ATOMIC IMAGES - A METHOD FOR MESHING DIGITAL IMAGES

Dave Hale

Landmark Graphics, Englewood, CO, USA   dhale@lgc.com

ABSTRACT

By combining a digital image with a lattice of points called atoms, in which atom coordinates are computed to minimize a potential energy function of the combination, we obtain a mesh suitable for further computations, such as flow simulation.

Each atom in the lattice contributes a potential function to an atomic potential field. The image represents another potential field. Total potential energy of the lattice is a weighted sum of the atomic and image potential fields, evaluated at atom coordinates. We exploit the uniform sampling of images to obtain a simple and efficient algorithm for computing the total potential energy.

Beginning with a pseudo-regular lattice, a generic optimization algorithm moves atoms to minimize this total potential energy. After lattice optimization (not before), we connect the atoms to form a mesh that tends to be aligned with image features.

Keywords: mesh generation, image processing

1 INTRODUCTION

Digital images are often analyzed to obtain meshes that facilitate further computation. For example, medical images of the human brain are analyzed to obtain meshes used to simulate blood flow in arteries [1]. Similarly, seismic images are analyzed to obtain geologic meshes used to simulate fluids flowing in subsurface reservoirs [2].

Computations such as flow simulations are typically performed on meshes that are not the uniform grids on which digital images are sampled and processed. For example, to reduce computation costs, a reservoir simulation mesh is typically sampled more coarsely, especially along horizontal dimensions, and its sampling may not be uniform. To better conform to subsurface geology, the reservoir simulation mesh may be unstructured — connections among mesh elements may be explicit, and not simply implied by array indices. A frequent goal of image analysis is to obtain such an irregularly sampled, unstructured mesh.

1.1 Images and meshing today

Today, image analyses often consist of the following sequence of steps [1][2]:

(1) Process an image to enhance features of interest.

(2) Find curves or surfaces in the image that bound regions of interest.

(3) Fill the space defined by those regions with a computational mesh.

(4) Simulate some process on the space-filling mesh.

For various reasons, each step in this sequence is today often taken independently, with little regard for the requirements of subsequent steps. For example, it is common in step (2) to produce a bounding curve or surface with more detail than can be represented in the space-filling mesh used in step (4). Seismic reflections representing geologic interfaces are routinely mapped with higher lateral resolution than can practically be used in meshes for fluid reservoir simulation.

This leads to the "scale up" or "upscaling" problem cited in [2].

This discrepancy in resolution is often accompanied by a discrepancy in data structure. For example, a two-dimensional (2-D) curve represented by a simple linked list of line segments in step (2) may become a relatively complex mesh of triangles in step (3). Such discrepancies today disrupt the analysis sequence, and may yield inconsistencies between the image processed in step (1) and the mesh used in step (4) that are difficult to quantify.

The disruptions are costly. In the example of seismic image analysis and reservoir simulation, one iteration of this sequence today may require a month or more of work. This high cost makes it difficult to perform multiple iterations in attempts to estimate uncertainties in simulation results.

1.2 Meshing with lattices and forces

The accuracy of most computations performed on meshes depends on the *regularity* of mesh elements. Simulations performed on highly regular triangular meshes, those with nearly equilateral triangles, are more accurate than those performed on irregular meshes with long thin triangles.

This is one reason that the *Delaunay triangulation* (e.g., [3]) is popular. Given a set of points representing the locations of nodes for a 2-D mesh, the 2-D Delaunay triangulation of those points, when compared with all other possible triangulations, yields triangles most nearly equilateral. However, Delaunay triangulation alone does not guarantee a regular mesh. For that, one must choose carefully the locations of the mesh nodes.

Outside the domain of image analysis, the problem of choosing optimal mesh node locations has been well studied. One solution to this problem is based on the observation [4][5] that the triangulation of a set of points defined by a simple crystal lattice of atoms yields a highly regular mesh. However, a uniform lattice of points can seldom be aligned exactly with the boundaries of objects to be meshed.

Therefore, some solutions (e.g., [4]) begin with an approximately uniform lattice, and then use numerical models of physical forces between atoms to automatically move mesh nodes (atoms) to more optimal locations. This method has been used to mesh geometric models that define precisely internal and external boundaries of objects. Here, we extend this method to the problem of computing meshes aligned with less well-defined features in digital images.

Models of physical forces are widely used to compute optimal locations for points that define curves and surfaces. For example, Witkin and Heckbert [6] use repulsive forces to sample and interactively manipulate free-form surfaces described by implicit functions. Closer to our context of image processing, Terzopoulis and others (e.g., [7][8]) find bounding curves and surfaces using active contours (also known as "snakes"), where initially simple curves or surfaces are deformed by artificial forces induced by image features. Here, we use models of physical forces to directly construct space-filling lattices (and meshes) aligned with image features, without first computing bounding curves or surfaces.

2 MESHING IMAGES

The method we propose replaces the image analysis sequence described above with the following sequence:

(1) Process an image to enhance features of interest.

(2) *Fill space with a computational mesh aligned with image features.*

(3) Simulate some process on the space-filling mesh.

Instead of finding boundaries of regions within images and then meshing those regions, one simply constructs a mesh that is aligned with the boundaries.

Figure 1 illustrates a seismic image of faults, discontinuities in subsurface geology. This 256 × 256-sample (6.4 × 6.4-km) 2-D image is a horizontal slice taken from a 3-D seismic image that was processed to enhance such discontinuities. The dark linear features in this image represent traces of faults that intersect this horizontal slice. The faults are approximately vertical, almost perpendicular to this horizontal slice.

Figure 2 shows a space-filling mesh that has been aligned with those faults. This mesh is highly regular, and its variable density smoothly conforms to the variable density of features in the image. Such a mesh could well be used for further computation, such as reservoir simulation.

The space-filling mesh shown in Figure 2 was computed in three steps:

(1) Fill the space spanned by the image with a pseudo-regular lattice of atoms. By "pseudo-regular", we mean that the nominal distance between an atom and its nearest neighbors varies smoothly, consistent with the density of features in the image.

(2) Move the atoms to minimize a total potential energy, defined to be a weighted sum of an atomic potential energy and an image potential energy.

(3) Connect the atoms using Delaunay (or some other) triangulation to form a mesh.

