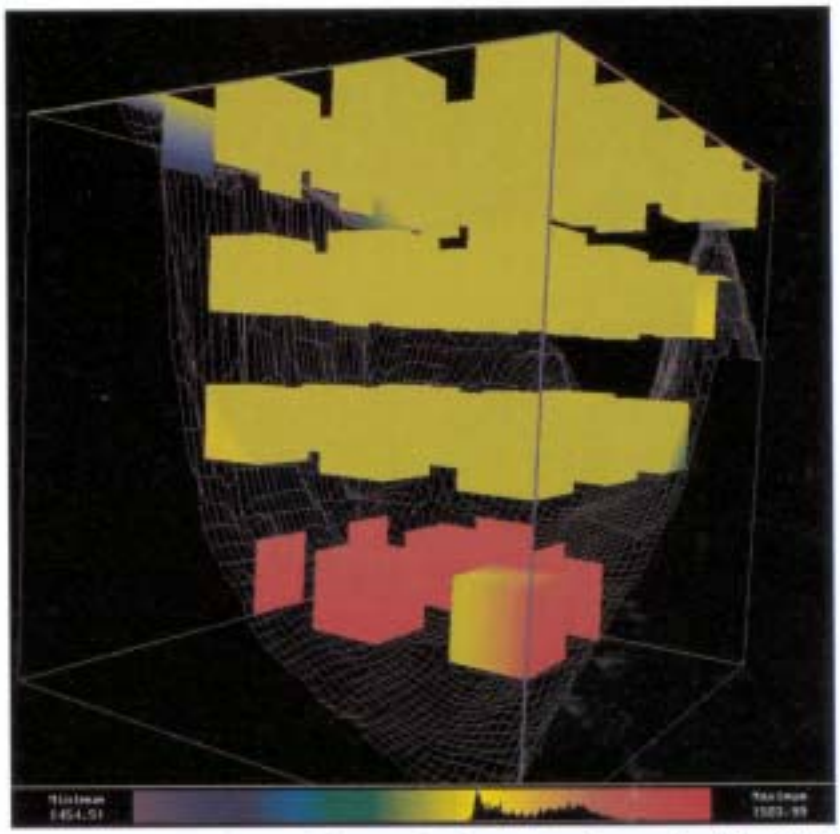
Title
A survey of grid generation methodologies and scientific visualization efforts

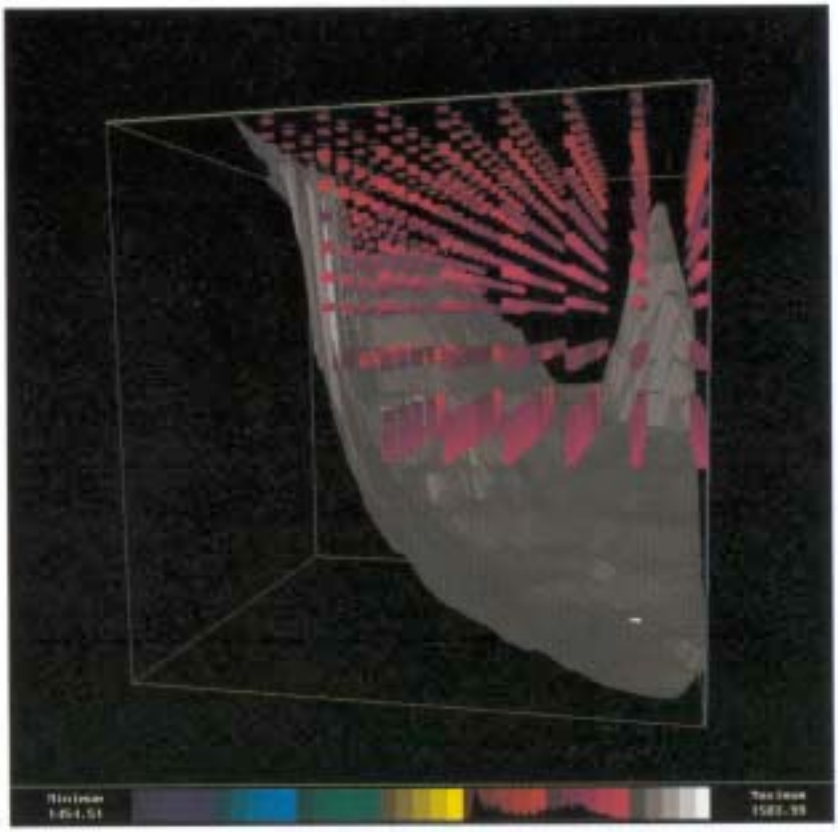
Authors
Hamann, Bernd
Moorhead, Robert J.

Publication Date
1997

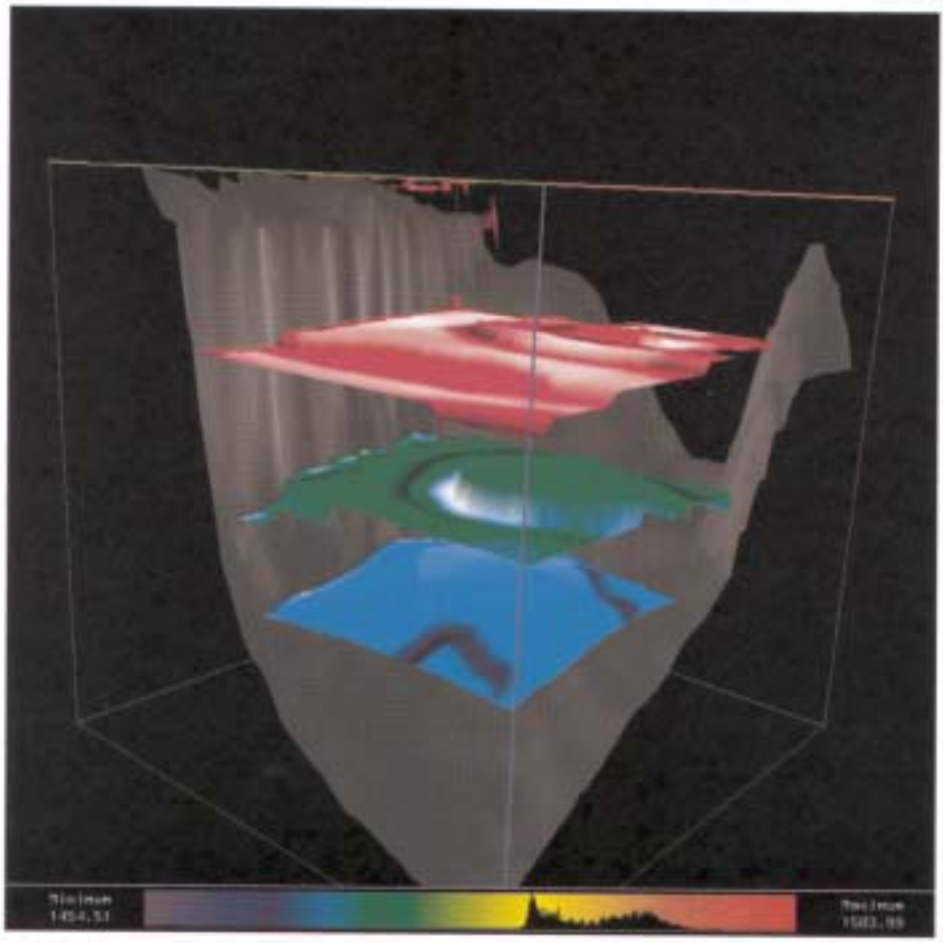
Peer reviewed



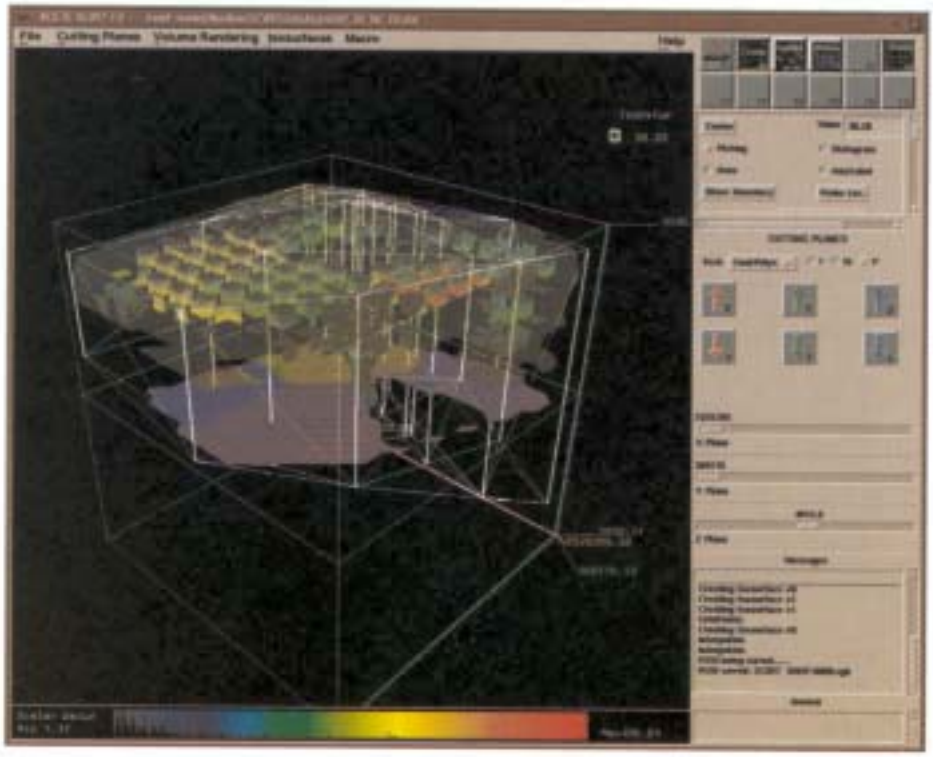
Color Plate 13: Minicubes method used for rendering sound speed with OVIRT system (uniform spacing and uniform cubes). (Figure 3.16, page 80.)



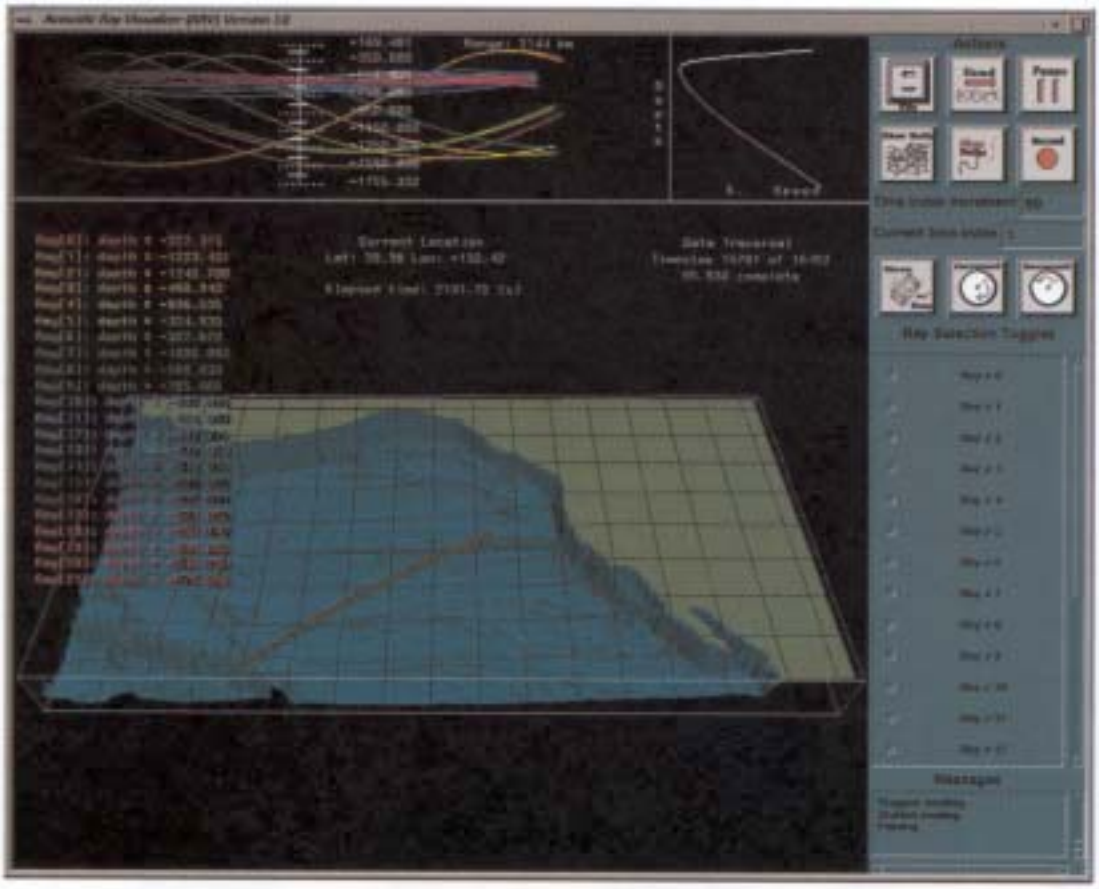
Color Plate 14: Minicubes method used for rendering sound speed with OVIRT system (nonuniform spacing and nonuniform cubes). (Figure 3.16, page 80.)



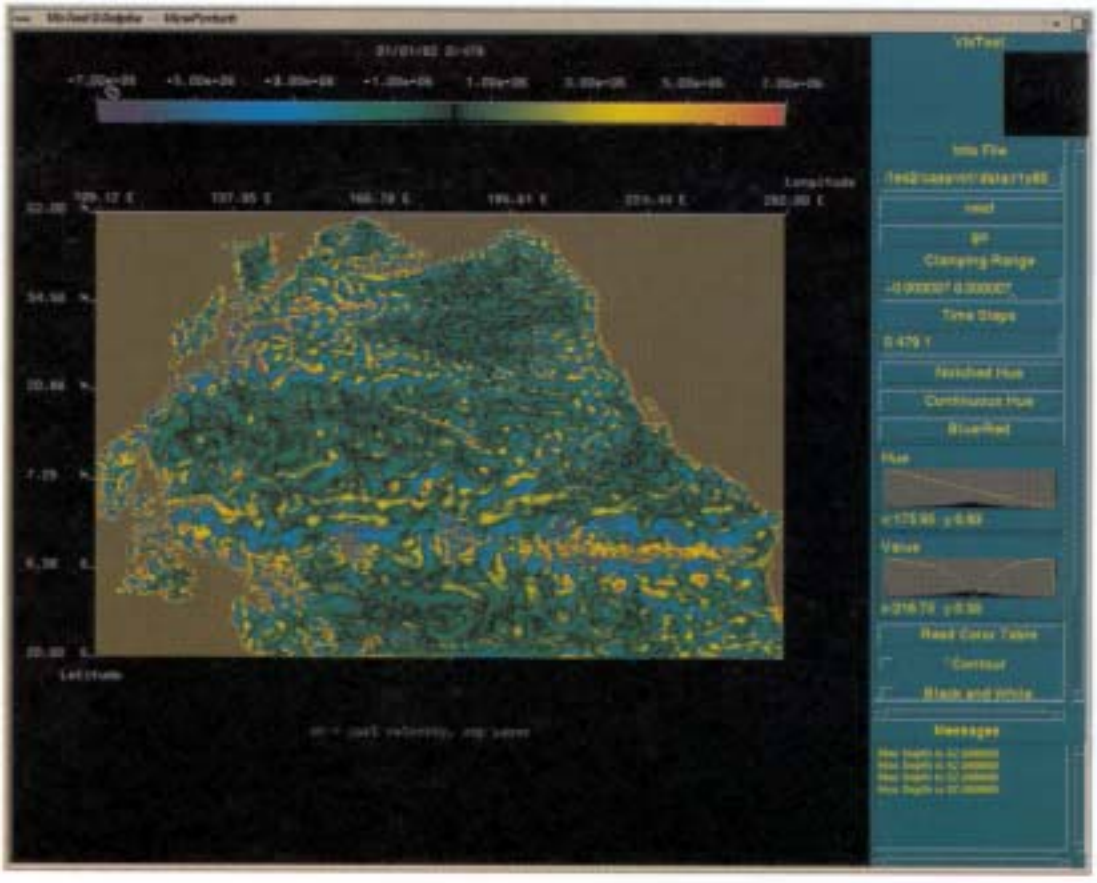
Color Plate 15: Sonic layer depth, deep sound channel axis, and critical depth generated and rendered with OVIRT system. (Figure 3.19, page 82.)



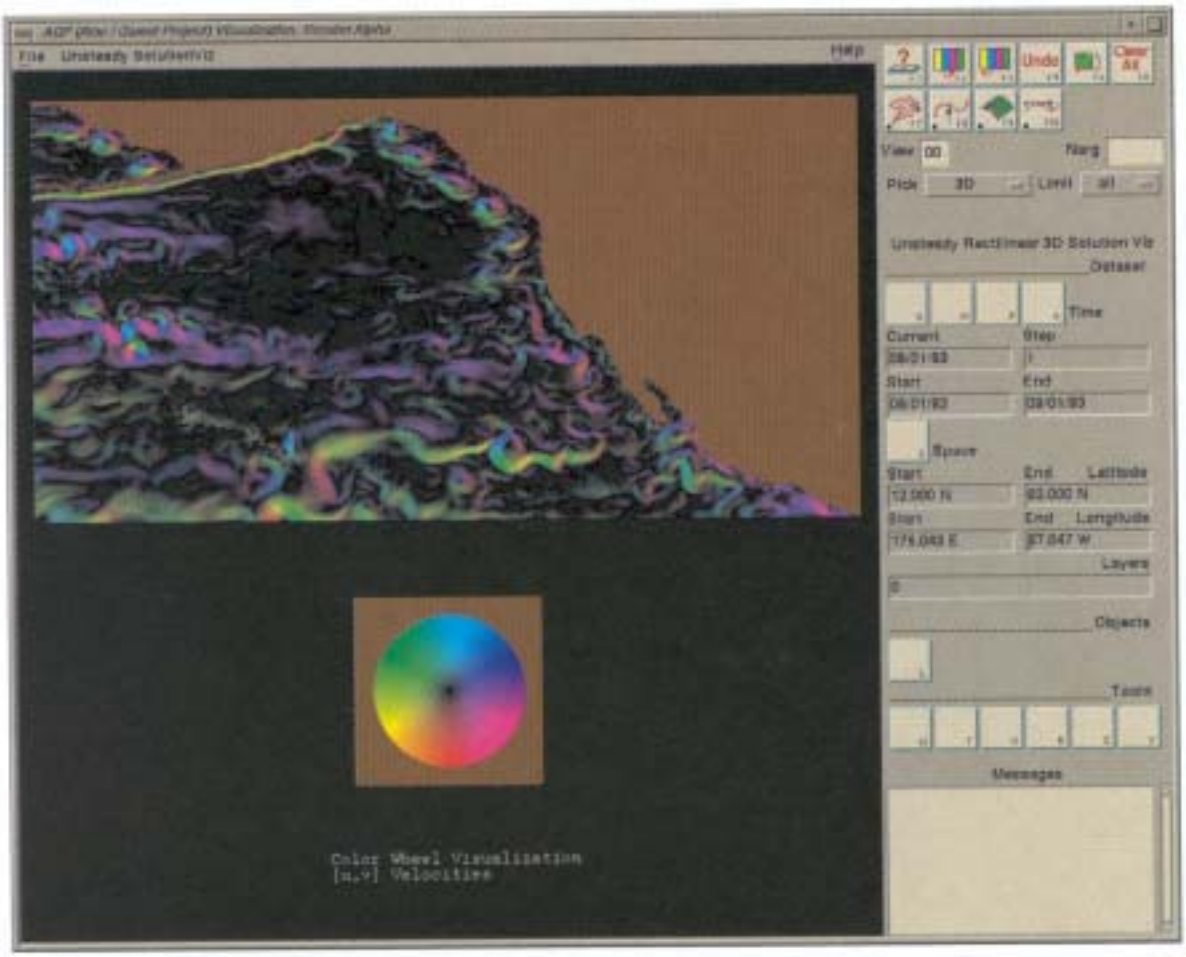
Color Plate 16: Transparent isosurfaces, minicubes, and cutting planes (clipped against convex hull) in a single image generated with SCIRT. (Figure 3.20, page 84.)



