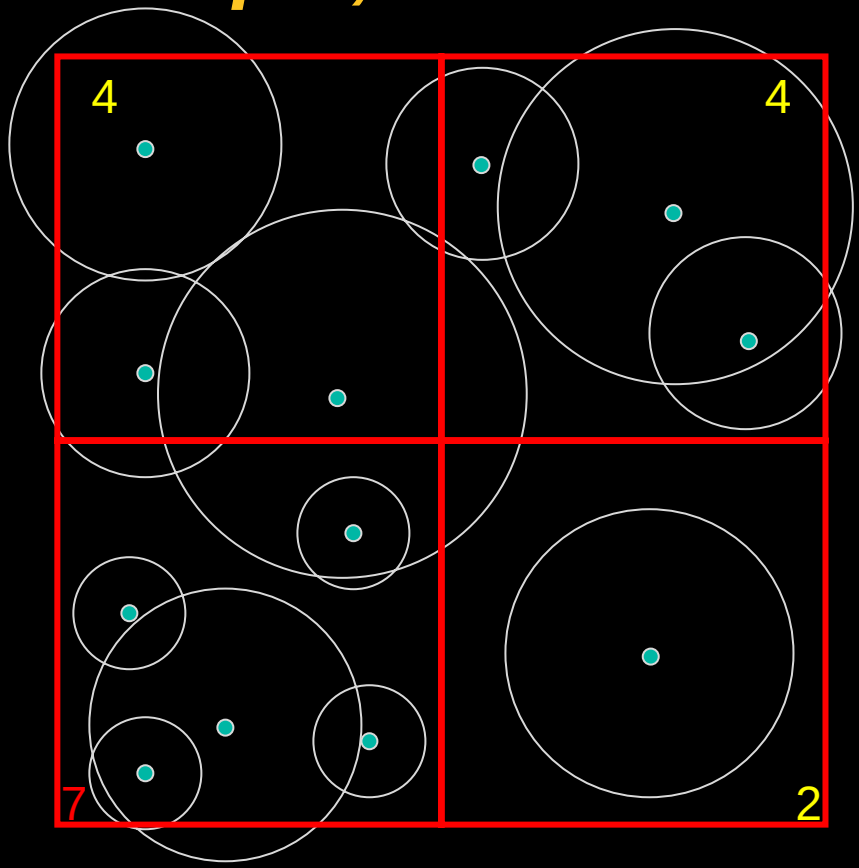


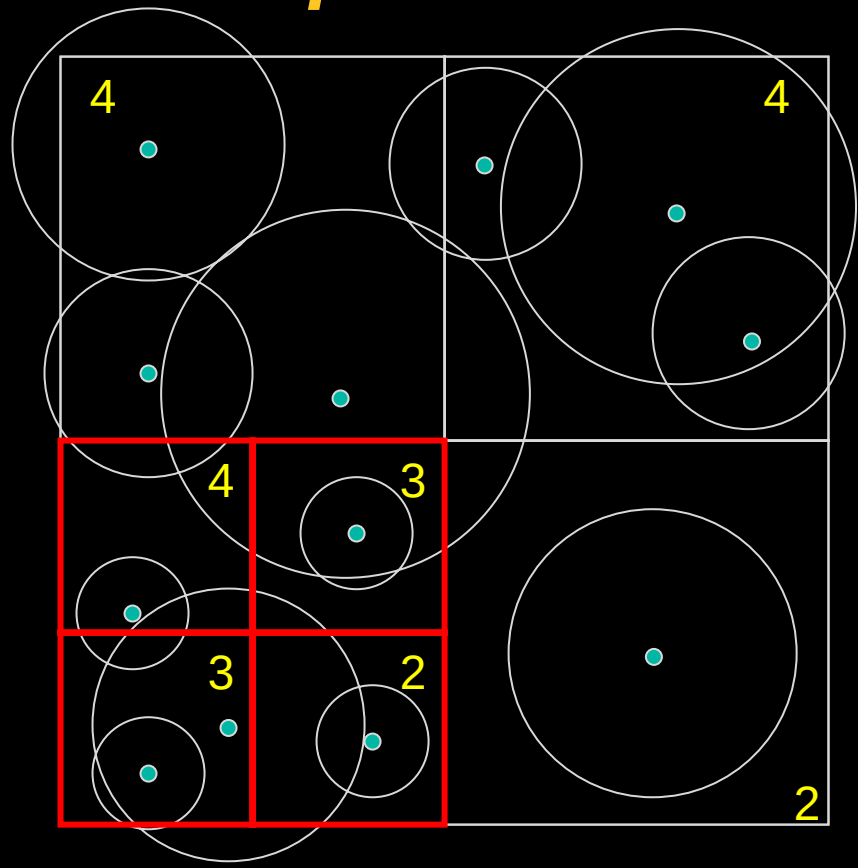


Spatial Data Structure:
Spatial RBF Distribution and Octree Generation

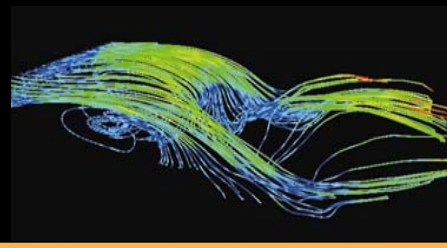


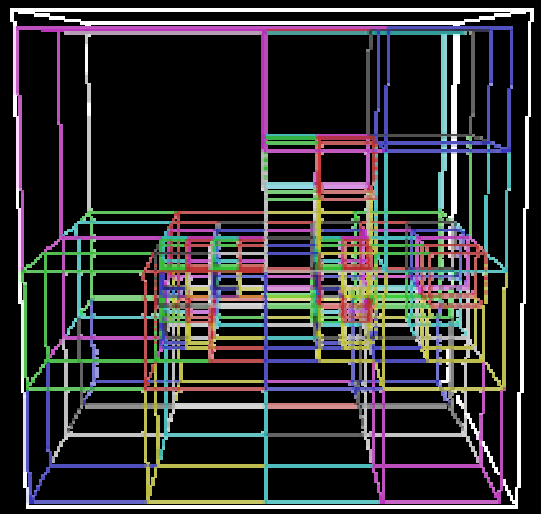
Example, Max number of RBF per cell = 4

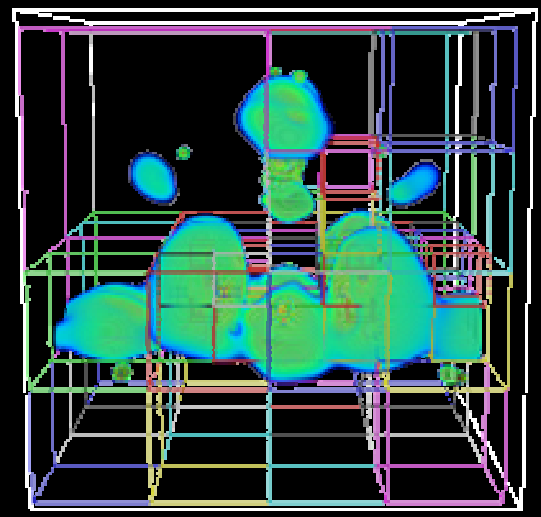




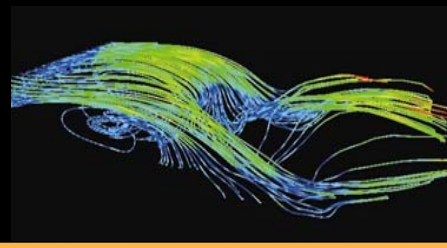
Spatial Data Structure: 3D Spatial RBF Distribution







Surface Generation and Visualization



Achieved by rendering slices

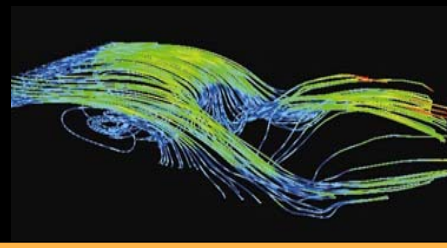
Mapping the reconstructed data into the alpha channel of fragment color

OpenGL alpha test to simulate the first-hit semantics of a volume ray caster

Pixel values are drawn only if

- They pass the z-buffer test
- The alpha values are larger than or equal to the selected iso-value

Direct Rendering from Encoding: Hardware Capabilities and Limit



nVdia high-level shading language Cg

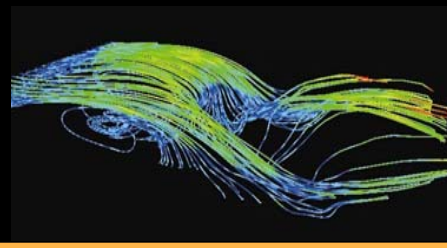
nVidia GeForceFX chip series

- Supports long fragment program with up to 1024 fragment program instructions

High memory bandwidth and parallel processing capability

Limited dynamic branching supported by GeForceFX fragment processing unit

Direct Rendering from Encoding: Splatting Approach



Rendering a polygon for each RBF center

- Polygon covers the influence disc of the basis function with respect to the rendered slice

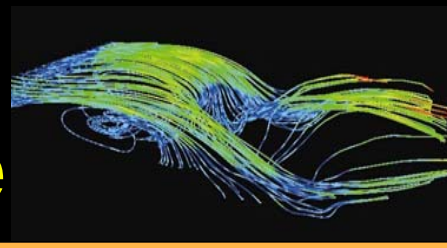
Accumulation of the polygons

- Blending is not supported for floating point p-buffer
- Ping-pong rendering by binding the result from previously rendered splats as a texture map
- Remove the continuous texture rebinds with feature of the GeForceFX

Subdivision approach

- Reduces the rasterization overhead

Direct Rendering from Encoding: Programmable Fragment Pipeline

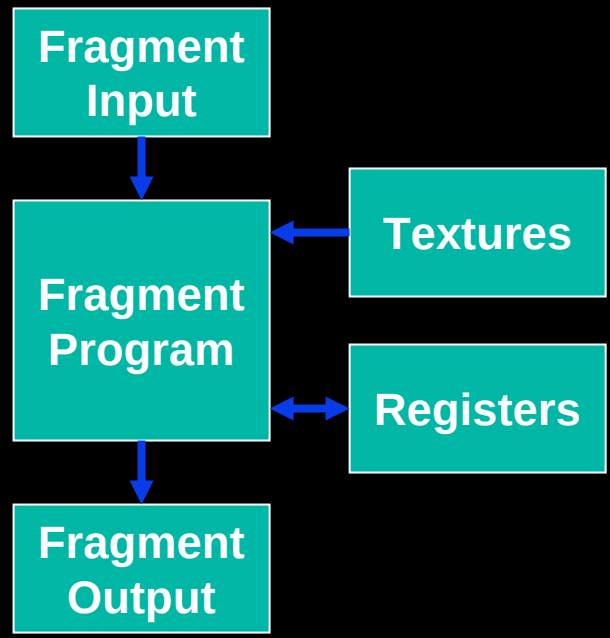


General, orthogonal instruction set

Floating-point data types

Resources

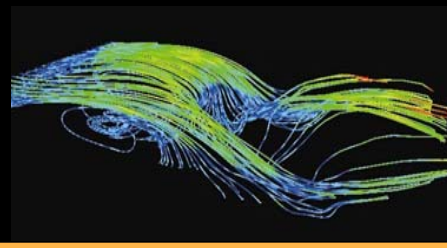
- Large number of registers
- Long programs
- Unlimited texture lookups
- Multiple levels of dependent texture lookup



High level programming languages

Very limited data dependent jumps or loops

Direct Rendering from Encoding: Fragment Based RBF Reconstruction



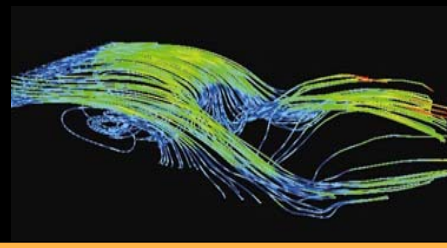
Similar to procedural textures

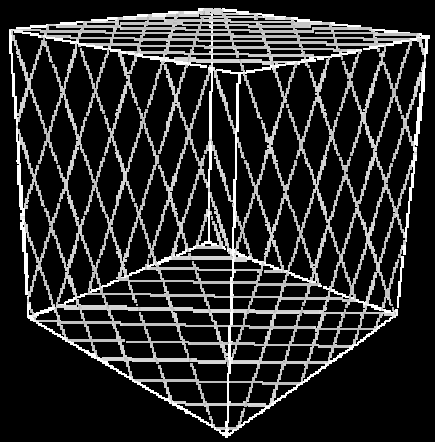
Decouple geometry from appearance

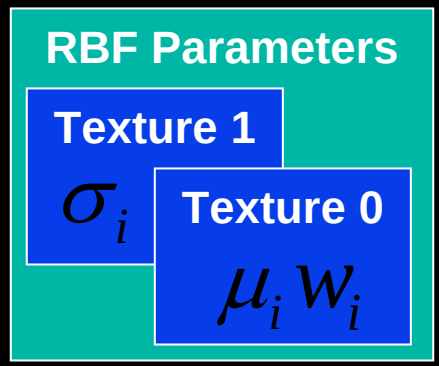
Compatible with various rendering/visualization algorithms