Note that this process constructs a mesh for the first time in step (3), only after lattice optimization in step (2). In the examples shown in this paper, we construct meshes using, or at least beginning with, a Delaunay triangulation of atom locations. Because Delaunay triangulation connects atoms with their nearest neighbors, and because, after step (2), atoms located on image features tend to be closer together than

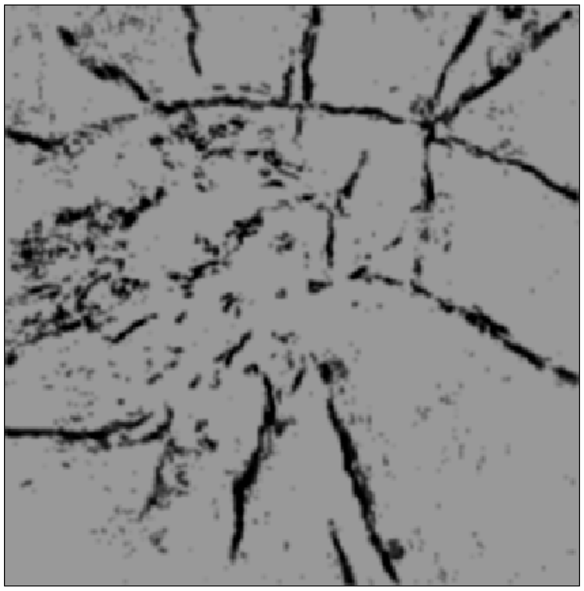


Figure 1: A seismic image of geologic faults.

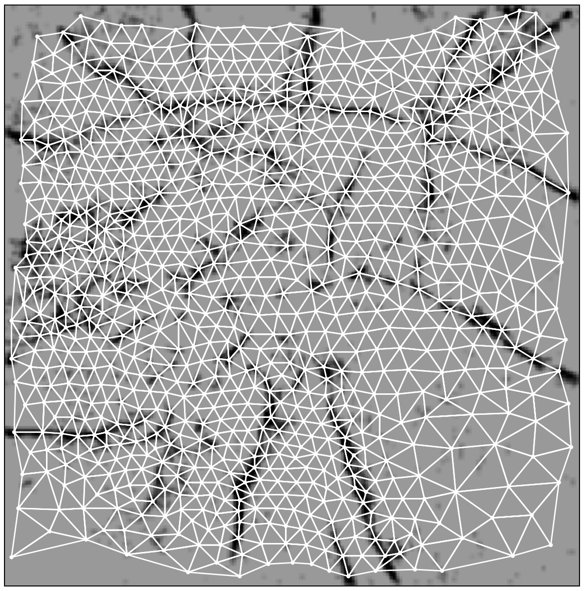


Figure 2: A space-filling variable-density mesh that has been automatically aligned with features in the seismic image.

atoms located elsewhere, Delaunay triangulation tends to create edges (in 2-D) and faces (in 3-D) aligned with image features, as illustrated in Figure 2. However, nothing in this process guarantees this alignment, and non-Delaunay triangulations, perhaps obtained by swapping edges or faces, may improve alignment of the mesh with image features.

Step (2) of this process is the most costly, as it requires repeated computations of the total potential energy and its partial derivatives with respect to the spatial coordinates of each atom. In the next section, we describe a simple and efficient algorithm for performing these computations.

2.1 Computing the potential energy

An atom in a two-dimensional (2-D) space has x and y coordinates, and an atom in a three-dimensional (3-D) space has x, y, and z coordinates. The vector $\mathbf{x}$ denotes the x and y (or x, y and z) spatial coordinates of a point in 2-D (or 3-D) space. Given two atoms with locations $\mathbf{x}_i$ and $\mathbf{x}_j$, $|\mathbf{x}_i - \mathbf{x}_j|$ denotes the Euclidean distance between them.

2.1.1 Pair-wise potential functions

For computational efficiency, we model the interaction among atoms with a simple pair-wise force function, so that the total force exerted on an atom by its neighbors is simply the sum of the forces exerted by each one of them. Even with this simplification, there exist many reasonable choices for the pair-wise force function.

To avoid having two or more atoms with the same, or nearly the same, coordinates, the force between them should be repulsive (positive) if they are too close to each other. Likewise, to prevent large empty spaces with no atoms, the force between two atoms should be attractive (negative) if they are too far away from each other. To facilitate numerical computations, the force should be bounded. To prevent every atom in the lattice from exerting a force on every other atom, the force should be zero beyond a cutoff distance. Furthermore, the force function should be continuous as a function of the inter-atomic distance. We use the force function proposed by Shimada [4], which has these properties.

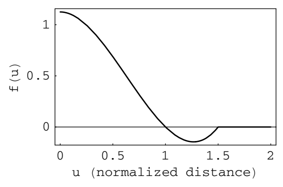
Let d denote the *nominal distance* between two atoms, the distance at which the force goes from being repulsive to being attractive. Then, the force f between two atoms located at $\mathbf{x}_i$ and $\mathbf{x}_j$ may be given by the cubic polynomial:

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

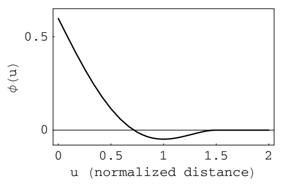
where u is the *normalized distance* between the two atoms defined by

$$u \equiv \frac{|\mathbf{x}_i - \mathbf{x}_j|}{d}.$$

The coefficients of this polynomial function ensure that the force is bounded and continuous, that it equals zero for $u = 1$ and $u \geq \frac{3}{2}$, and that it is positive for $0 \leq u < 1$ and negative for $1 < u < \frac{3}{2}$. Figure 3a illustrates this force function.



(a)



(b)

Figure 3: Pair-wise (a) force and (b) potential functions of normalized distance between any two atoms.

Generally, the force on an atom is a vector. Here, the direction of this vector is implied by the sign of $f(u)$, and by the locations $\mathbf{x}_i$ and $\mathbf{x}_j$ of the two atoms.

It is often more convenient to work with a scalar potential than with the multiple components of a vector force. Therefore, following a well-known convention, we define the force to be the negative of the gradient of a scalar potential:

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

The constant of integration $\frac{153}{256}$ has been chosen so that $\phi(u)$ is continuous at $u = \frac{3}{2}$. Figure 3b illustrates this potential function. As expected, the potential function $\phi(u)$ has a minimum at normalized distance $u = 1$, where the force function $f(u)$ is zero.

2.1.2 Potential energies and fields

Given a potential function $\phi(u)$ of normalized distance u, we define the *atomic potential energy* to be the following sum of pair-wise potentials:

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

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are the coordinates of n atoms in a lattice, and $d(\mathbf{x})$ is the nominal inter-atomic distance function of position $\mathbf{x}$. The nominal distance function $d(\mathbf{x})$ need not be constant; but, to ensure a smoothly graded lattice, we require that it be smooth. Specifically, we require that $|\boldsymbol{\nabla} d| \ll 1$, so that $d(\mathbf{x}_i) \approx d(\mathbf{x}_j)$ for $|\mathbf{x}_i - \mathbf{x}_j|/d$ less than the cutoff distance $\frac{3}{2}$ of the potential function $\phi(u)$. Then, the factor 1/2 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 the definition of the total potential energy A.

We may also define the atomic potential energy A in terms of an *atomic potential field*:

$$a(\mathbf{x}) \equiv \sum_{j=1}^{n} \phi \left[\frac{|\mathbf{x} - \mathbf{x}_j|}{d(\mathbf{x}_j)} \right], \tag{2}$$

so that

$$A = A(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \equiv \frac{1}{2} \sum_{i=1}^{n} a(\mathbf{x}_i).$$

In other words, the atomic potential energy is defined to be half the sum of values obtained by evaluating the atomic potential field at the atom coordinates.

