Wavelet Representation of Contour Sets

Martin Bertram[1,2] Daniel E. Laney[3] Mark A. Duchaineau[3] Charles D. Hansen[1] Bernd Hamann[4]
Kenneth I. Joy[4]

Abstract: We present a new wavelet compression and multiresolution modeling approach for sets of contours (level sets). In contrast to previous wavelet schemes, our algorithm creates a parametrization of a scalar field induced by its contours and compactly stores this parametrization rather than function values sampled on a regular grid. Our representation is based on hierarchical polygon meshes with subdivision connectivity whose vertices are transformed into wavelet coefficients. From this sparse set of coefficients, every set of contours can be efficiently reconstructed at multiple levels of resolution. When applying lossy compression, introducing high quantization errors, our method preserves contour topology, in contrast to compression methods applied to the corresponding field function. We provide numerical results for scalar fields defined on planar domains. Our approach generalizes to volumetric domains, time-varying contours, and level sets of vector fields.

Keywords: Contours, Geometry Compression, Isosurfaces, Level Sets, Multiresolution Methods, Wavelets

1 Introduction

Scientific visualization methods help us to explore and understand the nature of vast amounts of digital data produced by numerical simulations on supercomputers or by imaging technology like computer tomography. Visualizing scalar fields via exploration of their isosurface behavior is one of the most powerful ways to gain insight into a physical phenomenon. Our approach is driven by the need to explore very large scalar fields interactively by browsing through their continuous space of contours. In the past, multiresolution methods were developed for the modeling, rendering, and exploration of complicated two-manifold data, *e.g.*, large-scale isosurfaces [1]. In order to explore the entire contour space of a scalar field more powerful methods are required, as entire families of contours have to be extracted, represented, and rendered. The approach we are presenting here is driven by such considerations. We introduce a new framework for the multiresolution approximation of a multitude of contours defined by a single scalar field. This framework promises to have significant impact on state-of-the-art visualization and exploration of truly massive, tera-scale scalar field data.

Visualization methods often rely on continuous geometric models representing the relevant topological and geometric features of a data set. Multiresolution modeling techniques, like wavelet transforms [3, 16], provide efficient progressive access to local geometry. Wavelet transforms coupled with progressive coders for quantized coefficients are among the most efficient schemes for compression, error-driven querying, and progressive transmission of data defined on regularly gridded domains [15, 18]. When visualizing derived quantities or features, such as contours, these need to be extracted from a locally reconstructed geometric model. This extraction process can be very expensive, especially in the case of volume data, since an unknown surface topology needs to be recovered.

Standard wavelet compression algorithms [18] transform a function into wavelet coefficients of expectedly small absolute values. These coefficients are quantized (rounded to integers) or thresholded (selected by magnitude of absolute values) and compressed by a progressive coding scheme like *zero trees* [15]. When extracting contours from highly compressed data altered by a quantization error, there exists no guarantee of obtaining topologically correct contours. When reconstructing data from thresholded or quantized wavelet coefficients, for example, the resulting contours may even have additional components enclosing local extrema of the reconstruction error, see Figures 1 and 2.

The wavelet approach presented here overcomes this problem by compressing a parametrization of a field function that is induced by its contours, rather than compressing a field function directly. Our approach also simplifies the topology of represented contours. However, this simplification is performed in an initial step of our algorithm, where a finite set of selected contours, called *base contours*, is extracted. All other contours represented by our method have the topology of a corresponding base contour of the closest isovalue. Since the set of base contours can be chosen arbitrarily, our method introduces a predictable topological error reducing the quantity of topological changes that need to be stored. The topology of contours represented by our method is invariant under the level of detail such that the topological error is not augmented in coarse representations.

Starting with the set of base contours, we construct a coarse mesh structure, the *base mesh*, covering the domain of the underlying field function. This base mesh is recursively subdivided, and its vertices are projected onto intermediate contours. The resulting adaptive mesh structure is equivalent to a subdivision surface/volume with displacement of vertices correcting the geometry at finer levels of detail. Our algorithm represents these displacements compactly in the form of sparse wavelet coefficients. The contours produced by our subdivision process are either linear or cubic polynomials.

We represent a scalar field by a continuous parametrization of its domain that is defined by a subdivision surface/volume. This parametrization is a function mapping a manifold into Euclidean space. In the case of planar contours, our manifold domain has one global parameter specifying the isovalue and one local parameter traversing the corresponding contour. (In the case of isosurfaces of trivariate functions, our manifold has one parameter for the isovalue and two local parameters traversing an isosurface.) The coarsest level of resolution is defined by a base mesh providing both, the manifold topology and a coarse parametrization obtained by recursive subdivision. During the subdivision process, geometric detail can be expanded from wavelet coefficients resulting in representations at higher level of resolution.

Our representation of contour sets is equivalent to a representation of the underlying field function, but it provides rapid access to every contour at multiple levels of resolution. This is a highly desirable property for real-time visualization of contours, allowing

[1]University of Utah, SCI Institute, 50 S. Central Campus Drive, Room 3190, Salt Lake City, UT 84112, USA.

[2]University of Kaiserslautern, Department of Computer Science, P.O. Box 3049, D-67653 Kaiserslautern, Germany.

[3]Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA.

[4]Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California, Davis, CA 95616-8562, USA.

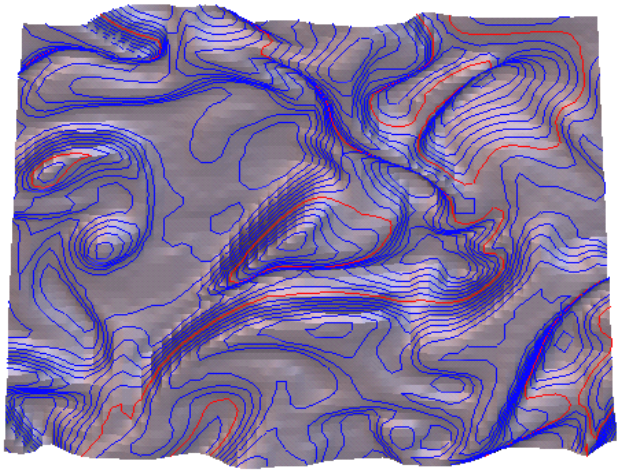


Figure 1: Contours of a 64×64 piecewise bilinearly interpolated slice of a volume data set, taken from a numerical simulation of a Rayleigh-Taylor instability.