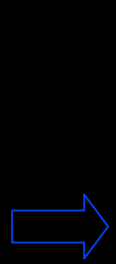
- Property texture for arbitrary geometry
  - *Pressure on airplane body*
- Individual cutting slices
- Texture based volume rendering
- Includes volume rendered isosurfaces

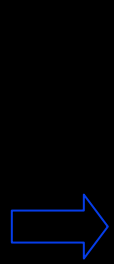
Direct Rendering from Encoding: RBF Reconstruction during Rasterization

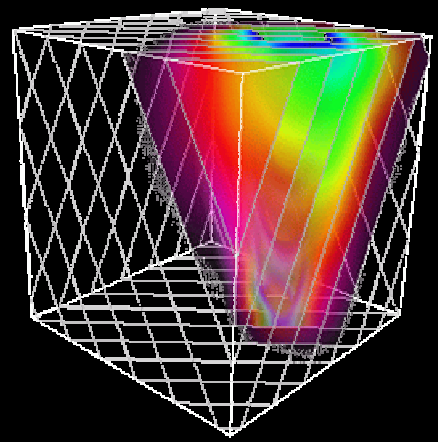




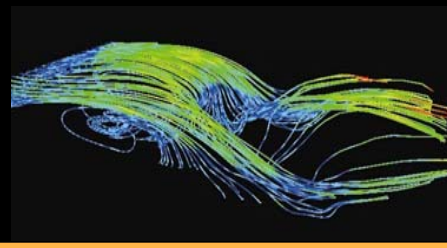








Direct Rendering from Encoding: High Level Rendering



Spatial decomposition (bricking)

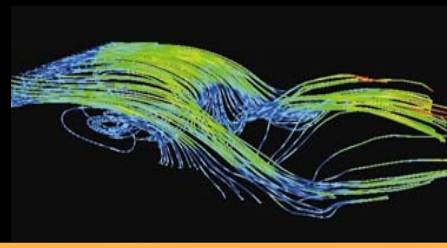
Multipass rendering (last pass for all cells)

- Hardware accelerated p-buffers
- Uses ping-pong rendering
- Active cell list

Switching to cell-based traversal for single pass reconstruction

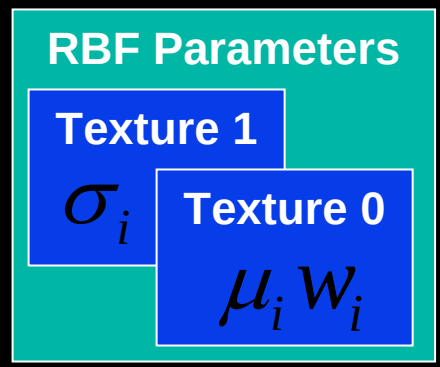
- Requires cell sorting by recursive depth-first traversal

Direct Rendering from Encoding: Texture Encoding



RBF parameters as two texture maps

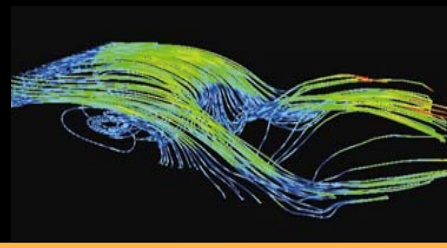
- Resides in the local graphics memory
- Full precision floating-point textures



RBF parameters of a single cell

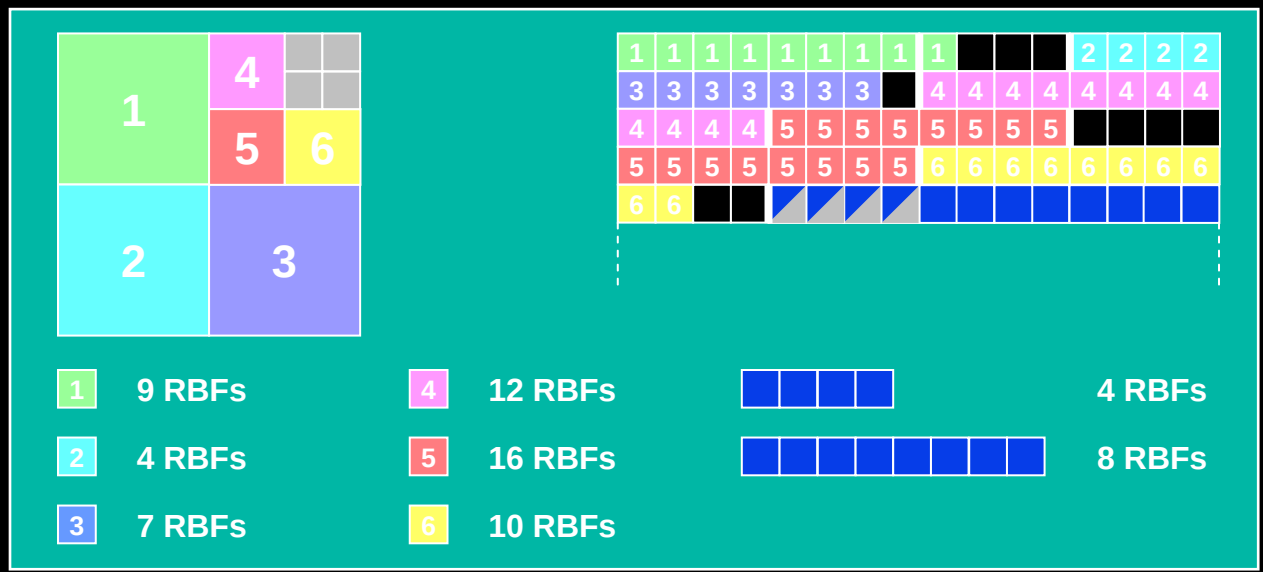
- Stored consecutively in the texture map
- Fragment program may access RBF parameter by lookup with increasing texture coordinate
- Avoid texture wrapping

Direct Rendering from Encoding: Texture Packing

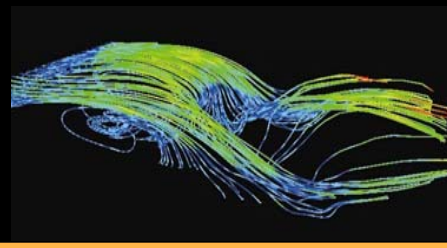


RBF chunks for multipass rendering

- Different chunk sizes for reducing rasterization
- Specialized fragment programs
- Padding

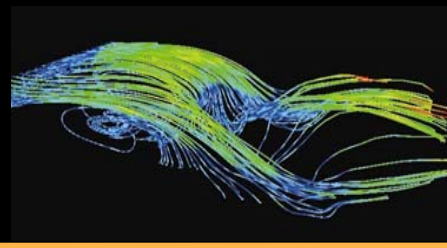


Direct Rendering from Encoding: Fragment Program



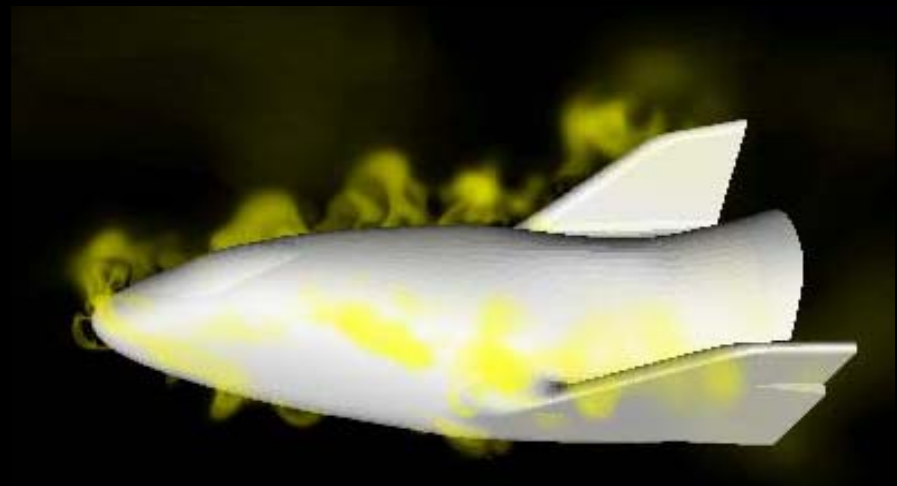
```
float4 main (…)
{
   float  val = 0.0;
   float4 texpos = texstart;

   for (float i = 0; i < CONST_NUMFUNCS; i++)
   {
       float4 tmp    = texRECT(rbfcenter, texpos.xy);
       float  s2_inv = texRECT(rbfwidth,  texpos.xy);
       float3 vec    = tmp.rgb – inpos.xyz;
       float  expval = - dot(vec, vec) * s2_inv;
       val    += tmp.a * ex2(expval);
       texpos += texinc;
   }

   val += bias + error;

   return tex1D(map, (val + mapSBA.r) * mapSBA.g);
}
```

Results:
X38 Crew Return Vehicle



Tetrahedral finite element viscous calculation on geometry

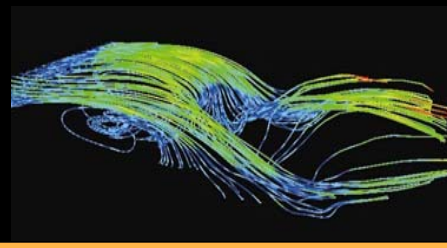
- Computed at Engineering Research Center at Mississippi State University by the Simulation and Design Center
- Single time step in the reentry process into atmosphere
- 1,943,483 tetrahedra at a 30 degree angle of attack



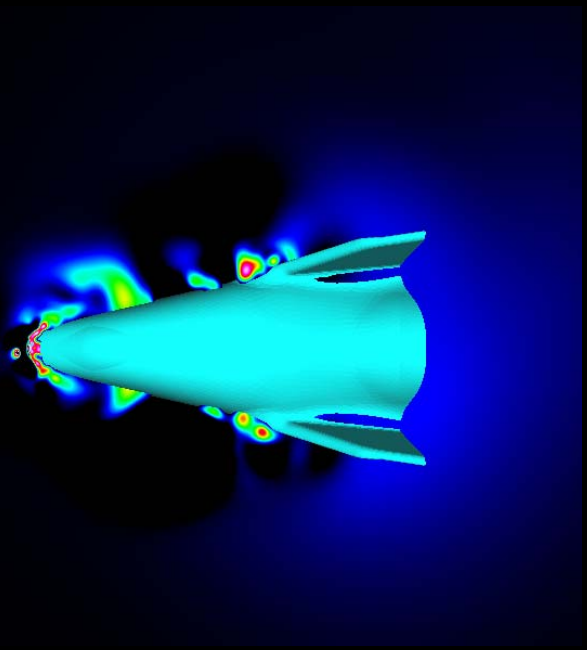
2,932 RBFs

Shock Volume Rendering representing normal Mach number around 1.0

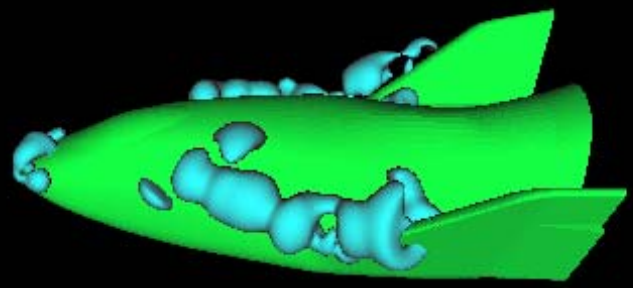
Results:
X38 Crew Return Vehicle



Cutting plane rendering of shock

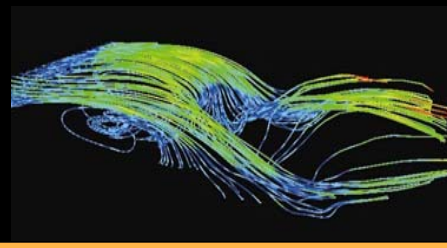


Volume isosurface rendering of density



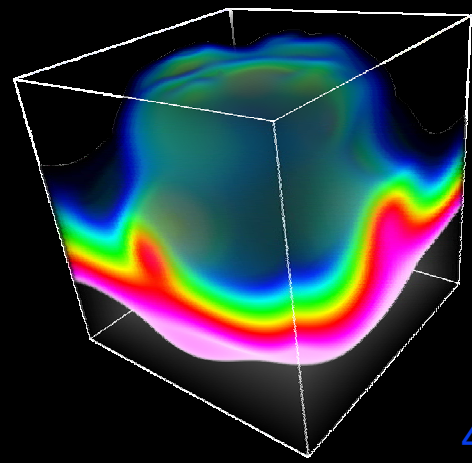
1,611 RBFs

Results:
Natural Convection in a box

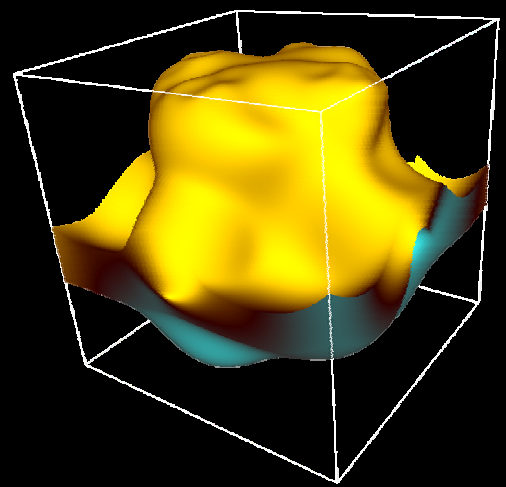


80th time step of temperature from a natural convection simulation

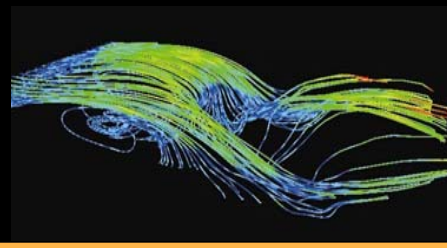
- A non-Newtonian fluid in a cube
- Developed at The University of Texas at Austin
- 48,000 tetrahedral elements