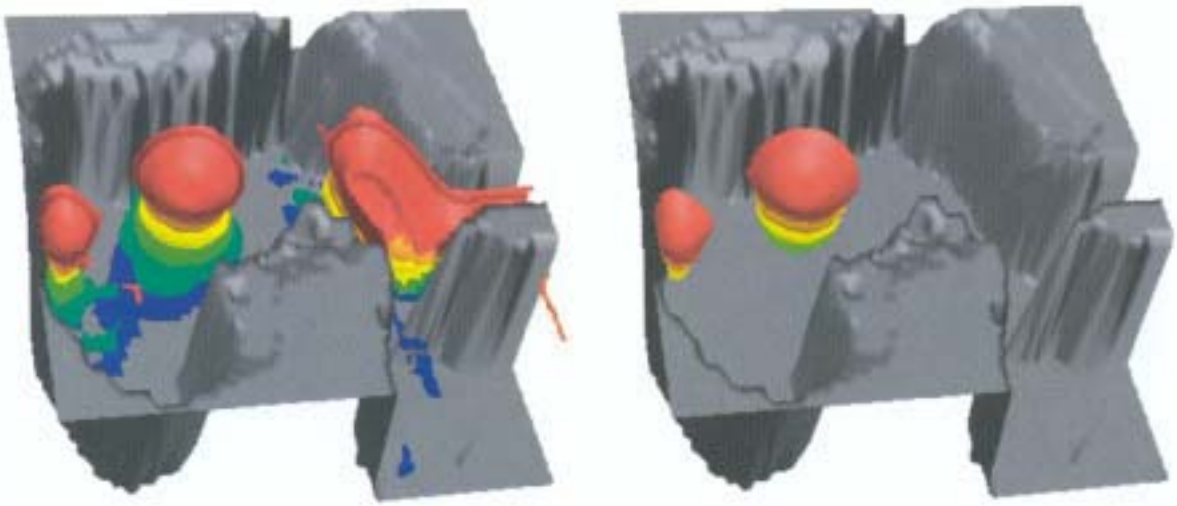
Color Plate 17: Snapshot of Acoustic Ray Visualizer (ARV). (Figure 3.21, page 86.)



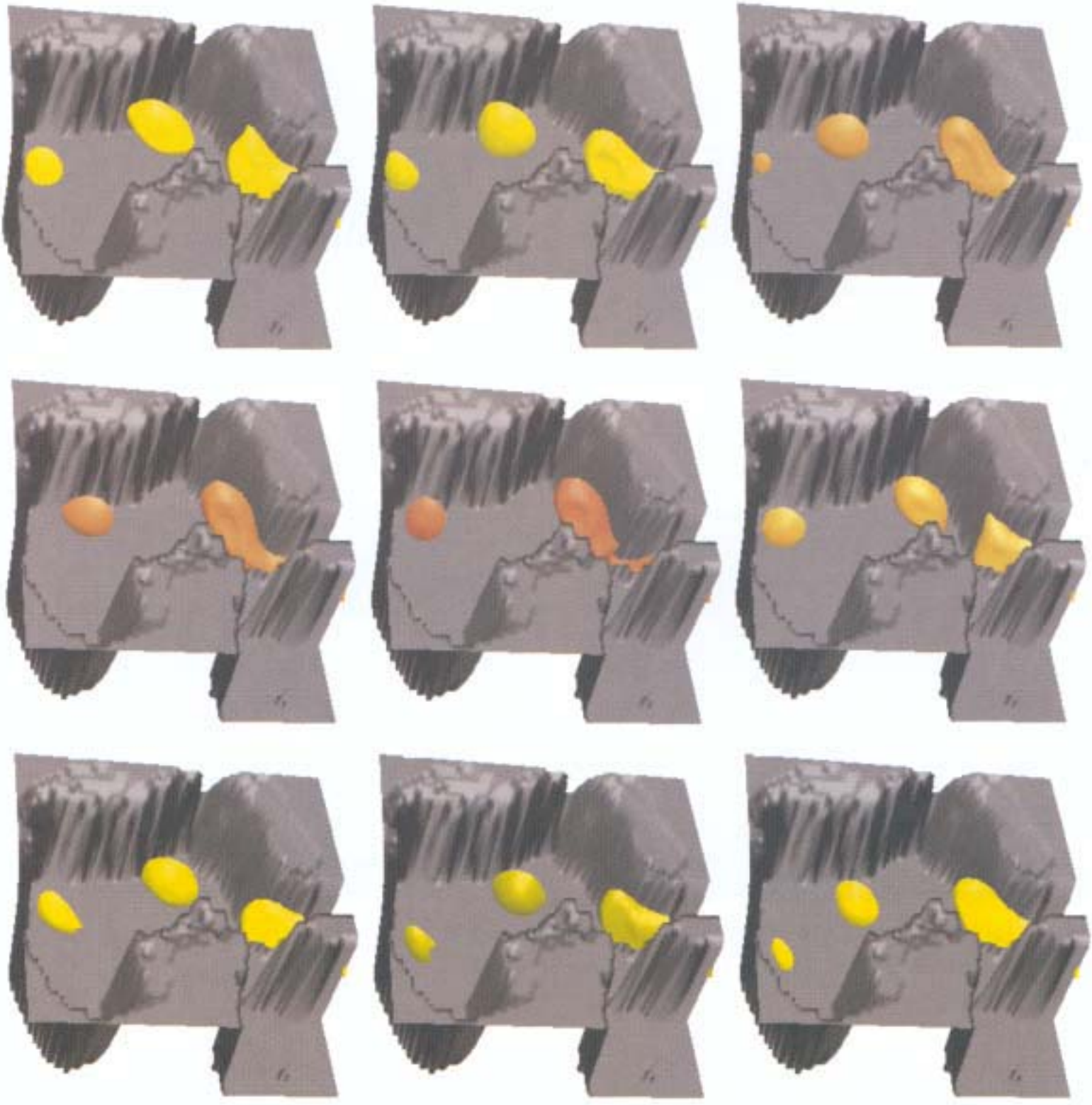
Color Plate 18: Relative vorticity for Pacific Ocean on January 1, 1982. (Figure 3.22, page 86.)



Color Plate 19: Flow in NE Pacific on August 1, 1983; direction of flow is mapped to a hue and magnitude to saturation and value; "rotated" color wheels close to coastline representing eddies. (Figure 3.23, page 87.)

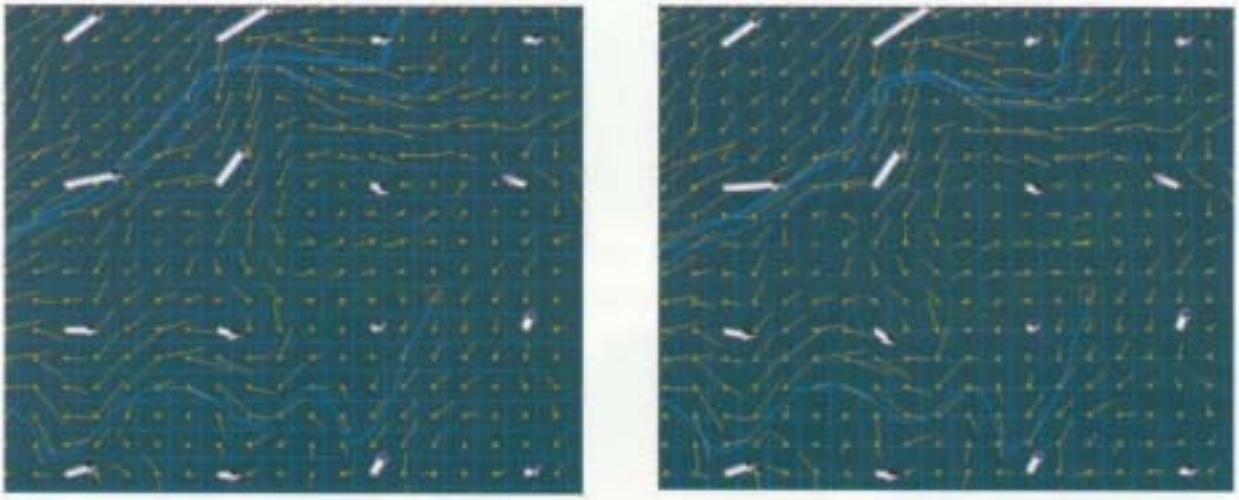


Color Plate 20: Difference in straightforward isosurfacing and enhanced edge-detection scheme. (Figure 3.24, page 88.)

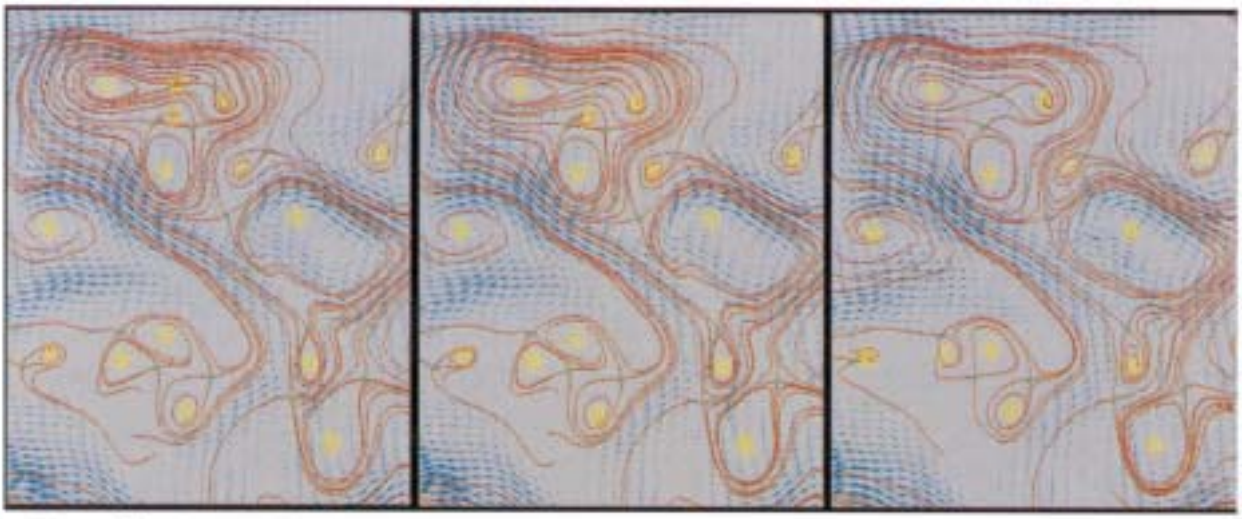


Color Plate 21: Example of eddies propagating through the Gulf of Mexico; sequence progresses from left to right and from top to bottom. (Figure 3.25, page 89.)

Color Plate 22: Layer thickness for NE Pacific Ocean. *(left)* original data; *(right)* reconstructed data (compression ratio 50:1). (Figure 3.28, page 93.)



Color Plate 23: Velocity data for NE Pacific Ocean. *(left)* original data; *(right)* reconstructed data (compression ratio 50:1). (Figure 3.29, page 94.)



Color Plate 24: Vector field topology extraction on reconstructed data. *(left)* original data; *(middle)* reconstructed data using 1/4 of the WT coefficients; *(right)* reconstructed data using 1/16 of the WT coefficients. (Figure 3.30, page 94.)

Chapter 3

A Survey of Grid Generation Methodologies and Scientific Visualization Efforts

Bernd Hamann and Robert J. Moorhead II

Abstract. *This survey chapter discusses recent and ongoing projects in grid generation and scientific visualization at the National Science Foundation (NSF) Engineering Research Center for Computational Field Simulation. The chapter provides a brief overview of standard grid generation techniques and summarizes the design requirements and current functionality of the National Grid Project, a universal, interactive grid generation system. Several projects that have led to powerful visualization systems are discussed.*

Keywords: advancing front, approximation, B-spline, CAD, Chimera grid, concurrent visualization, Delaunay triangulation, edge detection, elliptic grid generation, finite element method, hierarchical grid, hybrid grid, multiblock grid, NURBS, partial differential equation, ray casting, region growing, scattered data interpolation, scalar field visualization, structured grid generation, unstructured grid generation, vector field visualization, volume visualization, wavelets.

3.1 Introduction

The NSF Engineering Research Center for Computational Field Simulation (ERC-CFS) is an interdisciplinary research institution devoted to the simulation of field phenomena. Examples of field phenomena being studied are fluid flow, heat transfer, acoustic propagation, and electromagnetic radiation. Faculty members as well as full-time researchers from a variety of disciplines contribute to the mission of the ERC-CFS. Areas of expertise include aerospace engineering, electrical engineering, mechanical engineering, computer engineering, mathematics, and computer science. The ERC-CFS is divided into several research thrusts, among them the Grid Generation and the Scientific Visualization thrusts.

This survey chapter provides an overview of some of the efforts in these two thrusts and briefly outlines common grid generation methodologies.

Grid generation is the step required for solving the partial differential equations (PDEs) governing a model of some physical field phenomenon. Usually, CAD data describing some geometry is the input, and a finite set of points discretizing the given curves, surfaces, and possibly a surrounding volume, are the output. Scientific visualization techniques are required to analyze and display the—often time-varying—solution of the phenomenon being investigated. It is useful for visualization experts to have a thorough understanding of the grid types that are necessary to solve the PDEs and the ways different grid types are generated. This chapter describes the most common grid generation methods and grid types, discusses a universal grid generation system being developed at the ERC-CFS, and reports on various scientific visualization efforts. The chapter is divided into these sections:

- Grid generation methods and grid types used in CFS
- The National Grid Project (NGP)—A universal system for the generation of structured and unstructured grids (including geometry correction for CAD data with "gaps," intersecting surfaces, or overlapping surfaces)
- Visualization efforts at the ERC-CFS (scalar and vector field visualization with applications in aerodynamics, oceanography, acoustics, meteorology, and site characterization; feature extraction, wavelet-based data compression, and concurrent visualization)

3.2 Survey of Numerical Grid Generation Techniques

3.2.1 List of Grid Generation Methods and Grid Types

In this section, we discuss the main techniques used in numerical grid generation and provide a classification of the resulting grid types (see Figure 3.1). Essentially, one can distinguish between two grid types:

- *Structured grids*, classically generated by a combination of transfinite interpolation (TFI) and solving elliptic (or hyperbolic) systems of PDEs
- *Unstructured grids*, typically generated by a triangulation/tesselation algorithm (for example, Delaunay triangulation and Vornoï diagram) or an advancing front approach

In the grid generation community, the term *structured grid* is associated with a grid whose constituting elements (or cells) are topologically equivalent to a square (2D) or a cube (3D). In grid generation, the terms quadrilateral and hexahedron usually refer to a continuously deformed square or cube. When discretizing a single deformed square or cube by a structured grid, one usually chooses a curvilinear grid, that is, a grid whose edges are all straight line segments.

The connectivity among nodes in a structured grid is completely defined by the nodes' indices. For example, a node $\mathbf{x}_{i,j}$ in a 2D structured grid is connected with the nodes $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i+1,j}$, $\mathbf{x}_{i,j-1}$, and $\mathbf{x}_{i,j+1}$. This implied connectivity helps in designing highly efficient solution algorithms for the PDEs describing some phenomenon.

The term *unstructured grid* refers to any kind of grid that is not a structured grid, that is, is not of the finite difference type. The only unstructured grids of practical importance consist of triangles and/or quadrilaterals (2D) and tetrahedra, pentahedra (prisms), and/or hexahedra (3D). Thus, quadrilaterals and hexahedra can constitute an unstructured grid but the number of edges sharing a node is not restricted.

Combinations of structured and unstructured grid types are used as well. The most common grid types and combinations of various grid types are:

- *Structured grids*, consisting of quadrilateral and hexahedral elements whose node connectivity is implicitly defined by the nodes' indices
- *Multiblock structured grids*, consisting of multiple structured grids, each one associated with one of many *blocks* (connectivity among blocks not necessarily structured)
- *Unstructured grids*, consisting of quadrilateral, triangular, hexahedral, pentahedral, tetrahedral, and other types of polygonal/polyhedral elements; node connectivity explicitly defined for each node
- *Hybrid grids*, consisting of structured and unstructured grid regions
- *Chimera grids*, consisting of multiple structured grids with partially overlapping grid elements; overlap regions typically "resolved" using an appropriate interpolation schemes
- *Hierarchical grids*, generated by quadtree- and octree-like subdivision schemes (also referred to as *embedded grid* or *semi-structured grids*)

These grid types are illustrated for 2D in Figure 3.1.

Detailed information about these and other existing grid types and generation techniques/systems is provided in [5, 12, 19, 26, 53, 54].

3.2.2 Structured Grid Generation using TFI and Elliptic PDEs

One of the classical approaches for generating multiblock structured 2D (3D) grids for a finite space surrounding a geometry is based on TFI (see [10] and [15]) and solving elliptic systems of PDEs (see [53]). Once the block configuration around a geometry is established, an initial grid is generated inside each block by performing bilinear (trilinear) TFI of each block's boundary edges (faces). The TFI algorithm is performed in a discrete manner for a finite set of points on the boundary edges (faces) of each block. The points on the boundary edges (faces) are spaced according to a specified distribution function (for example, uniform in parameter space, uniform with respect to arc length, uniform with respect to integrated absolute curvature, and so on).

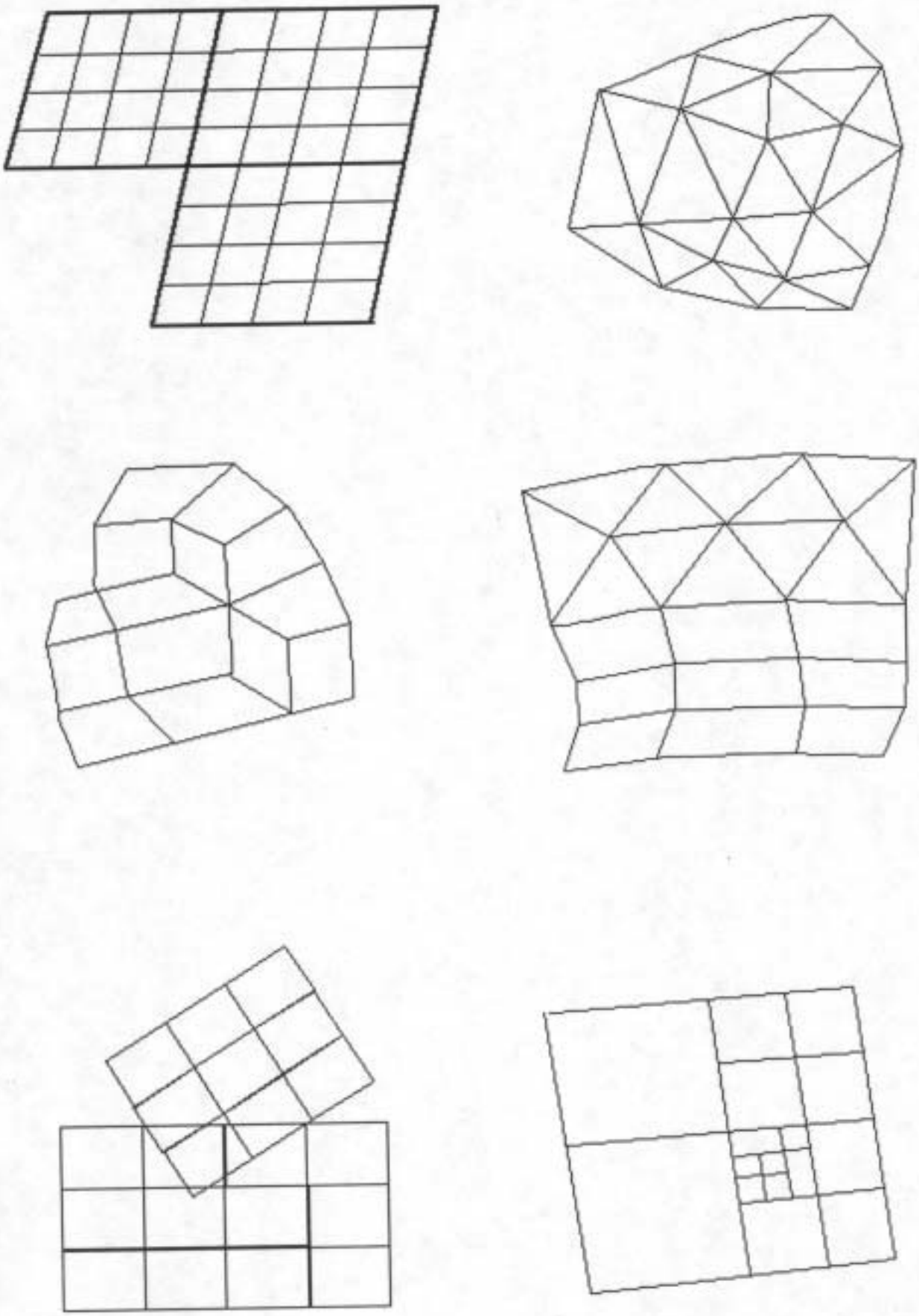


Figure 3.1: Multiblock structured *(upper-left)*, unstructured-triangular *(upper-right)*, unstructured-quadrilateral *(middle-left)*, hybrid *(middle-right)*, Chimera *(lower-left)*, and hierarchical grid *(lower-right)*.