Likewise, we define an *image potential energy*

$$B = B(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \equiv \sum_{i=1}^{n} b(\mathbf{x}_i), \tag{3}$$

where $b(\mathbf{x})$ is an *image potential field.*

In other contexts, an image potential field is simply an image (or a smoothed version of an image), typically represented by a 2-D (or 3-D) array of numbers stored in a computer memory. Here, we use the term "potential field" to emphasize (and later exploit) the similarity between the atomic and image potential fields.

To align atoms with features of interest in the image (as in Figure 2), we first process the image so that the image potential field attains a minimum value $b(\mathbf{x}) \approx -1$ within features of interest, and a maximum value $b(\mathbf{x}) \approx 0$ in uninteresting regions. Then, minimizing the image potential energy B is equivalent to moving atoms into minima corresponding to features in the image. Finally, connecting the atoms with Delaunay triangulation yields a mesh of triangles (or, in 3-D, tetrahedra) with edges (or faces) that tend to be aligned with image features.

We move atoms to minimize the following weighted sum of the atomic and image potential energies:

$$P = P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \equiv (1 - \beta)A + \beta B, \tag{4}$$

which we call the *total potential energy.* The scale factor β determines the relative contributions of A and B to the total potential energy P. When $\beta = 0$, atoms tend toward a perfectly regular 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. Typically, we choose $\beta \approx \frac{1}{2}$, and obtain an approximately regular lattice that respects image features.

In terms of the potential fields $a(\mathbf{x})$ and $b(\mathbf{x})$, the total potential energy is

$$P = \sum_{i=1}^{n} \frac{1}{2}(1 - \beta)a(\mathbf{x}_i) + \beta b(\mathbf{x}_i).$$

In terms of the *total potential field*, defined as

$$p(\mathbf{x}) \equiv (1 - \beta)a(\mathbf{x}) + \beta b(\mathbf{x}), \tag{5}$$

the total potential energy is

$$P = \frac{1}{2} \sum_{i=1}^{n} p(\mathbf{x}_i) + \beta b(\mathbf{x}_i). \tag{6}$$

Like the nominal distance function $d(\mathbf{x})$, the scale factor β in equations (5) and (6) may be a smoothly varying function of position $\mathbf{x}$. (As for $d(\mathbf{x})$, smoothness implies that derivatives of $\beta(\mathbf{x})$ are negligible.) This generalization enables the balance between lattice regularity and sensitivity to (attraction to or repulsion from) image features to vary spatially. Sensitivity of the lattice to image features may be more important in one part of the image than in some other part. For simplicity, here, we let β denote a constant scale factor.

The total potential energy P is a non-quadratic function of the atom coordinates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, with many local minima. (For example, note that interchanging the coordinates of any two atoms does not change P.) Therefore, any search for a minimum, usually one close to the initial lattice coordinates, must be iterative. In an efficient iterative search, we must evaluate repeatedly partial derivatives of P with respect to the atom coordinates. Consider, for example, the change in P with respect to the x-coordinate of the i'th atom:

$$\frac{\partial P}{\partial x_i} = (1-\beta)\frac{\partial A}{\partial x_i} + \beta\frac{\partial B}{\partial x_i}. \tag{7}$$

In evaluating the partial derivative $\partial A/\partial x_i$ of the atomic potential energy, we recall that the term $\phi[|\mathbf{x}_i - \mathbf{x}_j|/d(\mathbf{x}_j)] \approx \phi[|\mathbf{x}_j - \mathbf{x}_i|/d(\mathbf{x}_i)]$ appears twice in the double sum of equation (1). Therefore,

$$\begin{aligned}
\frac{\partial A}{\partial x_i} &= \sum_{j=1}^{n} \phi'\left[\frac{|\mathbf{x}_i - \mathbf{x}_j|}{d(\mathbf{x}_j)}\right] \frac{1}{d(\mathbf{x}_j)} \frac{x_i - x_j}{|\mathbf{x}_i - \mathbf{x}_j|} && (8) \\
&= \frac{\partial a}{\partial x}(\mathbf{x}_i) \\
\frac{\partial B}{\partial x_i} &= \frac{\partial b}{\partial x}(\mathbf{x}_i) && (9) \\
\frac{\partial P}{\partial x_i} &= \frac{\partial p}{\partial x}(\mathbf{x}_i). && (10)
\end{aligned}$$

Similar results may be obtained easily for partial derivatives with respect to the y (and, in 3-D, the z) coordinate of each atom.

2.1.3 A simple algorithm

Computation of the total potential energy P requires the computation of its components A and B. To compute the image potential energy B according to equation (3), we must evaluate the image potential field $b(\mathbf{x})$ for each atom location $\mathbf{x} = \mathbf{x}_i$. Images are typically sampled uniformly, and the simplest and most efficient approximation to $b(\mathbf{x}_i)$ is the value of the image potential field $b(\mathbf{x})$ at the image sample nearest to the point $\mathbf{x}_i$. More accurate approximations (interpolations) are possible, but we used this simple and fast nearest-neighbor interpolation in all of the examples shown here. Equation (3) implies that the cost (the computational complexity) of computing B is $O(n)$, where n is the number of atoms.

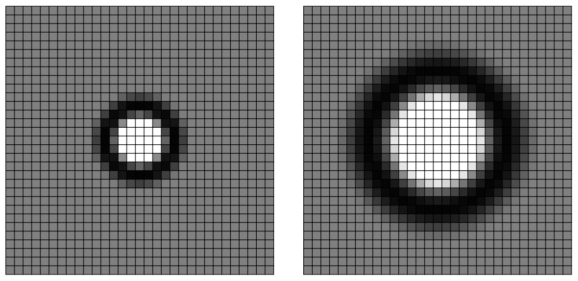
In contrast, the double sum in equation (1) implies that the cost of the most straightforward method for computing A is $O(n^2)$. In practical applications, n is large enough that an $O(n^2)$ cost for computing A would be much greater than the $O(n)$ cost of evaluating B. To reduce the cost of computing A, we may exploit our design of the potential function $\phi(u)$, which is zero for normalized distances u greater than the cutoff distance $\frac{3}{2}$. Only the atoms nearest to an atom located at position $\mathbf{x} = \mathbf{x}_i$ contribute to the atomic potential field $a(\mathbf{x}_i)$ at that position.

This observation leads to the problem of determining which atom neighbors are within a distance $\frac{3}{2}d(\mathbf{x}_i)$ of an atom located at $\mathbf{x} = \mathbf{x}_i$. Solution of this problem is non-trivial, because atoms move repeatedly during optimization of the lattice.

For example, if we construct lists of neighboring atoms, one list for each atom, we must update these lists (or at least check to see whether they require updating) whenever atoms move. For lattices with near constant density, the cost of constructing and updating such lists using simple data structures is $O(n)$. For variable-density lattices, more complex data structures are required and the cost becomes $O(n \log n)$ (e.g., [9]).

