Atomic Meshes: from Seismic Imaging to Reservoir Simulation

DAVE HALE
Landmark Graphics, 7409 S. Alton Ct., Englewood, CO, USA *dhale@lgc.com*

Abstract

Combining image processing and optimization techniques, we automatically align a lattice of points (atoms) with horizons and faults in a 3-D seismic image. Connecting these points yields an unstructured space-filling polyhedral (atomic) mesh. This single data structure can integrate multiple tasks, such as seismic interpretation, reservoir characterization, and flow simulation, thereby reducing work cycle times and errors.

Horizons and faults in seismic images often correspond to discontinuities in subsurface properties, such as permeability. Depending on the type of mesh elements desired, we align the lattice of atoms either on or alongside image features. 3-D Delaunay triangulation of atoms aligned *on* image features yields a tetrahedral mesh, with triangular faces of tetrahedra coincident with subsurface discontinuities. Alternatively, 3-D Delaunay triangulation of atoms aligned *alongside* image features yields a dual Voronoi polyhedral mesh in which polygonal faces coincide with subsurface discontinuities.

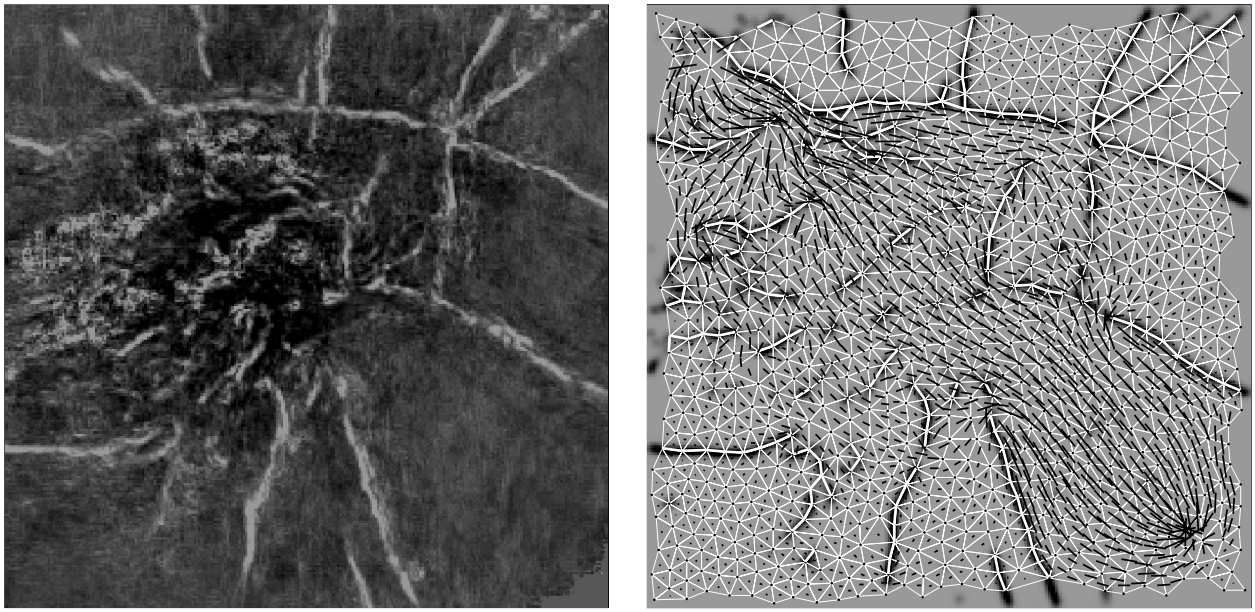
Introduction

Today's work cycle from seismic imaging to reservoir simulation requires a variety of data structures — simple arrays, triangulated surfaces, non-manifold frameworks, corner-point grids, etc. — to represent the earth's subsurface. Conversions among these different representations are both time consuming and error prone.

