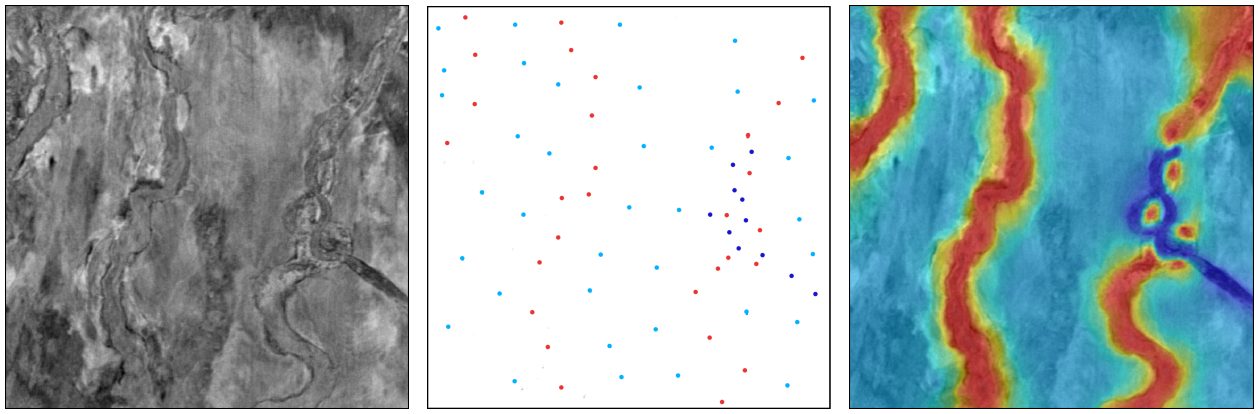
Image-guided blended neighbor interpolation

Dave Hale
Center for Wave Phenomena, Colorado School of Mines, Golden CO 80401, USA



(a) (b) (c)

Figure 1. A seismic image (a) and scattered data (b) used in image-guided blended neighbor interpolation (c).

ABSTRACT
Uniformly sampled images are often used to interpolate other data acquired more sparsely with an entirely different mode of measurement. For example, downhole tools enable geophysical properties to be measured with high precision near boreholes that are scattered spatially, and less precise seismic images acquired at the earth's surface are used to interpolate those properties at locations far away from the boreholes. *Image-guided interpolation* is designed specifically to enhance this process.
Most existing methods for interpolation require distances from points where data will be interpolated to nearby points where data are known. Image-guided interpolation requires non-Euclidean distances in metric tensor fields that represent the coherence, orientations and shapes of features in images. This requirement leads to a new method for interpolating scattered data that I call *blended neighbor interpolation*. For simple Euclidean distances, blended neighbor interpolation resembles the classic natural neighbor interpolation.

Key words: seismic image interpolation interpretation

1 INTRODUCTION

Interpolation of spatially scattered data is a classic problem. In the example shown in Figure 1, I guided this interpolation to conform to buried channels in a uniformly-sampled seismic image of the earth's subsurface. The 76 scattered sample values are represented by just three colors cyan, red and blue. Image-guided blended neighbor interpolation honors these values while computing a second transparent image overlay with a continuous spectrum of colors. The color yellow, not represented in the scattered data, corresponds to a value between the values for red and cyan; features painted yellow therefore lie between those painted red and cyan. In this example, I painted the scattered data values interactively.

More generally, the scattered data that we seek to

interpolate are a set

$$\mathcal{F} = \{f_1, f_2, \ldots, f_K\} \tag{1}$$

of K known sample values $f_k \in \mathbb{R}$ that correspond to a set

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\} \tag{2}$$

of K known sample points $\mathbf{x}_k \in \mathbb{R}^n$. Together these two sets comprise a set

$$\mathcal{K} = \{(f_1, \mathbf{x}_1), (f_2, \mathbf{x}_2), \ldots, (f_K, \mathbf{x}_K)\} \tag{3}$$

of K known samples. These samples are scattered in the sense that the n-dimensional sample points in the set $\mathcal{X}$ may have no regular geometric structure. The classic interpolation problem is to use the known samples in $\mathcal{K}$ to construct a function $q(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, such that $q(\mathbf{x}_k) = f_k$.