In the 2D case, for example, the boundary curve points $\mathbf{x}_{I,0}$, $\mathbf{x}_{I,N}$, $\mathbf{x}_{0,J}$, and $\mathbf{x}_{M,J}$ associated with parameter values $(u_{I,0}, v_{I,0})$, $(u_{I,N}, v_{I,N})$, $(u_{0,J}, v_{0,J})$, and $(u_{M,J}, v_{M,J})$, $I = 0, \ldots, M$, $J = 0, \ldots, N$, define the initial interior grid points by applying TFI:

$$\begin{aligned}\mathbf{x}_{I,J} &= [(1-u_{I,J}) \quad u_{I,J}] \begin{bmatrix} \mathbf{x}_{0,J} \\ \mathbf{x}_{M,J} \end{bmatrix} + [\mathbf{x}_{I,0}\, \mathbf{x}_{I,N}] \begin{bmatrix} (1-v_{I,J}) \\ v_{I,J} \end{bmatrix} \\ &\quad -[(1-u_{I,J}) \quad u_{I,J}] \begin{bmatrix} \mathbf{x}_{0,0} & \mathbf{x}_{0,N} \\ \mathbf{x}_{M,0} & \mathbf{x}_{M,N} \end{bmatrix} \begin{bmatrix} (1-v_{I,J}) \\ v_{I,J} \end{bmatrix}, \end{aligned} \tag{3.1}$$

where $u_{I,J}$ varies linearly between $u_{I,0}$ and $u_{I,N}$ and $v_{I,J}$ varies linearly between $v_{0,J}$ and $v_{M,J}$. The computation of surface grid points from a finite set of boundary curve points using TFI is illustrated in Figure 3.2.

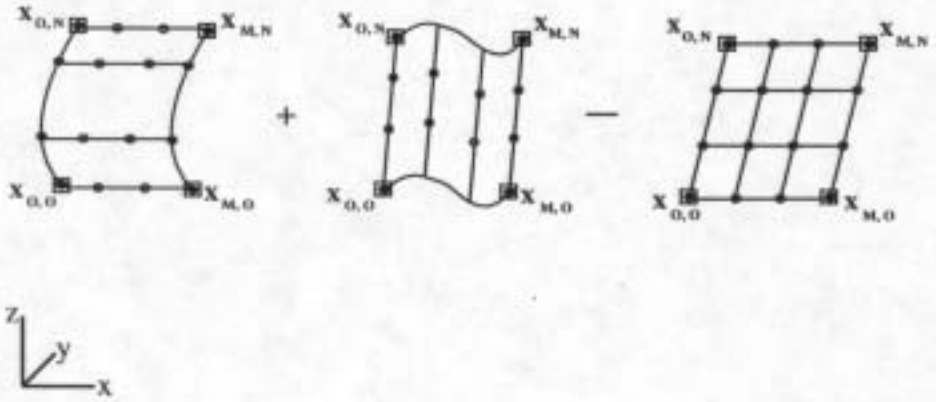


Figure 3.2: Principle of TFI: Two linear and one bilinear interpolation step.

The elliptic equations used for surface grid generation are a generalization of Laplace equations for harmonic mappings of planar regions. Let $\mathbf{x}$ be a simply connected parametric surface, written as

$$\begin{aligned}\mathbf{x}(u,v) &= (x(u,v), y(u,v), z(u,v)) \\ &= (x\,(u(\xi,\eta), v(\xi,\eta))\,, y\,(u(\xi,\eta), v(\xi,\eta))\,, z\,(u(\xi,\eta), v(\xi,\eta)))\,, \end{aligned} \tag{3.2}$$

where $0 \leq u, v \leq 1$. This notation implies that the *parametric variables* u and v are functions of the *computational variables* ξ and η. A rectilinear grid in the computational square ($\xi\eta$ space) generates a curvilinear grid in the parametric square (uv space), which maps to a curvilinear grid on the surface (xyz space or *physical variables*). Thus, a uniform grid in the computational space generates a curvilinear grid on the surface. Figure 3.3 shows the physical, parametric, and computational spaces associated with a single surface.

The elliptic system of partial differential equations defining the transformation between computational variables and parametric variables is related to conformal mappings on surfaces (see [32]). The derivation of the elliptic system of PDEs is based on conformal mappings. The approach utilizes the intrinsic orthogonality and uniformity properties that are inherent to a grid generated by a conformal mapping. Another advantage of using conformal coordinates is the fact that they allow solutions of differential equations on surfaces with the same ease as on a rectangle in the Cartesian plane.

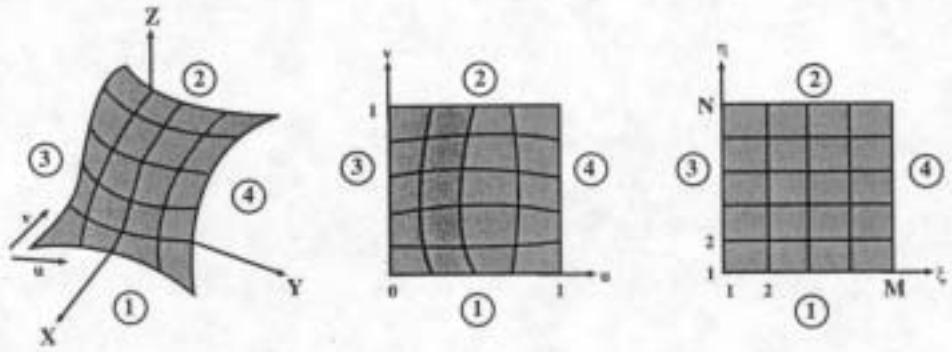


Figure 3.3: Physical, parametric, and computational spaces.

A surface grid generated by the conformal mapping of a rectangle onto the surface $\mathbf{x}$ is orthogonal and has a constant aspect ratio. These two conditions can be expressed as

$$\begin{aligned} \mathbf{x}_\xi \cdot \mathbf{x}_\eta &= 0 \quad \text{and} \\ M\,|\mathbf{x}_\xi| &= |\mathbf{x}_\eta|, \end{aligned}$$

where M is the *grid aspect ratio*. These two equations can be rewritten as

$$\begin{aligned} x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta &= 0 \quad \text{and} \\ M^2(x_\xi^2 + y_\xi^2 + z_\xi^2) &= x_\eta^2 + y_\eta^2 + z_\eta^2. \end{aligned}$$

Using the chain rule for differentiation, this system of equations can be expressed in the form

$$\begin{aligned} M u_\xi &= a v_\eta - b u_\eta \quad \text{and} \\ M v_\xi &= b v_\eta - c u_\eta, \end{aligned}$$

where

$$a = \frac{\overline{g}_{22}}{\overline{J}}, \quad b = -\frac{\overline{g}_{12}}{\overline{J}}, \quad c = \frac{\overline{g}_{11}}{\overline{J}}, \quad \overline{J} = \sqrt{\overline{g}_{11}\overline{g}_{22} - \overline{g}_{12}^2},$$
$$\overline{g}_{11} = \mathbf{x}_u \cdot \mathbf{x}_u, \quad \overline{g}_{12} = \mathbf{x}_u \cdot \mathbf{x}_v, \quad \text{and} \quad \overline{g}_{22} = \mathbf{x}_v \cdot \mathbf{x}_v.$$

If the parametric and computational variables are exchanged such that the parametric variables become the independent variables, the system becomes

$$\begin{aligned} -M \eta_u &= b\xi_u + c\xi_v \quad \text{and} \\ M \eta_v &= a\xi_u + b\xi_v. \end{aligned}$$

This first-order system is exactly *Beltrami's system of equations for the quasi-conformal mapping of planar regions*. It follows that the computational variables ξ and η are solutions of the following second-order linear homogeneous elliptic system ($\Phi = \Psi = 0$):

$$\begin{aligned} \overline{g}_{22}\xi_{uu} - 2\overline{g}_{12}\xi_{uv} + \overline{g}_{11}\xi_{vv} + (\Delta_2 u)\xi_u + (\Delta_2 v)\xi_v &= \Phi \quad \text{and} \\ \overline{g}_{22}\eta_{uu} - 2\overline{g}_{12}\eta_{uv} + \overline{g}_{11}\eta_{vv} + (\Delta_2 u)\eta_u + (\Delta_2 v)\eta_v &= \Psi. \end{aligned}$$

The Beltramians $\Delta_2 u$ and $\Delta_2 v$ are given by

$$\begin{aligned} \Delta_2 u = \overline{J}(a_u + b_v) &= \overline{J}\left[\frac{\partial}{\partial u}\left(\frac{\overline{g}_{22}}{\overline{J}}\right) - \frac{\partial}{\partial v}\left(\frac{\overline{g}_{12}}{\overline{J}}\right)\right] \quad \text{and} \\ \Delta_2 v = \overline{J}(b_u + c_v) &= \overline{J}\left[\frac{\partial}{\partial v}\left(\frac{\overline{g}_{11}}{\overline{J}}\right) - \frac{\partial}{\partial u}\left(\frac{\overline{g}_{12}}{\overline{J}}\right)\right]. \end{aligned}$$

This system is the basis for most elliptic grid generation methods. The source terms (or control functions) Φ and Ψ are added to allow control over the distribution of grid points on the surface. Typically, points are given in computational space and points in parametric space must be computed. Therefore, it is convenient to exchange variables again such that the computational variables ξ and η are the independent variables. This leads to the following quasi-linear elliptic system:

$$\begin{aligned} g_{22}(u_{\xi\xi} + Pu_\xi) - 2g_{12}u_{\xi\eta} + g_{11}(u_{\eta\eta} + Qu_\eta) &= J^2\Delta_2 u \quad \text{and} \\ g_{22}(v_{\xi\xi} + Pv_\xi) - 2g_{12}v_{\xi\eta} + g_{11}(v_{\eta\eta} + Qv_\eta) &= J^2\Delta_2 v, \end{aligned} \tag{3.3}$$

where

$$\begin{aligned} g_{11} &= \overline{g}_{11}u_\xi^2 + 2\overline{g}_{12}u_\xi v_\xi + \overline{g}_{22}v_\xi^2, \\ g_{12} &= \overline{g}_{11}u_\xi u_\eta + \overline{g}_{12}(u_\xi v_\eta + u_\eta v_\xi) + \overline{g}_{22}v_\xi v_\eta, \\ g_{22} &= \overline{g}_{11}u_\eta^2 + 2\overline{g}_{12}u_\eta v_\eta + \overline{g}_{22}v_\eta^2, \quad \text{and} \\ J &= u_\xi v_\eta - u_\eta v_\xi. \end{aligned}$$

The elliptic system is solved using a finite difference approach using either Dirichlet (or Neumann) boundary conditions. Since P and Q control the grid point distribution, $P = Q = 0$ leads to a uniformly spaced grid in the absence of boundary curvature. These two control functions are estimated from an initial grid having a point distribution that is close to the desired one. It is possible to compute the functions P and Q by performing TFI from user-specified boundary curve distributions. They are commonly defined in terms of relative spacing between grid points. When grid line orthogonality is desired, the grid points are allowed to move on the boundary curves.

A detailed discussion of elliptic structured grid generation using NURBS surfaces is given in [25]. Parametric surfaces/volumes and their relation to grid generation applications are dealt with in [2, 10, 22].

Structured grids can be generated by using either an elliptic or a hyperbolic approach. Earlier approaches use elliptic systems of PDEs (see [53]). In this case, the outer (or far-field) boundary must be known. The alternative approach uses hyperbolic systems (see [6]), where the outer boundary is not prescribed but is a result of the grid generation process.

Sometimes it is advantageous to decompose a field to be discretized into multiple structured grids (multiple blocks). The entire field is first divided into blocks (that is, boundaries of each block are defined), and a structured grid is computed for each block. Certain continuity conditions (for example, C^0, C^1, or C^2 continuity) are often imposed at block interfaces. The curves and surfaces used for the block boundaries can consist of multiple curve segments and surface patches. Figure 3.4 shows a multiblock configuration around a 2D geometry.

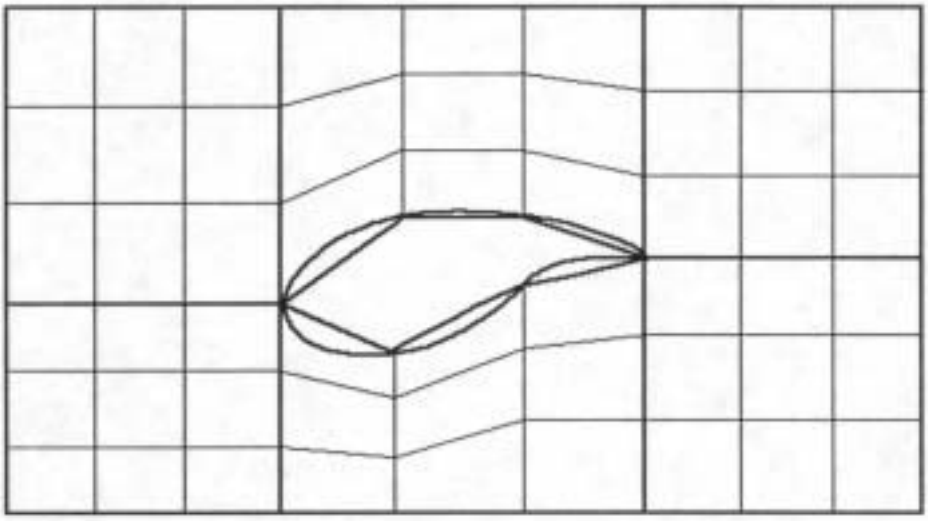


Figure 3.4: Multiblock structured grid around 2D geometry.

3.2.3 Unstructured Grid Generation Based on the Delaunay Triangulation

Most algorithms for generating unstructured-triangular and unstructured-tetrahedral grids are based on the *Delaunay triangulation* or the so-called *advancing front* method. Unstructured grids consisting of types of elements other than triangles and tetrahedra are not considered here.

The Delaunay triangulation has been used for various applications, including scattered data interpolation. In that context, function values f_i are given at scattered locations $\mathbf{x}_i$ for which no connectivity is given. Typically, the Delaunay triangulation is computed for the points $\mathbf{x}_i$ and triangular (tetrahedral) interpolants are constructed. The Delaunay triangulation is characterized by the fact that the circle (sphere) passing through the vertices of any triangle (tetrahedron) does not contain any point of the original point set in its interior. In the 2D case, the Delaunay triangulation is the max-min angle triangulation of the given point set, that is, it maximizes the minimum angle in the triangulation (see [42]). This property is very desirable for many applications, particularly for grid generation.

The Delaunay triangulation has found wide popularity in the finite element method (FEM) and unstructured grid generation communities. In general, triangles (tetrahedra) must be generated for a simply connected region in 2D (3D). They are generated in two steps. The first step is the generation of a boundary grid for the boundary curves (boundary surfaces) of the space to be discretized, that is, a set of boundary conforming line segments (triangles) is computed. The second step is the generation of triangles (tetrahedra) inside this boundary grid (see [57, 58, 60]).

The unstructured grid can be generated by inserting points and performing local retriangulation iteratively or by first generating the entire point set and then triangulating it, possibly using parallel programming paradigms. Grid points are placed such that certain quality measures (edge lengths, areas, volumes, angles, ratios thereof, and such) and specified distributions are satisfied.

The grid points are typically chosen according to geometrical properties of the boundary (for example, arc length or absolute curvature) and distribution functions. Spacing

parameters are used for the boundary point distribution, and so-called *sources* (that is, point, line, and plane sources) are used to further control grid point distributions in the interior. Grid point densities decrease in a predefined fashion with increasing distance from the sources. Grids must be *graded* in this fashion due to the sometimes sudden occurrence of discontinuities in certain field parameters, for example, shocks. Figure 3.5 shows 2D triangular grids with point and line sources.

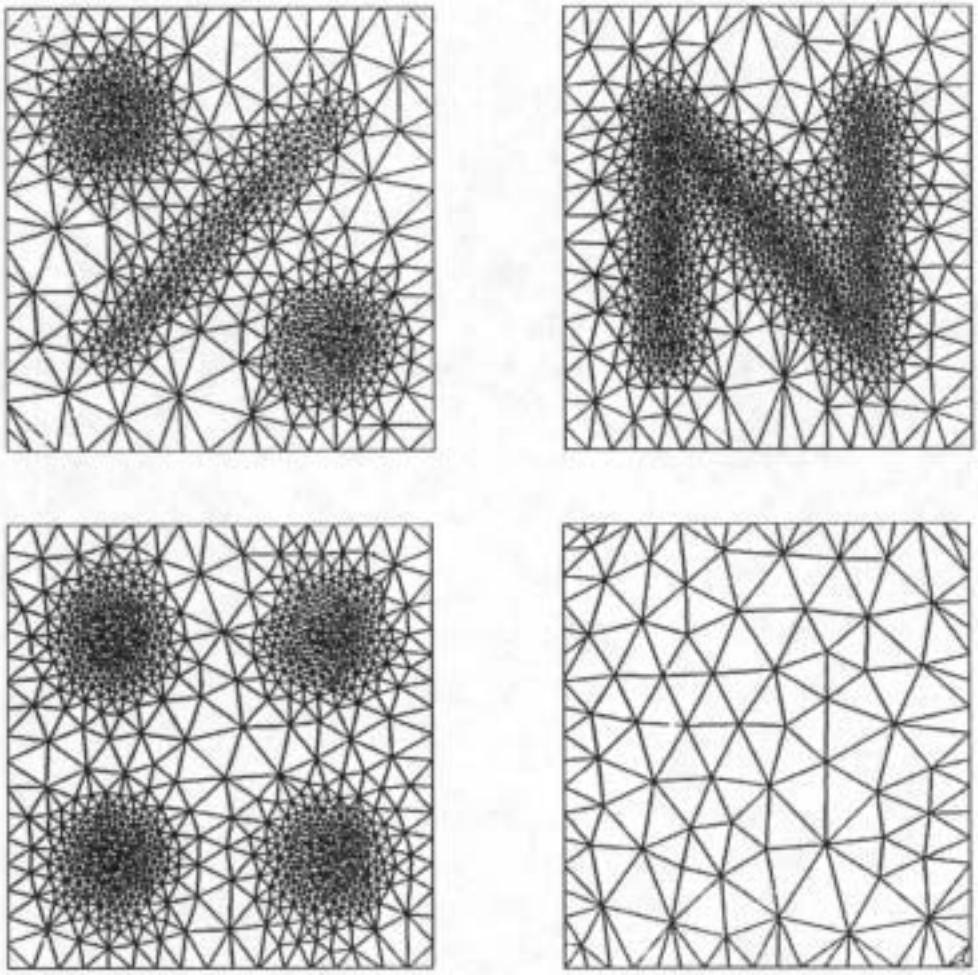


Figure 3.5: Unstructured-triangular grids with point and line sources (courtesy of N.P. Weatherill, University College of Swansea, U.K.).

3.2.4 Unstructured Grid Generation Based on Element Size Optimization

An automatic algorithm for the generation of 3D unstructured-tetrahedral grids is described in [18]. This algorithm can be applied to any closed geometry, that is, a geometry consisting of surfaces whose boundary curves are each shared by exactly one other surface boundary curve. The method is based on intersecting the edges of an initial (coarse) tetrahedral grid with the given geometry, clipping this initial grid against the geometry by extracting the (parts of) tetrahedra on the outside (or inside) of the geometry, and iteratively inserting grid points.

3.2.5 Unstructured Grid Generation Based on the Advancing Front Method

The advancing front method generates unstructured grids in a completely different way. The input—for the generation of a triangular (tetrahedral) grid—is a set of oriented line segments (oriented triangles) discretizing the boundary curves (surfaces) of some geometry. Triangles (tetrahedra) are then constructed by advancing a *front* into the interior of the field until it is completely filled with elements. Depending on local edge and angle configurations of the current front, new elements are created by connecting existing points or by inserting new points—according to some distribution function—and connecting them with existing ones (see [29]). The approach is similar to hyperbolic grid generation.

The desired point distribution is typically defined by a set of points with associated spacing parameters (background grid). Interpolating these spacing parameters yields the desired spacing at any point in the field. Points are inserted such that the desired spacing is optimally satisfied. The advancing front strategy is also used for the generation of unstructured grids consisting of quadrilaterals (hexahedra). Figure 3.7 shows various stages of the advancing front algorithm.