We may reduce both work cycle times and errors by performing tasks such as seismic interpretation, reservoir characterization, and flow simulation using a single space-filling mesh. For example, consider the horizontal slice of a 3-D seismic image of geologic faults shown in Figure 1a. Figure 1b shows a space-filling mesh of triangular elements that is aligned with those faults (bold white lines). Flow vectors (black line segments) indicate both the direction and velocity of fluid flowing from an injector in the upper left to a producer in the lower right part of the image. *We performed both the fault interpretation and the flow simulation on the same space-filling mesh.*

Today, one typically interprets faults by drawing curves on a seismic image, interpolating those curves to form surfaces, stitching those surfaces together to form a topologically consistent framework, all *before* creating a mesh within that framework. In contrast, we interpret faults *after* aligning a space-filling mesh with relevant features in a seismic image. In other words, we interpret seismic images using a data structure that works for flow simulation.

The flow simulation illustrated in Figure 1b is simple. Permeability is constant and isotropic, except at the faults, which we assumed to be impermeable, Flow is single-phase and steady-



(a) (b)

Figure 1: From seismic imaging (a) to flow simulation (b). The seismic image of geologic faults in (a) is a 256×256-sample (6.4×6.4-km) horizontal slice taken from a 3-D seismic image that was processed to enhance such discontinuities. The space-filling mesh in (b) was automatically aligned with features in the seismic image, and then used to identify the faults and simulate the flow of fluid around them.

state (computed using a discontinuous Galerkin method [1]). These limitations merely simplify the illustration, and are not required by the meshing process.

We call the mesh of triangles in Figure 1b an *atomic mesh*, because we compute the locations of mesh nodes (atoms) using simple models of inter-atomic forces. Attractive and repulsive forces among atoms cause them to arrange themselves in a geometrically regular lattice, as in a crystal.

Crystal lattices provide a useful analogy for geologic modeling. In this analogy, layers of atoms in crystals correspond to geologic layers, and dislocations in crystals correspond to geologic faults. We manipulate geologic models by moving atoms. As we move one atom, atoms nearby move along with it, thereby preserving the regularity of the lattice. (Physical models such as this are commonly used in meshing [2] [3].) Here, we move atoms to align a mesh with features in an image.

In this context, an image is simply a uniformly sampled 2-D or 3-D array of numbers. Atomic meshing is most relevant for images that contain edges or ridges with which mesh elements should be aligned, so that discontinuities in imaged properties are well represented by the mesh. For example, given a CT scan of a core sample, we might align a mesh with pore space boundaries. In another example, we might align a mesh with discontinuities in permeability, in an image representing a reservoir model [3]. In all cases, we assume that images are imperfect, and that interactive manipulation of meshes must be feasible.

In this paper, we demonstrate atomic meshing using seismic images, in which the relevant features correspond to geologic faults and horizons. We describe the meshing process in detail sufficient for its implementation by others. We show that the same process can generate meshes of either Delaunay triangles and tetrahedra or Voronoi polygons and polyhedra for 2-D and 3-D images, respectively.

To align a mesh with an image, we perform the following sequence of steps:

1. *Process the image to detect and normalize features of interest.*
2. *Fill space spanned by the image with a pseudo-random lattice of atoms.*
3. *Move the atoms to minimize a potential energy function of the lattice.*
4. *Connect the atoms to create either a Delaunay or Voronoi mesh.*

Image processing

We begin by processing an image to detect and normalize features with which to align a mesh. Typically, those features are either edges or ridges, and techniques for such processing are well known. In the example of Figure 1a, the faults appear as (white) ridges, so we describe our image processing for this type of feature.

We first compute Gaussian derivatives [4] of the image to obtain, for each image sample, the two components of its gradient. In the example of Figure 1a, we chose a Gaussian filter of width $\sigma = 3$ samples. We then compute outer products of those derivatives, and smooth those products with a Gaussian filter (here with width $\sigma = 12$ samples) to obtain a 2×2 symmetric and positive-definite matrix. For each image sample, the eigenvector corresponding to the largest eigenvalue of that matrix is normal to linear features (the faults) in the image [5]. We use the direction of this eigenvector, again for each image sample, to compute a directional derivative. Zero crossings of that derivative, for samples where image sample values exceed a computed threshold, correspond to ridges that we mark with sample value 1. This process yields a normalized binary image, with sample values equal to 0 or 1.