Our expression of the atomic potential energy A in terms of the atomic potential field $a(\mathbf{x})$ suggests a simpler solution. We interpret equation (2) as a recipe for computing an atomic potential field $a(\mathbf{x})$ that is sampled like the image potential field $b(\mathbf{x})$. Specifically, we represent $a(\mathbf{x})$ with a 2-D (or 3-D) array having the same dimensions as the array used to represent the image $b(\mathbf{x})$. We first initialize $a(\mathbf{x})$ to zero for all $\mathbf{x}$ sampled. Then, for each atom located at position $\mathbf{x} = \mathbf{x}_j$, we accumulate a sampled potential function $\phi[|\mathbf{x} - \mathbf{x}_j|/d(\mathbf{x}_j)]$. This accumulation is limited spatially to samples inside the circle (or sphere) of radius $\frac{3}{2}d(\mathbf{x}_j)$ centered at position $\mathbf{x}_j$, where the contribution of the sampled potential function is non-zero.

Figures 4a and 4b illustrate two such potential functions, for nominal distances $d = 4$ and $d = 8$, respectively. Gray levels between black and white correspond to sampled function values between -0.05 and 0.05, respectively. The atomic potential field $a(\mathbf{x})$ is simply the accumulation of many such functions.



(a) (b)

Figure 4: The contributions, for nominal distances (a) 4 and (b) 8, of one atom to a sampled atomic potential field.

For computational efficiency, we pre-compute and tabulate sampled potential functions for different nominal distances d. Then, given $d(\mathbf{x})$ for any location $\mathbf{x}$, we determine the appropriate sampled potential function by table lookup.

Our sampling of an atomic potential field and accumulation of sampled potential functions are similar to techniques used by others in different contexts. For example, Turk and Banks [10] accumulate sampled functions (low-pass-filtered curves) to compute an energy in their algorithm for optimally placing streamlines in visual displays of vector fields.

One advantage of using sampled potential functions is that we need not determine the nearest neighbors of atoms when computing the total potential energy and its partial derivatives. In our simplest algorithm, we use pre-computed tables of potential functions and equations (2) and (5) to compute the total potential field $p(\mathbf{x})$. We then use equation (6) to compute the total potential energy, and equation (10) to compute its partial derivatives. The following pseudo-code listing describes this algorithm in detail:

Algorithm 1: compute P, $\frac{\partial P}{\partial x_i}$, $\frac{\partial P}{\partial y_i}$, and $\frac{\partial P}{\partial z_i}$

```
initialize total potential field $p(\mathbf{x}) = \beta b(\mathbf{x})$
for all atom locations $\mathbf{x}_j = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ {
  for all $\mathbf{x}$ such that $|\mathbf{x} - \mathbf{x}_j| < \frac{3}{2} d(\mathbf{x}_j)$ {
    accumulate $p(\mathbf{x}) = p(\mathbf{x}) + (1-\beta)\phi[|\mathbf{x} - \mathbf{x}_j| / d(\mathbf{x}_j)]$
  }
}
initialize total potential energy $P = 0$
for all atom locations $\mathbf{x}_i = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ {
  accumulate $P = P + \frac{1}{2}[p(\mathbf{x}_i) + \beta b(\mathbf{x}_i)]$
  compute $\frac{\partial P}{\partial x_i} = \frac{1}{2}[p(x_i+1, y_i, z_i) - p(x_i-1, y_i, z_i)]$
  compute $\frac{\partial P}{\partial y_i} = \frac{1}{2}[p(x_i, y_i+1, z_i) - p(x_i, y_i-1, z_i)]$
  compute $\frac{\partial P}{\partial z_i} = \frac{1}{2}[p(x_i, y_i, z_i+1) - p(x_i, y_i, z_i-1)]$
}
```

The first loop over atom locations accumulates the contribution of the atomic potential field $a(\mathbf{x})$ to the total potential field $p(\mathbf{x})$, sampled like the image potential field $b(\mathbf{x})$. From the latter two fields $p(\mathbf{x})$ and $b(\mathbf{x})$, the second loop over atom locations computes the total potential energy P and its partial derivatives $\frac{\partial P}{\partial x_i}$, $\frac{\partial P}{\partial y_i}$, and $\frac{\partial P}{\partial z_i}$. In this second loop, we compute the values of the total potential field and the image potential field at atom location $\mathbf{x}_i$ as described above, by selecting the corresponding field values at the nearest image sample position. Also in the second loop, we compute the partial derivatives from the total potential field using simple centered-finite-difference approximations. For definiteness, this pseudo-code assumes a 3-D coordinate space. For a 2-D space, one simply ignores the z coordinates and partial derivatives $\frac{\partial P}{\partial z_i}$.

Assuming that atom locations are consistent with the nominal distance function $d(\mathbf{x})$, the computational cost of Algorithm 1 is $O(N)$, where N is the number of samples in the image. Recall that we sample the total potential field like the image, and that each atom contributes a spatially limited potential function (like those in Figures 4) to those samples in the total potential field that lie nearest to the atom. Therefore, the cost of accumulating the contributions from all atoms is proportional to the number of samples N in the field.

The cost of this algorithm is comparable to that of conventional image processing, and it requires data structures no more complex than the simple array that represents the image.

One can easily show that, for constant nominal distance d, the number of floating point operations (additions) required to compute the total potential field is approximately $8N$ for 2-D and $20N$ for 3-D. Furthermore, this cost is roughly the same for non-constant nominal distance functions $d(\mathbf{x})$.

2.2 Lattice initialization

As noted above, the total potential energy P is a non-quadratic function of the atom coordinates, with many local minima. During lattice optimization, we move atoms iteratively in search of a minimum. In practice, we neither seek nor find the global minimum. Rather, we find an optimized lattice of atoms that is close to an initial lattice. Therefore, we seek an initial lattice that

- minimizes (locally) the atomic potential energy,
- is highly regular, and
- is consistent with the nominal distance function $d(\mathbf{x})$.

For constant nominal distance d, we can easily construct an initial lattice with these properties. For example, in 3-D, the face-centered-cubic (FCC) lattice satisfies the criteria listed above for a desirable initial lattice. For a non-constant nominal distance function $d(\mathbf{x})$, the initial arrangement of atoms is more difficult.

The first complication is that the function $d(\mathbf{x})$ must be computed, if not otherwise specified. In some applications, the nominal distance function $d(\mathbf{x})$ may be specified explicitly or constructed interactively using a computer program for painting images. The actual manner in which the function $d(\mathbf{x})$ is computed is likely to depend on the application. As stated above, we require only that this function be smooth; i.e., that $|\boldsymbol{\nabla} d| \ll 1$.