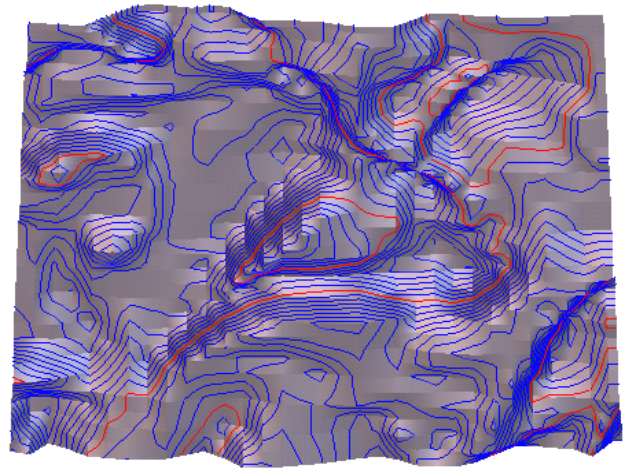


Figure 2: Contours of the scalar field shown in Figure 1, reconstructed form thresholded wavelet coefficients (12 percent), using a linear spline wavelet. This compressed representation does not preserve the topology of extracted contours.

for interactively changing isovalues and rendering multiple transparent isosurfaces at once. Our representation provides additional flexibility for algorithms processing contours with the goal of improving the underlying field function. For example, *constrained fairing* of all contours of a field function is a non-trivial operation that becomes fairly simple when using our approach.

2 Related Work

Multiresolution contouring schemes extract isosurfaces from hierarchical scalar field representations providing multiple levels of detail. Weber *et al.* [19] present an efficient construction method for crack-free isosurfaces from adaptively refined hexahedral domains. A similar approach using a hierarchical octree structure for interactive view-dependent contouring is presented by Westermann *et al.* [20]. A real-time rendering approach for multiple transparent isosurfaces reconstructed from a tetrahedral grid hierarchy is described by Gerstner [4].

Wood *et al.* [21] use a surface wavefront propagation method for constructing a coarse base mesh approximating an isosurface with correct topology. Their approach provides a semi-regular triangular subdivision hierarchy of an isosurface that is useful for wavelet compression. In previous work, we have constructed quadrilateral base meshes with subdivision hierarchy that were used for wavelet compression of isosurfaces [1]. Our wavelet construction for subdivision surfaces [2] generalizes to higher dimensions, *e.g.*, volumes of manifold topology, like level sets and time-varying surfaces. Wavelet constructions for subdivision surfaces were initially described by Lounsbery *et al.* [8, 16].

When using wavelet approaches for geometry compression [6], it becomes important to construct smooth surface parametrizations by improving the regularity of control meshes. For triangle meshes, such regular parametrizations are constructed by the *MAPS algorithm* described by Lee *et al.* [7]. Similar algorithms need to be developed for pseudo-regular meshing of three-dimensional level sets. A multiresolution approach for matching contours defined on different cutting planes is presented by Meyers [10]. Efficient meshing algorithms for level sets are described by Sethian [14].

To our knowledge, previous methods have not attempted to reparametrize sets of contours for the purpose of wavelet compression. Hence, our general approach is innovative, combining individual techniques from different fields, such as contour extraction, mesh generation, and subdivision surface wavelets.

3 Adaptively Representing Contour Sets

This section describes our novel multiresolution approach for sets of contours. We describe our algorithm in the context of bivariate scalar fields and provide extensions to volumetric domains, time-varying contours, and level sets.

3.1 Overview of the Algorithm

Our algorithm first constructs a coarse base mesh induced by certain base contours. This mesh is then regularly subdivided, and the new vertices are projected onto intermediate contours. Finally, we use a wavelet transform for compression and multiresolution modeling of this mesh structure, defining smooth sets of contours by recursive subdivision. Our algorithm consists of the following steps that are illustrated in Color Plates (a–f):

1. Extraction of a prescribed set of *base contours*, using, for example, uniformly distributed isolevels. This set of contours defines the topology of all intermediate contours represented by our scheme.

2. Sampling *base vertices* distributed uniformly with respect to arc lengths from the extracted base contours. These vertices will represent the coarsest level of detail for our parametrization. Hence, the set of base contours selected in step 1 should not be too dense.

3. Constructing links between base vertices on adjacent base contours and relaxing these links by moving the base points on their corresponding contours.

4. Filling the space between adjacent base contours and their links with convex polygons that have low numbers of edges. The resulting *base mesh* serves as coarsest level of detail, defining a smooth parametrization of contours when recursively subdivided (using, for example, Catmull-Clark subdivision [9]).

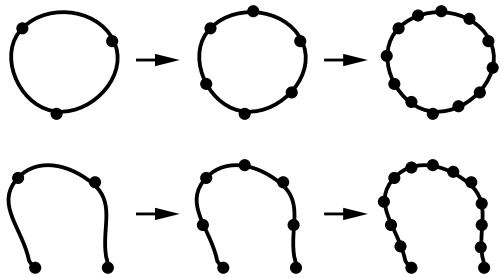


Figure 3: Dyadic refinement of a closed and an open contour component

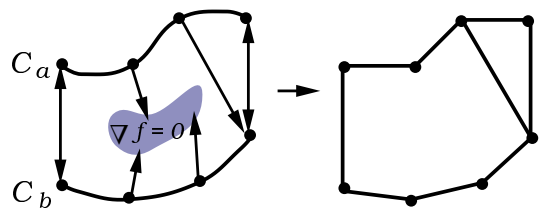


Figure 4: Constructing links between contours C_a and C_b by following the gradient starting at base vertices. Some vertices cannot be linked due to field regions of zero gradient.