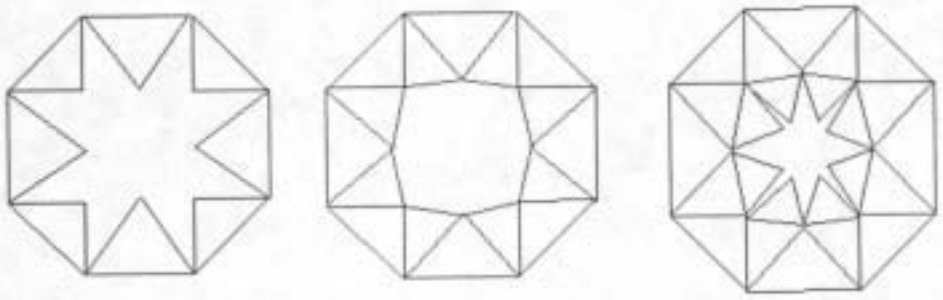


Figure 3.7: Different stages of the advancing front algorithm for 2D example.

3.2.6 Chimera Grids

Chimera grids (overlapping structured grids) are a collection of partially overlapping structured "component grids." Chimera grids are produced by generating a separate structured grid around or above each component of a geometry. The component grids are overlapping, and information is passed from one component grid to the other via an appropriate interpolation scheme. Often, Chimera grids are generated with a hyperbolic grid generation method and "growing" each component grid from the associated geometry component. The approach is described in detail in [3] and [8]. Figure 3.8 shows an example.

3.2.7 Hybrid Grids

Hybrid grids are the combination of structured and unstructured grids. When generating a hybrid grid, a field is discretized by structured grid topologies wherever possible, and regions between the different structured grids are discretized by unstructured grids. Thus, this approach leads to grids that combine the strength of structured grids (computational

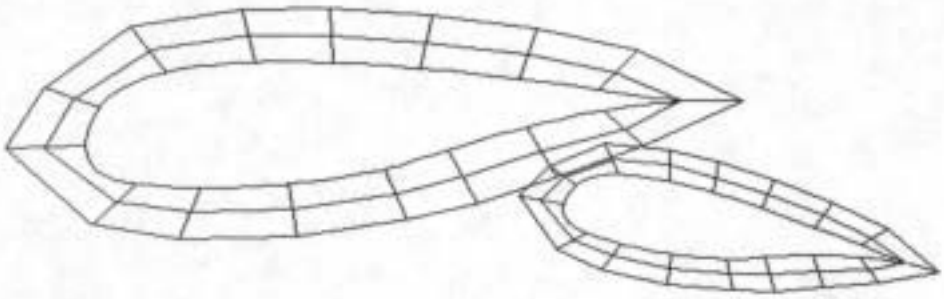


Figure 3.8: Chimera grid for 2D wing configuration.

efficiency and high-aspect ratio elements) and unstructured grids (flexibility). Figure 3.9 shows a hybrid grid.

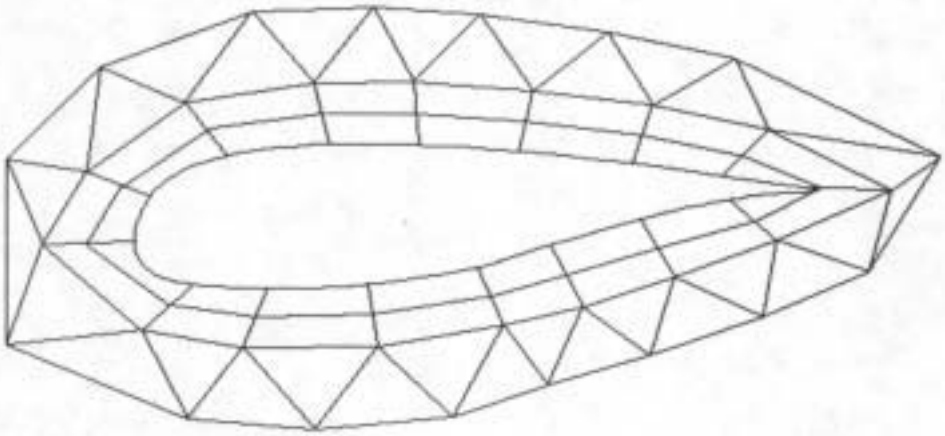


Figure 3.9: Hybrid grid around 2D wing configuration.

3.2.8 Hierarchical Grid Generation Based on Quadtrees and Octrees

By applying a subdivision technique based on quadtrees (octrees) to certain elements of a grid, multiple levels of different resolution are defined. This approach leads to so-called hierarchical (embedded) grids. Depending on the local complexity of the geometry or the local complexity of an intermediate field solution, more elements are adaptively created by splitting parent elements.

In the structured case, a quadrilateral (hexahedron) is subdivided into four (eight) quadrilaterals (hexahedra) which are defined by splitting the edges at their midpoints (see [33, 45]). The subdivision algorithm must ensure that no "cracks" are introduced in physical xyz space when splitting edges in the parameter space of a parametric surface. As a result of the subdivision process, nodes are created that lie in the interior of edges (faces). They are usually called "hanging nodes."

Hierarchical subdivision schemes based on quadtrees (2D) and octrees (3D) are also used for the automatic generation of entire unstructured grids inside simply connected regions. For this purpose, the boundary of the region to be discretized must be given by a set of line segments (triangles), and the bounding box of the region is recursively subdivided

until no cuboid contains more than one of the points used in the boundary discretization. The cuboids lying in the interior of the region are then subdivided into triangles (tetrahedra). Special care is necessary to preserve the region's boundary. This approach is discussed in [61, 62, 63]. Figure 3.10 shows the quadtree implied by the boundary of a simply connected region.

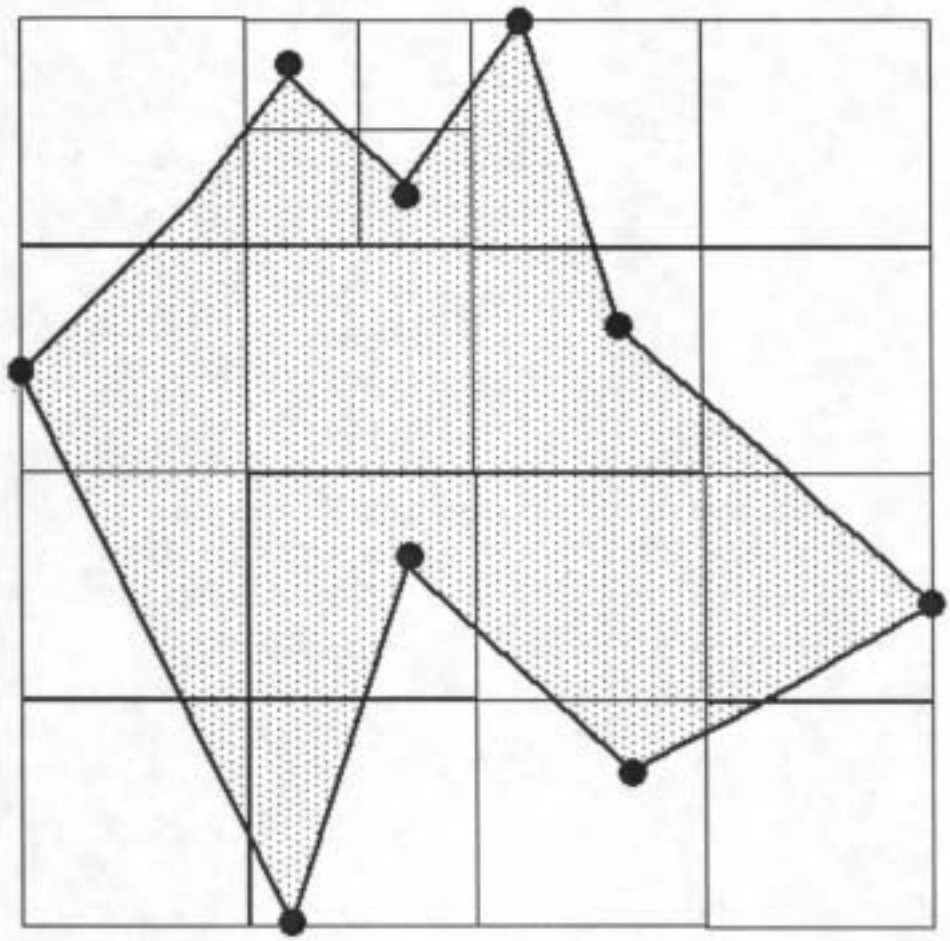


Figure 3.10: Quadtree implied by boundary line segments of 2D region—interior (shaded) must be triangulated.

3.3 The National Grid Project (NGP)

3.3.1 Survey

The National Grid Project (NGP) started in 1991. The purpose of this project is the development of a next-generation, interactive grid generation system for a consortium of U.S. industrial and governmental laboratories. This effort has led to a grid generation system that combines geometric modeling (CAD), structured grid generation, and unstructured grid generation functionality. The interdisciplinary character of the ERC-CFS has made it possible to design and develop this general grid generation system.

Nonuniform rational B-splines (NURBS) are used throughout the system for the representation of geometries. The user interface is based on MOTIF and X, and Iris GL is

used for graphics. The system interfaces with CAD systems by converting given CAD data (IGES format) to NURBS and vice versa. NURBS approximations of original data are generated only if an exact conversion is impossible. The NGP system provides various CAD functions for the creation and manipulation of geometries, which can also be used for the definition of field boundary curves and surfaces. In addition, the CAD system provides a method for the continuous approximation of geometries with discontinuities.

CAD functionality that is helpful for the block definition and grid generation process is continuously added to the system. Currently, most CAD operations can be performed only on a single NURBS surface, that is, it is not possible to perform all CAD operations across multiple NURBS surfaces with different parameter spaces. Algorithms that allow certain operations across surface boundaries will be developed. The NGP system allows the user to

- interface with CAD systems via IGES
- create geometries via a NURBS-based CAD subsystem
- perform geometry-processing operations, such as surface-surface intersection, trimming, and surface subdivision
- correct geometries with discontinuities ("gaps," overlapping surfaces, or intersecting surfaces)
- automatically generate the topology information of the boundary representation (B-rep) of a CAD model, that is, generate the connectivity information for curves and surfaces
- construct unstructured grids (triangular, quadrilateral, and tetrahedral elements) and multiblock structured grids (quadrilateral and hexahedral elements), including Chimera and hybrid grids
- analyze—according to various criteria—and visualize the quality of 2D and 3D grids (see [41])
- save the state of the system at any time and provide a script language that can be edited.

Figure 3.11 shows an example of an aircraft geometry and the far-field boundary created with the NGP CAD system. It also shows the general layout of the NGP interface.

Most of the requirements that were defined by the NGP consortium are satisfied at this point, three years after the start of the project. Currently, the system can handle only four-sided surfaces connected in a full-face interface fashion, that is, surfaces must share entire boundary edges and not parts thereof. Partial edge (and partial face) matching will be added to the NGP system. When dealing with trimmed surfaces (that is, surfaces containing holes defined by so-called trimming curves), the user must generate a set of new four-sided surfaces from each trimmed surface. Thus, each trimmed surface is replaced by a set of new surfaces without trimming curves (holes). This process will eventually be automated. The current capabilities of the NGP system are described in more detail in [43].

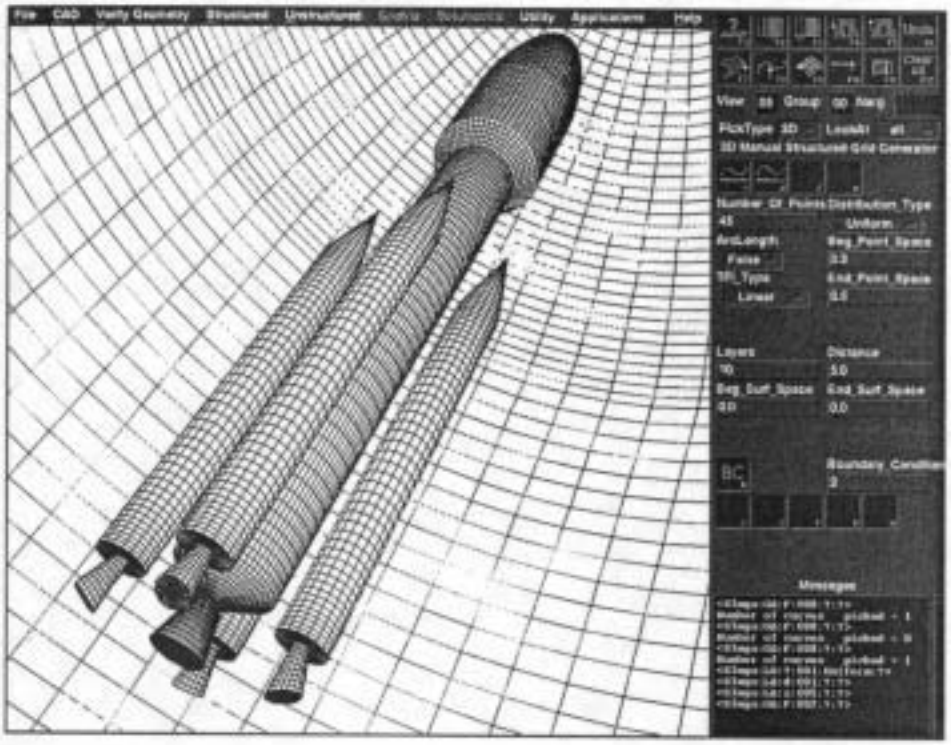


Figure 3.11: Geometry and far-field boundary created with the NGP CAD system.

3.3.2 Approximation of Geometries with Discontinuities

For grid generation purposes, a geometry that is to be discretized must be error-free, that is, it must not contain discontinuities (overlapping surfaces, "gaps" between surfaces, or surface intersections). Unfortunately, CAD data imported from other CAD systems very often contains such errors.

The NGP system provides an interactive geometry correction technique that locally approximates a given geometry by a B-spline surface. The technique must be applied to all regions containing discontinuities. Eventually, a continuous geometry is obtained that consists partially of original NURBS surfaces and partially of B-spline approximations. The continuous geometry is then used for grid generation.

The geometry correction technique is based on constructing an initial local surface approximation (a bilinearly blended Coons patch) which is projected onto the given surfaces. The user must ensure that the B-spline surface approximations are connected with each other and with certain given NURBS surfaces in a full-face interface fashion. This can done with the CAD system. Generating a local surface approximation requires these steps:

1. Definition of four surface boundary curves
2. Generation of $N \times N$ points on the bilinearly blended Coons patch defined by the four surface boundary curves
3. Projection of the $N \times N$ points onto the given surfaces
4. Generation of "artificial projections" whenever certain points of the Coons patch cannot be projected onto any original surface

When trying to project a point of the Coons patch onto the given surfaces, a projection might or might not be found. If one or more projections are found within a small distance to the Coons patch, the one closest to the Coons patch is chosen. If no projection is found, an "artificial projection" is approximated by applying a scattered data interpolation scheme to the projections that have been found.

5. Interpolation of the points resulting from steps (3) and (4) by a bicubic B-spline surface

Points on the Coons patch are denoted by $\mathbf{x}_{i,j}$ and are obtained by applying the bilinear blending procedure to a finite set of points on the four specified boundary curves (see Section 3.2.2, Equation (3.1)). The points $\mathbf{x}_{i,j}$ are then projected onto the given surfaces in the normal direction $\mathbf{n}_{i,j}$ of the Coons patch, which is approximated by

$$\mathbf{n}_{i,j} = \frac{(\mathbf{x}_{i+1,j} - \mathbf{x}_{i-1,j}) \times (\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j-1})}{\| (\mathbf{x}_{i+1,j} - \mathbf{x}_{i-1,j}) \times (\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j-1}) \|}, \tag{3.5}$$

where $\| \ \|$ is the Euclidean norm. A family of line segments, defined by the point pairs in

$$\{\mathbf{a}_{i,j} = \mathbf{x}_{i,j} - d\mathbf{n}_{i,j},\ \mathbf{b}_{i,j} = \mathbf{x}_{i,j} + d\mathbf{n}_{i,j} \mid d > 0,\ i,j = 1,\ldots,N\} \tag{3.6}$$

is intersected with the given geometry. If a line segment $\overline{\mathbf{a}_{i,j}\mathbf{b}_{i,j}}$ has multiple intersections with the given geometry, the one closest to $\mathbf{x}_{i,j}$ is chosen as the projection of $\mathbf{x}_{i,j}$.

The original surfaces might be discontinuous, and, in this case, certain points $\mathbf{x}_{i,j}$ cannot be projected onto an original surface. Each projection $\mathbf{p}_{i,j}$ that has been found can be represented as the linear combination $\mathbf{p}_{i,j} = (1-t_{i,j})\mathbf{a}_{i,j} + t_{i,j}\mathbf{b}_{i,j}$. Thus, the computation of "artificial projections" reduces to the construction of a scattered data interpolant. Hardy's reciprocal multiquadric method is used for interpolating the known values $t_{i,j}$ (see [11]). The system of linear equations to be solved is

$$t_{i,j} = \sum_{I,J \in \{1,\ldots,N\}} c_{I,J} \left((u_{I,J} - u_{i,j})^2 + (v_{I,J} - v_{i,j})^2 + R \right)^{-0.5}, \quad i,j \in \{1,\ldots,N\}, \tag{3.7}$$

where all values $t_{i,j}$, $u_{i,j}$, and $v_{i,j}$ are considered for which a projection has been found. The value for $R > 0$ is chosen according to the point spacing on the Coons patch. The NGP system uses a localized version of this method. The solution of the system (3.7) is used to approximate the required "artificial projections." The approximation of "artificial projections" from the ones that can be found is illustrated in Figure 3.12.

The resulting $N \times N$ points (projections and "artificial projections") are interpolated by a C^1 continuous, bicubic B-spline surface. An error estimate is computed for each local surface approximant. Curves on the original geometry, for example, surface boundary curves or trimming curves, can be preserved by this method.

The algorithm is described in detail in [17, 24, 50]. The underlying concepts are well known in geometric modeling and can be found in [10] and [22]. Figure 3.13 shows the original CAD data of a car body geometry with "holes" (upper two images) and its continuous approximation (lower two images).

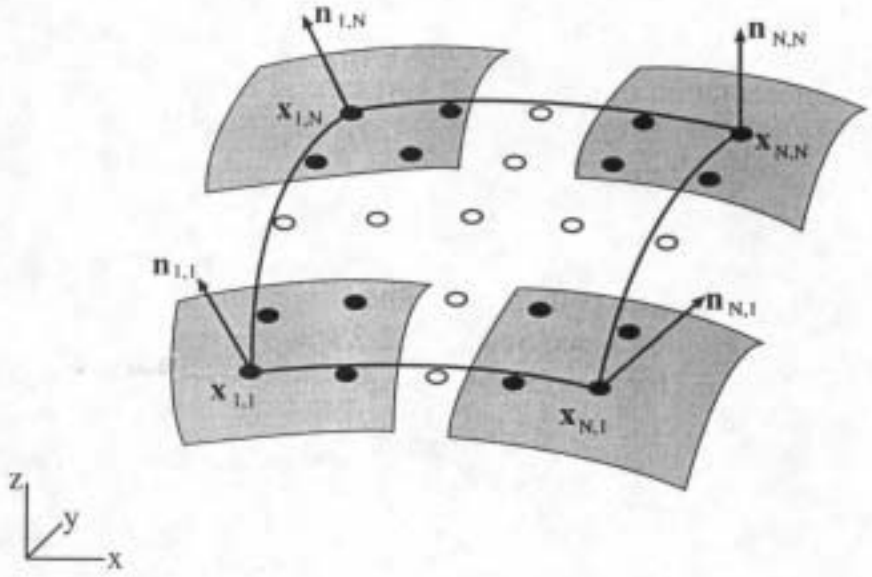


Figure 3.12: Projections (bullets) and estimated "artificial projections" (circles).

Figure 3.13: Car body with "holes" *(top)* and its approximation *(bottom)*.

3.3.3 Grid Generation with the NGP System

The structured and unstructured grid generation modules of the NGP system are based on NURBS. The representation of curves and surfaces as NURBS has turned out to be beneficial for the grid generation process. The generation of a structured grid for a single NURBS surface requires two steps. The first step is the generation of boundary curve points considering user-specified distribution functions. The second step is the generation of grid points in the surface's interior. An initial surface grid is computed by applying TFI to the boundary points. This surface grid is then "smoothed" yielding a surface grid with nearly orthogonally intersecting grid lines (see Section 3.2.2 and [53]).

The same principle is used for the generation of 3D structured volume grids. Currently, the user has to define the vertices, edges, and faces of the blocks surrounding a 3D geometry. All blocks must be connected in a full-face interface fashion; partial face matching is not possible at this time. It is planned to use automatic procedures for the block definition phase (see [7]). Figure 3.14 shows the initial TFI surface grid and the smoothed surface grid for the space shuttle.