The second complication is that of arranging atoms in a lattice consistent with the nominal distance function $d(\mathbf{x})$. We use the following algorithm for initializing such a *pseudo-regular* lattice:

Algorithm 2: initialize a lattice

```
initialize an array of boolean flags $w(\mathbf{x}) = false$
construct an empty $list$ of atoms
construct an empty $queue$ of atom sites
append location $\mathbf{x}_i$ of the image center to the $queue$
while the $queue$ is not empty {
  get and remove the first site $\mathbf{x}_i$ from the $queue$
  if $\mathbf{x}_i$ lies within the coordinate bounds of the image {
    set $sphere$ = spherical region
      with center $\mathbf{x}_i$ and diameter $\gamma d(\mathbf{x}_i)$
    if, for all samples inside the $sphere$, $w(\mathbf{x}) = false$ {
      for all samples inside the $sphere$, set $w(\mathbf{x}) = true$
      append an atom with coordinates $\mathbf{x}_i$ to $list$
      append ideal sites for neighbors to end of $queue$
    }
  }
}
```

The array $w(\mathbf{x})$ is a temporary work array, with dimensions equal to those of the image. Its sole purpose is to mark locations of atoms in the lattice as they are generated by the algorithm, to ensure that locations so marked will not be marked again. This array can be the same as that used to store the total potential field $p(\mathbf{x})$ in Algorithm 1, so that no additional memory is required.

To mark the location of an atom, the algorithm sets the flags $w(\mathbf{x}) = true$ within a spherical region, centered at the atom's location, with diameter proportional to the value of the nominal distance function $d(\mathbf{x})$ at that location. The constant of proportionality is the factor γ. By choosing this factor to be less than one, we permit some atoms in the initial lattice to be closer together than the nominal distance function $d(\mathbf{x})$ implies, knowing that other atoms will be further away. We determined experimentally that the factor $\gamma = 0.8$ yields pseudo-regular lattices consistent with smooth nominal distance functions $d(\mathbf{x})$.

The *ideal sites* in Algorithm 2 are the locations of atom neighbors in a regular (e.g., FCC) lattice. The ideal distance to any neighbor of an atom located at $\mathbf{x} = \mathbf{x}_i$ is simply $d(\mathbf{x}_i)$. Therefore, for constant d, Algorithm 2 yields a regular lattice. For non-constant $d(\mathbf{x})$, it yields a pseudo-regular lattice.

In any case, the processing of ideal sites placed in the *queue* causes the lattice to grow outward from the first site placed in the *queue*. Therefore, the first site acts as a seed from which the lattice is grown. Algorithm 2 chooses that first site to be the center of the image. Alternative seed locations may be used. For example, if a mesh for simulation of fluid flow is desired, then the locations of one or more fluid sources or sinks may be used to seed the growth of a lattice.

Figure 5 illustrates a nominal distance function $d(\mathbf{x})$ and the corresponding initial pseudo-regular lattice computed with Algorithm 2. In this example, the function $d(\mathbf{x})$ is simply a smoothed, scaled, and biased version of the image shown in Figure 1. The darkest regions in this figure correspond to a minimum distance of $d = 6$ samples, and the lightest regions correspond to a maximum distance of $d = 18$ samples. We specified these minimum and maximum distances explicitly, based on the level of detail we observed in the image. Distances $d(\mathbf{x})$ are smallest in the middle-left portion of the image, and largest in the lower-right portion. The average distance is approximately 9.7 samples.

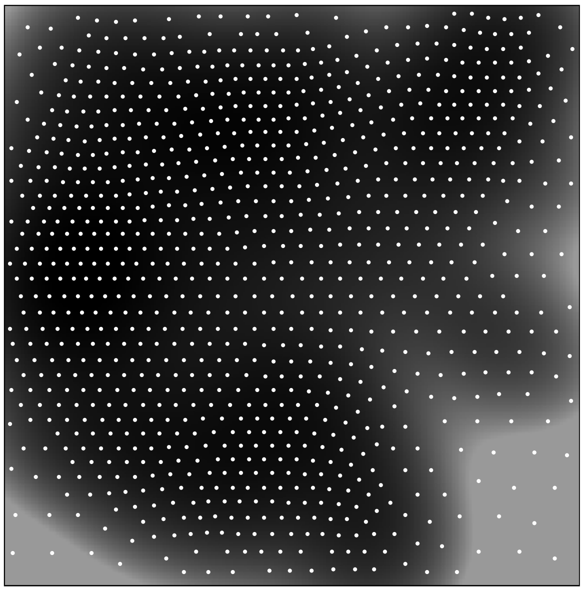


Figure 5: A pseudo-regular lattice and nominal distance function $d(\mathbf{x})$, for the image of Figure 1.

2.3 Lattice optimization

Lattice optimization moves atoms in an initial lattice to obtain a lattice that minimizes the total potential energy P. Given the value of the function P, which depends on the atom coordinates, and the values of its partial derivatives with respect to each of those coordinates, any generic function minimizer may be used to find a minimum. We use the 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) is significantly more costly than the other computations performed by the minimizer. Therefore, the L-BFGS minimizer is well-suited to our algorithm for lattice optimization:

Algorithm 3: optimize the lattice

```
get the initial lattice atom coordinates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
construct a potential energy computer
construct a minimizer
compute the initial lattice total potential energy $P$
do {
  set $P_o = P$
  randomly perturb $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
  do {
    set $P_i = P$
    let minimizer decrease $P$ by adjusting $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
  } while $P_i - P > \epsilon|P_i|$
} while $P_o - P > \epsilon|P_o|$
```

This algorithm begins with an initial lattice of atoms, such as the pseudo-regular lattice produced by Algorithm 2. It then constructs a potential energy computer (e.g., Algorithm 1) responsible for computing the total potential energy P and its partial derivatives. It then constructs a minimizer, which will use the potential energy computer to minimize P. (Construction of the potential energy computer and the minimizer includes allocation of memory and initialization of a number of variables and tables.) It then computes the total potential energy P of the initial lattice.

The remainder of the algorithm consists of two nested loops. The inner loop lets the minimizer adjust the atom coordinates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ to decrease the total potential energy P. This loop continues until the decrease in P becomes insignificant, as determined by the small threshold ϵ. A typical threshold is $\epsilon = 0.001$.

Recall that the total potential energy is a function with many local minima. The inner loop begins with the current atom coordinates, and tends toward the minimum nearest to those coordinates. We have observed that this local minimum may have a total potential energy larger than that of another minimum nearby.

The outer loop enables the algorithm to move from one local minimum to another one, until the decrease in total potential energy P is insignificant. The random perturbations of atom coordinates in Algorithm 3 are small, typically no more than 10% of the nominal distance $d(\mathbf{x}_i)$, for each atom location $\mathbf{x}_i$. We use a commonplace pseudo-random number generator to compute these perturbations. Subsequent iterations of the inner minimization loop typically yield a significant decrease in total potential energy P.