Sample values equal to 1 in the binary image correspond to the ridges in the image of Figure 1a. To facilitate lattice optimization, as described below, we negate and smooth this normalized image with another Gaussian filter (with, in this example, width $\sigma = 1.5$ samples). The image in the background of Figure 1b is the final result of our processing. Most sample values in this processed image are zero; the dark features correspond to samples with negative values. Thus, the effect of our processing has been to detect significant ridges of various heights and and convert them into valleys with constant depth and width. As discussed below, we typically choose the valley width to be 1/4 the desired minimum distance between atoms.

Lattice initialization

Initially, we distribute atoms pseudo-randomly in the space spanned by the image, consistent with a specified density defined in that space. That density may be constant, but to enhance lattice optimization, it should vary with the density of features in the image.

Let the vector $\mathbf{x}$ denote image sample coordinates. For a 2-D image, $\mathbf{x}$ has two components x and y. For a 3-D image, $\mathbf{x}$ has three components x, y, and z. Then, let $b(\mathbf{x})$ denote the normalized image, a function of sample coordinates $\mathbf{x}$. For reasons described below, we want the nominal distance $d(\mathbf{x})$ between a pair of atoms near $\mathbf{x}$ to satisfy two conditions:

$$d(\mathbf{x}) = \frac{c}{\sqrt{|\boldsymbol{\nabla}^2 b(\mathbf{x})|}} \qquad \text{and} \qquad |\boldsymbol{\nabla} d(\mathbf{x})| \ll 1, \tag{1}$$

where c is a constant.

The first condition implies that the minimum value of $d(\mathbf{x})$ is inversely proportional to the maximum value of $\sqrt{|\boldsymbol{\nabla}^2 b(\mathbf{x})|}$. That maximum value, in turn, is inversely proportional to the width σ of the final Gaussian smoothing filter that we apply during image processing. Typically, we choose the constant c so that the minimum distance $d_{min} \approx 4\sigma$.

The second condition ensures a smoothly graded lattice and mesh. In practice, we first compute $d(\mathbf{x})$ using the first condition and then smooth it to satisfy the second condition. When smoothing $d(\mathbf{x})$, we take care to preserve the minimum distance d_{min}, which occurs near features detected during image processing.

Recall that, in the final smoothing for Figure 1b, we used $\sigma = 1.5$ samples, which implies a minimum distance of $d_{min} \approx 6$ samples. For clarity in this figure, we increased the sizes of mesh elements by specifying a larger, constant nominal distance $d(\mathbf{x}) = 9$ samples. We illustrate the use of non-constant $d(\mathbf{x})$ in examples below.

Given the nominal distance function $d(\mathbf{x})$, we create a pseudo-random initial distribution of atoms. We favor a pseudo-random distribution, because it is isotropic. Any orientation of the lattice should be induced by image features, not by the initial distribution of atoms.

We create this initial distribution of atoms by first placing one atom at each corner of the image and one atom in a pseudo-random location far away from those corners. We then compute the Delaunay triangulation (e.g., [6]) of these atom locations. This creates a small number of triangles (in 2-D, or tetrahedra, in 3-D). The circumcenters of these triangles (tetrahedra) are prospective locations for new atoms. For each triangle (tetrahedron), we compute the ratio r/d of the radius r of its circumcircle (circumsphere) to the nominal distance function $d = d(\mathbf{x})$ evaluated at its circumcenter.

We then add these triangles (tetrahedra) to a priority queue. We remove the triangle (tetrahedron) with largest r/d from the queue, add an atom at its circumcenter, and update the Delaunay triangulation. This update creates new triangles (tetrahedra). It also destroys some old ones, which we ignore as we process the queue. As we create each triangle (tetrahedron), we compute its ratio r/d. If r/d is larger than a constant value (0.760 in 2-D, 0.803 in 3-D), we add it to the queue. We continue in this way until the queue is empty.

This initialization scheme is an adaptation of methods developed for Delaunay mesh refinement [7][8]. The resulting initial lattice of atoms is isotropic and consistent with the nominal distance function $d(\mathbf{x})$. Our next step is to align the atoms with features in the image $b(\mathbf{x})$, while maintaining their nominal distances $d(\mathbf{x})$.