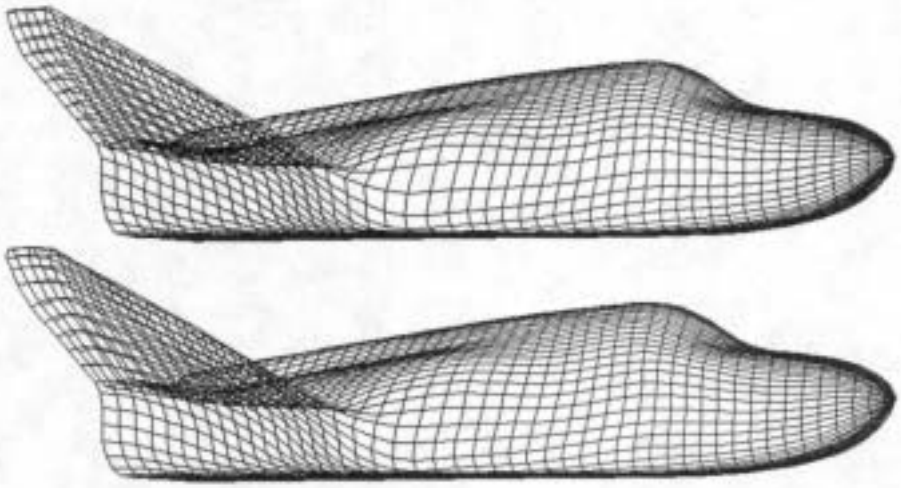


Figure 3.14: Initial grid obtained by transfinite interpolation *(top)* and elliptically smoothed grid *(bottom)* for space shuttle.

The generation of unstructured-triangular surface grids is based on the Delaunay triangulation. Once surface boundary curve discretizations are established, scattered points are generated in a surface's interior, and the Delaunay triangulation is computed. Point insertion and triangulation are performed in a surface's parameter space while considering metric properties of the surface in 3D space. Poorly parameterized surfaces can sometimes lead to poor surface triangulations. Appropriate reparameterization of parametric surfaces is currently being investigated.

The unstructured grid generation module can handle surfaces with any number of closed, nonintersecting trimming curves. Usually, the point distributions on the boundary and trimming curves imply the point distribution in a surface's interior. Local variations in point distribution can be achieved by using point, line, and plane sources. Point densities decrease with increasing distance to a source. Volume grids are computed by the Delaunay triangulation as well. The resulting triangulations are clipped against the boundaries of the field to obtain boundary conformity.

The unstructured grid generation module requires little user input and operates highly automated. The underlying concepts and theories are discussed in [54, 59, 60]). Figure 3.15 shows an unstructured-tetrahedral grid around a car body.

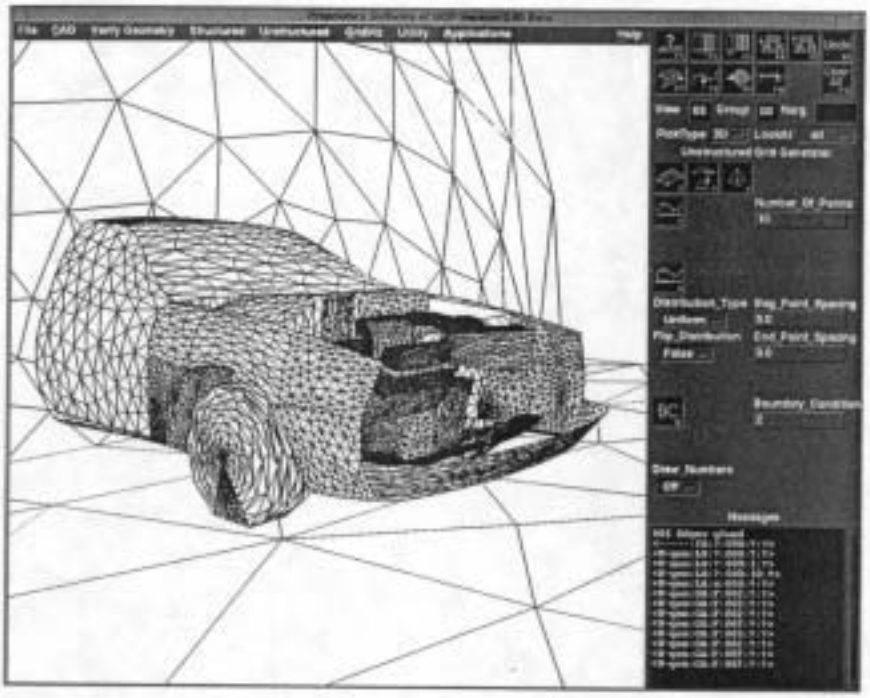


Figure 3.15: Unstructured-tetrahedral grid generated with the NGP system.

The NGP system also allows for hyperbolic grid generation and supports the Chimera approach for multiple blocks. These two approaches are often combined (see [6]).

3.4 Visualization Efforts at the ERC-CFS

3.4.1 The Scientific Visualization Thrust

The Scientific Visualization Thrust at the ERC-CFS provides image-oriented analytical tools to allow scientists and researchers to better understand and explain physical phenomena. To accomplish this goal, both measured and computational data is visualized. The application focus has been diverse and has included oceanographic, aerodynamic, hydrodynamic, meteorological, geophysical, and acoustic work. The work has also spanned the functional gamut—from visualization to allow application scientists and engineers to understand a measured phenomenon or the results of a computational model—to visualization to allow application scientists and engineers to explain their results to their supervisors and funding sources. In some cases, the understanding must be obtained quickly so that the next sample may be taken in an operational scenario; in other cases, the understanding can be obtained only after extensive and rigorous analysis in a laboratory.

Unfortunately, it is still true that many revered and respected scientists and researchers seldom actually create visualizations themselves, but instead direct "visualization scientists and engineers" to accomplish effective visualizations. Thus, although photorealism is usually a secondary goal, since the visualization is not done interactively, high-quality imagery

is usually expected if not demanded. For example, Gouraud shading and anti-aliasing is expected, that is, "jaggies" are unacceptable. Even so, interactivity or the ability to explore a data set is very important to the visualization engineer in this case, because often the application scientist is not exactly sure for what he or she is searching and because new or better understanding is often obtained by data exploration. Interactivity is also crucial for "field-deployment" scenarios, in which rapid visualization of measured data is important, and for concurrent visualization, in which the computational scientist seeks to save expensive supercomputer time by looking for mistakes in an evolving simulation.

The quantity and quality of the visualization technology is another metric that can be applied to the scenarios for which we develop visualization techniques and toolkits. Within the ERC-CFS, the personnel have access to state-of-the-art equipment such as:

- Powerful visualization computers
- Equipment to convert computer-generated imagery to video format
- Various video recording equipment (for example, video laser disks and Betacam SP, U-matic SP, S-VHS, and VHS VTRs)
- Immersive display devices

However, the visualization technology available to users in other environments constrains much of the research and development. For example, as will be described in the section on SCIRT below, a rapid data approximation and visualization scheme had to be created to allow engineers in the field to decide where to take the next sample. This software had to function on a lower-end graphics workstation in the field, yet allow engineers in the lab to postprocess the data to quantify more precisely the measured environment. In another scenario described below (one of the AGP environments), there is only one visualization workstation in a lab with multiple desktop workstations and an FDDI-attached supercomputer for computational field simulations. In this environment, the scientist's visualizations typically consist of lower-end, image-processing (2D)-type analysis and batch-mode-oriented, high-quality, video-based animations. Minimal expenditure has been made to allow the scientist to interactively visualize the results of the simulation of time-varying phenomena in an animated fashion on a workstation that supports high-spatial, high-intensity, and high-temporal resolution.

Thus the research and development efforts have spanned the gamut from high-end, nonreal-time imagery production to computationally efficient, interactive, near-real-time visualization. In the next few sections, some of that work is described in hopes of showing the diversity and value of analytical scientific visualization.

3.4.2 The Oceanographic Visualization Interactive Research Tool (OVIRT)

In 1992 and 1993, an interactive 3D scalar field visualization system, called the Oceanographic Visualization Interactive Research Tool (OVIRT), was developed at the ERC-CFS for the Naval Oceanographic Office (NAVO), Stennis Space Center, Mississippi. The purpose of this project was the development of effective visualization tools for oceanographic sound speed data.

The scalar field visualization, scattered data modeling, and isosurface generation techniques on which OVIRT is based are discussed in [16, 38, 39, 40]. OVIRT is described in more detail in [34].

OVIRT is used for visualizing environmental ocean data. It extends some of the classical 2D oceanographic displays into a 3D visualization environment. Its five major visualization tools are *cutting planes*, *minicubes*, *isosurfaces*, *sonic surfaces*, and *direct volume rendering* (DVR).

The cutting planes tool provides three orthogonal planes which can be independently and interactively moved through the volume. The minicubes routine renders small cubes inside the given volume. Their faces are shaded according to function value, for example, sound speed, pressure, temperature, or salinity. The isosurface tool allows the generation of constant value surfaces (see [38]). The DVR techniques give a global, translucent view of the data.

The sonic surfaces are an extension of 2D surfaces which have been classically used to display acoustic propagation paths and inflection lines within the ocean. The surfaces delineate the extent and axis of the sound channels. Other features in OVIRT include the ability to overlay the shoreline and inlay the bathymetry. Traditional analysis of sound velocity in the ocean involves graphical depictions of horizontal and vertical slices of the ocean. The limitation of those techniques is that the analyst views only a limited portion at once. To achieve a total view requires assimilating multiple executions of the display routine and mentally composing the results. The principle function of OVIRT is to visualize a 3D ocean sound speed field as a volume.

The user interface is based on X and MOTIF, and rendering is done with Iris GL using mixed-mode programming. The data is given on a rectilinear grid which is uniformly spaced in the x and y directions (latitude and longitude) and nonuniformly in the z direction (depth). In Figure 3.16, sound speed is rendered by the minicubes method ($10 \times 10 \times 10$ resolution). Some of the cubes are clipped against the bathymetry (wire frame). Trilinear interpolation is used for generating the function values at the cubes' vertices.

In Figure 3.17, the original nonuniform spacing in the z direction is used to define the vertices of the small cubes, creating cubes which increase in height as the depth increases, but which reflect the original sampling grid.

In an attempt to extend the visualization techniques with which the analyst is familiar, the 2D diagrams of sound speed versus depth (sound speed profiles) were extended to create 3D visualizations in which all the sound speed profiles for a particular latitude or longitude can be simultaneously viewed. The technique has been named *marching wiggles*. Traditionally, the analyst could not see the sonic profiles or sonic surfaces inlaid in the ocean. The sonic surfaces of particular interest to an oceanographer are (from the surface down):

- the *shallow sound channel* (SSC) *axis*
- the *sonic layer depth* (SLD)
- the *deep sound channel* (DSC) *axis*
- the *critical depth* (CD)

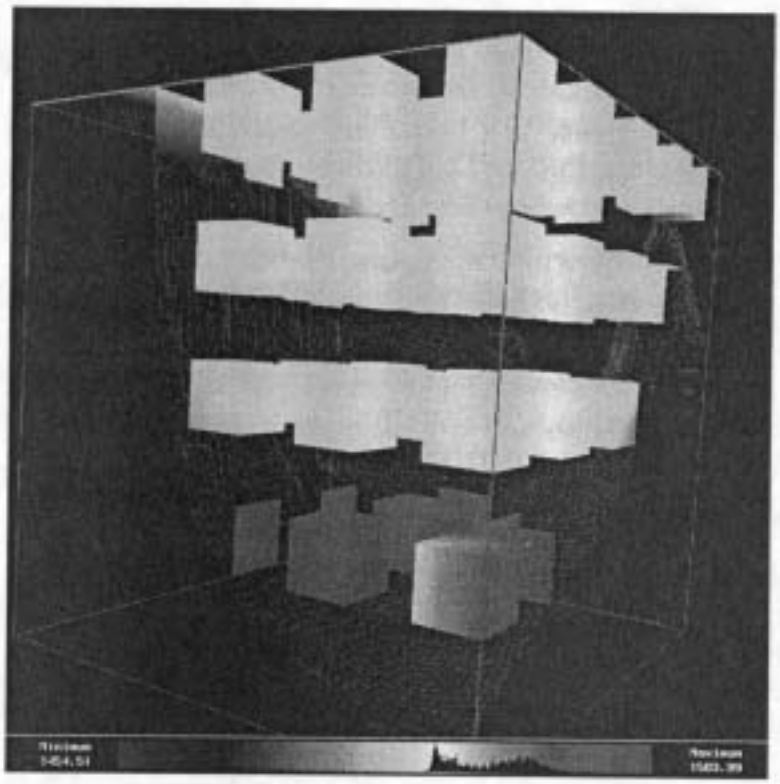


Figure 3.16: Minicubes method used for rendering sound speed with OVIRT system (uniform spacing and uniform cubes). See Color Plate 13.

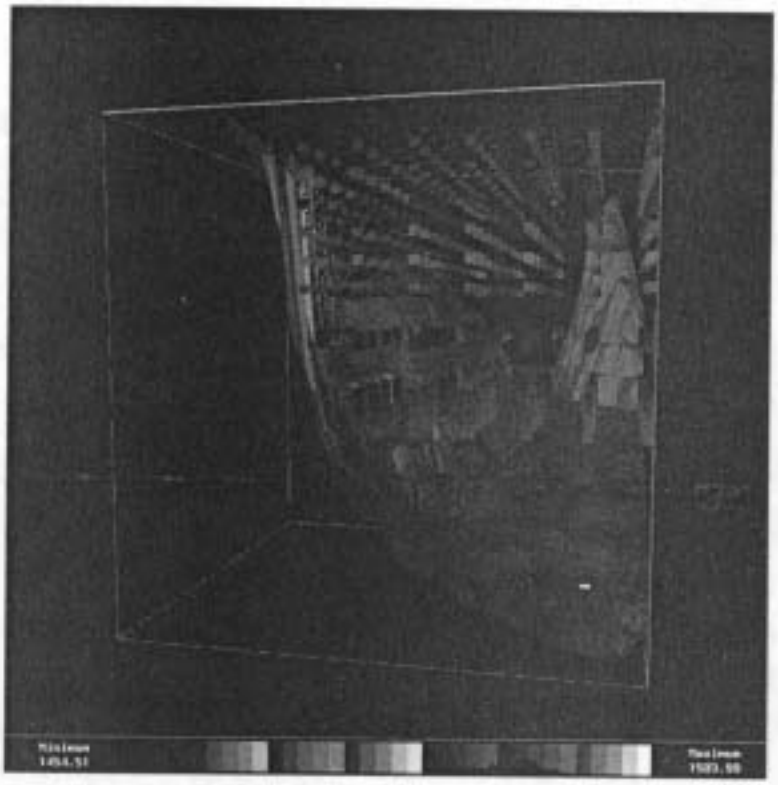


Figure 3.17: Minicubes method used for rendering sound speed with OVIRT system (nonuniform spacing and nonuniform cubes). See Color Plate 14.

These surfaces can be extracted from the collection of sound speed profiles. In general, they bound or are the axis of a sound channel. Figure 3.18 illustrates the surfaces for a 2D longitude slice.

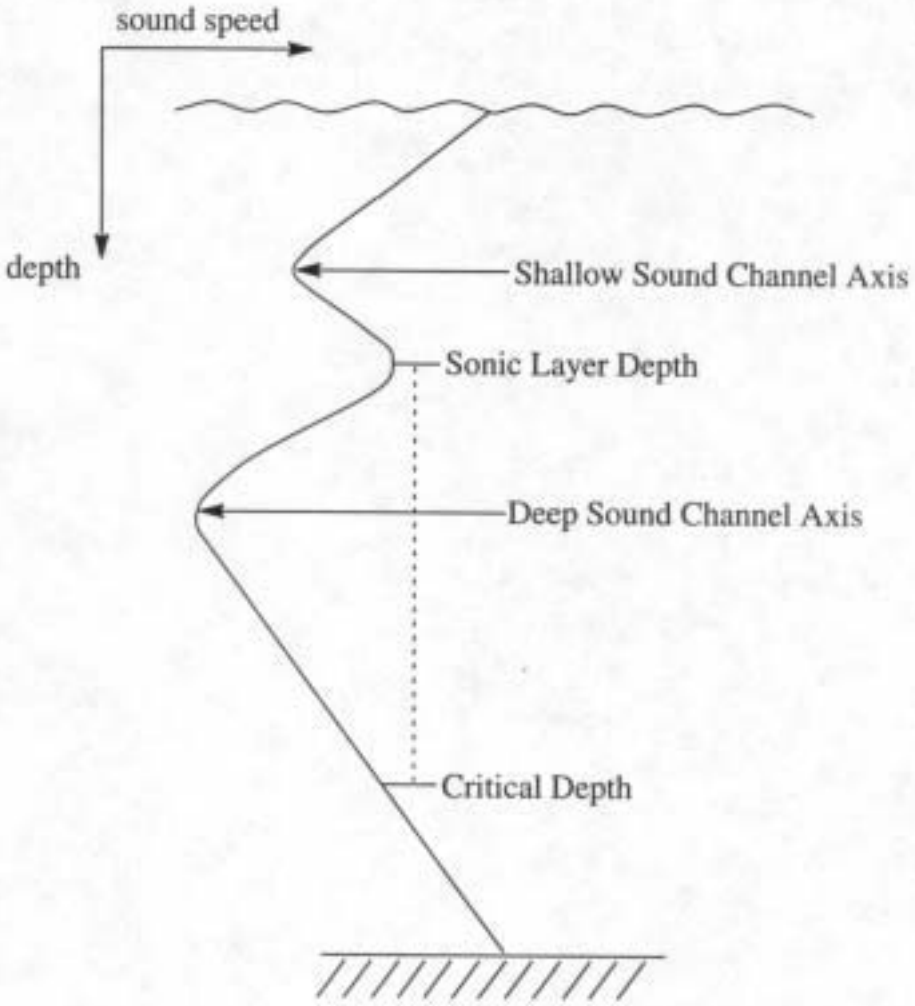


Figure 3.18: Shallow sound channel axis, sonic layer depth, deep sound channel axis, and critical depth.

Considering just a single sound speed profile (sound speed versus depth), the SSC is the region between the sea surface and the depth of the first sound speed minimum. The SLD is defined as the depth at which the first relative maximum of sound speed occurs in the upper levels of the ocean. In general, the next relative maximum is the CD (the sound speed is the same at the SLD and the CD). Sound rays will bounce between the SLD and the CD when emitted from a source between these two surfaces. If the acoustic source is above the SLD (below the CD), the sound rays will bend upward (downward). Figure 3.19 shows the SLD (red), the DSC axis (green), and the CD (blue).

The isosurface generation module is based on the algorithms given in [30] and [40]. Two DVR techniques are included in OVIRT, based on the methods in [28] and [44]. It has been found that DVR techniques do not convey adequate information regarding the location of sonic surfaces in the ocean. Better opacity mapping functions combined with robust classification techniques are necessary (see [9]). OVIRT has the capability to display

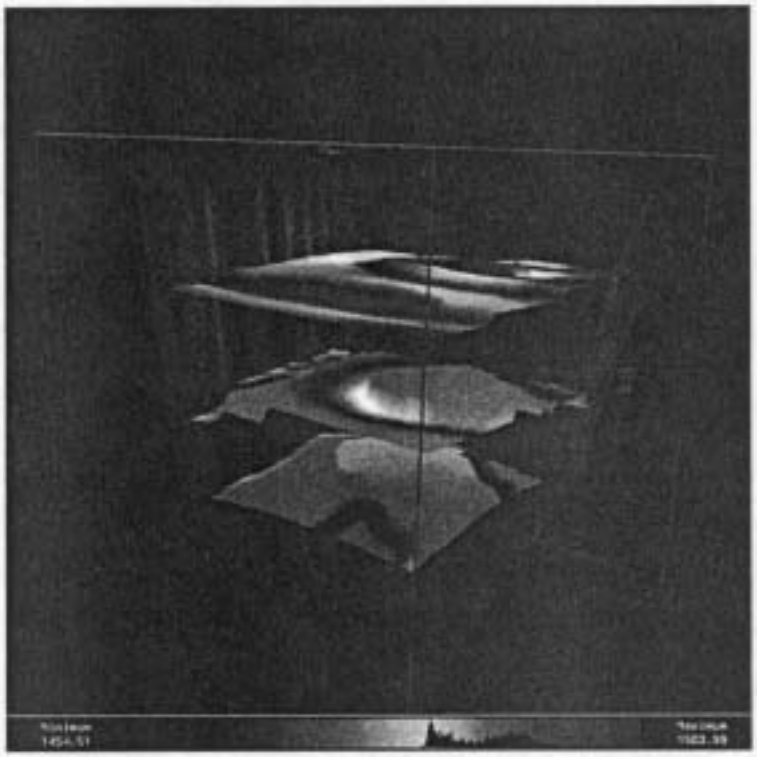


Figure 3.19: Sonic layer depth, deep sound channel axis, and critical depth generated and rendered with OVIRT system. See Color Plate 15.

animations of DVR imagery since, in general, DVR is not an interactive technique, whereas the other techniques are.