5. Regular subdivision of the base mesh by inserting new vertices at the centroid of every polygon and in the middle of every edge. The vertices obtained by subdivision are connected to define a quadrilateral mesh structure that is recursively refined. Instead of applying stationary subdivision, which would result in incorrect intermediate contours, the new vertices are projected onto corresponding contours. This subdivision process terminates at a resolution slightly finer than the grid resolution of the underlying field function.

6. Subdivision-surface wavelets [1, 2] are used to generate a hierarchy of continuous parametrizations. The differences between individual levels of detail are compactly represented by wavelet coefficients. Data compression can be achieved by thresholding or by encoding quantized coefficients [11, 15].

3.2 Constructing Base Meshes

As a first step we extract a finite set of base contours using a standard approach. The corresponding isovalues can be uniformly distributed or they can be more densely sampled in certain regions of interest. Then we define base vertices by re-sampling the base contours at approximately equidistant intervals of arc length Δ. The value of Δ depends on the number n_s of dyadic refinement levels, see Figure 3, that we will compute and on the finest sampling distance δ, which should be slightly smaller than the edge length of the regular grid defining the field function. Hence, we use

$$\Delta = 2^{n_s}\delta. \tag{3.1}$$

All boundary points of contours need to be base vertices, such that the base contours can be completely generated by dyadic refinement. Additionally, we require every contour component to have at least three base vertices, to avoid degenerate base polygons.

The next task is to fill the space between every adjacent pair of base contours, say C_a and C_b, with convex polygons. Therefore, it is desired to connect matching pairs of base vertices from both

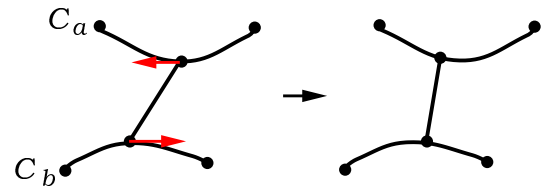


Figure 5: Relaxing a link between base vertices on contours C_a and C_b by displacing these vertices along their contours.

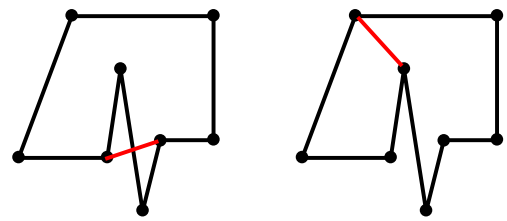


Figure 6: Splitting a non-convex polygon. Left: invalid split; right: correct split.

contours, which will improve the fairness of our final parametrization. We use Newton-iteration to propagate the base vertices of C_a onto the contour C_b. In each step of this iteration, the movement of a vertex is restricted to a maximal distance of δ, to avoid divergence due to shallow gradients. Some vertices will not converge to the contour C_b, since they may get stuck at local extrema or zero-gradient areas, and the iteration must terminate after a prescribed number of steps. Those vertices that converge to contour C_b are linked to the closest base point on C_b and their initial position on contour C_a is restored. To find all possible links, this step of the algorithm is repeated with the vertices of C_b, iterating towards contour C_a, see Figure 4. Base vertices located on the boundary of the data set are simply connected by traversing this boundary.

The length of the individual links between every pair of base vertices is minimized by an iterative procedure, allowing the base vertices to move a certain distance along their corresponding contours, see Figure 5. Here, we restrict the maximal displacement of a vertex to the value $\frac{\Delta}{3}$, to avoid coincidence of adjacent base vertices. This step is necessary to improve smoothness of the final parametrization and to avoid intersections of polygon strips defined by the base vertices of every contour component. Isolated components, "islands", are connected by one additional link to the closest base vertex on its surrounding contour component.

The mesh structure resulting from this procedure already defines a set of closed polygons covering the scalar field domain. However, some polygons may still be very large and non-convex and need to be subdivided further. Additionally, we need represent these polygons explicitly. For this purpose, we traverse every polygon in counter-clockwise orientation of edges and record the participating base vertices. We use every base vertex as a starting point for constructing a potential polygon. Every edge in the mesh has two associated flags for traversal in each direction, which are set when a polygon is constructed. These flags are tested for every traversal to avoid multiple constructions of the same polygon. The constructed polygons are then recursively split until they are convex and consist of no more than five edges. Splitting a polygon is performed by connecting a pair of close, non-adjacent vertices, avoiding self-intersections and augmentation of the enclosed region in case of a non-convex polygon, see Figure 6. The resulting set of polygons is a convex tessellation of the domain, the base mesh.

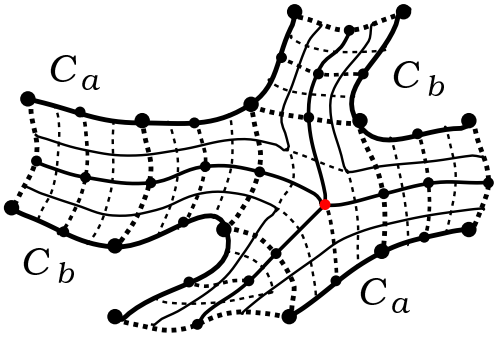


Figure 7: Regular mesh refinement near a saddle point located in the center of the five-sided patch.

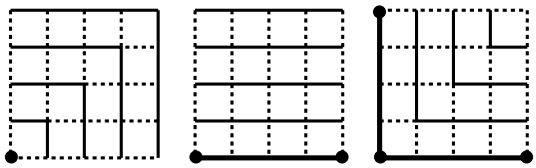


Figure 8: Mapping intermediate isovalues to vertices defined by subdivision. The first refinement step generates quadrilateral patches that have either one, two, or three vertices located on a base contour.

3.3 Regular Mesh Refinement

Once we have generated our base mesh, we apply recursive subdivision using the refinement connectivity of Catmull-Clark subdivision inserting vertices at the centroids of polygons and on their edges. The first subdivision step generates quadrilaterals that are regularly refined in the subsequent steps, as illustrated in Figure 7. Instead of applying stationary subdivision rules to compute the coordinates for vertices on finer levels, we place them on intermediate contours. The subdivision process terminates at a resolution slightly finer than the resolution of the initial rectilinear grid defining the field function. This mesh hierarchy is then compressed using wavelets.