The inner and outer loops in Algorithm 3 use the same test for convergence. Both loops terminate when the decrease in total potential energy P becomes insignificant. Alternative convergence criteria are commonplace in numerical optimization, and our choices here serve only as examples. For example, one might terminate these loops when the maximum change in atom coordinates is less than some threshold.

Figures 6 and 7 show the total potential fields $p(\mathbf{x})$ and atom locations $\mathbf{x}_i$ for initial and optimized lattices, respectively. (These lattices correspond to Figures 1 and 5). They demonstrate the movement of atoms into potential valleys created by adding the image potential field $b(\mathbf{x})$ to the atomic potential field $a(\mathbf{x})$. Recall equation (5), which states that the total potential field $p(\mathbf{x})$ is a weighted sum of atomic and image potential fields. In this example, we used an image weight $\beta = 0.3$.

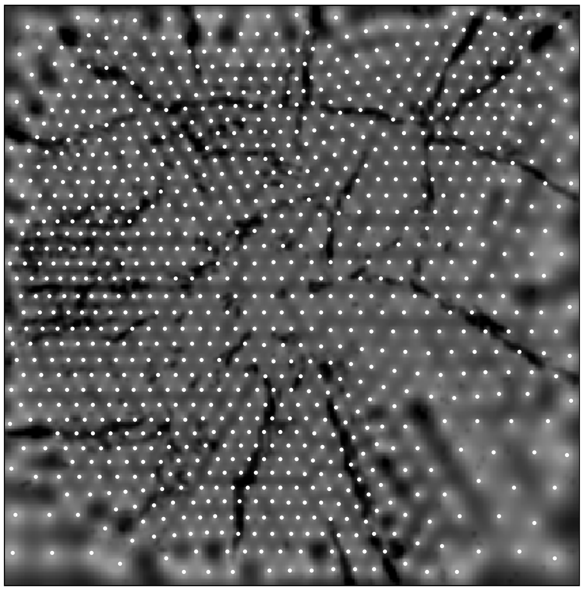


Figure 6: Total potential field $p(\mathbf{x})$ for an initial lattice.

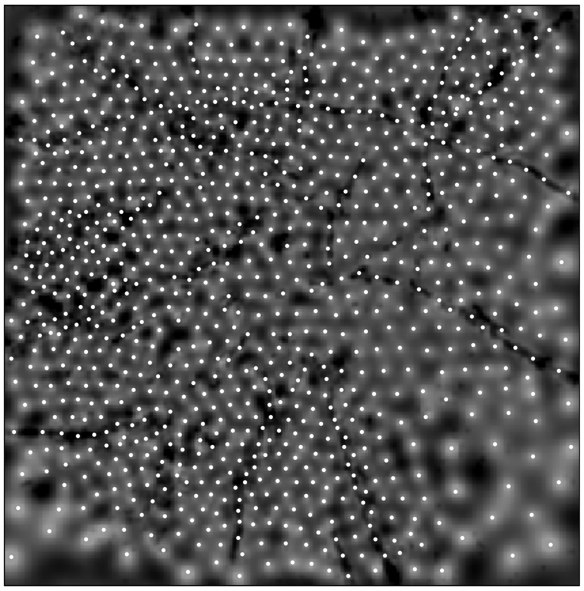


Figure 7: Total potential field $p(\mathbf{x})$ for an optimized lattice.

For images with small or narrow features, it may be useful to perform the first few iterations of Algorithm 3 using a slightly smoothed version of the image. These first iterations enable atoms to move towards features that might otherwise be missed, because they are too far away from initial lattice locations. Atoms are attracted initially to the smoothed features, and then, in subsequent iterations, to high-resolution features in the original unsmoothed image.

The mesh shown in Figure 2 is simply the result of connecting the atoms in the lattice via Delaunay triangulation. After removal of sliver triangles near the convex hull of the lattice, the mesh is both highly regular and well-aligned with image features.

3 APPLICATIONS

Recall that a primary goal of this work is to reduce the effort required to construct meshes suitable for further computations, such as flow simulation.

3.1 Flow simulation

Figure 8 illustrates this application of image meshing. Using the algorithms described above, we first aligned a lattice of atoms with the seismic image. (In this example, we used a constant nominal distance $d(\mathbf{x}) = 8$.) Delaunay triangulation of the atom locations yielded the mesh shown here.

Then, we automatically selected those edges coincident with image samples having values below a specified threshold (here, -0.2). These edges represent faults and are highlighted in Figure 8.

Finally, assuming the faults to be no-flow boundaries, and assuming constant isotropic permeability elsewhere, we computed a steady-state flow solution for a source in the upper-left-hand corner and a sink in the lower-right-hand corner of the image. (To easily handle discontinuities in pressure across faults, we used a variant of the discontinuous Galerkin method [12].) The black lines in Figure 8 represent flow vectors; they are perpendicular to contours of constant pressure, and have length proportional to the pressure gradient.

Although this example is simple in many ways, it demonstrates the complexity of fluid flow in a model with complex boundaries. It also demonstrates the ease with which faults may be modeled in a space-filling mesh derived from a seismic image.

Our method for selecting edges is particularly crude, but the fact that we select them from a space-filling mesh is significant. For one thing, it ensures that their intersections are well defined. Edges either intersect or they do not. Instead of creating topologically consistent boundaries from a collection of independent curves (in 2-D, or surfaces, in 3-D), and then constructing a mesh that conforms to those boundaries, we simply select the boundaries from the mesh.

Most importantly, this example demonstrates that a space-filling mesh aligned with image features is a framework that may enhance subsequent image analyses. The unstructured mesh augments the structured

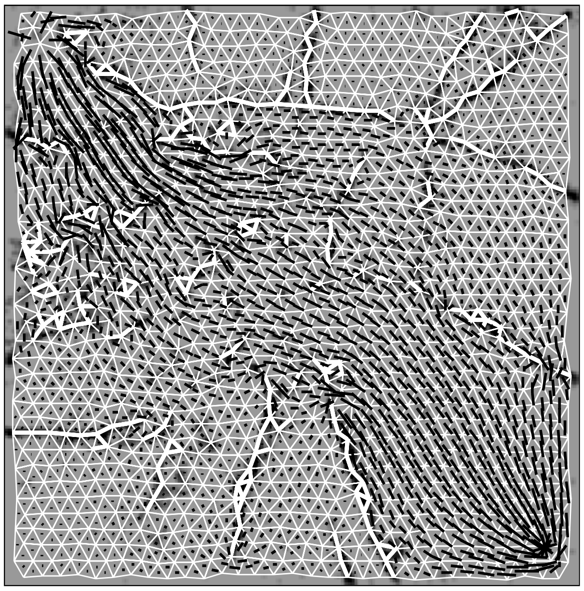


Figure 8: Flow vectors for a faulted reservoir model derived from a seismic image.

grid on which the image is sampled. The mesh is both coarser and more aligned with features of interest than the image sampling grid, and these properties may be exploited in future applications.

3.2 3-D image meshing

Although the examples discussed above illustrate 2-D image meshing, the atomic images method may be used to mesh 3-D images, such as those commonly acquired in seismic and medical imaging.