435 RBFs

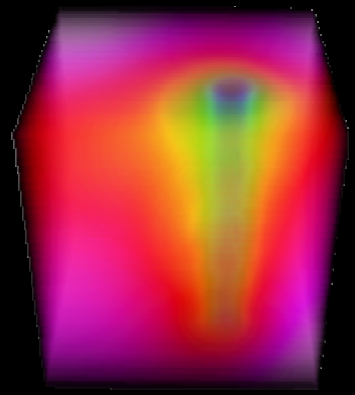


Results:
Black Oil Reservoir Simulation

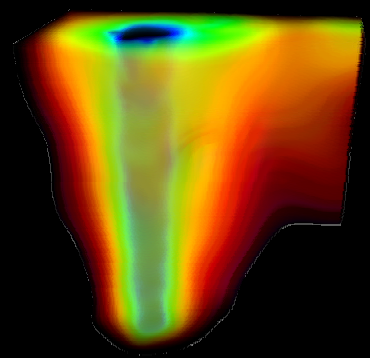


A simulation for prediction of placement of water injection wells to maximize oil from production wells

- Computed by the Center for Subsurface Modeling at The University of Texas at Austin
- 156,642 tetrahedra containing water pressure values for the injection well

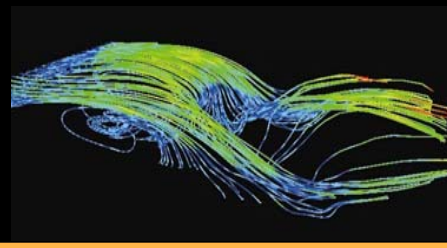


458 RBFs

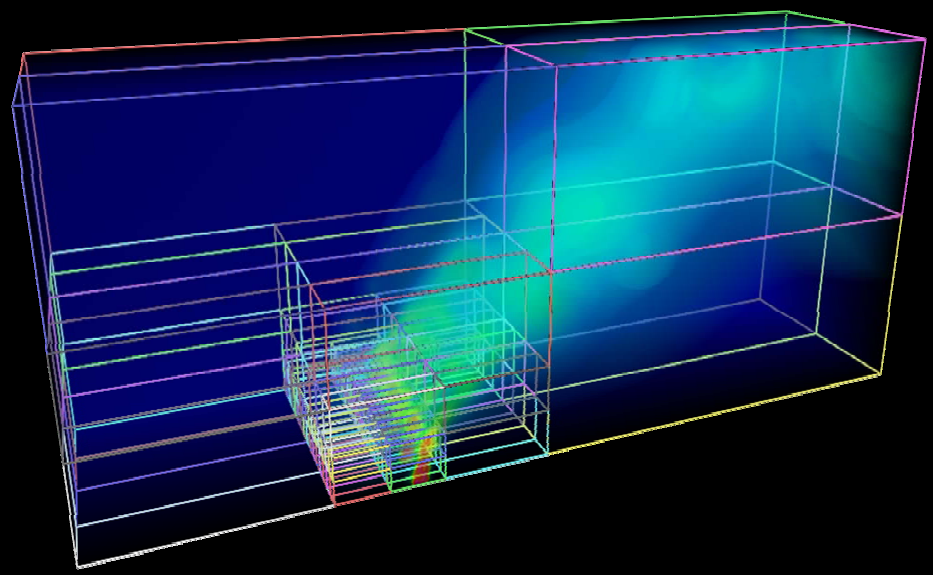


222 RBFs
49 Cells

Results:
Blunt fin

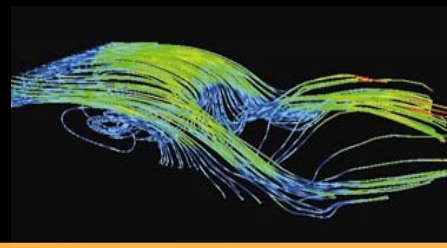


Volume rendering and RBF spatial distribution



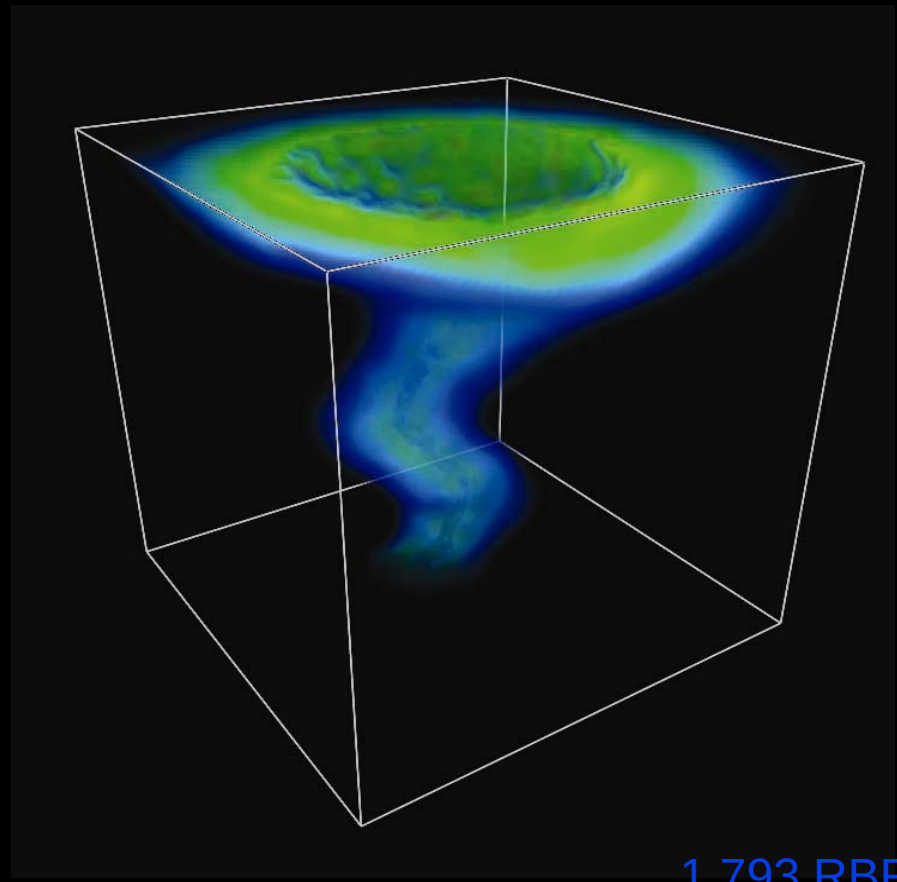
695 RBFs
238 Cells
< 60 RBFs/Cell

Results: Tornado



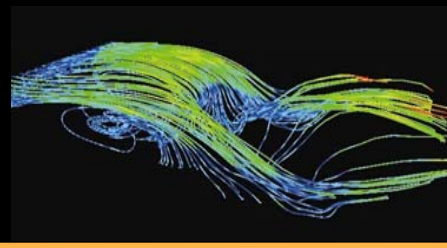
Synthetic dataset

- Courtesy of Roger Crawfis from The Ohio State University
- 32,768 cells
- Visualization of velocity magnitude



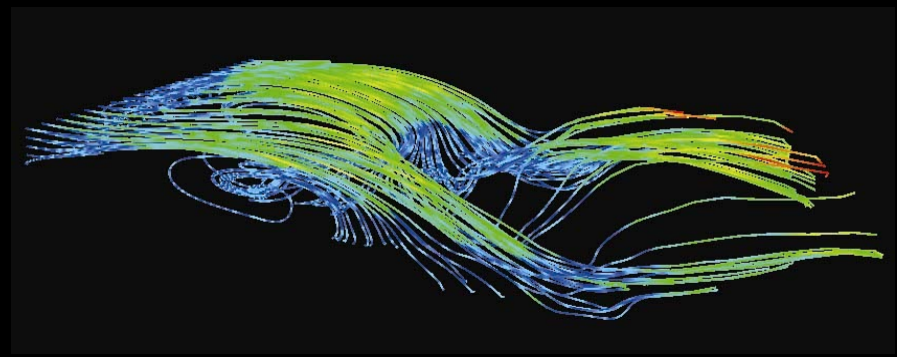
1,793 RBFs

Results:
Turbulent Channel Flow



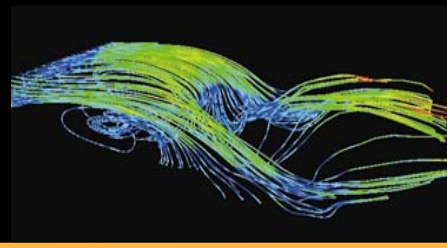
Experiment studying of laminar-turbulent boundary layer transition in a water channel

- Provided by the Institute for Aerodynamics and Gasdynamics of the University of Stuttgart
- 32,085 cells



2,105 RBFs
110 particles

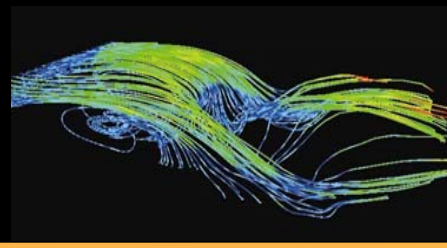
Results: System



Intel Pentium 4 2.80 GHz process, 2 GB memory

256 MB nVidia GeForce 6800GT graphics board

Results: Performance



Limited by the rasterization of the graphics card

Single cutting planes

>> 30 fps even for several thousands of RBFs per fragment

Volume Rendering by splatting approach

- 6.4 – 44.3 fps with 64 slices on 400 x 400 viewport

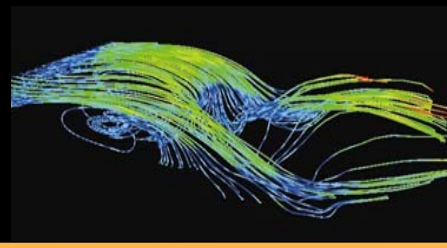
Volume Rendering by texture-based approach

- 0.96 – 10.5 fps with 64 slices on 400 x 400 viewport
- Limited by multipass rendering
- Isosurface shading

Optimization for nv40

- Code not optimized for new capabilities

Conclusion

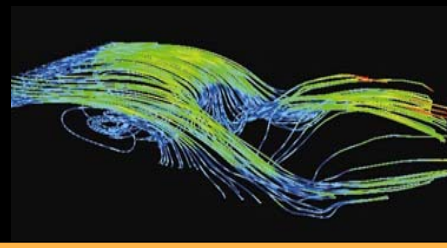


Effective encoding of scalar and vector fields

Novel approach for interactive reconstruction and visualization of arbitrary 3D fields

Allows interactive exploration of large datasets from a variety of sources

Future Work

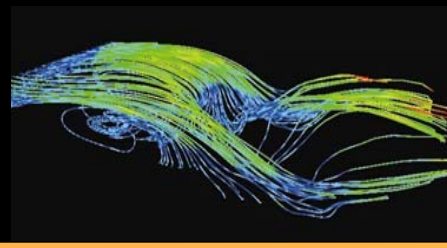


Improve rendering and increase image quality by incorporating pre-integrated volume rendering

Improve RBF encoding techniques for improved performance

Better error measurement methods for vector encoding

Acknowledgements

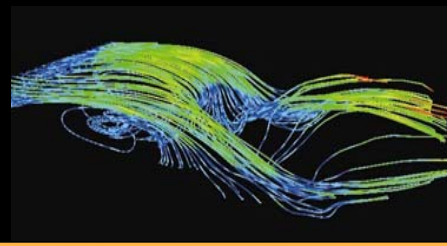


This work is supported by the US National Science Foundation under grants NSF ACI-0081581 and NSF ACI-0121288