OVIRT provides a macro capability which allows the user to create and execute macro commands. Applications can be executed remotely, yet visualization can be done on a local workstation. Computationally intensive modules have been ported so that they can be executed remotely on SGI and Sun machines, as well as the Cray YMP. Other features include

- script and journal capabilities
- interactive query capability of both location and function value via picking
- rendering modes for combined rendering of bathymetry, cutting planes, and sonic surfaces

3.4.3 The Site Characterization Interactive Research Toolkit (SCIRT)

In 1993 and 1994, a scattered data interpolation and visualization system, called the Site Characterization Interactive Research Toolkit (SCIRT), was developed at the ERC-CFS for the U.S. Army Corps of Engineers, Waterways Experiment Station, Vicksburg, Mississippi. The purpose of the project was the development of a visualization tool for scattered data collected by the Site Characterization and Analysis Penetrometer System (SCAPS).

At each point, the soil classification number (SCN) and a concentration value are given. The data is usually collected by probing vertically into the ground. SCIRT provides techniques for the visualization of the SCN and concentration value. The system provides various scattered data interpolation techniques, clipping against the convex hull of the given 3D data points, and the scalar field visualization methods provided in OVIRT with extensions, for example, the visualization of concentration isosurfaces inside transparently rendered soil types and the indication of probe locations.

Since the visualization algorithms provided by OVIRT require a rectilinear grid, the given SCN and concentration data sets cannot be rendered directly. Scattered data interpolation and evaluation on a rectilinear grid are done in a preprocessing step. The system contains these modules:

- Convex hull and triangulation algorithms for scattered 3D points
- Global and local scattered data interpolation methods
- Overlay and transparency methods for the visualization of concentration isosurfaces inside surrounding soil types
- Scalar field visualization techniques supported by OVIRT with the capability of combining various scalar field visualizations into one image
- Local-viewing feature by applying visualization techniques to a user-specified box inside the data set

The visualization results depend greatly on the particular scattered data interpolation method used for the creation of the required rectilinear data set. The problem is to find a trivariate function $f(\mathbf{x}) = f(x, y, z)$ which interpolates the given values f_i at the sites $\mathbf{x}_i$, $i = 1, \ldots, N$. Several techniques have been developed over the last three decades for the interpolation (and approximation) of scattered data (see [11]).

Some scattered data interpolation techniques do not require a grid for the given data. Examples are Shepard's and Hardy's multiquadric methods. Shepard's global scattered data interpolant is the function

$$f(\mathbf{x}) = \frac{\sum_{i=1}^{N} \frac{f_i}{d_i^2}}{\sum_{i=1}^{N} \frac{1}{d_i^2}}, \tag{3.8}$$

where $f(\mathbf{x}_i) = f_i$ and $d_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2$ is the square of the Euclidean distance between $\mathbf{x}$ and a data point $\mathbf{x}_i$. Hardy's multiquadric method is based on constructing the function

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \left((x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 + R\right)^{0.5}, \tag{3.9}$$

$R > 0$, by solving the $N \times N$ system of equations given by the condition $f(\mathbf{x}_i) = f_i$. Usually, Hardy's multiquadric method yields much better results.

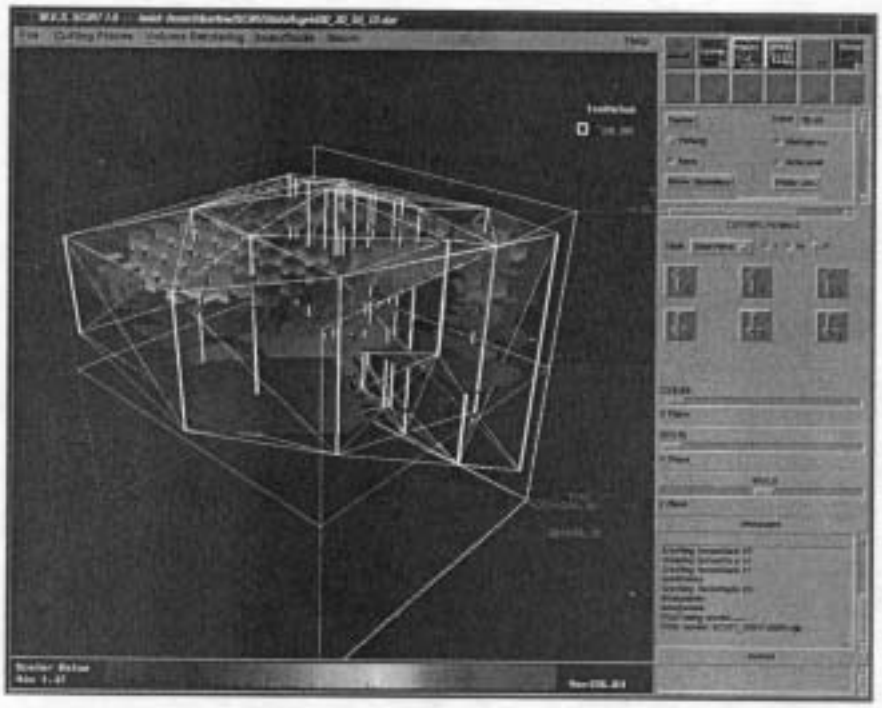


Figure 3.20: Transparent isosurfaces, minicubes, and cutting planes (clipped against convex hull) in a single image generated with SCIRT. See Color Plate 16.

Some techniques require a triangulation of the data sites $\mathbf{x}_i$. Triangulation-based techniques can be used to construct interpolants for each tetrahedron with a specified degree of continuity at the interfaces. Performing piecewise linear interpolation leads to a C^0 continuous function, and its value inside a particular tetrahedron is given by

$$f(\mathbf{x}) = \sum_{j=1}^{4} u_j f_j, \tag{3.10}$$

where u_1, u_2, u_3, and u_4 are the barycentric coordinates of the point $\mathbf{x}$ inside the tetrahedron with vertices $\mathbf{x}_j$, $j = 1, 2, 3, 4$, and function values f_j at its vertices. Higher-degree continuity for triangulation-based methods is achieved by considering derivative estimates at the given data points. Interpolation methods for C^1 and C^2 continuous interpolation will be added to SCIRT.

The SCIRT system supports Shepard's method, Hardy's multiquadric method, a localized version of Hardy's multiquadric method, and the triangulation-based piecewise linear method. The interpolants are evaluated on a rectilinear grid and are then visualized using the techniques provided by OVIRT. Examples are shown in Figure 3.20.

A future research issue is the reduction of the given scattered data sets. Typically, the number of holes used for probing is rather small, and the number of measurements in depth is extremely large.

3.4.4 The ATOC-GAMOT Project (AGP)

In late 1993, work was initiated on a time-varying volume visualization system in support of a large environmental project attempting to measure global warming by using acoustic

tomography. The underlying physics is that the speed of sound in water is proportional to the temperature, pressure, and salinity. If the pressure and salinity are known and the transmission time is measured with sufficient accuracy, the temperature can be determined with similar accuracy. For example, travel time variations of 0.2 sec over 3300 km paths indicate temperature changes of 0.1° Celsius. By utilizing many intersecting transmission paths, tomography can be used to determine the temperature throughout the ocean volume, much as computer-aided tomography (CAT) is used in medicine. The visualization task is twofold: visualize time-varying acoustic propagation in the ocean and visualize ocean model data. The ocean model is used to forecast the state of the ocean through which the acoustic signal will pass. The acoustic signal is the basis of the whole experiment.

Visualization of the acoustic propagation has required a number of different visualization approaches. One area of interest has been which eigenrays, that is, which acoustic projections, arrive at the receiver, with how much intensity, and at what time. Not only do the conditions vary within the sound channel over time, but the rays that deviate the most from the sound channel axis travel the fastest. Thus the signal does not arrive all at once, but over a finite time interval (for example, 3.5 sec) that varies diurnally. The computational model generates depth, distance, and time-traveled triples for each step of every ray that is determined to strike the receiver. The environmental context (bathymetry, shoreline, and so on) and conditions (salinity, temperature, flow regime, and such) are all important information. To visualize this data, a system called Acoustic Ray Visualizer (ARV) has been developed. ARV automatically determines the appropriate section of bathymetry and topology to use from the data output by the computational model. The user options include pausing the propagation, moving with the propagating ray, varying the number of time steps between each image, visualizing only some of the eigenrays, and recording the image corresponding to each time step to create a movie. A snapshot of the propagating rays is shown in Figure 3.21. The top 2D window gives a side view of the propagating rays, so that a more precise visual representation of the different propagation rates can be seen. The large 3D window shows the rays moving over the bathymetry.

Another interest is the variance of the acoustic path due to varying ocean conditions over transoceanic paths. Even with fixed source and receiver locations, these acoustic paths are known to deviate by 10 to 15 km due to seasonal variations. This is another time-varying 3D problem, but with much longer time scales. Associated with this interest is a desire to simply see the time-varying sound speed anomalies, which are derived by perturbing climatological data sets with dynamic ocean models. A very flexible program, ViewPerturb, has been developed to visualize this type of data. The interface allows the user to design his or her own color map based on the HSV color model, to clip the range of data mapped into the color map, and to create a movie from the rendered images. A number of different readers have been attached to the system, so that other time-varying 2D fields, for example, vorticity magnitude, sea-surface height, or salinity, can be visualized. An example of a relative vorticity field visualized using ViewPerturb is shown in Figure 3.22.

Visualization of the dynamic ocean models requires a significant understanding of computational fluid dynamics. The biggest hindrance to effective visualization is data overload. The ocean model (see [23]) has a 989×657 time-invariant uniform rectilinear grid in the horizontal with a latitude/longitude spacing of 0.125/0.176 degrees and six time-varying isopycnal layers. The internal time step of the model is on the order of 24 minutes, but data is only saved every 3.05 model days. This generates about 3GB of data per model year and

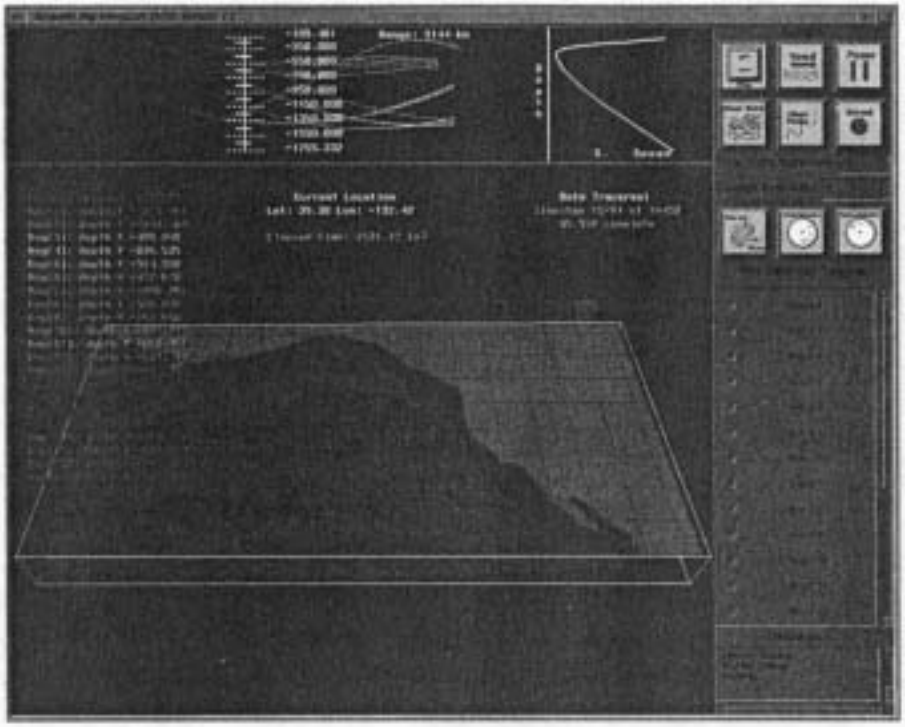


Figure 3.21: Snapshot of Acoustic Ray Visualizer (ARV). See Color Plate 17.

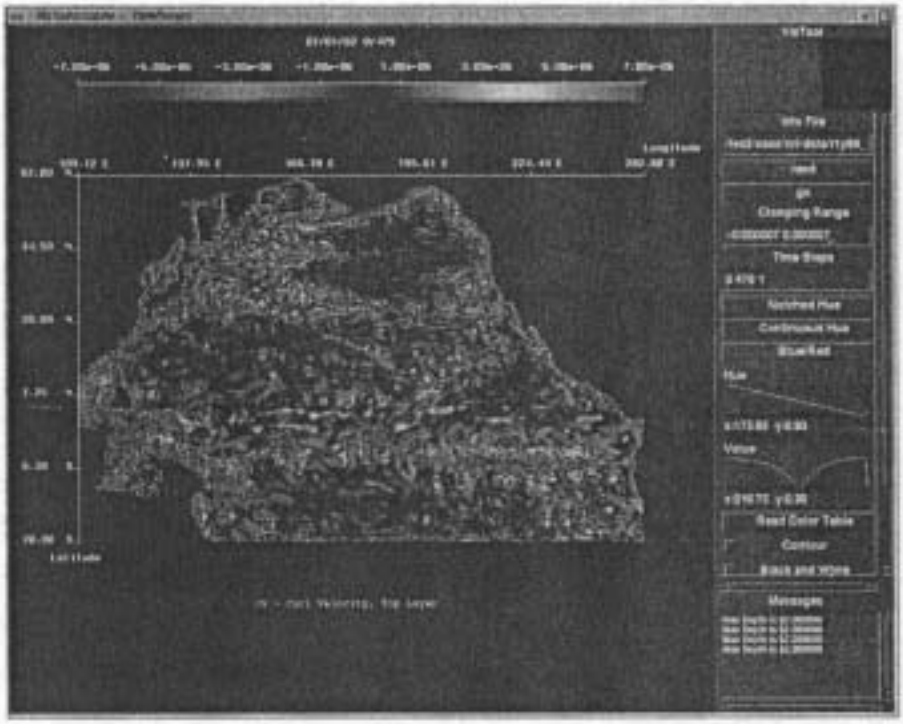


Figure 3.22: Relative vorticity for Pacific Ocean on January 1, 1982. See Color Plate 18.

at present, the model has been run for 13.5 model years. The visualizations that are needed are time-varying features such as waves, currents, and eddies. The geographic context must be given, since the flow physics is a function of the geographical location. The flow can be represented using moving arrows or a partially redundant mapping into the HSV color space. An example of this color wheel visualization scheme is shown in Figure 3.23. The direction of flow is mapped to a hue and the magnitude is mapped to both saturation and value (brightness). Thus high-magnitude southerly flows are bright red and small northerly flows are black with a slight tint of cyan.

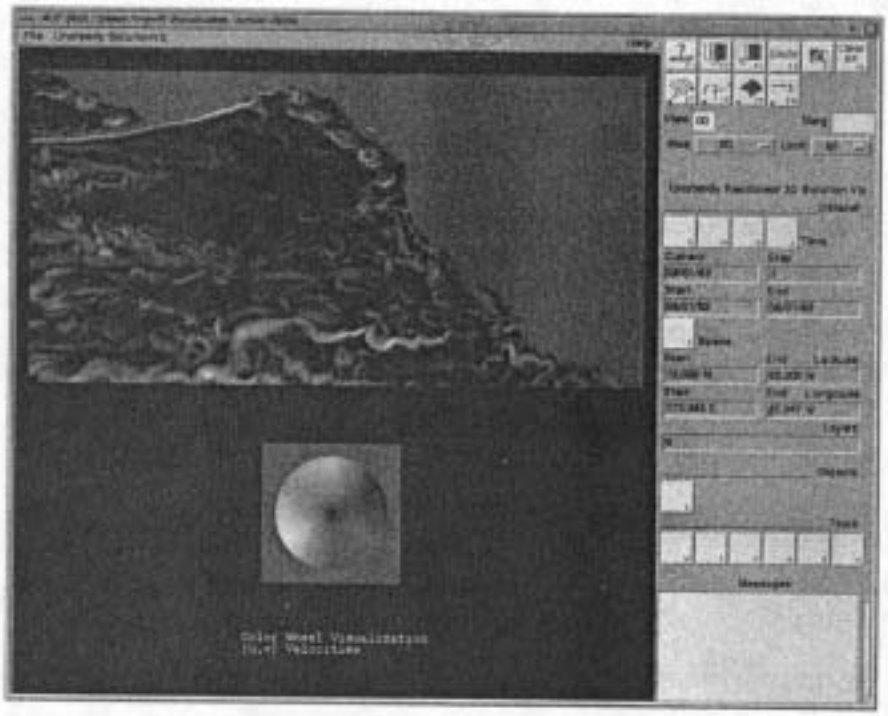


Figure 3.23: Flow in NE Pacific on August 1, 1983; direction of flow is mapped to a hue and magnitude to saturation and value; "rotated" color wheels close to coastline representing eddies. See Color Plate 19.

3.4.5 Feature Extraction from Oceanographic Data Sets

The ability to automatically recognize and track underlying features such as eddies and fronts from oceanographic data sets is an excellent tool, especially if important features can be detected automatically during the visualization process. The existence and therefore the locations of features are sometimes unknown even to a knowledgeable user at the beginning of the analysis phase. Without an automatic mechanism, intensive and time-consuming searches must be performed, otherwise features may not be revealed. In the analysis of the evolution and interaction of features, their temporal motions and fluctuations and their cross-correlations over multiparameter fields are best observed through animation. This again complicates the searching (via data classification) problem. In order to best characterize the features, locally optimized data classification must be achieved at each time step, at each depth level, and for each parameter. This is a difficult, if not impossible, task with traditional data classification methods.

A requirement of the feature extraction algorithm is the ability to operate effectively in noisy data sets. Although there is no commonly accepted measurement, the fundamental criterion is that the "signal-to-noise ratio" should be maximized at the output of the feature extractor (see [14]). The feature extraction methods for volumetric data sets can be considered an extension of those for computer vision and image processing. They can be classified into region-based and edge-based approaches. Region-based methods deal only with functional values. Examples can be found in [47, 48, 46]. The entire process usually contains extrema detection, thresholding, and feature isolation algorithms. Region-based methods are reliable even in low "contrast" situations, since they operate only on the original functional values. They are good if the possible features have unique and known functional values, especially when the unique values are extrema.

Edge-based methods attempt to detect features directly by tracing the boundary of possible features. Edge-based methods assume that features have some values that are significantly different from those outside the feature; therefore features must exhibit obvious spatial edges along the boundaries. Edges can be detected by computing the gradients. Edge-based methods are the most intuitive way to track features that can be best profiled by their edges. However, little work had been done in this area. The probable reasons include: 1) unlike images, data volumes are not necessarily regularly spaced, or even rectilinear, making the edge computation more complicated; and 2) 3D feature recognition algorithms are much more complicated. In [35], the 3D image edge operator was first extended and made applicable to nonregular 3D data volumes. That 3D image edge operator handles rectilinear data sets with irregular grid intervals in one direction, as are often used in environmental simulations. The edge operator was developed for locating, contouring, and tracking oceanic features, such as eddies (see [65]). It was later applied to cyclone and jet stream detection in meteorological data. The algorithms track features over space and time (see [64]).

Figure 3.24 shows the difference in simply detecting the edges and applying the filtering techniques based on parameters such as eccentricity and shape presented in [35]. Figure 3.25 shows eddies propagating through the Gulf of Mexico.

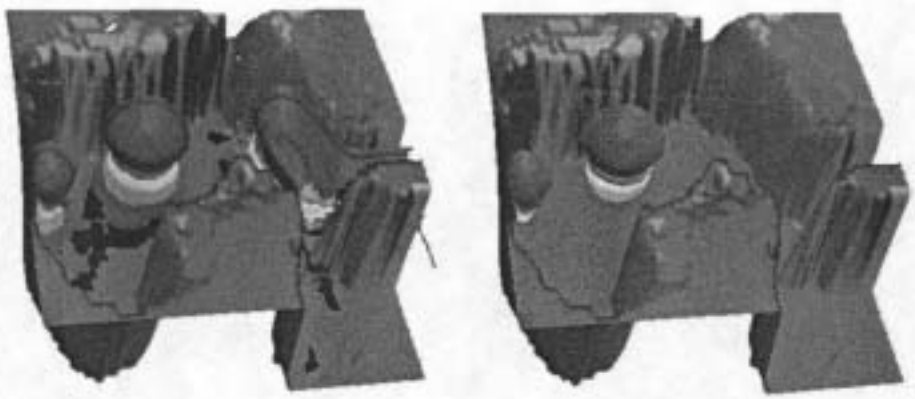


Figure 3.24: Difference in straightforward isosurfacing and enhanced edge-detection scheme. See Color Plate 20.