Figure 9 illustrates the application of the method to a 3-D seismic image. The image displayed here is a nonlinear combination of a conventional seismic image and that same image after processing to enhance faults. During lattice optimization, atoms are attracted to both the faults and the more horizontal seismic horizons.

The intersection of three orthogonal slice planes with a tetrahedral mesh yields a web of triangles and quadrilaterals, many of which are long and thin. However, the corresponding tetrahedra are highly regular. (3-D Delaunay triangulation often yields sliver tetrahedra [13]. Here, we used Joe's local transformations [14] to remove those slivers. One might also use such transformations to improve alignment of triangular faces with image features, although we have not done so here.) In 3-D, mesh nodes (atoms) have x, y, and z coordinates, and they rarely lie precisely in any of the image sampling planes. Here, we show nodes that lie within one sample distance to those planes.

Each white line in a slice plane represents one triangular face of a tetrahedron intersected by that plane. Figure 9 suggests that many of these triangles are aligned with image features. Subsets of contiguous

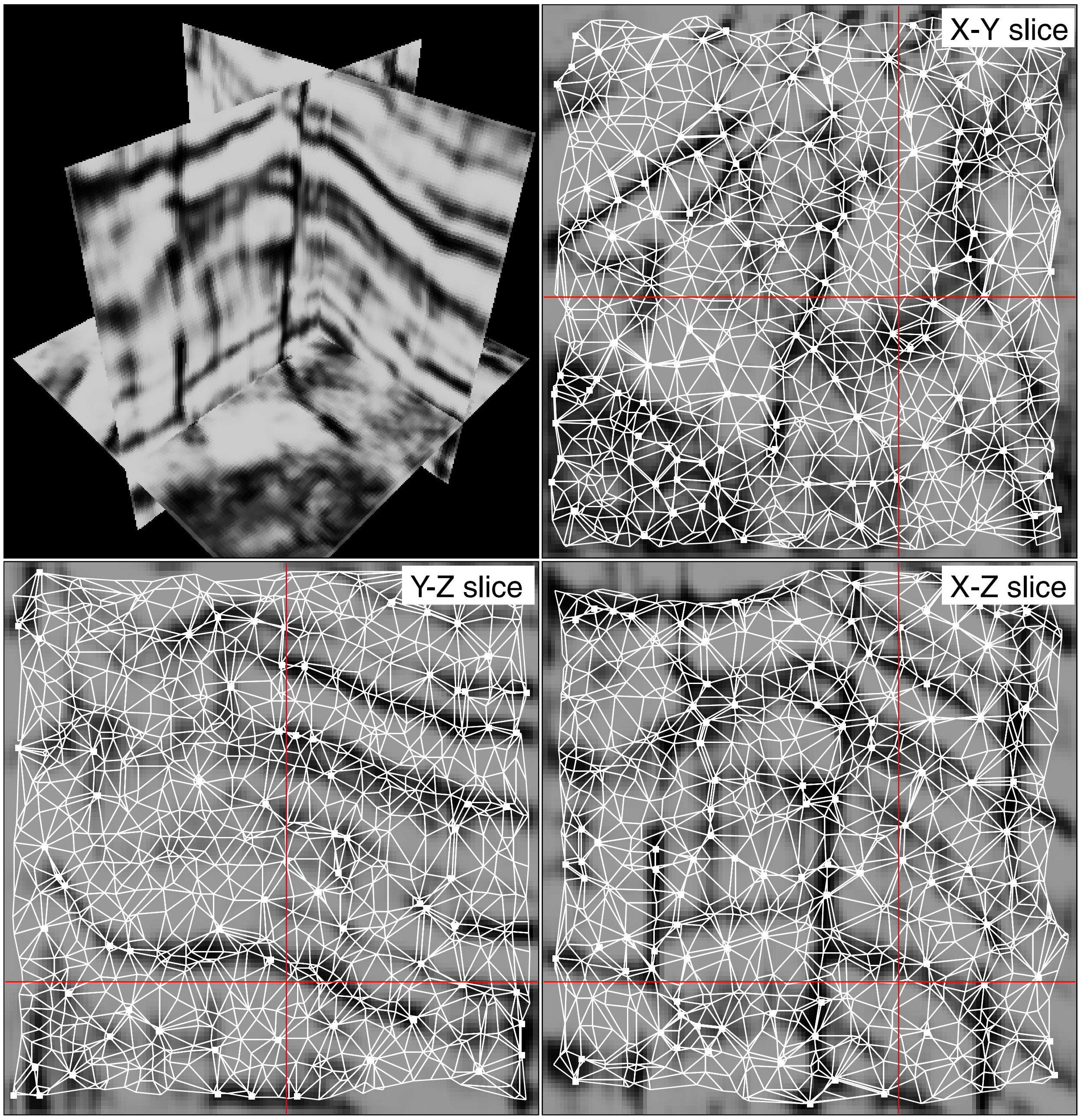


Figure 9: Orthogonal slices from a 3-D tetrahedral mesh that has been aligned with features in a 3-D seismic image. The nearly vertical features in the 3-D image represent faults. The nearly horizontal features represent seismic horizons, reflections caused by changes in rock properties. The tetrahedral mesh has been aligned with both types of features.

triangles so aligned represent surfaces, analogous to the curves represented by subsets of contiguous line segments in Figure 8. Figure 10 illustrates two such surfaces, selected from the space-filling mesh using the simple thresholding test described above. One surface was selected using the fault-enhanced 3-D seismic image. The other was selected using the conventional 3-D seismic image shown here.

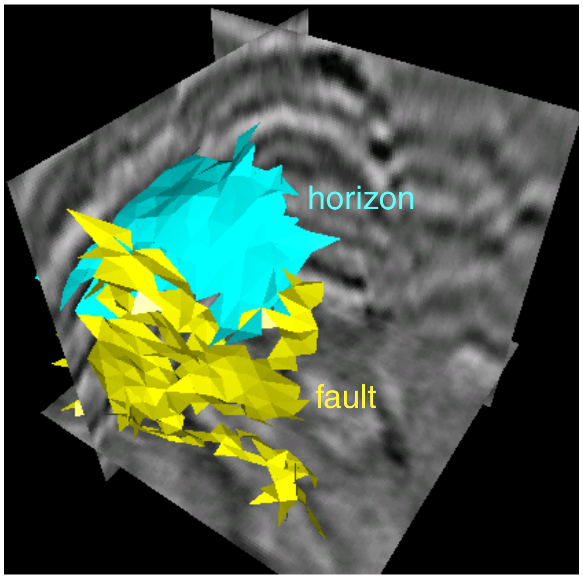


Figure 10: Two geologic surfaces extracted from the space-filling tetrahedral mesh of Figure 9.