PURPL
Purdue University

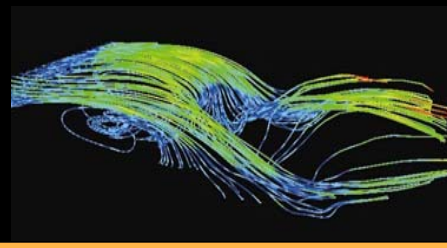
VIS 2004 austin, texas

References: RBF Encoding



- R. L. Hardy, Multiquadric equations of topography and other irregular surfaces, Journal of Geophysical Research, 1971
- R. L. Hardy, Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968-1988, Computers and Mathematics with Applications, 1990
- R. Franke, Scattered Data Interpolation: Tests of Some Method, Mathematics of Computation, 1982
- R. Franke and G. M. Nielson, Scattered data interpolation and applications: A tutorial and survey, Geometric Modelling, Methods and Applications, 1991
- R. Franke and H. Hagen, Least squares surface approximation using multiquadrics and parametric domain distortion, Computer Aided Geometric Design, 1999
- J. R. McMathon and R. Franke, Knot Selection for Least Squares Thin Plate Splines, SIAM, 1992
- J. Carr et al., Reconstruction and Representation of 3D Objects With Radial Basis Functions, Proceedings of ACM SIGGRAPH 2001
- C. S. Co et al., Hierarchical Clustering for Unstructured Volumetric Scalar Fields, Proceedings of IEEE Visualization 2003

References: RBF Encoding



- V. Savchenko et al., Function Representation of Solids Reconstructed from Scattered Surface Points and Contours, Computer Graphics Forum, 1995
- G. Turk and J. O'Brien, Shape Transformation Using Variational Implicit Functions, Proceedings of SIGGRAPH 99
- G. Turk and J. O'Brien, Modelling with implicit surfaces that interpolate, ACM Transactions on Graphics, 2002
- B. S. Morse et al., Interpolating Implicit Surfaces From Scattered Surface Data Using Compactly Supported Radial Basis Functions, Proceedings Shape Modeling International, 2001
- A. Goshtasby, Grouping and parameterizing irregularly spaced points for curve fitting, ACM Transactions on Graphics, 2000
- Co et al., Meshless Isosurface Generation from Multiblock Data, *VisSym 2004*, May 2004
- Ohtake et al., 3D Scattered Data Approximation with Adaptive Compactly supported Radial Basis Functions, Shape Modeling International 2004, 2004

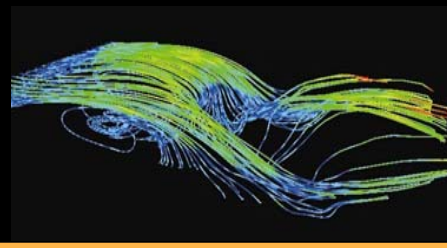
Feature Analysis using Functional Encoding

Kelly Gaither

Texas Advanced Computer Center

TACC

Motivation

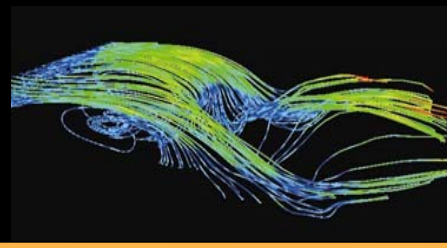


Recent computational performance increases

Massive dataset from advanced computing simulations

Difficulty in direct analysis of large datasets

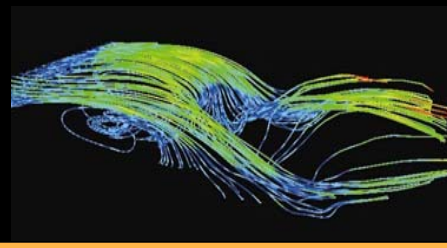
Motivation



Feature detection

- Powerful means of automatically detecting regions of interest
- Automates data analysis
- Extracts the salient features

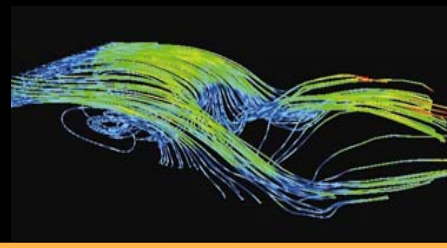
Survey of Techniques



J. Helman and L. Hesselink

- Evaluation of Flow Topology from Numerical Data, 1987
  - *Two dimensional topology using critical points*
  - *Attachment and separation surfaces in three dimensional flows*
- Representation and display of vector field topology in fluid flow data sets, 1989
  - *Representation of global topology based on the analysis of critical points*
- Visualizing vector field topology in fluid flows, 1991
  - *Combining simplicity of scheme depiction with curves and surfaces directly from the data*

Survey of Techniques



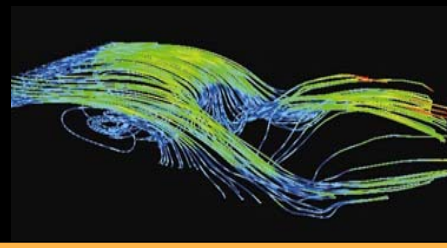
Globus et al.

- A tool for visualizing the topology of three-dimensional vector fields, 1990
  - *Numerical analysis and graphical display of topological aspects of vector fields*
  - *Critical points, their invariant manifolds, and integral curves*

Jeong and Hussain

- On the identification of a vortex, 1995
  - *Identifying a vortex core referred to as the λ_2-definition*

Survey of Techniques



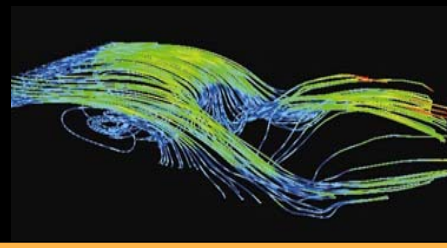
Lovely and Haimes

- Shock detection from computational fluid dynamics results, 1999
  - *Locating shocks in transient and steady state solution using flow physics*
  - *Removing false shock detection using a set of filtering algorithms*

Haimes and Kenwright

- On the velocity gradient tensor and fluid feature extraction, 1999
  - *Identifying global features using local analytical tests based on critical point theory, phase plane analysis, and the velocity gradient tensor*

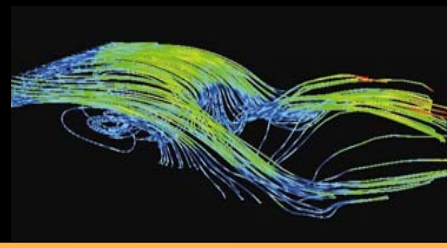
Survey of Techniques



Silver and Wang

- Tracking scalar features in unstructured datasets, 1998
  - *Visualization of time-varying datasets and tracking volume features in unstructured scalar datasets*
  - *Determining history of time-varying features difficult*

Feature Definition



Critical point

- Stationary point
- Location in the vector field **v** where **v**=0

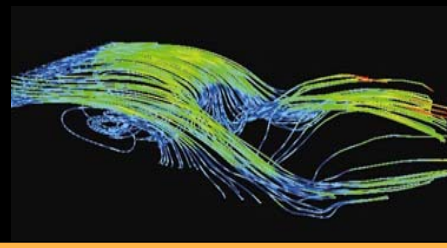
Vortex core

- Central core region of a vortex

Shock

- Connected regions of sharp discontinuities
- Very thin region in a supersonic flow

Critical Points



Integral manifolds

- Combination of vector field topology consisting of key points, curves, and surfaces

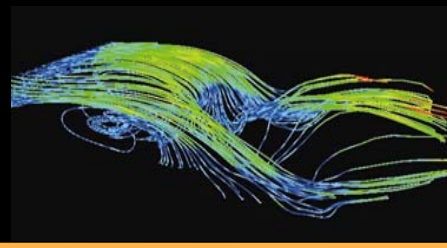
With a few exceptions, all integral manifolds must begin and end at zeros in the vector fields

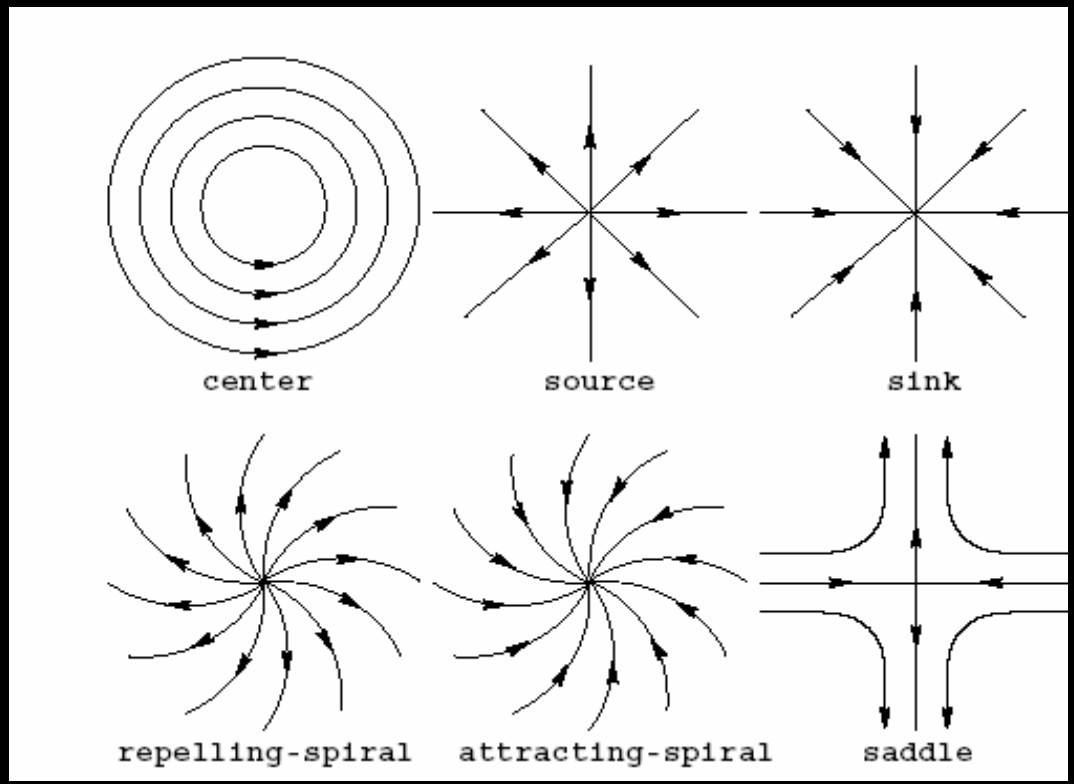
These zeros form the critical manifolds

Critical manifolds allow us to characterize the flow in the areas surrounding the critical points

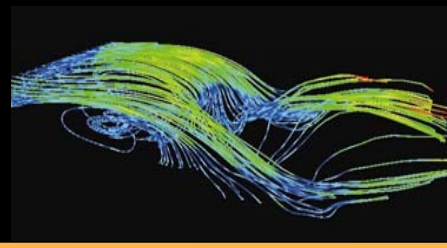
Critical point contains a greater probability of a region of interest

Critical Points





Critical Points Detection



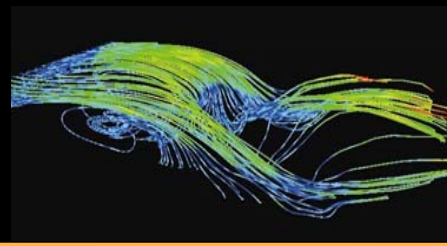
Taylor series expansion of the vector field v

$$v_i = v_i^{(0)} + \left(x_j - x_j^{(0)}\right)\frac{\partial v_i}{\partial x_j} + O\left(\Delta x_k \Delta x_l\right)$$