Lattice optimization

Our goal in lattice optimization is a balance between the regularity of the lattice and its alignment with image features. We achieve this balance by minimizing a *total potential energy* P of the lattice, which we define (as in [9]) by the weighted sum

$$P = P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \equiv (1-\beta)A + \beta B, \tag{2}$$

where A is the *atomic potential energy* defined by

$$A = A(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \equiv \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \phi\left[\frac{|\mathbf{x}_i - \mathbf{x}_j|}{d(\mathbf{x}_j)}\right], \tag{3}$$

and B is the *image potential energy* defined by

$$B = B(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \equiv \sum_{i=1}^{n} b(\mathbf{x}_i). \tag{4}$$

In all of these definitions, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are the coordinates of n atoms in the lattice.

The weight β in equation (2) determines the relative contributions of A and B to the total potential energy P. When $\beta = 0$, atoms tend toward a regular (face-centered cubic) lattice that is not aligned with the image. When $\beta = 1$, atoms are sensitive to only image sample values; they will congregate in the minima and vacate the maxima in the image, yielding a highly irregular lattice. Provided that the conditions in equations (1) are satisfied, choosing $\beta \approx \frac{1}{2}$ yields a lattice that is both regular and aligned with image features.

We define the atomic potential energy A in equation (3) in terms of a potential function $\phi(u)$, where $u \equiv |\mathbf{x}_i - \mathbf{x}_j|/d(\mathbf{x}_j)$ is the normalized distance between two atoms. Suitable potential functions $\phi(u)$ have been developed by others working in different contexts [2][10]. To ensure that the optimized lattice remains consistent with the nominal distance function, we choose $\phi(u)$ to be large for $u \ll 1$ and to have a minimum at $u = 1$. To reduce the cost of computing A, we limit the range of interactions among atoms by choosing $\phi(u)$ to be zero for $u > \frac{3}{2}$. The function $\phi(u)$ defined by Shimada [2],

$$\phi(u) \equiv \begin{cases} \frac{153}{256} - \frac{9}{8}u + \frac{19}{24}u^3 - \frac{5}{16}u^4, & 0 \le u < \frac{3}{2}, \\ 0, & \frac{3}{2} \le u, \end{cases}$$

has these properties. Recalling the condition $|\boldsymbol{\nabla} d(\mathbf{x})| \ll 1$, the factor 1/2 in our definition of atomic potential energy A compensates for the appearance of $\phi[|\mathbf{x}_i - \mathbf{x}_j|/d(\mathbf{x}_j)] \approx \phi[|\mathbf{x}_j - \mathbf{x}_i|/d(\mathbf{x}_i)]$ twice in that definition.

The image potential energy B in equation (4) is relatively easy to evaluate. For each atom location $\mathbf{x}_i$, we simply look up (or interpolate) the image sample value at that location. The sum of these values is B.

Given our definition of lattice potential energy, we use a generic optimization algorithm to search efficiently for atom locations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ that minimize that energy. Specifically, we use a limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) minimizer. (See [11]. Note, particularly, the simple two-loop recursion on page 17.) Like other minimizers, the L-BFGS method iteratively evaluates a function and its partial derivatives, in its search for a minimum.