3.3 Medical image meshing

The atomic images method may also be applied to medical images, such as that shown in Figure 11. (This image is freely available from the U.S. National Library of Medicine's Visible Human Project.) Before meshing this image, we used a simple and well-known Prewitt edge enhancement algorithm to compute the image potential field (not shown). The resulting mesh, shown in Figure 12, is well-aligned with edges apparent in the image.

4 CONCLUSION

The task of meshing images differs from that of meshing geometric (e.g., CAD) models. Boundaries in the latter are defined precisely, and the goal in meshing is to preserve those boundaries while filling the space between them. In contrast, images are noisy, imprecise representations of something. In an image, features of interest, or boundaries between them, are often poorly defined. Indeed, as the examples above illustrate, image meshing itself may be used to define those features or boundaries.

As we prepared this paper, we learned of recent work [15] in materials science, in which images of a material's microstructure are meshed to create models for finite-element analysis. While our algorithms

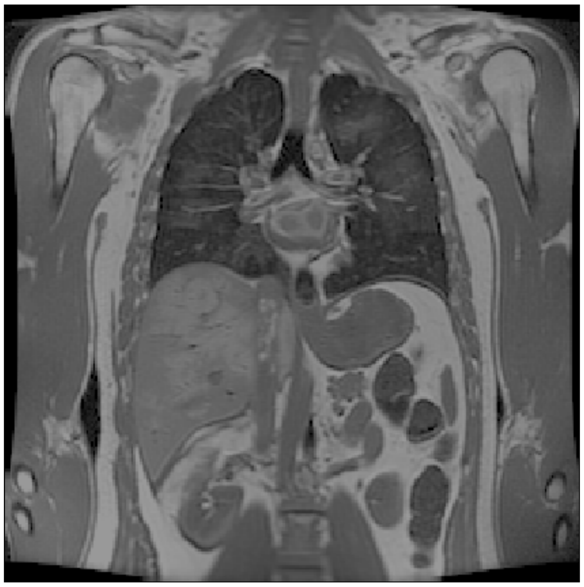


Figure 11: A magnetic resonance image (MRI) of a human torso.

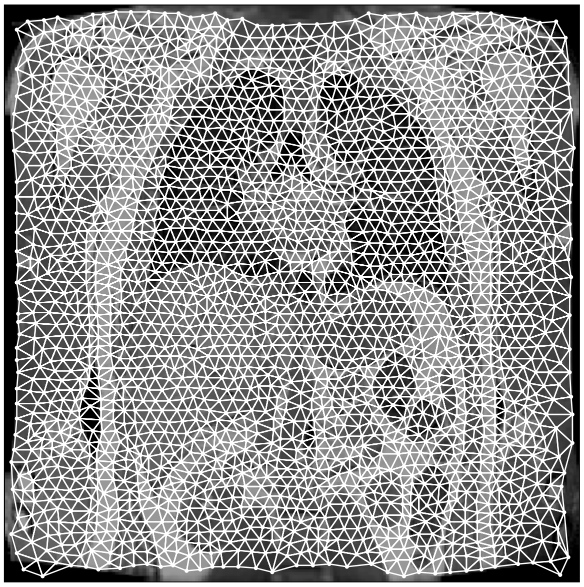


Figure 12: A mesh of the MRI in Figure 11.

for image meshing differ (as they were developed independently), our motives are strikingly similar:

> Remember that the image is an approximate representation of the physical system, the material image is an approximation of the image, and the generated mesh is an approximation to the material image. Forcing the boundary to be well defined and then approximating the boundary by the boundaries of the finite elements just introduces another level of approximation into the existing hierarchy. [15]

We agree, and add that efficiency, as well as accuracy, may be gained by meshing images directly.

Our method for meshing images is based on ideas developed in other contexts. In particular, like Shimada [4], we do not connect mesh nodes (atoms) until *after* we have optimized their locations, thereby avoiding the relatively high cost of repeatedly constructing and destroying those connections. In extending such ideas to the problem of meshing images, we exploit the fact that images are uniformly sampled to obtain a simple and efficient algorithm for optimizing a space-filling lattice of mesh nodes.

Recently, Shimada and others have extended their methods to the problems of anisotropic [16] and quadrilateral [17] meshing. We expect the atomic images method may be extended likewise.

ACKNOWLEDGMENTS

The author is grateful for the comments of two anonymous reviewers, particularly one who highlighted a variety of work related to this. Thanks go also to numerous colleagues at Landmark Graphics for their suggestions, especially Bob Howard for his advice on automatic surface extraction algorithms, and John Killough and Bill Watts for insights on flow simulation.

REFERENCES

[1] J. R. Cebral and R. Löhner, "From Medical Images to CFD Meshes", *Proceedings of the 8th International Meshing Roundtable*, pp. 321–331 (1999).

[2] S. Garrett, S. Griesbach, D. Johnson, D. Jones, M. Lo, W. Orr, and C. Sword, "Earth Model Synthesis", *First Break*, vol. 15, no. 1, pp. 13–20 (1997).

[3] 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).

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

[5] U. T. Mello, and P. R. Cavalcanti, "A Point Creation Strategy for Mesh Generation Using Crystal Lattices as Templates", *Proceedings of the 9th International Meshing Roundtable*, pp. 253–261 (2000).

[6] A. P. Witkin, and P. S. Heckbert, "Using particles to sample and control implicit surfaces", *Computer Graphics Proceedings, Annual Conference Series*, (SIGGRAPH '94), pp. 269–278 (1994).

[7] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models" *International Journal of Computer Vision*, vol. 1, pp. 321–331 (1987).

[8] T. McInerney, and D. Terzopoulos, "Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation", *IEEE Transactions on Medical Imaging*, vol. 18, no. 10, pp. 840–850 (1999).

[9] J. L. Bentley, and J. H. Friedman, "Data Structures for Range Searching", *Computing Surveys*, vol. 11, no. 4, pp. 397–409 (1979).

[10] G. Turk, and D. Banks, "Image-Guided Streamline Placement", *Computer Graphics Proceedings, Annual Conference Series*, (SIGGRAPH '96), pp. 453–460 (1996).

[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] 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).

[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).

[14] B. Joe, "Construction of Three-Dimensional Improved-Quality Triangulations Using Local Transformations", *SIAM Journal of Scientific Computing*, vol. 6, pp. 1292–1307 (1995).

[15] S. A. Langer, E. R. Fuller, Jr., and W. C. Carter, "OOF: An Image-Based Finite-Element Analysis of Material Microstructures", *Computing in Science & Engineering*, vol. 3, no. 3, pp. 15–23 (2001).

[16] S. Yamakawa and K. Shimada, "High Quality Anisotropic Tetrahedral Mesh Generation via Ellipsoidal Bubble Packing", *Proceedings of the 9th International Meshing Roundtable*, pp. 263–273 (2000).

[17] N. Viswanath, K. Shimada, and T. Itoh, "Quadrilateral Meshing with Anisotropy and Directional Control via Close Packing of Rectangular Cells", *Proceedings of the 9th International Meshing Roundtable*, pp. 227–238 (2000).