3x3 coefficient matrix

$$\left(\Delta v\right)_{ij} = \frac{\partial v_i}{\partial x_j}$$

Critical Points Detection



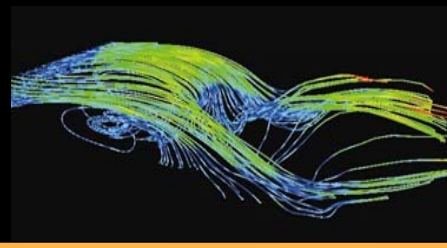
Classification of critical points

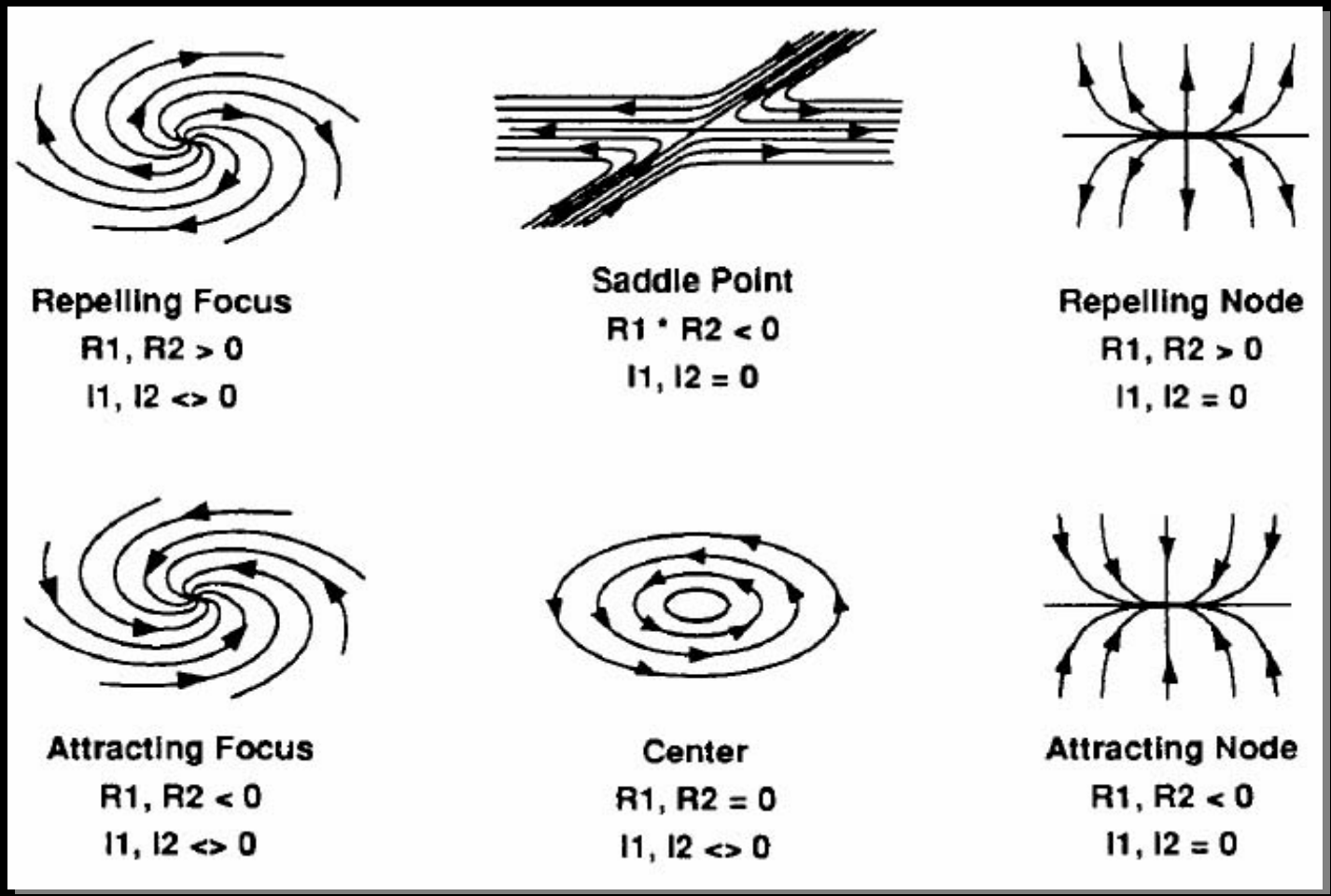
- Eigenvalues of the coefficient matrix

$$R_1 + iI_1, \quad R_2 + iI_2, \quad R_3 + iI_3$$

- Combination of real part and imaginary part
  - *Positive real part*
    - Repelling direction
  - *Negative real part*
    - Attracting direction
  - *Imaginary part*
    - Circulation
- Real eigenvalues all having same signs
  - *Purely repelling node*
  - *Purely attracting node*

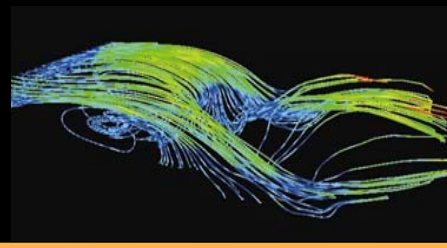
Critical Point Detection





[Hesselink and Helman, 1991]

Vortex Core Detection



Velocity gradient tensor $J = \nabla v$

- Symmetric part, strain-rate tensor S

$$S = \frac{J + J^T}{2}$$

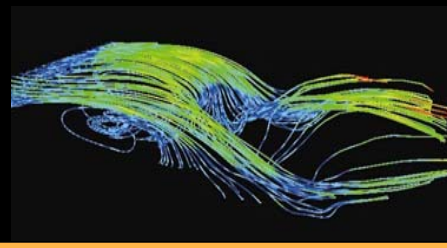
- Asymmetric part, spin tensor Ω

$$\Omega = \frac{J - J^T}{2}$$

Eigenvalues of $S^2 + \Omega^2$

$$\lambda_1 \geq \lambda_2 \geq \lambda_3$$

Vortex Core Detection



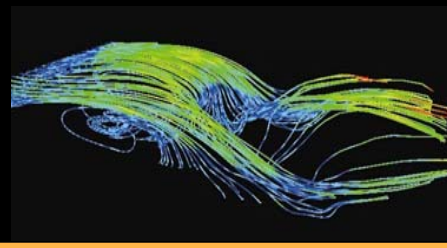
Vortex

- Connected region where $S^2 + \Omega^2$ has two negative eigenvalues

Vortex core

- Points having negative λ_2

Shock Detection



Calculate quantities: [Marcum and Gaither, 1997]

$$E_1 = \max\left(\frac{v}{|v|}\cdot\Delta U, 0\right)$$

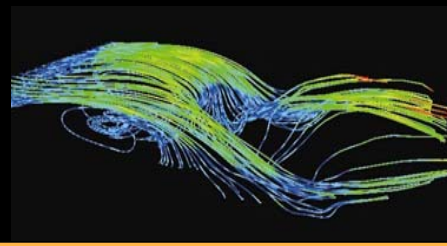
$$E_2 = \min\left(\frac{v}{|v|}\cdot\Delta U, 0\right)$$

$$E_3 = \left|\Delta U - \left[\frac{v}{|v|}\right] v\cdot\Delta U\right|$$

U *represents*

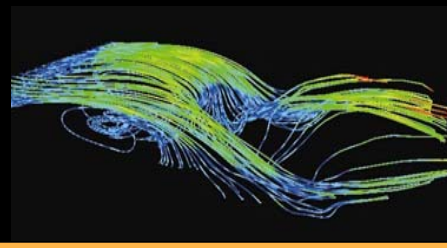
- *P*, pressure
- *ρ*, density
- *M*, mach number

Shock Detection



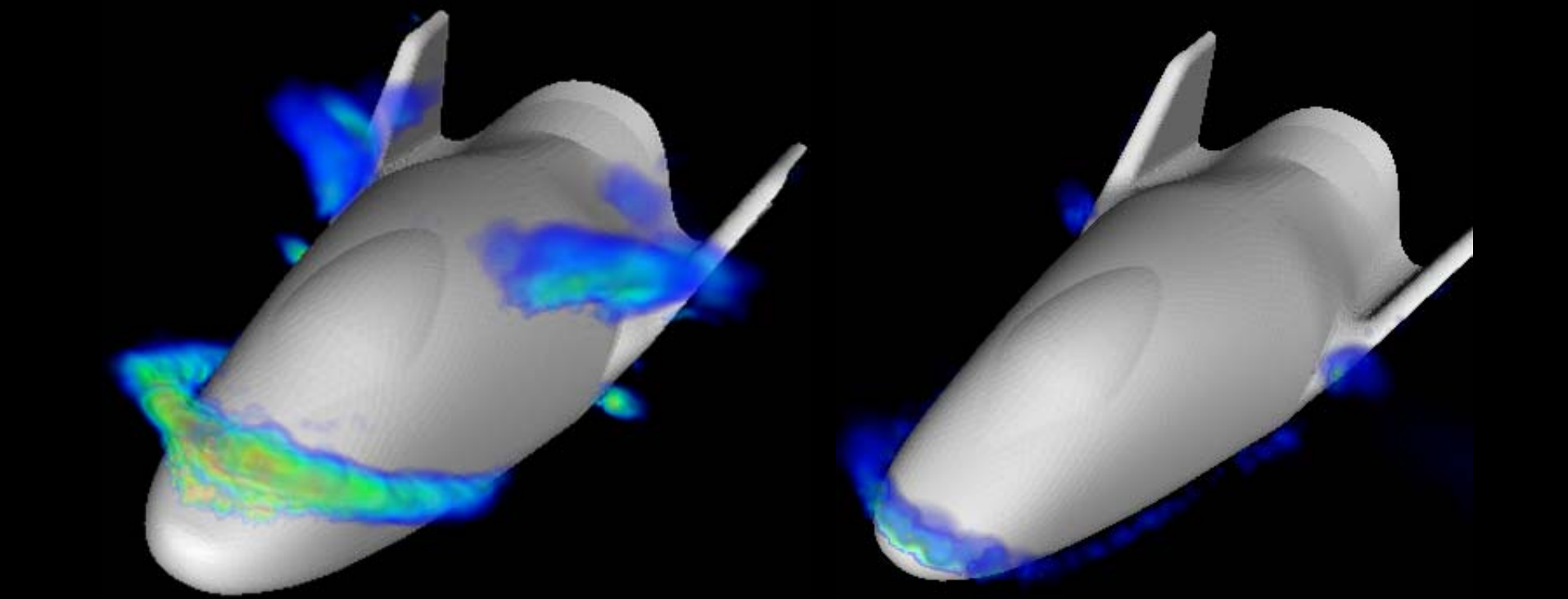
	P	ρ	M
E_1	*Positive E_1 : Compression shock*	*Positive E_1 : Expansion shock*	*Positive E_1 : Expansion shock*
E_2	*Negative E_2 : Expansion shock*	*Negative E_2 : Compression shock*	*Negative E_2 : Compression shock*
E_3	*Representing shear shock or contact discontinuities orthogonal to the flow direction*		

Shock Detection

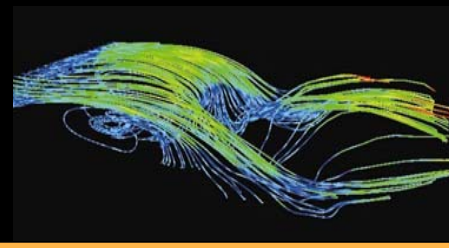


Compression Shock of X38

Expansion Shock of X38



Shock Detection: *U = P*

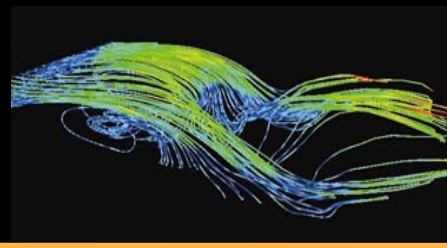


$$p = 0.4 \times \left(e - \frac{0.5}{\rho} \times \left(m_x^2 + m_y^2 + m_z^2 \right) \right)$$

p	Pressure
e	Energy
ρ	Density
m_x^2, m_y^2, m_z^2	x, y, z components of momentum

$$\begin{aligned} E_{1,2} &= (u,v,w) \cdot \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right) \\ &= 0.4 \times u \cdot \left[\frac{\partial e}{\partial x} - 0.5 \times \frac{\partial \rho}{\partial x} \times (u^2 + v^2 + w^2) - \rho \times \left(u\frac{\partial u}{\partial x} + v\frac{\partial v}{\partial z} + w\frac{\partial w}{\partial z} \right) \right] \\ &\quad + 0.4 \times v \cdot \left[\frac{\partial e}{\partial y} - 0.5 \times \frac{\partial \rho}{\partial y} \times (u^2 + v^2 + w^2) - \rho \times \left(u\frac{\partial u}{\partial y} + v\frac{\partial v}{\partial y} + w\frac{\partial w}{\partial y} \right) \right] \\ &\quad + 0.4 \times w \cdot \left[\frac{\partial e}{\partial z} - 0.5 \times \frac{\partial \rho}{\partial z} \times (u^2 + v^2 + w^2) - \rho \times \left(u\frac{\partial u}{\partial z} + v\frac{\partial v}{\partial z} + w\frac{\partial w}{\partial z} \right) \right] \end{aligned}$$

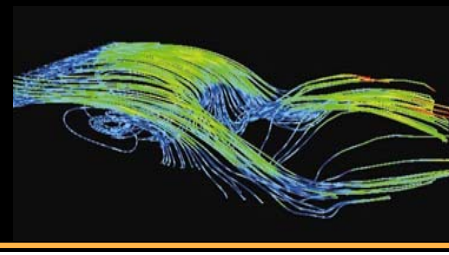
Shock Detection: $U = \rho$



$$E_{1,2} = (u, v, w) \cdot \left(\frac{\partial \rho}{\partial x}, \frac{\partial \rho}{\partial y}, \frac{\partial \rho}{\partial z} \right)$$

$$= u \cdot \frac{\partial \rho}{\partial x} + v \cdot \frac{\partial \rho}{\partial y} + w \cdot \frac{\partial \rho}{\partial z}$$