Before we can project the new vertices onto intermediate contours, we have to define an isovalue for every vertex. After the first subdivision step, the resulting vertices are either located on a base contour or placed in the space between two base contours. In the latter case, these vertices will be associated with the average of both corresponding isovalues. For the subsequent levels of regular, rectilinear refinement, we use the templates illustrated in Figure 8: vertices located on edges are assigned the average isovalue of both incident vertices. Vertices located inside a quadrilateral are assigned the average of the minimal and the maximal isovalues of the quadrilateral's four corner vertices.

Every vertex is projected onto a contour with the correct isovalue. For this purpose, we use a constrained Newton iteration coupled with Laplacian smoothing of the mesh (moving every vertex to the centroid of its neighbors). In every step of the Newton iteration, a vertex is propagated along the gradient of the field and subsequently relaxed orthogonal to the gradient by projecting the Laplacian displacement onto a vector/plane orthogonal to the gradient. Again, the maximal displacement is limited by the distance δ. Due to the topology simplification imposed by the choice of base contours, some vertices cannot be projected onto the correct contour, since a nearby component of this contour does not exist. In this case, the relaxation prevents the mesh from entangling. The iteration process must terminate after a finite number of steps.

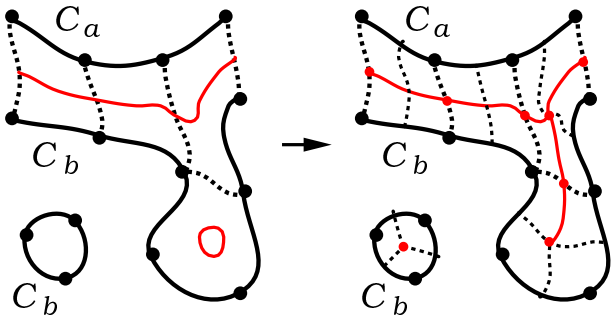


Figure 9: Topology of contour C_a changes into topology of C_b at an intermediate contour. The topology of this intermediate contour (left) cannot be represented correctly (right).

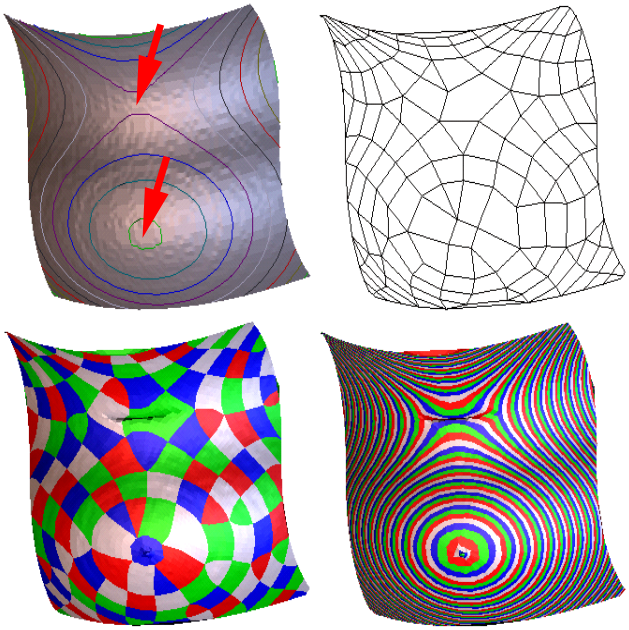


Figure 10: Algorithm applied to a surface with two critical points (a saddle point and a local minimum). Small regions of the mesh collapse near these points, due to topological error.

Besides critical points (points where contour topology changes, *e.g.*, saddle points and local extrema), the worst-case scenario are long "headlands" in the scalar field, where the mesh is either collapsed or stretched along a ridge, see Figure 9. However, the geometric error of every mesh vertex is bounded by one half of the sampling distance for base isovalues. The behavior of our meshing algorithm is shown in Figure 10 and Color Plates (a–f).

From the mesh structure we have constructed, every contour can be derived immediately by linear interpolation of its closest contours that are explicitly represented in the mesh. Alternatively, we can use a subdivision scheme, like Catmull-Clark, to refine the mesh smoothly. We do not need to store the isovalues associated with every vertex, since these can be recovered from the base mesh.

3.4 Subdivision-surface Wavelets

Starting with our regularly refined mesh hierarchy composed of vertices located on certain contours, we can efficiently derive any set of contours using subdivision and linear interpolation. For compres-

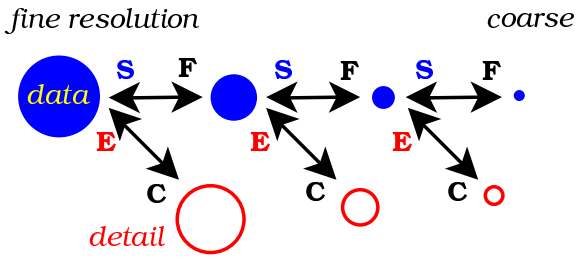


Figure 11: Modeling paradigm of a wavelet transform.

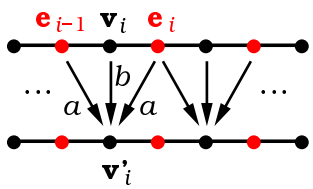


Figure 12: Vertex manipulation defined by an s-lift operation.

sion purposes and level-of-detail rendering, we need a multiresolution representation of this mesh structure providing these operations:

S Subdivision. This operation defines stationary subdivision rules providing a continuous limit surface when applied recursively. The mesh vertices correspond to control points of smooth basis functions.

E Expanding detail. At every level of refinement, geometric detail can be added to a subdivision surface. This detail is compactly stored in the form of wavelet coefficients and can be expanded from these.

F Fitting. This operation reverses a subdivision step. Based on all vertices on a fine level, the vertices on the next coarser level are predicted such that they provide a good approximation to the fine level when applying subdivision, again.