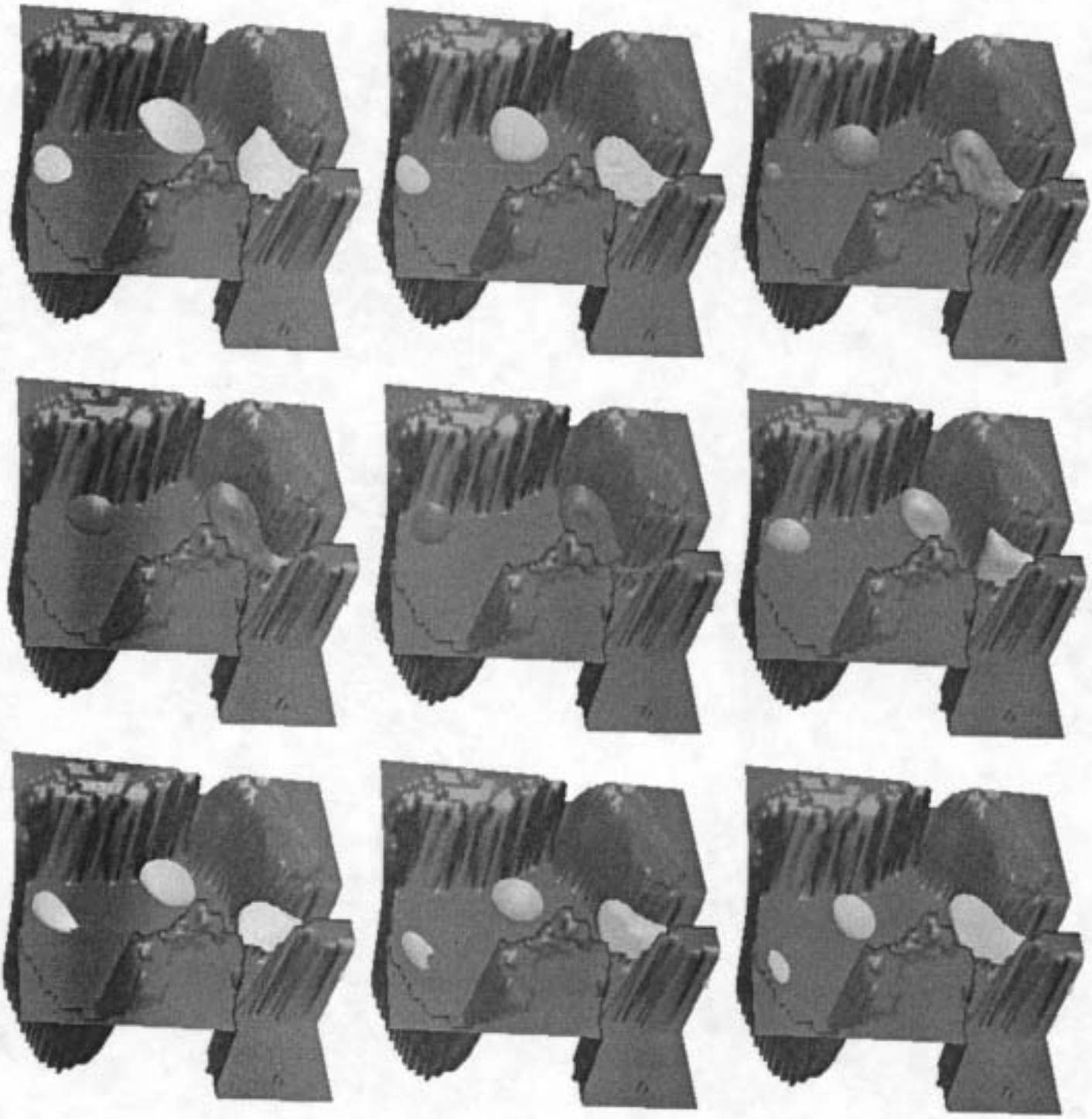


Figure 3.25: Example of eddies propagating through the Gulf of Mexico; sequence progresses from left to right and from top to bottom. See Color Plate 21.

3.4.6 Data Compression for Oceanographic Data Sets

As mentioned above, many real data sets, whether created by measurement or simulation, are "too big to visualize interactively." Thus an important issue in scientific visualization systems is to reduce the storage space and the access time of these data sets in order to speed up the visualization process. Progressive transmission offers a solution. The principle is to transmit the least amount of data necessary to generate a usable approximation of the original data. If the user requires further refinement, more data is fetched into the memory and a higher-resolution data set is reconstructed.

A progressive refinement scheme consists of four steps:

1. data encoding
2. data decoding
3. a refinement strategy
4. the rendering algorithm

State-of-the-art data encoding/decoding schemes consist of a wavelet transform (WT) technique and entropy coding. Because of its multiresolutional nature, the WT is very suitable for hierarchical manipulations such as progressive data transmission and tree-structured graphics rendering. Sophisticated methods for entropy coding and refinement control have been previously developed (see [55, 4, 31]). The need was for a better WT, which was more computationally efficient and produced more precise reconstruction of functional values in continuous space.

Previous work (see [36]) applied the truncated version of Battle-Lemarié (BL) wavelets to volume data and proposed a fast superposition algorithm for reconstruction of functional values in continuous space as a progressive transmission scheme. However, the BL wavelets are not perfect. First, they are computationally slow. This greatly limits their value in applications with strict speed requirements. Second, in most visualization techniques such as isosurface rendering and fluid topology extraction, the functional values in the continuous space must be calculated conveniently. By using an infinitely supported function—as is necessary with the BL wavelets—for the superposition basis, a complex approximation algorithm has to be utilized. A third drawback is that lossless data compression is difficult because the denominators of the coefficients are not in the form of 2^n.

A new progressive transmission scheme using spline biorthogonal wavelet bases was developed (see [51]). Because all the filter coefficients in the biorthogonal spline WT are dyadic rationals, multiplication and division operations can be simplified to addition and subtraction of the exponent for floating-point numbers, or to shifting of the exponent for integer numbers. This makes the transform very fast. This property also makes lossless coding of floating-point numbers possible. Symmetry is more easily achieved, resulting in a better handling of the data boundary and halving the number of multiplications required.

To combine a progressive transmission scheme with a visualization algorithm such as isosurface generation or ray casting, the functional values in the continuous volume must be approximated efficiently from the transform coefficients. Because the transform basis functions are compactly supported and can be explicitly formulated in a polynomial form, the reconstruction of functional values from the WT coefficients is straightforward.

With the given technique, the exact functional value at any real x can be computed through linear interpolation of the function values at the two discrete neighbors which surround x. In this case, the resulting $\overline{f}(x)$ is C^0 continuous. By increasing the filter length to k, $\overline{f}(x)$ will be C^{k-2} continuous. If the tensor product of the one-dimensional bases is used as the basis for the multidimensional WT, the same conclusion can be derived.

The scheme has been applied to data from an ocean model (see [23]). Model output from a 337×468 horizontal section of the Northeast (NE) Pacific was used. Layer thickness and current data in floating-point format—"prognostic variables" from the model—were visualized. Each of the six time-varying layers were coded independently.

To apply the biorthogonal WT to the data set, the boundary conditions had to be considered. In each frame (time step) of ocean model data, valid functional values are available only within the ocean area of the rectangular region (Figure 3.26a). This area is constant over time. To apply a 2D WT the ocean area must be approximated using square blocks. The block size used was $2^n \times 2^n$, where n was larger than the level of the WT in each data frame. The approximated ocean area had to cover the entire original ocean area (Figure 3.26b). For grid points in the approximated ocean area but not in the original area, interpolation and extrapolation techniques are used to yield function values. This introduces some discontinuity at the boundary. In the implementation, second-order Lagrange interpolation is used. Since all the WT bases are symmetric, the function values outside the boundary are achieved simply by reflection.

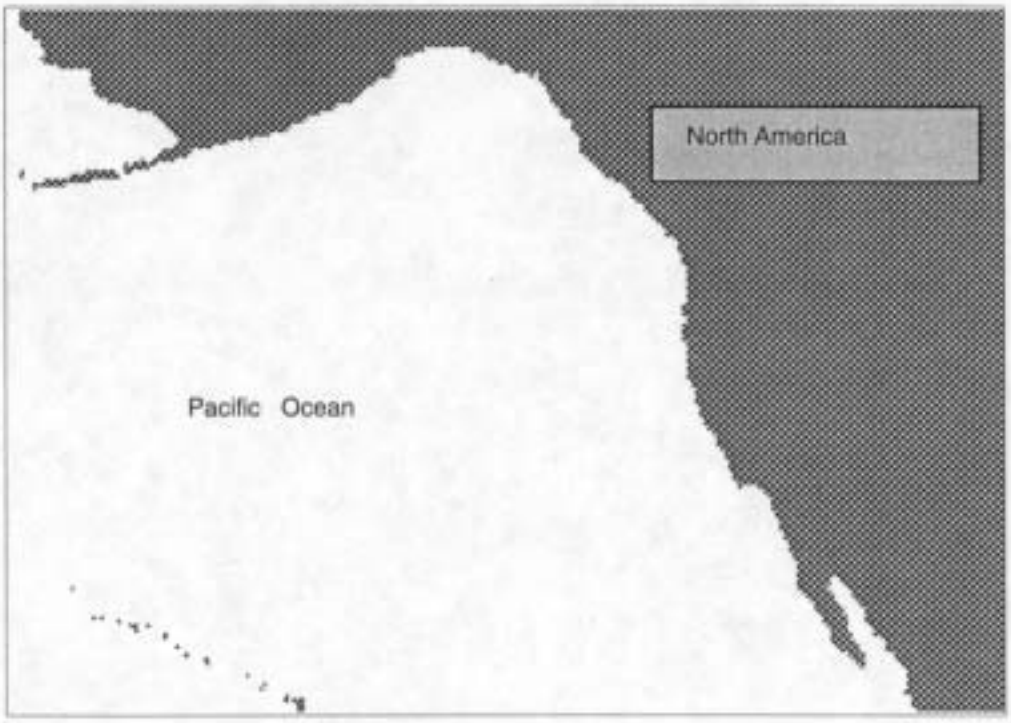
As shown in Figure 3.27, after applying the transform twice in each spatial dimension in a frame and once in the temporal dimension, only 1/32 of the transform coefficients are transmitted to the decoder. The reconstructed data has good quality and meets the requirements of further scientific visualization processes. By using the refinement control strategy proposed by Blandford (see [4]), only a small amount of data is needed for a better reconstruction.

In Figure 3.28, layer thickness compressed at the rate of 50:1 is shown. For a 2D vector field, the two components are encoded independently. Streamlines are generated both for the original field and the field reconstructed using only 1/32 of the WT coefficients; the results are shown in Figure 3.29. The additional compression (from 32:1 to 50:1) is obtained by using entropy encoding.

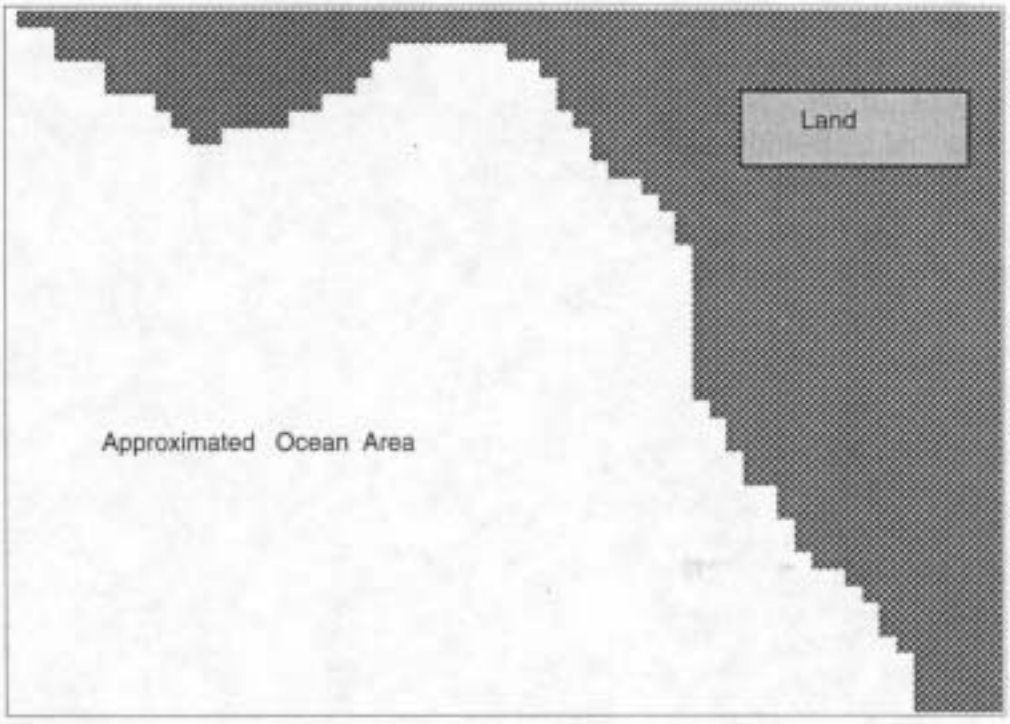
Applying a topology extraction algorithm (see [20]) to this vector field, the global and the stable topology structures remained relatively unchanged, even at a 16:1 compression ratio. With progressive refinement, more and more local and unstable features can be extracted (Figure 3.30). This is the low-pass effect of the WT.

Lossless compression was obtained by recording the necessary extra bits in an associated octree. The lossless compression ratio is about 1.5:1. The decoding speed, an important factor in interactive visualization system, is about 10 frames/sec on an SGI $Indigo^2$ machine. This satisfies the need of fast volume rendering in the application.

In summary, the WT scheme discussed above has several features that make it attractive. The transform itself is fast, and high-compression ratios can be achieved. The reconstructed data is of good quality and can be refined with a small amount of additional data. Boundaries can be handled gracefully. The reconstruction of functional values in continuous space from the WT coefficients requires only a simple polynomial, which is especially useful for scientific visualization applications.



(a)



(b)

Figure 3.26: Land and ocean areas in each data frame. *(a)* Ocean area in original data (white area); *(b)* Approximation of ocean area using 8×8 blocks. Approximated version covers original ocean area entirely.

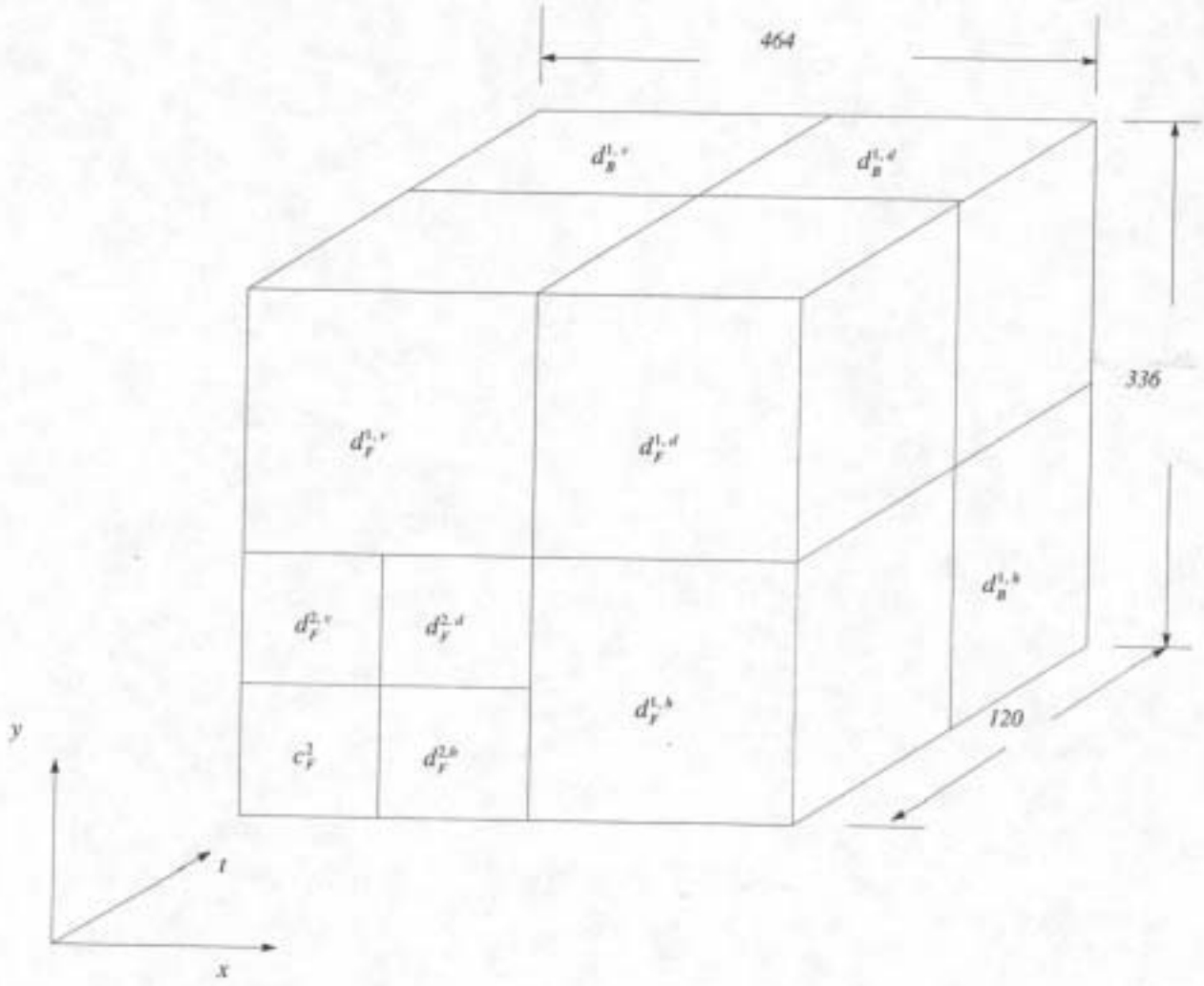


Figure 3.27: Wavelet transform coefficients of 3D ocean model data.

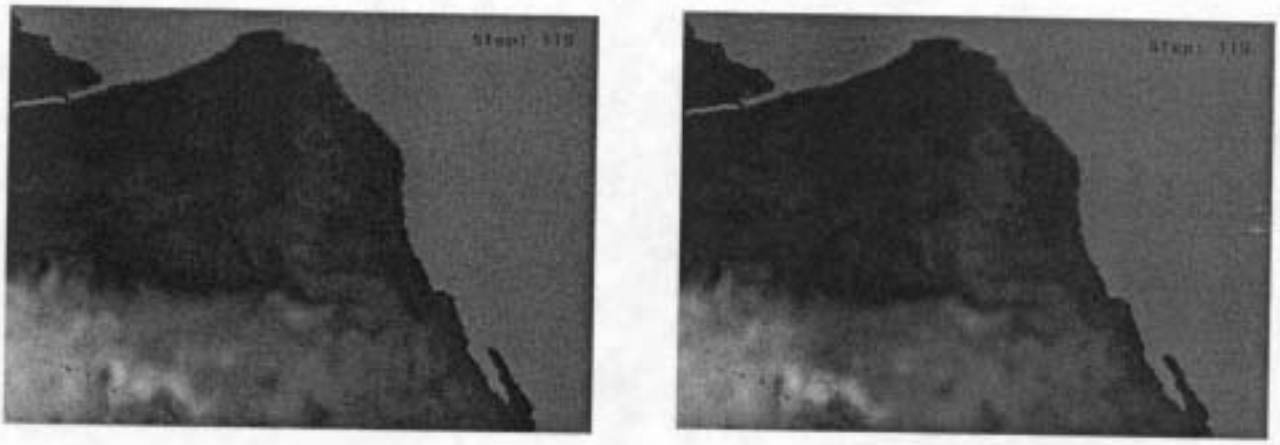


Figure 3.28: Layer thickness for NE Pacific Ocean. *(left)* original data; *(right)* reconstructed data (compression ratio 50:1). See Color Plate 22.