Shock Detection: *U = M*



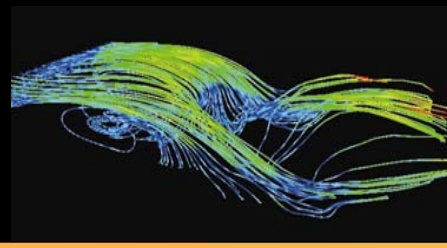
$$M = \frac{\sqrt{m_x^2 + m_y^2 + m_z^2}}{\rho \times \sqrt{1.4 \times \frac{p}{\rho}}} = \frac{\sqrt{m_x^2 + m_y^2 + m_z^2}}{\sqrt{1.4 \times p \times \rho}}$$

$$\frac{\partial M}{\partial x} = \frac{1}{2.8 \times p \times \rho \times M}\left[2m_x\left(\frac{\partial \rho}{\partial x}u + \rho\frac{\partial u}{\partial x}\right) + 2m_y\left(\frac{\partial \rho}{\partial x}v + \rho\frac{\partial v}{\partial x}\right) + 2m_z\left(\frac{\partial \rho}{\partial x}w + \rho\frac{\partial w}{\partial x}\right) - 1.4 \times \left(\frac{\partial p}{\partial x} \times \rho \times M^2 + p \times \frac{\partial \rho}{\partial x} \times M^2\right)\right]$$

$$\frac{\partial M}{\partial y} = \frac{1}{2.8 \times p \times \rho \times M}\left[2m_x\left(\frac{\partial \rho}{\partial y}u + \rho\frac{\partial u}{\partial y}\right) + 2m_y\left(\frac{\partial \rho}{\partial y}v + \rho\frac{\partial v}{\partial y}\right) + 2m_z\left(\frac{\partial \rho}{\partial y}w + \rho\frac{\partial w}{\partial y}\right) - 1.4 \times \left(\frac{\partial p}{\partial y} \times \rho \times M^2 + p \times \frac{\partial \rho}{\partial y} \times M^2\right)\right]$$

$$\frac{\partial M}{\partial z} = \frac{1}{2.8 \times p \times \rho \times M}\left[2m_x\left(\frac{\partial \rho}{\partial z}u + \rho\frac{\partial u}{\partial z}\right) + 2m_y\left(\frac{\partial \rho}{\partial z}v + \rho\frac{\partial v}{\partial z}\right) + 2m_z\left(\frac{\partial \rho}{\partial z}w + \rho\frac{\partial w}{\partial z}\right) - 1.4 \times \left(\frac{\partial p}{\partial z} \times \rho \times M^2 + p \times \frac{\partial \rho}{\partial z} \times M^2\right)\right]$$

Shock Detection



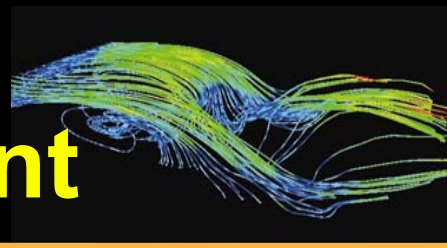
Bunning's technique [Haimes 1999]

- Normal mach number
- Stationary shock

$$M_n = \frac{\vec{V}}{a} \cdot \frac{\nabla p}{|\nabla p|} = \vec{M} \cdot \frac{\nabla p}{|\nabla p|} = 1.0$$

M_n	Normal Mach number
$\vec{V}$	Velocity vector
a	Speed of sound
∇p	Pressure gradient
M	Mach number
$\vec{M} = \frac{\vec{V}}{\|\vec{V}\|} M$	Mach vector

Calculation of Velocity gradient



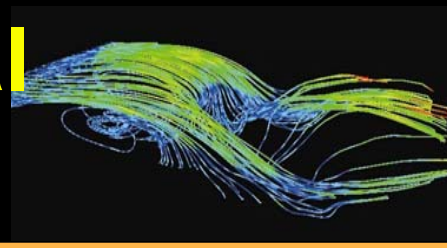
In the tetrahedral cell, 4 nodes are used to calculate velocity gradient

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} + \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} c_1 & \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ c_2 & \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ c_3 & \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} c_1 \\ \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_1 & z_3 \\ 1 & x_4 & y_1 & z_4 \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

Computing Features in Functional Domain



Possible to compute features analytically

$$s = f(x)$$

Functional representation $\hat{s}$ ***using RBF***

$$\hat{s}(x) = w_0 + \sum_{i=1}^{M} \lambda_i \Phi(r)$$

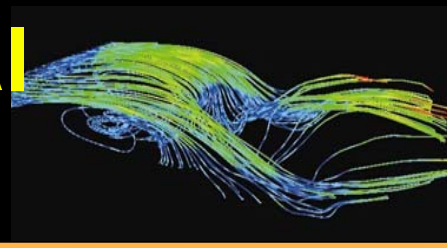
$$\Phi(r_i) = \exp\left(-\frac{r_i^2}{2\sigma_i^2}\right)$$

$$r_i = \|x - \mu_i\|$$

TACC

VIS 2004 austin, texas

Computing Features in Functional Domain

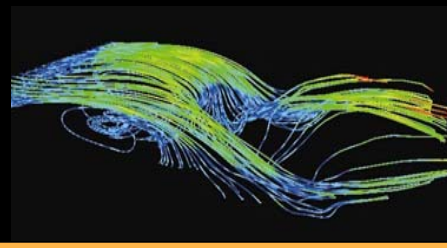


Partial derivatives of function s

$$\nabla s(x) = -\sum_{i=1}^{M} \frac{x - \mu_i}{\sigma_i^2} \lambda_i \exp\left(-\frac{\|x - \mu_i\|}{2\sigma_i^2} \right)$$

Approximations of partial derivatives are used to compute a wide variety of features

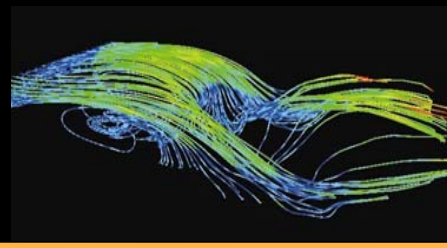
References



Feature detection

- Hesselink, L. and Helman, J., Evaluation of flow topology from numerical data, AIAA-87-1811, 1987
- Helman, J., Hesselink, L., “Representation and display of vector field topology in fluid flow data sets”, IEEE Computer, Vol. 22 , Issue 8, page 27-36, 1989
- Helman, J., Hesselink, L., “Visualizing vector field topology in fluid flows”, IEEE Computer Graphics & Applications, Vol. 11, Issue 3, page 36-46, 1991
- Globus et al., “A tool for visualizing the topology of three-dimensional vector fields”, Proceedings of the 2nd conference on Visualization, 1991
- J. Jeong and F. Hussain, "On the Identification of a Vortex“, J. Fluid Mech. 285, pp. 69-94, 1995

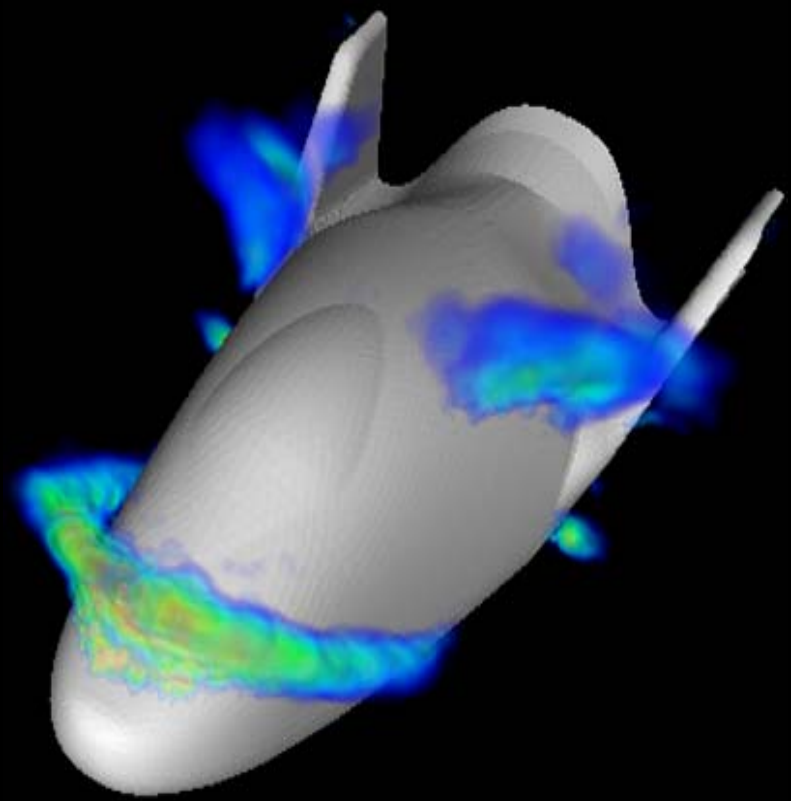
References

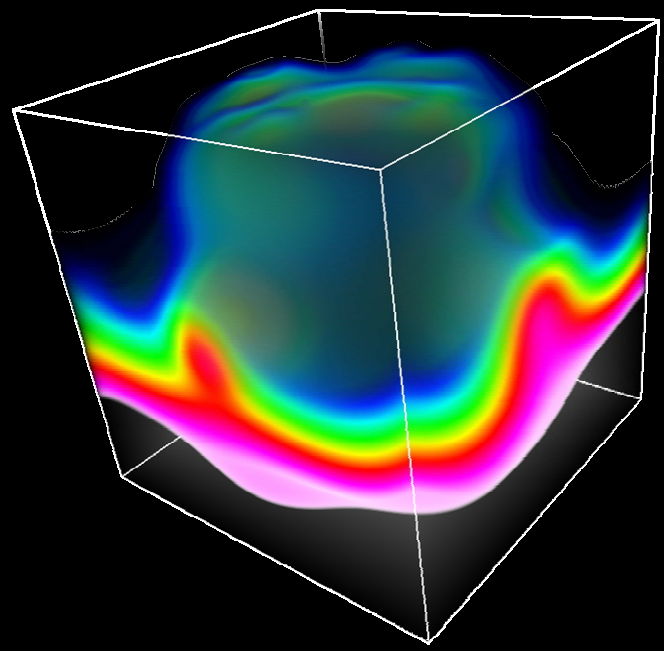


Feature detection

- D. Lovely and R. Haimes, "Shock Detection from Computational Fluid Dynamics Results", AIAA Paper 99-3285, 1999
- R. Haimes and D. Kenwright, “On the Velocity Gradient Tensor and Fluid Feature Extraction", AIAA Paper 99-3288, 1999
- *Deborah Silver, Xin Wang, “*Tracking Scalar Features in Unstructured Datasets “, Proceedings of IEEE Visualization 1998
- Marcum, D. L. and Gaither, K. P., "Solution Adaptive Unstructured Grid Generation Using Pseudo-Pattern Recognition Techniques," AIAA-97-1860 , 1997

Applications

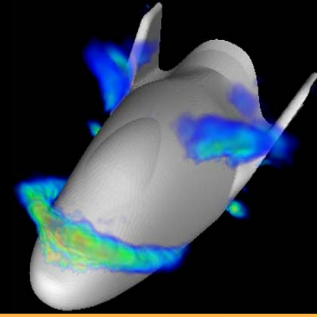




Kelly Gaither

Texas Advanced Computing Center

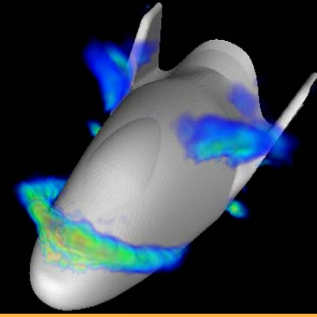
Meshless Method



Shift from scattered data approximation to numerical solution of partial differential equations (PDEs)

Mesh-free nature of RBFs: Motivation for dealing with PDEs

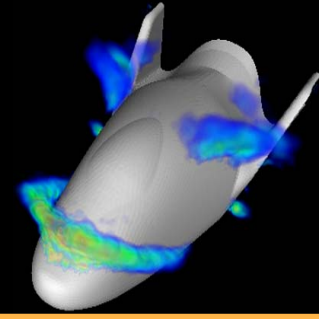
Meshless Method



Scattered data approximation

- Globally supported RBFs for relative small number of points (400-500)
- For large datasets (> 10,000)
  - *Domain decomposition using fast multipole method [Beatson et al. 2000]*
  - *Domain localization [Nielson 1993]*
  - *Partition of unity [Ohtake et al. 2003]*

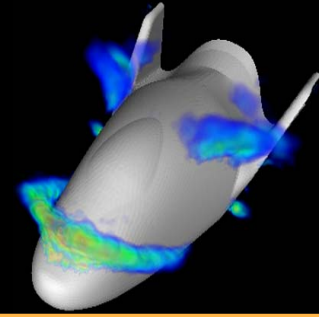
Meshless Method



Scientific visualization

- Providing a crucial role in the development and understanding of computational simulations
- Current focus:
  - *improved understanding of results from traditional grid-based techniques (e.g., rectilinear, tetrahedral, curvilinear, and hierarchical grid structures)*

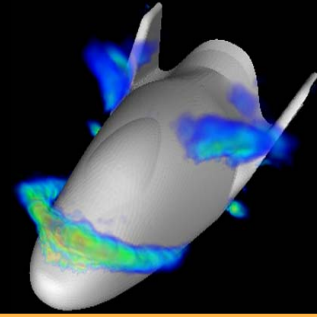
Meshless Method



Traditional methods for modeling numerical system

- Generation of an underlying grid structure (e.g., finite element methods (FEM), finite volume methods (FVM), and finite difference methods (FDM))
- Time consuming creation of “good” meshes
- Prohibitively expensive to solve excessive change scale model by traditional FEMs (e.g., crack propagation)