C Compacting detail. The difference between two levels of resolution, *i.e.*, the displacement of mesh vertices when applying **F** followed by **S**, is compactly stored in form of wavelet coefficients that replace the vertices removed by **F**.

The modeling paradigm of such a multiresolution representation is illustrated in Figure 11. These four operations define a wavelet transform for subdivision surfaces. We have constructed wavelets for bilinear and bicubic subdivision generalized to arbitrary meshes with regular refinement [2] and used the bicubic wavelet transform for multiresolution modeling of large-scale isosurfaces [1]. We summarize the details necessary to implement these transforms in the remainder of this section.

Our wavelet transforms are computed by a few local vertex manipulations, called *lifting* operations [17], since they can be used to manipulate the shape of basis functions. Considering a polygon strip composed of vertices $\mathbf{v}_i$ and its dyadic refinement with vertices $\mathbf{e}_i$ located on the edges $\mathbf{v}_i\mathbf{v}_{i+1}$, we define two lifting operations:

s-lift(a, b)**:**

$$\mathbf{v}_i \leftarrow a\mathbf{e}_{i-1} + b\mathbf{v}_i + a\mathbf{e}_i. \tag{3.2}$$

w-lift(a, b)**:**

$$\mathbf{e}_i \leftarrow a\mathbf{v}_i + b\mathbf{e}_i + a\mathbf{v}_{i+1}. \tag{3.3}$$

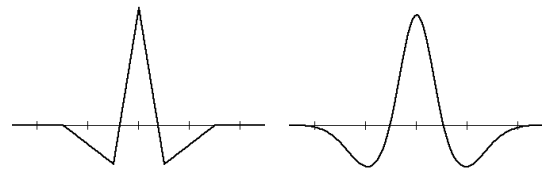


Figure 13: Linear and cubic B-spline wavelets

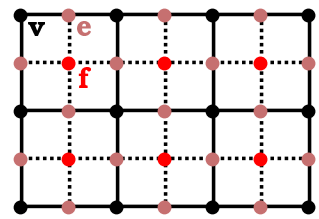


Figure 14: Regularly refined mesh, composed of vertices of types $\mathbf{v}$ (coarse resolution mesh), $\mathbf{e}$, and $\mathbf{f}$ (wavelet coefficients).

An s-lift operation is illustrated in Figure 12. This operation manipulates coefficients associated with *scaling functions* representing the individual levels of resolution, and a w-lift operation manipulates coefficients associated with *wavelets* representing geometric detail, *i.e.*, displacements between two levels.

The operations **S** and **E** define the *reconstruction* or *synthesis*, which is one step of an inverse wavelet transform. The vertices $\mathbf{v}_i$ represent initially a coarse level of resolution and the vertices $\mathbf{e}_i$ contain wavelet coefficients. After a reconstruction step, all vertices represent the next finer level of resolution. An inverse wavelet transform is computed by repeated reconstruction starting with a coarse base polygon. The reconstruction procedures for wavelet transforms based on dyadic refinement of linear and cubic B-splines are defined as follows:

Linear B-spline wavelet reconstruction:

$$\begin{aligned} &\mathrm{s-lift}(-\tfrac{1}{4}, 1); \\ &\mathrm{w-lift}(\tfrac{1}{2}, 1). \end{aligned}$$

Cubic B-spline wavelet reconstruction:

$$\begin{aligned} &\mathrm{s-lift}(-\tfrac{3}{8}, 1); \\ &\mathrm{w-lift}(\tfrac{1}{2}, 1); \\ &\mathrm{s-lift}(\tfrac{1}{4}, \tfrac{1}{2}), \end{aligned}$$

The basis functions of the transform corresponding to a wavelet coefficient are depicted in Figure 13.

.

The operations **F** and **C** represent wavelet *decomposition* or *analysis*, which is the inverse of a reconstruction step. The decomposition formulae for our one-dimensional wavelet constructions are defined by the inverse of every individual lifting operation applied in reverse order. Decomposition is defined as follows:

Linear B-spline wavelet decomposition:

$$\begin{aligned} &\mathrm{w-lift}(-\tfrac{1}{2}, 1); \\ &\mathrm{s-lift}(\tfrac{1}{4}, 1). \end{aligned}$$

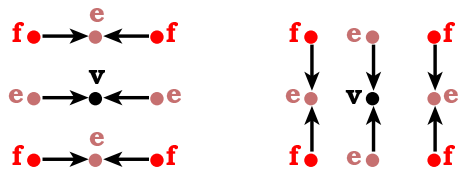


Figure 15: A two-dimensional s-lift is computed by applying its one-dimensional equivalent to the rows and columns of a regularly refined grid.

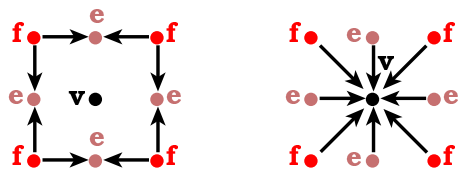


Figure 16: Two-dimensional s-lift operation performed in different order of vertex updates.

Cubic B-spline wavelet decomposition:

$$\begin{aligned} &\mathrm{s-lift}(-\tfrac{1}{2}, 2); \\ &\mathrm{w-lift}(-\tfrac{1}{2}, 1); \\ &\mathrm{s-lift}(\tfrac{3}{8}, 1). \end{aligned}$$

A wavelet transform is computed by repeated decomposition, starting with a fine resolution and terminating at the coarse resolution of a base polygon.

We now describe the generalization of these lifting operations to polygon meshes with regular subdivision hierarchy. The refinement of a regular, rectilinear mesh is illustrated in Figure 14. The vertices corresponding to wavelet coefficients are located on edges and polygons (faces) of the coarse mesh and are denoted by $\mathbf{e}$, and $\mathbf{f}$, respectively. On a completely regular mesh, a lifting operation is performed by applying the corresponding one-dimensional operation to the rows and columns, see Figure 15. Instead of updating the vertices $\mathbf{v}$ twice in an s-lift operation, we can change the order of computation such that every vertex is modified only once, as illustrated in Figure 16.