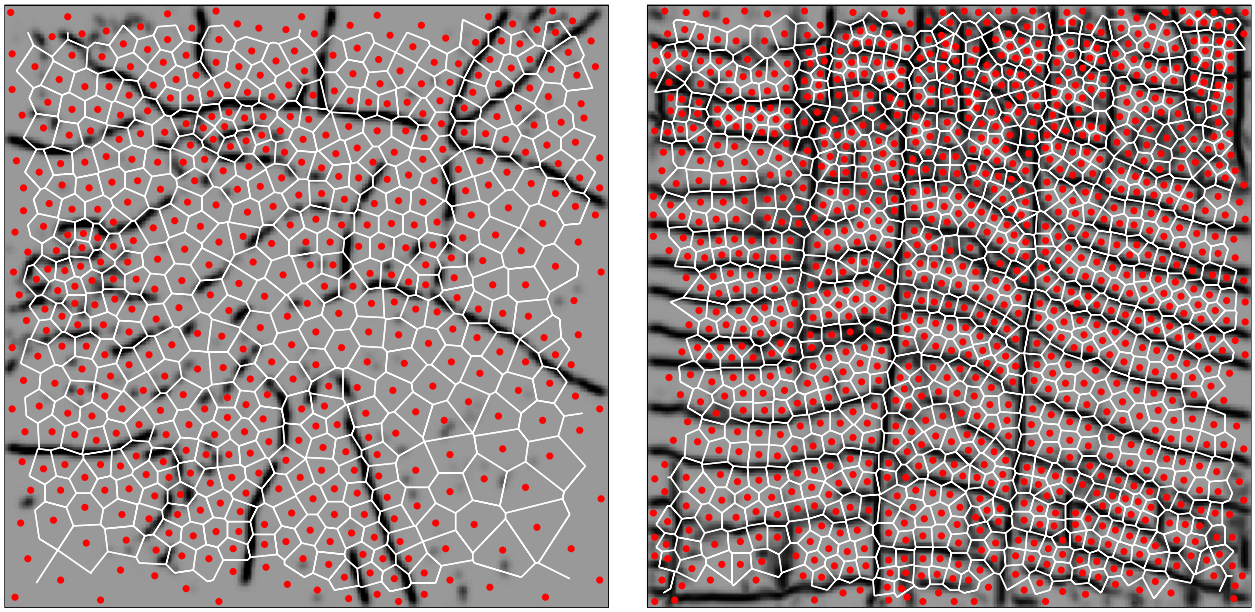
The L-BFGS minimizer requires more computer memory but fewer function evaluations than the well-known method of conjugate gradients. However, the additional memory required is insignificant when compared with the memory required to represent an image. Furthermore, the cost of each function evaluation (computing the lattice total potential energy P) is significantly more costly than the other computations performed by the minimizer. Therefore, the L-BFGS minimizer is well-suited to lattice optimization.

Note that we need not find the global minimum of total potential energy P. After satisfying the conditions in equations (1), a local minimum, with atoms close to their initial locations, yields a lattice that is well aligned with image features.

Delaunay or Voronoi meshes

Our description of lattice optimization above implicitly assumes that atoms are to be aligned *on* image features. During image processing, we create normalized valleys (potential minima) corresponding to those features. Those valleys attract atoms; minimizing the total potential energy causes atoms to move into them. Delaunay triangulation of the optimized lattice then creates triangles (in 2-D, tetrahedra in 3-D) with edges (faces) that tend to be aligned with image features, as illustrated in Figure 1.

Alternatively, we may align atoms *alongside* image features, by simply changing the sign of the normalized image $b(\mathbf{x})$. This change causes valleys to become ridges that repel atoms. Inter-atomic forces keep atoms from moving too far away, and thereby maintain lattice regularity. We illustrate this alternative in Figure 2. Here, we display Voronoi polygons, with edges well aligned with image features. These polygons correspond to Delaunay triangles not shown. In the context of reservoir simulation, Voronoi meshes like these are also known as *PEBI grids* [12].