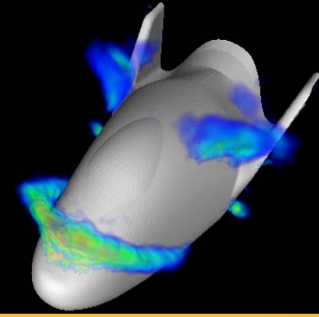
Meshless Method



Traditional methods for modeling numerical system:

- Construction of airtight geometry
  - *free of cracks or holes*
- Generation of surface mesh given
  - *geometry description*
  - *set of point distributions*
- Generation of a volume with elements (e.g., hexahedra, pyramids, prisms, tetrahedra) adhering to desired spatial transitions

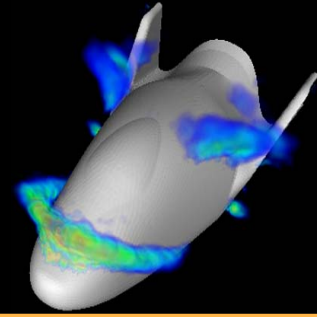
Meshless Method



Traditional methods for modeling numerical system

- Grid generation technique balancing
  - *Manpower time required to generate the grid*
  - *Resulting size of data set with proper resolution and spacing needed to maintain accuracy and convergence*
- Storage of a set of discretized points with either implicit or explicit connectivity

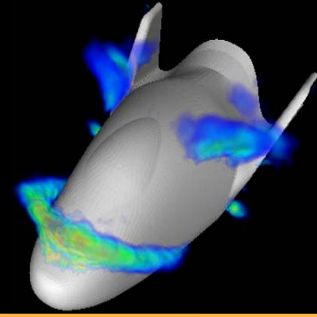
Meshless Method



Additional reasons for exploring meshless techniques [Kansa, 1990]

- Too much user tuning for multi-dimensional moving mesh schemes
- Modification of physics to accommodate the numerical schemes, rather than modification of numerical schemes to accommodate physics
- Very slow convergence of numerical scheme

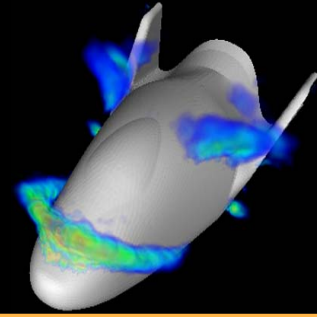
Meshless Method



Development of "meshless" methods

- Growing research area
- Providing a fundamental shift away from the traditional grid-based simulation techniques
- Modeling the domain of interest
- Governing equations of a series of basis functions

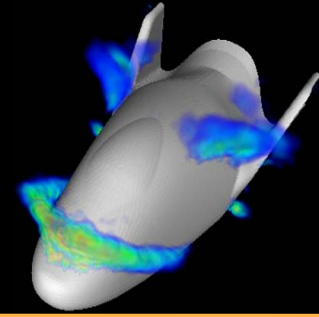
Meshless Method



Advantages over the traditional grid-based techniques

- No dependence on a large underlying grid structure with explicit connectivity
- Increasing rates of convergence when solving the numerical PDEs
- Higher order continuity across the global domain of interest
- Superior methods for computing physical systems that have excessive variation in scale and large deformations (e.g., crack propagation and fragmentation)

Meshless Method



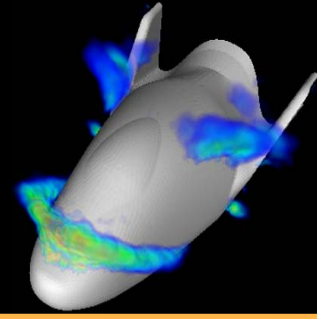
Meshfree models and particle systems

- Often better suited to cope with geometric changes in the underlying domain (e.g., free surfaces and large deformations)

Elimination of cost of grid generation

Weak dependence on a coarsely defined background mesh that support numerical quadrature calculations

Meshless Method



Pure functional representations

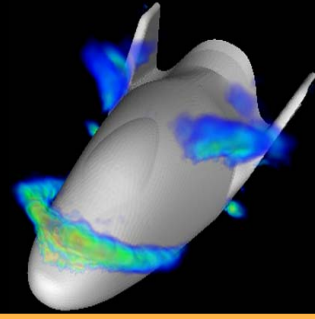
- Gaussian basis functions
  - *Relatively localized basis set for modeling rapidly varying functions*
  - *Easy addition in the regions of clusters*
  - *Most suitable for the evaluation of quantum mechanical operators between wave functions*

$$G(\alpha,l,m,n,x,y,z) = N \cdot e^{(-\alpha r^2)} x^l y^m z^n$$

N	= normalization factor
x, y, z	= coordinates of the electron
α	= contraction coefficient
r	= radius of the electron orbit from the nucleus
l, m, n	= power coefficients

Maybe this equation is not needed

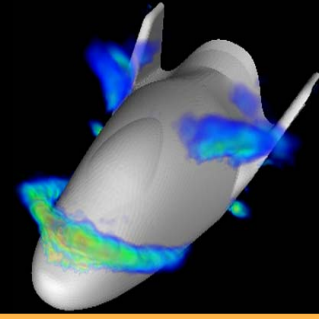
Meshless Method: Pure Functional Representations



Wavelets

- Excellent mathematical framework for the systematic decomposition of data into levels of detail (LOD)
- Perfect reconstruction while simultaneously maintaining a sparse data representation
- Multi-wavelets
  - *Approximation of a scalar function by expanding several scaling and wavelet functions*
  - *Direct construction of a vector wavelet transform*

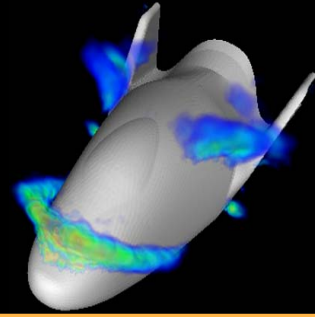
Meshless Method: Pure Functional Representations



Smooth Particle Hydrodynamics (SPH)

- Lagrangian method for modeling a variety of computational fluid dynamics simulations
- Approximation of materials by particles that are free to move rather than being fixed at grid locations
- Convert PDEs governing forces (e.g. gravitational forces) into equations of motion
- Advantages
  - *Handles momentum dominated flows well*
  - *Natural modeling for complex free surfaces*
  - *Easy addition of complicated multi-phase physics, realistic equations of state, compressibility, radiation, and solidification*
  - *Easy handing of complex geometries in two and three dimensions*

Meshless Method: Hybrid Methods

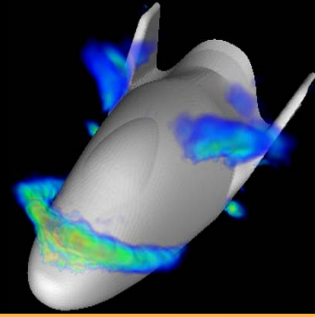


Functions + Coarse background mesh

Radial basis functions (RBFs)

- Numerical solutions of PDEs via the method of collocation by Kansa
- No requirement of adaptively finer controls of both space and polynomial order
- Two-dimensional incompressible Navier-Stokes equations [Florez and Power, 2002]
- Elliptic boundary value problems [Chan and Ke, 2002]
- Transient nonlinear Poisson problem [Balakrishnan et al. 2002]

Meshless Method: Hybrid Methods



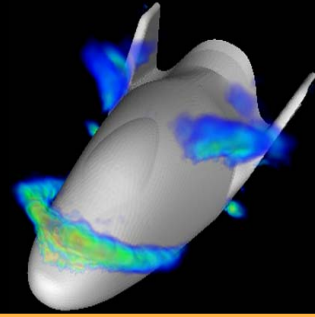
Element-Free Galerkin methods (EFGM)

- Solve PDEs using a
  - *Set of points in the domain of interest*
  - *geometric description of the body to form a discrete model*
- Background grid to support numerical quadrature calculations
- Moving Least Squares Methods as a mean for approximating the function
- Computationally expensive
- Advantages over FEM
  - C^1 *continuity over the domain*
  - *Rapid convergence rates*
  - *No dependence on an explicitly defined grid structure*
- Attractive in the analysis of crack problem without interactive refining underlying mesh

TACC

VIS 2004 austin, texas

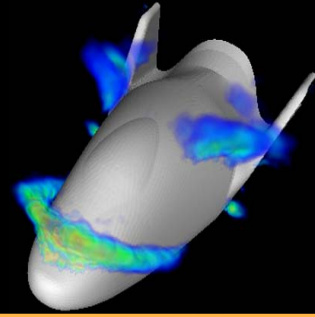
Meshless Method: Hybrid Methods



h-p cloud

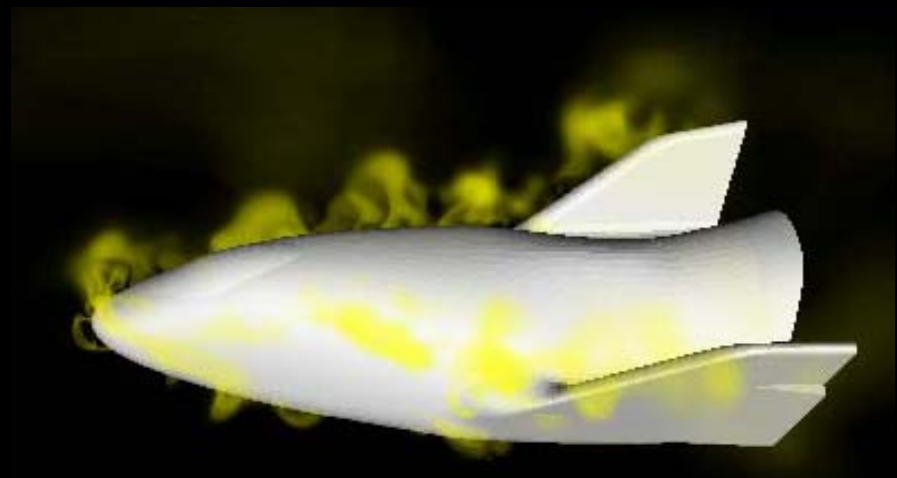
- Relies on key principal of a signed partition of unity
- As smooth as desired
- No need to partition the domain of interest into smaller domain
- Only an arbitrarily placed set of nodes needed
- Moving Least Squares Method for all numerical interpolation
- Suitable for large deformation problems exhibiting a significant change in shape by some load or force
- Crack propagation and vehicle crashworthiness
- Avoids the solution instabilities due to mesh distortion

Examples of using Functional Encoding



X38 Crew return vehicle

- Tetrahedral finite element viscous calculation on geometry
- 1,943,483 tetrahedra at a 30 degree angle of attack
- Computed at Engineering Research Center at Mississippi State University by the Simulation and Design Center



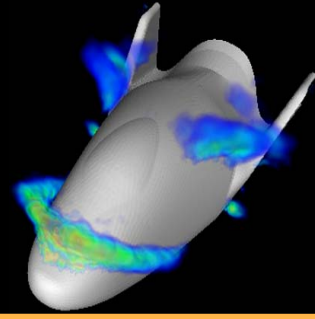
2,932 RBFs

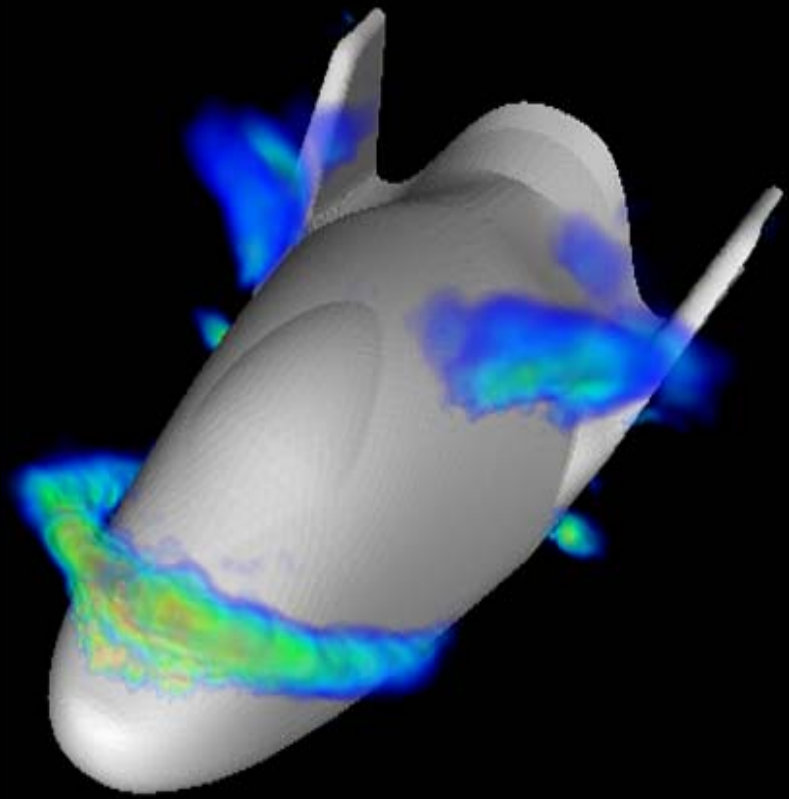
Shock Volume Rendering representing normal Mach number around 1.0