The corresponding two-dimensional lifting operations can be defined in a notation without indices, where $\overline{x}_y$ denotes the average of the vertices of type x that are adjacent to vertex y (or that belong to the closest stencil around y). For example, $\overline{\mathbf{v}}_\mathbf{e}$ is the midpoint of an edge and $\overline{\mathbf{v}}_\mathbf{f}$ is the centroid of a polygon. Using this notation, the two-dimensional lifting operations are defined like this:

2D s-lift(a, b)**:**

$$\begin{aligned} \mathbf{e} &\leftarrow b\mathbf{e} + 2a\overline{\mathbf{f}}_\mathbf{e}; \\ \mathbf{v} &\leftarrow b^2\mathbf{v} - 4a^2\overline{\mathbf{f}}_\mathbf{v} + 4a\overline{\mathbf{e}}_\mathbf{v}. \end{aligned} \tag{3.4}$$

2D w-lift(a, b)**:**

$$\begin{aligned} \mathbf{e} &\leftarrow b\mathbf{e} + 2a\overline{\mathbf{v}}_\mathbf{e}; \\ \mathbf{f} &\leftarrow b^2\mathbf{f} - 4a^2\overline{\mathbf{v}}_\mathbf{f} + 4a\overline{\mathbf{e}}_\mathbf{f}. \end{aligned} \tag{3.5}$$

The advantage of these index-free definitions is that they can be used for irregular meshes with regular refinement. not only applicable to regular meshes. These lifting operations are well defined for extraordinary vertices (vertices that do not have four incident edges) and for arbitrary polygons in a base mesh. For a correct transformation of mesh boundaries, we apply the one-dimensional lifting operations to all vertices located on these boundaries. The overall wavelet transform is analogous to the one-dimensional transform, except that for all inner mesh vertices equations (3.2) and (3.3) are substituted by equations (3.4) and (3.5), respectively. A two-dimensional scaling function and a wavelet are depicted in Figure 17.

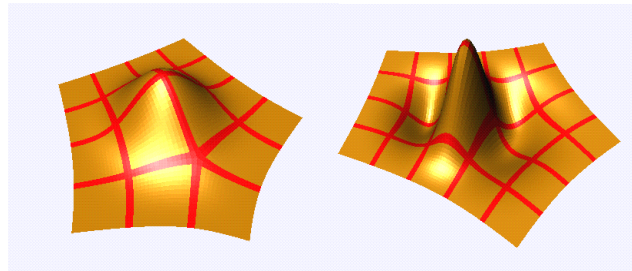


Figure 17: Generalized bicubic scaling function and wavelet.

Starting with the finest-level mesh structure constructed in the previous section, we compute our wavelet decomposition repeatedly until we reach the base mesh. The base vertices then represent a coarse approximation that is obtained by subdivision without expansion of detail. All vertices that are not base vertices contain wavelet coefficients that can be used to reconstruct the subdivision level where these vertices were introduced. For compression purposes, we can quantize the wavelet coefficients and compress them using, for example, arithmetic coding. We note that all coefficients have two coordinates, since they represent points and vectors in the plane.

4 Extensions of our Algorithm

We outline some modifications to our algorithm that are necessary to represent time-varying contours and to represent three-dimensional manifolds.

4.1 Time-varying Contours and Level Sets

Time-varying contours and level sets, *i.e.*, surfaces evolving over time like shock waves and material interfaces in fluid simulations, can be represented and compressed in the same way as contour sets. A major difference of time-varying curves/surfaces is that they can become self-intersecting over time, whereas contours propagate locally in only one direction when their isovalues are monotonically changed. Our algorithm for constructing base meshes cannot be used for curves/surfaces of this type, since it assumes that the set of base polygons provides a planar tessellation without self-intersections.

In general, it is possible to construct meshes with manifold topology approximating time-varying objects. For this purpose, we need to construct a mapping between objects from consecutive base time steps. For the case of one-dimensional contours, a multiresolution tiling algorithm is presented by Meyers [10]. This algorithm constructs polygons connecting contours on different planes corresponding to different times steps. We could use this method for generating base meshes of manifold topology that could then be subdivided recursively and iteratively displaced onto contours at intermediate time steps. Level set and efficient marching methods for meshing time-dependent surfaces are described by Sethian [14].

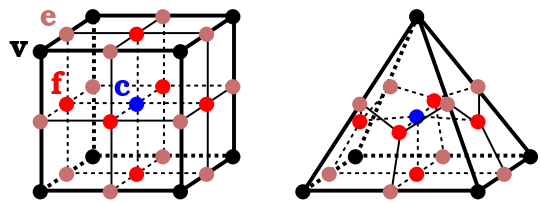


Figure 18: Regular subdivision of polyhedra. Subdividing a pyramid results in for hexahedra and one type-4 cell.

Transform	Level	No. of coeff.	L^2-error	L^1-error
none	3	40947	0.37	0.06
bicubic	2	10331	0.65	0.29
bicubic	1	2631	1.18	0.72
bicubic	0	656	2.61	1.89
bilinear	2	10331	0.63	0.29
bilinear	1	2631	1.00	0.61
bilinear	0	656	1.82	1.29

Table 1: Geometric error of represented contours relative to increment of isovalue at finest subdivision level (index three).

4.2 Wavelet Representation of Three-manifolds

In the case of time-varying surfaces or sets of static isosurfaces, lattices composed of polyhedral cells need to be constructed, connecting two surface components of consecutive base time steps or filling the space between two adjacent base isosurfaces. These lattices are recursively refined by placing new vertices inside each cell, on every face, and on every edge, see Figure 18. A generalization of Catmull-Clark surfaces to this type of volumetric subdivision is provided by MacCracken and Joy [9].

Many types of polyhedra, like prisms and tetrahedra, produce hexahedra after the first subdivision step, allowing for regular refinement. Unfortunately, some polyhedra, like pyramids, produce so-called *type-n* cells composed of $2n + 2$ vertices and $2n$ faces. These reproduce two type-n cells when subdivided. To keep the mesh structure simple, it is desired to avoid type-n cells, except for the case $n = 3$ (hexahedra).