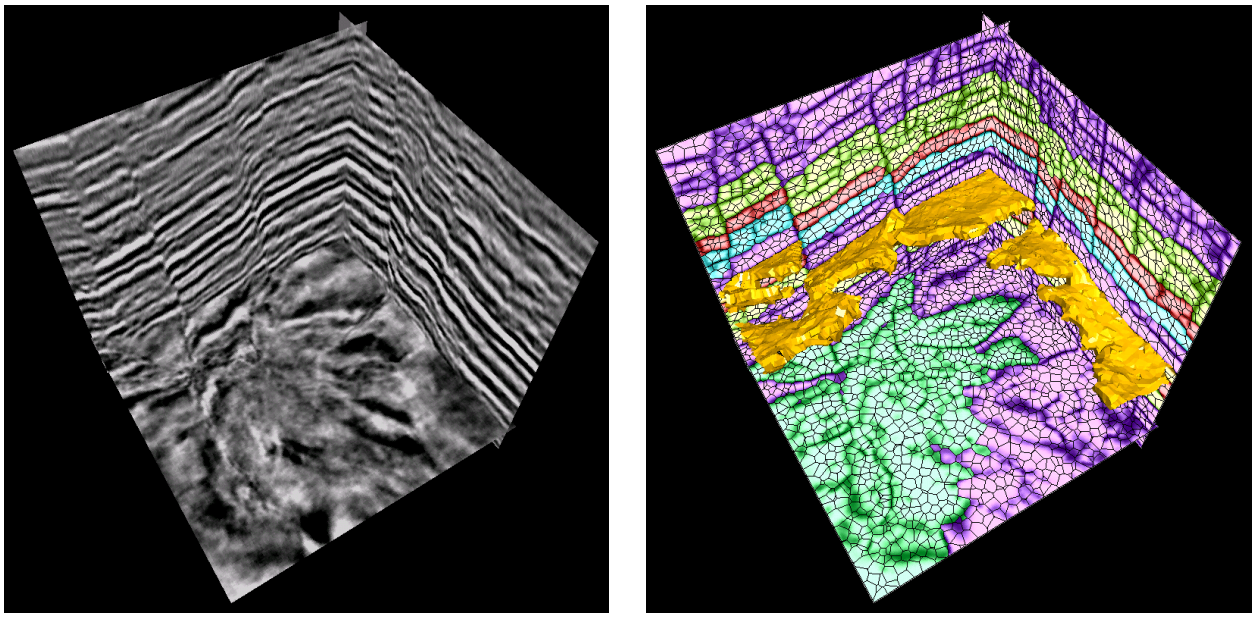
(a) (b)

Figure 2: Voronoi polygonal meshes aligned with (a) the faults imaged in Figure 1a and (b) a combination of both horizons and faults in vertical (256×256-sample, 6.4×0.8-km) slices extracted from two 3-D seismic images.

Figure 3 shows an example of a 3-D Voronoi polyhedral mesh. After meshing, we interpreted the 3-D seismic image by interactively painting polyhedra, so that atoms within the same interpreted unit have the same color. (The interpretation is best viewed in the full-color electronic version of this abstract.) We then selected one unit painted gold and computed the surface surrounding all atoms with that color. Parts of that surface are shown in Figure 3b. Large displacements have caused this unit to become discontiguous across several fault blocks.

Either type of mesh — Delaunay or Voronoi — is capable of representing discontinuities in imaged properties. In practice, however, we find Voronoi meshes, such as those in Figures 2 and 3, preferable, for several reasons.

First, Voronoi polygons vary smoothly with small perturbations of atom locations, whereas Delaunay triangles do not. In a Delaunay triangulation, a slight change in atom location may cause existing triangles to be destroyed and new triangles to be created. Second, properties that we associate with Voronoi polygons (e.g., permeability) are implicitly associated with atoms, and those properties remain well-defined as we move atoms. In contrast, properties associated with Delaunay triangles may become undefined as triangles are destroyed and created. Third, in a 3-D atomic mesh, there are about six times as many Delaunay tetrahedra as there are Voronoi polyhedra. Finally, 3-D Voronoi polyhedra tend to be well shaped, even as some 3-D Delaunay tetrahedra (called *slivers* [6] [13]) tend to be poorly shaped, with large surface area but small volume.



(a) (b)

Figure 3: A Voronoi polyhedral mesh of a 3-D seismic image. Three orthogonal slices (a) of the $256 \times 256 \times 256$-sample, ($6.4 \times 6.4 \times 0.8$-km) image illustrate significant faulting. We automatically aligned the 3-D mesh (b) with both faults and horizons in that image, and then interpreted geologic layers by interactively painting polyhedra. (See the full-color electronic version.)

Some of the problems associated with Delaunay tetrahedral meshes may be solved by using an alternative 3-D triangulation. Other triangulations are feasible, but their use leads to other problems, particularly in the context of interactive editing.