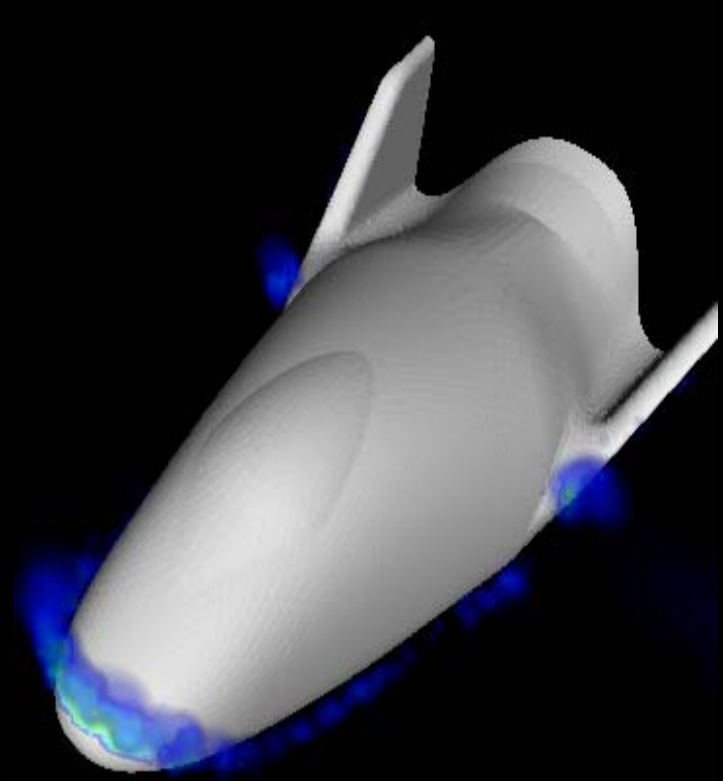
TACC

VIS 2004 austin, texas

Examples of using Functional Encoding







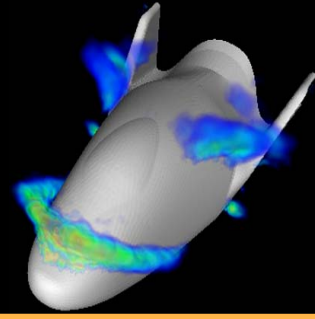
Compression Shock
5,703 RBFs

Expansion Shock
6,750 RBFs

TACC

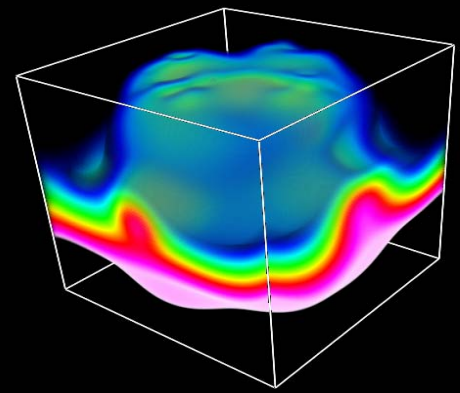
VIS 2004 austin, texas

Examples of using Functional Encoding

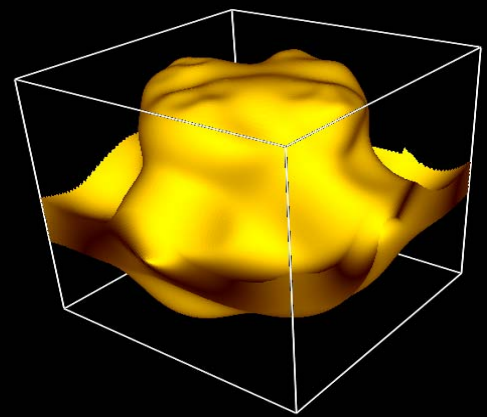


Natural Convection in a box

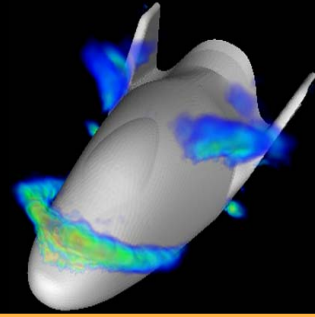
- 80th time step of temperature from a natural convection simulation
- 48,000 tetrahedral elements
- Developed at The University of Texas at Austin



435 RBFs

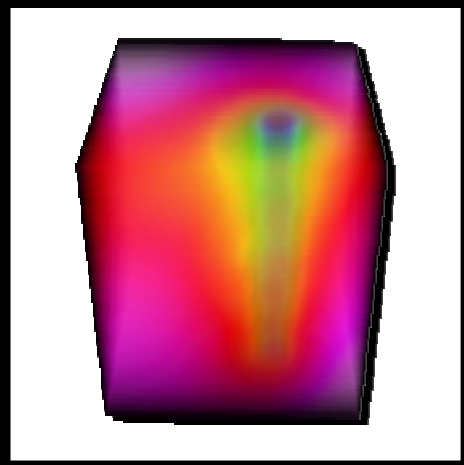


Examples of using Functional Encoding

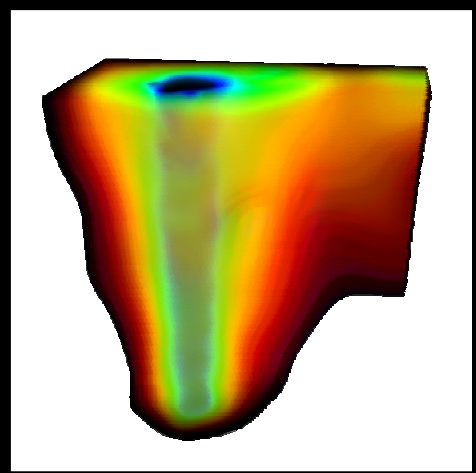


Black Oil Reservoir Simulation

- A simulation for prediction of placement of water injection wells to maximize oil from production wells
  - *156,642 tetrahedra containing water pressure values for the injection well*
  - *Computed by the Center for Subsurface Modeling at The University of Texas at Austin*



458 RBFs

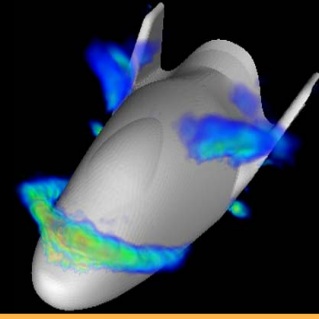


222 RBFs

TACC

VIS 2004 austin, texas

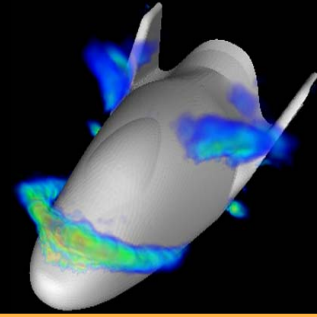
References



Meshless techniques

- G.E. Fasshauer, "Solving partial differential equations by collocation with radial basis functions", Surface fitting and multiresolution methods, pp. 131-138, 1997
- G.E. Fasshauer, "On the numerical solution of differential equations with radial basis functions", Boundary Element Technology XIII, pp. 291-300, 1999
- G.E. Fasshauer, "Nonsymmetric Multilevel RBF Collocation within an Operator Newton Framework for Nonlinear PDEs", Trends in Approximation Theory, pp. 103-112, 2001
- Y.C. Hon and Z.M. Wu, “A quasi-interpolation method for solving stiff ordinary differential equations”, International Journal for Numerical Methods in Engineering, Vol. 48, pp. 1187-1197, 2000

References



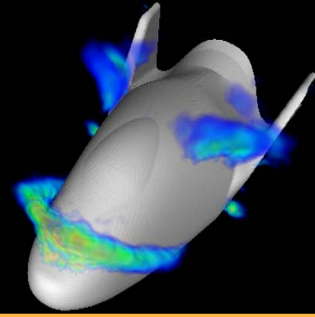
Meshless techniques

- E.J. Kansa, “A scattered data approximation scheme with applications to computational fluid-dynamics - I: Surface Approximations and Partial Derivative Estimates”, Computers and Mathematics with Applications, Vol. 19, Num. 8, pp. 127-145, 1990
- E.J. Kansa, “A scattered data approximation scheme with applications to computational fluid-dynamics - II: Solutions to parabolic, hyperbolic and elliptic partial differential equations”, Computers and Mathematics with Applications, Vol. 19, Num. 8, 147-161, 1990
- E.J. Kansa and Y.C. Hon, “Circumventing the ill-conditioning problem with multiquadric radial basis functions: applications to elliptic partial differential equations”, Computers and Mathematics with Applications, Vol. 29, pp. 123-137, 2000
- B. Forberg and T.A. Driscoll, “Interpolation in the limit of increasingly flat radial basis functions”, Computational and Applied Mathematics, Vol. 43, pp. 413-422, 2002

TACC

VIS 2004 austin, texas

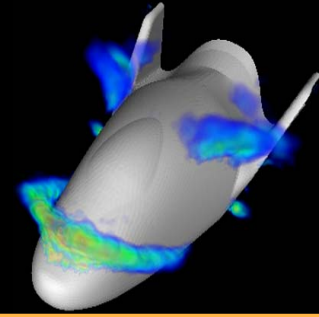
References



Meshless techniques

- Chen et al., “Dual reciprocity method using for Helmholtz-type operators”, Boundary Elements, Vol. 20, pp. 495-504, 1998
- W. Chen and M. Tanaka, “A meshless, integration-free, and boundary-only RBF technique”, Computational Mathematics Applications, Vol. 43, pp. 379-391, 2002
- O. M. Nielsen, “Wavelets in Scientific Computing”, in Department of Mathematical Modeling, Technical University of Denmark, pp. 224, 1998
- J. Fowler and Hua L., “Omnidirectionally Balanced Multiwavelets for Vector Wavelet”, Proceedings of Data Compression Conference, 2002
- J. J. Monaghan, “Smoothed particle hydrodynamics”, Journal of Computational Physics, Vol. 110, pp. 229-406, 1994

References



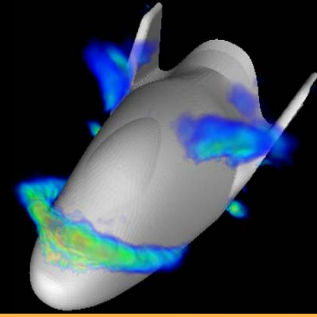
Meshless techniques

- W. F. Florez and Power, H., “DRM multi-domain mass conservative interpolation approach for the BEM solution of the two-dimensional Navier-Stokes equations”, Computer & Mathematics with Applications, Vol. 43, 2002
- C. Y. Chan and Ke L., “Numerical computations for singular semilinear elliptic boundary value problems”, Computer & Mathematics with Applications, Vol. 43, 2002
- K. Balkrishnan et al. “An operator splitting method radial basis function method for the solution of transient nonlinear Poisson problems”, Computer & Mathematics with Applications, Vol. 43, 2002

TACC

VIS 2004 austin, texas

References



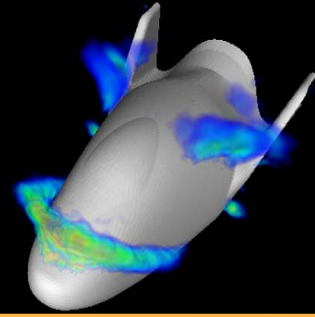
Meshless techniques

- M. Fleming et al., “Enriched Element-Free Galerkin Methods for Crack Tip fields”, International Journal for Numerical Methods in Engineering, Vol. 40, 1997
- N. Sukumar et al., “An element-Free Galerkin Method for Three-dimensional Fracture Mechanics”, Computational Mechanics, Vol. 20, pp.170-175, 1997
- T. Belytschko et al., “Element-Free Galerkin Methods”, International Journal for Numerical Methods in Engineering, Vol. 27, pp. 229-256, 1994
- T. Belytschko et al., “Crack propagation by elementfree galerkin methods”, Advanced Computational Methods for Material Modeling, Vol. 180, pp. 191-205, 1993

TACC

VIS 2004 austin, texas

References



Large scale techniques

- R.K. Beatson and W. A. Light, “Fast Evaluation of radial basis functions: Method for 2-dimensional polyharmonic splines”, IMA Journal of Numerical Analysis, Vol. 17, pp. 343-372, 1997
- Beatson et al., “Fast fitting of radial basis functions: Methods based on precondition GMRES iteration”, Advanced in Computational Mathematics, Vol. 11, pp. 253-270, 1999
- Beatson et al., “Fast Solution of the Radial Basis Function Interpolation Equations: Domain Decomposition Methods”, SIAM Journal of Scientific Computing, Vol. 22, Num. 5, pp. 1717-1740, 2000
- M.J.D. Powell, “Radial basis function methods for interpolation to functions of many variables”, Proceedings of the 5th Hellenic-European Conference on Computer Mathematics and its Applications, pp. 2-24, September, 2001