Our wavelet transform generalizes nicely to volumetric (and higher-dimensional) subdivision, since the individual lifting operations can be computed by a sequence of vertex-manipulations for every type of vertex, analogously to the two-dimensional case. When applied to a regularly gridded domain, these lifting operations define tensor-product basis functions.

5 Results

We have implemented and tested our algorithm for scalar fields defined on planar domains. As an example data set, we have used a slice of rich geometric detail taken from a three-dimensional numerical simulation of a *Rayleigh-Taylor instability*, courtesy of Lawrence Livermore National Laboratory. The initial slice is defined by $64 \times 64 = 4096$ byte samples given on a regularly gridded domain. We extracted nine base contours at uniformly distributed isovalues, re-sampled at a resolution δ of half the length of a grid edge. We used $n_s = 3$ levels of subdivision for mesh generation.

Our algorithm generated a base mesh composed of 656 vertices and 661 polygons, resulting in 40947 vertices (corresponding to wavelet coefficients) after three levels of subdivision, which corresponds to an over-sampling factor of about ten. For our wavelet representation, we need to store the coordinates of the 656 base vertices (their isovalues can be recorded by grouping vertices of same contours together into a list), the connectivity of the base mesh, and the wavelet coefficients, which can be quantized and encoded at high compression rates. Hence, our over-sampled representation of contours may use less storage space than the original data set. This becomes crucial when converting large-scale data sets into our representation. The computationally expensive part of our algorithm is the projection of vertices onto contours, which required less than ten seconds on an SGI O^2 workstation using a 180 MHz R5000 processor.

We used our generalized bilinear and bicubic wavelet transforms to compute different levels of resolution, obtained by removing wavelet coefficients on the highest-resolution levels and reconstructing the mesh at the finest level of refinement, obtained after three subdivisions. The reconstructed meshes are depicted in Color Plates (g–l). The finest-resolution mesh is shown in Color Plate (f). We rendered these meshes by assigning the same color to all quadrilaterals located between each pair of adjacent contours at the finest level.

The geometric errors of all contours that are explicitly represented in the mesh at finest level are shown in Table 1. These errors represent the difference between the isovalue associated with a mesh vertex and the real function value of the underlying scalar field at the vertex location. All errors are relative to the difference of two adjacent contours represented in the finest mesh. An error larger than one means that adjacent contours may be intersecting and the mesh no longer defines a unique parametrization of the domain. In the case of lossy compression, this can be avoided by appropriately choosing a threshold for quantization of wavelet coefficients. We note that this problem does not occur in the case of time-varying contours, where self-intersections over time are natural.

The geometric error at the finest-resolution mesh is caused by regions of incorrect topology where vertices could not be projected onto their corresponding contour. For the majority of vertices, the geometric error is zero, which explains why the L^1-error (the average of individual errors) is much smaller than the L^2-error (the square-root of the averaged squared errors).

6 Conclusions

Our approach supports the exploration of scalar fields via their contours. A key issue of our approach is the construction of a base mesh of manifold topology that is induced by a set of originally extracted contours. This base mesh defines a subdivision surface/volume from which all intermediate contours can be reconstructed efficiently. During this subdivision process, geometric detail is expanded from wavelet coefficients increasing the level of detail. For efficiently representing very large data sets it will be crucial to select and construct a locally optimal set of base contours and to blend the resulting local base meshes to a global representation. A solution to this challenging problem might be the consideration of topological characteristics of a field function, like critical points and separatrices, which can be constructed explicitly for scalar and vector fields [12, 13].

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. It was also supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Ad-

vanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of ALSTOM Schilling Robotics and SGI.

We thank the members of the Scientific Computing and Imaging (SCI) Institute at the University of Utah, the scientists at the Center for Applied Scientific Computing (CASC) at Lawrence Livermore National Laboratory, and the members of the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

References

[1] M. Bertram, M.A. Duchaineau, B. Hamann, and K.I. Joy, *Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization*, Proceedings of Visualization 2000, IEEE, 2000, pp. 389–396 & 579.

[2] M. Bertram, M.A. Duchaineau, B. Hamann, and K.I. Joy, *Generalized B-spline subdivision surface wavelets for geometry compression*, submitted to Computer Aided Geometric Design, 2001.

[3] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1992.

[4] T. Gerstner, *Fast Multiresolution Extraction of Multiple Transparent Isosurfaces*, Proceedings of VisSym '01, Ascona, Swizerland, May 2000 (to appear).

[5] B. Jawerth and W. Sweldens, *An overview of wavelet based multiresolution analysis*, Society for Industrial and Applied Mathematics (SIAM) Rev., Vol. 36, No. 3, 1994, pp. 377–412.

[6] A. Khodakovsky, P. Schröder, and W. Sweldens, *Progressive Geometry Compression*, Computer Graphics, Proceedings of Siggraph 2000, ACM, 2000, pp. 271–278.

[7] A.W.F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, *MAPS: multiresolution adaptive parameterization of surfaces*, Computer Graphics, Proceedings of SIGGRAPH '98, ACM 1998, pp. 95–104.

[8] M. Lounsbery, T.D. DeRose, and J. Warren, *Multiresolution analysis for surfaces of arbitrary topological type*, ACM Transactions on Graphics, Vol. 16, No. 1, ACM, Jan. 1997, pp. 34–73.

[9] R. MacCracken and K.I. Joy, *Free-form deformations with lattices of arbitrary topology*, Computer Graphics, Proceedings of SIGGRAPH '96, ACM, 1996, pp. 181–188.

[10] D. Meyers, *Multiresolution tiling*, Computer Graphics Forum, Vol. 13, No. 5, December 1994, pp. 325–340.

[11] A. Moffat, R.M. Neal, and I.H. Witten, *Arithmetic coding revisited*, ACM Transactions on Information Systems, Vol. 16, No. 3, ACM, July 1998, pp. 256–294.