In all of the examples shown in this paper, we aligned meshes with image features automatically, without manual editing. However, in regions that are poorly imaged, we may need to interactively add, remove, or move atoms. As we do so, we reoptimize the lattice and update mesh elements for only those atoms near the atom being edited. We restrict reoptimization to a small region by giving each atom a temperature t between 0 and 1. An atom's temperature scales the forces applied to it during lattice reoptimization. Atoms nearest to the atom being manipulated have temperature $t = 1$; they move freely. Temperature decreases to $t = 0$ for atoms further (say, 5 $d(\mathbf{x})$) away; atoms not in the neighborhood of the atom being edited are frozen and cannot move.

Conclusion

Using well-known image processing and optimization techniques, we may automatically align a lattice of points on or alongside features in 3-D images. After connecting those points to form Delaunay tetrahedra or Voronoi polyhedra, we obtain space-filling meshes with edges and faces aligned with image features. Those meshes, in turn, facilitate other tasks, such as seismic interpretation and flow simulation.

Integration of multiple tasks is the primary goal of this research. In fact, this work began with the question, "what if, during seismic interpretation, fault framework building, reservoir modeling, all that stuff, we were forced to use a data structure that would work for flow simulation?" Our answer in this paper is incomplete, particularly with respect to reservoir property modeling. But part of the answer is illustrated in Figure 3, where our use of a

space-filling polyhedral mesh has enhanced seismic interpretation, by enabling interactive 3-D painting of geologic layers.

Furthermore, though beyond the scope of this paper, computations like those used in flow simulation enable us to perform much of this painting automatically. (Think of paint flowing within geologic layers, but not across faults.) The resulting segmentation of the mesh implies a domain decomposition that may, in turn, enable more efficient flow computations. Such cross-fertilization follows from our use of a common data structure.

References

[1] J. T. Oden, I. Babuska, and C. E. Baumann, "A Discontinuous hp Finite Element Method for Diffusion Problems", *Journal of Computational Physics*, vol. 146, pp. 491–519 (1998).

[2] K. Shimada, "Physically-Based Mesh Generation: Automated Triangulation of Surfaces and Volumes via Bubble Packing", *Ph.D. thesis*, Massachusetts Institute of Technology (1993).

[3] M. H. Garcia, A. G. Journel, and K. Aziz, "Automatic Grid Generation for Modeling Reservoir Heterogeneities" *SPE Reservoir Engineering*, pp. 278–284 (1992).

[4] I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter", *Signal Processing*, vol. 44, no. 2, pp. 139-151 (1995).

[5] L. J. van Vliet and P. W. Verbeek, "Estimators for orientation and anisotropy in digitized images", *ASCI'95, Proc. First Annual Conf. of the Advanced School for Computing and Imaging*, pp. 442–450 (1995).

[6] M. Bern, and D. Eppstein, "Mesh Generation and Optimal Triangulation", in *Computing in Euclidean Geometry*, D.-Z. Du and F.K. Hwang, eds., World Scientific (1995).

[7] J. R. Shewchuk, "Tetrahedral Mesh Generation by Delaunay Refinement", *Proceedings of the 14th Annual Symposium on Computational Geometry*, pp. 86–95 (1998).

[8] J. R. Shewchuk, "Delaunay Refinement Algorithms for Triangular Mesh Generation", *Computational Geometry: Theory and Applications*, vol. 22, no. 1–3, pp. 86–95 (2002).

[9] D. Hale, "Atomic images - a method for meshing digital images", *Proceedings of the 10th International Meshing Roundtable*, pp. 185–196 (2001).

[10] D. Tonnesen, "Dynamically Couples Particle Systems for Geometric Modeling, Reconstruction, and Animation", *Ph.D. thesis*, University of Toronto (1998).

[11] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of Quasi-Newton Matrices and Their Use in Limited Memory Methods", *Technical Report NAM-03*, Northwestern University, Department of Electrical Engineering and Computer Science, (1996).

[12] Z. E. Heinemann, C. W. Brand, M. Munka, and Y. M. Chen, "Modeling Reservoir Geometry With Irregular Grids" *SPE Reservoir Engineering*, pp. 225–232 (1991).

[13] J. C. Cavendish, D. A. Field, and W. H. Frey, "An Approach to Automatic Three-Dimensional Finite Element Mesh Generation", *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 329–347 (1985).