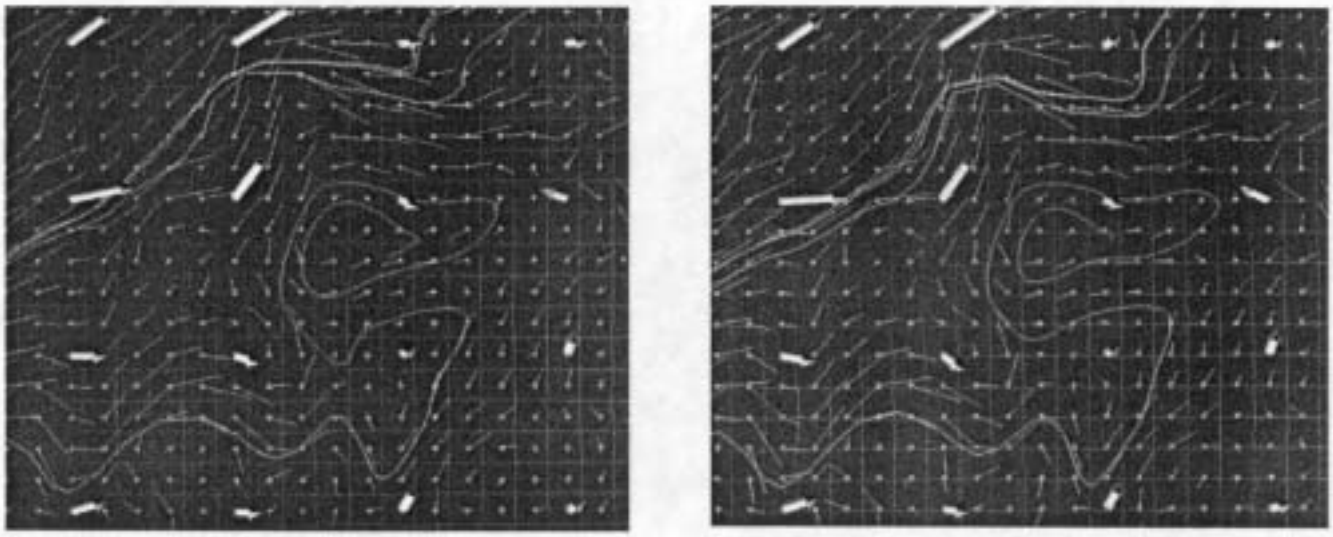


Figure 3.29: Velocity data for NE Pacific Ocean. *(left)* original data; *(right)* reconstructed data (compression ratio 50:1). See Color Plate 23.

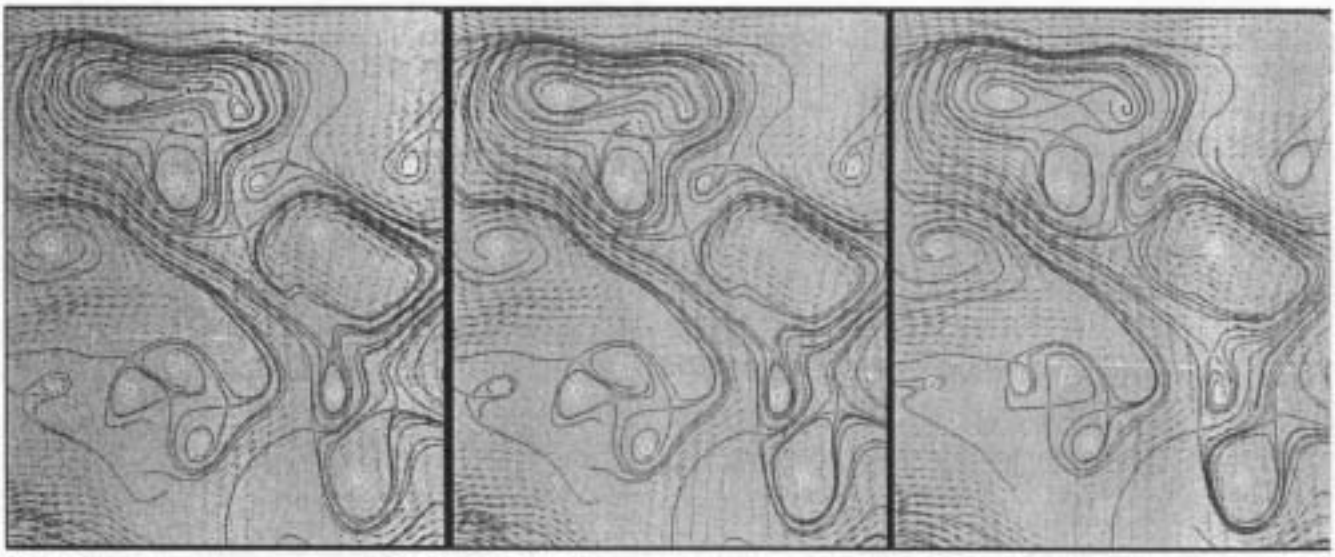


Figure 3.30: Vector field topology extraction on reconstructed data. *(left)* original data; *(middle)* reconstructed data using 1/4 of the WT coefficients; *(right)* reconstructed data using 1/16 of the WT coefficients. See Color Plate 24.

3.4.7 Concurrent Visualization

Computational field simulation visualization has typically been a postprocess. However, concurrent visualization allows the user to more effectively monitor the evolving simulation. The problem has typically been that the creator of the computational field simulation program does not know graphics programming or have access to adequate and/or easy-to-use visualization libraries. A secondary problem is that there is always a need for more compute power or more memory on the machine on which the simulation is being run. In an attempt to develop a concurrent visualization scheme, FAST (see [1] and [56]) was modified to take as input a script from a remote process. FAST was chosen because it was designed to deal with aerodynamic data on 3D curvilinear grids, the source program was available, it has a powerful and extensive scripting facility, and it had been used in previous visualization development work at the ERC-CFS.

Concurrent visualization can be described as either integrated or distributed (see [27]). Integrated systems combine the CFS solution program(s) and visualization modules into one program. This allows a shared memory space and simplified synchronization of the visualization and solution routines. Distributed systems separate the solution program(s) and visualization modules into different programs which communicate with each other. Since there is no longer shared memory, a large amount of communication and data transfer must take place; however, a distributed system offers the opportunity to exploit powerful computational engines connected via high-speed networks with high-performance graphics workstations.

The need to visualize time-varying data was the motivation for this work. Since FAST Version 1.1a did not support time-dependent solutions directly, a concurrent visualization system was developed which visualized the data as it was generated. The system is a master-slave design where the simulation software controls FAST via network communication. The network communication is based on the DTM library (see [37]). DTM handles the message processing required when communicating between dissimilar architectures. The simulation software is part of a family of programs known as UNCLE (see [52]). The original program, written in FORTRAN, was augmented with several C routines allowing the software to indirectly control the full functionality of FAST. Thus, instead of simply generating large simulation solution files which are cumbersome to visualize, the software has the ability to generate potentially smaller visualization files and the user has the ability to watch simulations develop.

Control of FAST was obtained via its scripting facility and named pipes (see [49]). FAST reads from the pipe and waits for data until an EOF signal is read. In order to keep the pipe open, another program, FASTd (FAST daemon), is started which receives messages from the network and translates them into script commands which are written to the named pipe. When a special message is received, signaling that the simulation is complete, FASTd closes the pipe and exits. The visualization entities created by FAST can be ARCGraph files (see [21]), which, as visualization objects (see [13]) and not images, allow the viewing angle and color map to be modified in postproduction.

This approach has shown great promise for utilizing distributed computing techniques to interactively visualize simulation results. The greatest hindrance to concurrent visualization is the lack of standard file and data formats for time-varying scientific data.

3.5 Conclusions

Various recent and ongoing research projects conducted in grid generation and scientific visualization at the ERC-CFS have been discussed. The most common grid generation techniques have been reviewed due to their importance for the development of visualization techniques for different grid types. The survey attempts to create more interest in grid generation aspects among visualization experts.

The basic principles of structured and unstructured grid generation have been pointed out. The NGP system is an example of a state-of-the-art system incorporating a wide variety of geometric modeling and grid generation methods to solve complex, real-world CFS problems.

The survey also provides insight into many scientific visualization projects, the most prominent one being the ATOC-GAMOT project dealing with the visualization of ocean currents and ocean sound propagation.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under contract EEC-8907070 (ERC-CFS) and ASC-9210439 (Research Initiation Award), the Strategic Environmental Research and Development Program (SERDP) and the Advanced Research Projects Agency (ARPA) via contract EEC-8907070 (ERC-CFS), the National Grid Project (NGP) consortium, the Office of Naval Research (ONR) under research grant N00014-92-J-4109, the Naval Oceanographic Office (NAVO) via contract NAS13-330, and the U.S. Army Corps of Engineers under contract DACA 39-93-K-0055.

Special thanks go to the members of the Grid Generation and Scientific Visualization thrusts and the research and development team of the NGP system at the ERC-CFS.

Bibliography

[1] G.V. Bancroft, F.J. Merritt, T.C. Plessel, P.G. Kelaita, R.K. McCabe, and A. Globus, "FAST: A Multiprocessing Environment for Visualization of CFD," *Visualization '90*, A. Kaufman, editor, IEEE Computer Society Press, Los Alamitos, Calif., 1990, pp. 14–27.

[2] R.E. Barnhill, G. Farin, and B. Hamann, "NURBS and Grid Generation," *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, IMA Volumes in Mathematics and its Applications 75, I. Babuska, J.E. Flaherty, W.D. Henshaw, J.E. Hopcroft, J.E. Oliger, and T. Tezduyar, editors, Springer-Verlag, New York, N.Y., 1995, pages 1–22.

[3] J.A. Benek, P.G. Buning, and J.L. Steer, *A 3-D Chimera Grid Embedding Technique*, AIAA paper 85-1523, AIAA, 1985.

[4] R.P. Blandford, "Wavelet Encoding and Variable Resolution Progression Transmission," *NASA Space and Earth Science Data Compression Workshop Proceedings*, 1993, pp. 25–34.

[5] J.E. Castillo, *Mathematical Aspects of Numerical Grid Generation*, SIAM, Philadelphia, Pa., 1991.

[6] W.M. Chan and J.L. Steger, "Enhancements of a Three-Dimensional Hyperbolic Grid Generation Scheme," *Applied Mathematics and Computation*, Vol. 51, 1992, pp. 181–205.

[7] J.F. Dannenhoffer, *A New Method for Creating Grid Abstractions for Complex Configurations*, AIAA paper 93-0428, AIAA, 1993.

[8] F.C. Dougherty, J.A. Benek, and J.L. Steger, *On the Application of a Chimera Grid Scheme to Store Separation*, NASA paper TM-88193, NASA, 1985.

[9] T. Elvins, "A Survey of Algorithms for Volume Visualization. *Computer Graphics*, Vol. 26, No. 3, 1992, pp. 194–201.

[10] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, third edition, Academic Press, San Diego, Calif., 1993.

[11] R. Franke, "Scattered Data Interpolation: Tests of Some Methods," *Math. Comp.*, Vol. 38, 1982, pp. 181–200.

[12] P.L. George, *Automatic Mesh Generation*, Wiley & Sons, New York, N.Y., 1991.

[13] A. Globus, *A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamic Results*, Technical Report RNR-92-031, NASA Ames Research Center, 1992.

[14] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley, New York, N.Y., 1992.

[15] W.J. Gordon, "Blending-Function Methods of Bivariate and Multivariate Interpolation and Approximation," *SIAM Journal of Numerical Analysis*, Vol. 8, No. 1, 1971, pp. 158–177.

[16] B. Hamann, "Modeling Contours of Trivariate Data," *Modélisation Mathématique et Analysis Numérique [Mathematical Modelling and Numerical Analysis]*, Vol. 26, No. 1, 1992, pp. 51–75.

[17] B. Hamann, "Construction of B-Spline Approximations for Use in Numerical Grid Generation," *Applied Mathematics and Computation*, Vol. 65, Nos. 1 & 2, special issue, 1994, pp. 295–314.

[18] B. Hamann, J.L. Chen, and G. Hong, "Automatic Generation of Unstructured Grids for Volumes Outside or Inside Closed Surfaces," *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, Pineridge Press Ltd., Swansea, UK, 1994, pp. 187–197.

[19] B. Hamann and R.F. Sarraga, "Finite Elements, Grid Generation, and Geometric Design," *Computer Aided Geometric Design*, Vol. 12, No. 7, special issue, Elsevier Science Publishing Co. Inc., New York, N.Y., 1995, pp. 647–784.

[20] J.L. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets," *Computer*, Vol. 22, No. 8, 1989, pp. 27–36.

[21] E.A. Hibbard and G. Makatura, *ARCGraph System User's Manual, Version 7.0*, NASA Ames Research Center, 1988.

[22] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Ltd., Wellesley, Mass., 1993.

[23] H.E. Hurlburt, A.J. Wallcraft, Z. Sirkes, and E.J. Metzger, "Modeling of the Global and Pacific Oceans: On the Path to Eddy-Resolving Ocean Prediction," *Oceanography*, Vol. 5, No. 1, 1992, pp. 9–18.

[24] B.A. Jean and B. Hamann, "Interactive Techniques for Correcting CAD/CAM Data," *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, Pineridge Press Ltd., Swansea, UK, 1994, pp. 317–328.

[25] A. Khamayseh and B. Hamann, "Elliptic Grid Generation Using NURBS Surfaces," *Computer Aided Geometric Design*, Vol. 13, No. 4, 1996. pp. 369–386.

[26] P.M. Knupp and S. Steinberg, *Fundamentals of Grid Generation*, CRC Press, Inc., Boca Raton, Fla., 1993.

[27] J.C. Lakey and R.J. Moorhead, "Concurrent Visualization of Time-Varying CFD Simulations," *Visual Data Exploration and Analysis*, Vol. 2178, R.J. Moorhead, D.E. Silver, and S.P. Uselton, editors, SPIE—The International Society for Optical Engineering, Bellingham, Wash., 1994, pp. 123–126.

[28] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, Vol. 8, No. 3, 1988, pp. 29–37.

[29] R. Löhner and P. Parikh, "Three-Dimensional Grid Generation by the Advancing Front Method," *International Journal for Numerical Methods in Fluids*, Vol. 8, 1988, pp. 1135–1149.

[30] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol. 21, No. 4, 1987, pp. 163–169.

[31] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, 1989, pp. 674–693.

[32] C.W. Mastin and J.F. Thompson, "Quasiconformal Mappings and Grid Generation," *SIAM Journal on Scientific and Statistical Computing*, Vol. 5, 1984, pp. 305–310.

[33] D. Meagher, "Geometric Modelling Using Octree Encoding," *Comp. Graph. Image Proc.*, Vol. 19, 1982, pp. 129–147.

[34] R.J. Moorhead, B. Hamann, C. Everitt, S.C. Jones, J. McAllister, and J.H. Barlow, "Oceanographic Visualization Interactive Research Tool (OVIRT)," *Visual Data Exploration and Analysis*, Vol. 2178, R.J. Moorhead, D.E. Silver, and S.P. Uselton, editors, SPIE—The International Society for Optical Engineering, Bellingham, Wash., 1994, pp. 24–30.

[35] R.J. Moorhead and Z. Zhu, "Feature Extraction for Oceanographic Data Using a 3D Edge Operator," *Visualization '93*, G.M. Nielson and D. Bergeron, editors, IEEE Computer Society Press, Los Alamitos, Calif., 1993, pp. 402–405.

[36] S. Muraki, "Volume Data and Wavelet Transforms," *IEEE Computer Graphics and Applications*, Vol. 13, No. 4, 1993, pp. 50–56.

[37] *Data Transfer Mechanism Programming Manual, Version 2.3*, National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, 1992.

[38] G.M. Nielson, T.A. Foley, B. Hamann, and D.A. Lane, "Visualizing and Modeling Scattered Multivariate Data," *IEEE Computer Graphics and Applications*, Vol. 11, No. 3, 1991, pp. 47–55.

[39] G.M. Nielson and B. Hamann, "Techniques for the Interactive Visualization of Volumetric Data," *Visualization '90*, A. Kaufman, editor, IEEE Computer Society Press, Los Alamitos, Calif., 1990, pp. 45–50.

[40] G.M. Nielson and B. Hamann, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes," *Visualization '91*, G.M. Nielson and L.J. Rosenblum, editors, IEEE Computer Society Press, Los Alamitos, Calif., 1991, pp. 83–91.

[41] K.L. Parmley, J.F. Dannenhoffer, and N.P. Weatherill, *Techniques for the Visual Evaluation of Computational Grids*, AIAA paper 93-3353, AIAA, 1993.

[42] F.P. Preparata and M.I. Shamos, *Computational Geometry*, third printing, Springer-Verlag, New York, N.Y., 1990.

[43] M.G. Remotigue, "The National Grid Project: Making Dreams into Reality," *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, N.P. Weatherill, P.R. Eiseman, J. Häuser, and J.F. Thompson, editors, Pineridge Press Ltd., Swansea, UK, 1994, pp. 429–439.

[44] P. Sabella, "A Rendering Algorithm for Visualizing 3D Scalar Fields," *Computer Graphics*, Vol. 22, No. 4, 1988, pp. 51–58.

[45] H. Samet, "The Quadtree and Related Hierarchical Data Structures," *Computing Surveys*, Vol. 16, No. 2, 1984, pp. 187–285.

[46] R. Samtaney, D.E. Silver, N.J. Zabusky, and J. Cao, "Visualizing Features and Tracking their Evolution," *Computer*, Vol. 27, No. 7, 1994, pp. 20–27.

[47] D.E. Silver and N.J. Zabusky, "3D Visualization and Quantification of Evolving Amorphous Objects," *Extracting Meaning from Complex Data: Processing, Display, Interaction II*, Vol. 1459, E.J. Farrell, editor, SPIE—The International Society for Optical Engineering, Bellingham, Wash., 1991, pp. 97–108.

[48] D.E. Silver and N.J. Zabusky, "Quantifying Visualizations for Reduced Modeling in Nonlinear Science: Extracting Structures from Data Sets," *Journal of Visual Communications and Image Representation*, Vol. 4, No. 1, 1993, pp. 46–61.

[49] M. Sobell, *A Practical Guide to the Unix System*, Benjamin-Cummings, 1989.

[50] B.K. Soni and B. Hamann, "Computational Geometry Tools in Grid Generation," *Advances in Hydro-Science & Engineering*, Volume I, Part B, S.S.Y. Wang, editor, 1993, pp. 2004–2009.

[51] H. Tao and R.J. Moorhead, "Progressive Transmission of Scientific Data Using a Biorthogonal Wavelet Transform," *Visualization '94*, R.D. Bergeron and A. Kaufman, editors, IEEE Computer Society Press, Los Alamitos, Calif., 1994.

[52] L.K. Taylor and D.L. Whitfield, *Unsteady Three-Dimensional Incompressible Euler and Navier-Stokes Solver for Stationary and Dynamic Grids*, AIAA paper 91-1650, AIAA, 1991.

[53] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin, *Numerical Grid Generation*, North-Holland, New York, N.Y., 1985.

[54] J.F. Thompson and N.P. Weatherill, *Aspects of Numerical Grid Generation: Current Science and Art*, AIAA paper 93-3539-CP, AIAA, 1993.

[55] K.H. Tzou, "Progressive Image Transmission: A Review and Comparison of Techniques," *Optical Engineering*, Vol. 26, No. 7, 1987, pp. 581–589.

[56] P. Walatka, J. Clucas, K. McCabe, T. Plessel, and R. Potter, *FAST User's Guide, FAST 1.1a*, NASA Ames Research Center, NAS Division, RND Branch, 1994.

[57] N.P. Weatherill, *The Generation of Unstructured Grids Using Dirichlet Tessellations*, MAE Report 1715, Princeton University, 1985.

[58] N.P. Weatherill, "A Method for Generating Irregular Computational Grids in Multiply Connected Planar Domains," *International Journal for Numerical Methods in Fluids*, Vol. 8, 1988, pp. 181–197.

[59] N.P. Weatherill, "The Delaunay Triangulation in CFD," *Computers and Mathematics with Applications*, Vol. 24, 1992, pp. 129–150.

[60] N.P. Weatherill, O. Hassan, D.L. Marcum, and M.J. Marchant, "Grid Generation by the Delaunay Triangulation," *von Karman Institute for Fluid Dynamics 1993-1994 Lecture Series*, 1994.

[61] M.A. Yerri and M.S. Shephard, "A Modified Quadtree Approach to Finite Element Mesh Generation," *IEEE Computer Graphics and Applications*, Vol. 3, 1983.

[62] M.A. Yerri and M.S. Shephard, "Automatic 3D Mesh Generation by the Modified-Octree Technique," *Int. J. Num. Meth. Eng.*, Vol. 20, 1984, pp. 1965–1990.

[63] M.A. Yerri and M.S. Shephard, "Automatic Mesh Generation for Three-Dimensional Solids," *Comp. Struct.*, Vol. 20, 1985.

[64] Z. Zhu and R.J. Moorhead, "Exploring Feature Detection Techniques for Time-Varying Volumetric Data," *IEEE Workshop on Visualization and Machine Vision*, Seattle, Wash., 1994, pp. 45–54.

[65] Z. Zhu, R.J. Moorhead, H. Anand, and L.R. Raju, "Feature Extraction and Tracking in Oceanographic Visualization," *Visual Data Exploration and Analysis*, Vol. 2178, R.J. Moorhead, D.E. Silver, and S.P. Uselton, editors, SPIE—The International Society for Optical Engineering, Bellingham, Wash., 1994, pp. 31–39.