[12] X. Tricoche, G. Scheuermann, and H. Hagen, *Topology-Based Visualization of Time-Dependent 2D Vector Fields*, Proceedings of VisSym '01, Ascona, Swizerland, May 2000 (to appear).

[13] X. Tricoche, G. Scheuermann, and H. Hagen, *A Topology Simplification Method for 2D Vector Fields*, Proceedings of Visualization 2000, IEEE, 2000, pp. 359–366 & 576.

[14] J.A. Sethian, *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, 1999.

[15] J.M. Shapiro, *Embedded image coding using zerotrees of wavelet coefficients*, IEEE Transactions on Signal Processing, Vol. 41, No. 12, Dec. 1993, pp. 3445–3462.

[16] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin, *Wavelets for Computer Graphics–Theory and Applications*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

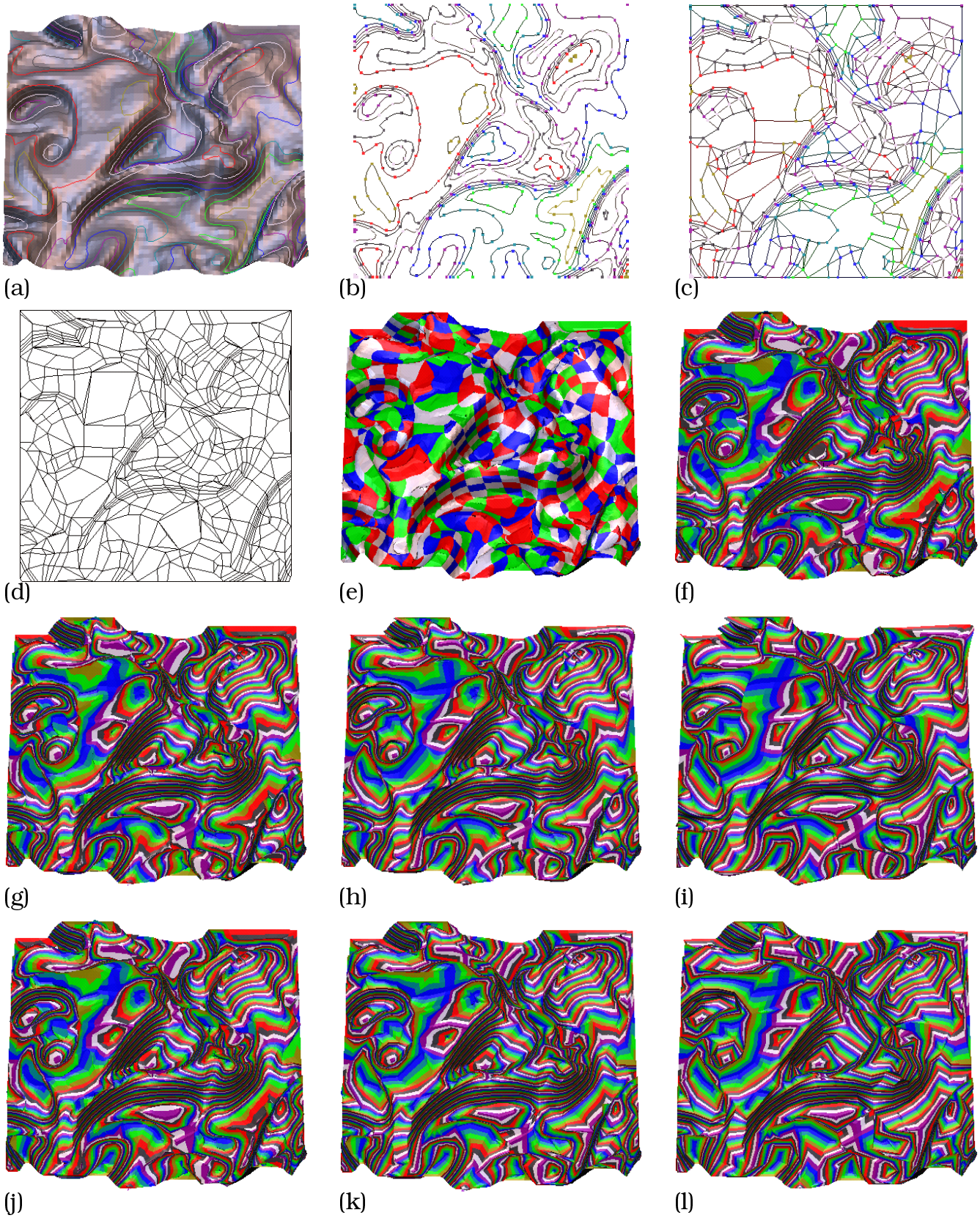
[17] W. Sweldens, *The lifting scheme: a custom-design construction of biorthogonal wavelets*, Applied and Computational Harmonic Analysis, Vol. 3, No. 2, 1996, pp. 186–200.

[18] H. Tao and R.J. Moorhead, *Progressive transmission of scientific data using biorthogonal wavelet transform*, Proceedings of Visualization '94, IEEE, 1994, pp. 93–99.

[19] G.H. Weber, O. Kreylos, T.J. Ligocki, J.M. Shalf, H. Hagen, B. Hamann, and K.I. Joy, *Extraction of Crack-free Isosurfaces from Adaptive Mesh Refinement Data*, Proceedings of VisSym '01, Ascona, Swizerland, May 2000 (to appear).

[20] R. Westermann, L. Kobbelt, and T. Ertl, *Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces*, The Visual Computer Vol. 15, No. 2, 1999, pp. 100–111.

[21] Z.J. Wood, M. Desbrun, and P. Schröder, and D. Breen: *Semi-regular mesh extraction from volumes*, Proceedings of Visualization 2000, IEEE, 2000, pp. 275–282.



Color Plate. (a) Slice of trivariate scalar field; (b) base contours; (c) linking base vertices; (d) base polygons; (e) patches defined by fitted mesh; (f) mesh with colored contours; (g–i) bicubic wavelet reconstructions using 10331, 2631, and 656 coefficients, respectively; (j–l) corresponding bilinear wavelet reconstructions.