This problem has no unique solution, as there exist an infinite number of functions $q(\mathbf{x})$ that satisfy the interpolation conditions $q(\mathbf{x}_k) = f_k$. Additional criteria may include measures of smoothness, robustness, and efficiency. Because tradeoffs exist among such criteria, a variety of methods for interpolating scattered data are commonly used today. Foster and Evans (2008) provide a recent evaluation of several methods in the context of a geoscience application.

In this paper I add the requirement that the interpolation should somehow conform to a uniformly sampled image, as in the example of Figure 1. Mindful of this additional requirement, I first review some popular interpolation methods, before combining elements of them to construct a new method for image-guided interpolation.

1.1 PDE methods for interpolation

One popular family of interpolation methods is derived from partial differential equations (e.g., Gáspár, 1999). For example, the harmonic interpolant is the solution to

$$\begin{aligned} \nabla^2 q(\mathbf{x}) &= 0, \quad \mathbf{x} \notin \mathcal{X}; \\ q(\mathbf{x}_k) &= f_k, \quad \mathbf{x}_k \in \mathcal{X}. \end{aligned} \tag{4}$$

The biharmonic interpolant is defined similarly as the solution to

$$\begin{aligned} (\nabla^2)^2 q(\mathbf{x}) &= 0, \quad \mathbf{x} \notin \mathcal{X}; \\ q(\mathbf{x}_k) &= f_k, \quad \mathbf{x}_k \in \mathcal{X}. \end{aligned} \tag{5}$$

Examples of both interpolants are illustrated in Figure 2. Intuitively, these interpolants are made smooth by zeroing high-order derivatives of the interpolating function $q(\mathbf{x})$ at points $\mathbf{x} \notin \mathcal{X}$.

Both the harmonic and biharmonic interpolants are defined explicitly to satisfy the interpolation conditions $q(\mathbf{x}_k) = f_k$. Moreover, because the Laplacian of any linear function is zero, both interpolants have linear precision; they will exactly recover a linear function $f(\mathbf{x}) = f_0 + \mathbf{g}_0^T(\mathbf{x} - \mathbf{x}_0)$ (for some constants f_0, $\mathbf{g}_0$ and $\mathbf{x}_0$) from samples $f_k = f(\mathbf{x}_k)$.

However, as shown in Figures 2a and 2b, the harmonic interpolant has sharp cusps at the sample points. In contrast, the biharmonic interpolants in Figures 2c and 2d are much smoother; if smoothness were our only desirable feature, then we would certainly favor biharmonic interpolation.

Unfortunately, the increased smoothness of biharmonic interpolation is obtained at the cost of decreased computational efficiency. To understand why, first note that we cannot analytically solve equations 4 or 5 for arbitrary sets $\mathcal{K}$ of scattered data. Instead, we must discretize these equations for some region of interest and thereby obtain a large sparse system of linear equations to be solved numerically. The number of equations equals the number of points $\mathbf{x}$ at which we want to interpolate. Because that number is typically large (10^5 or higher), iterative methods are most efficient in solving these equations.

When the number of known samples is much less than the number of samples to be interpolated, a simple iterative conjugate-gradient solver requires many iterations to converge to a sufficiently accurate numerical solution to equations 4 and 5. The number of iterations required is roughly proportional to the largest distance $\|\mathbf{x} - \mathbf{x}_k\|$. If I iterations are required to solve the harmonic equations 4, then roughly I^2 iterations are required to solve the biharmonic equations 5. Therefore, more efficient iterative solvers such as multigrid methods (Gáspár, 1999) are typically used for biharmonic interpolation.

In the context of image-guided interpolation, methods based on solutions to partial differential equations are tempting because they can be easily modified for metric tensor fields $\mathbf{D}(\mathbf{x})$ derived from images. We simply replace the homogeneous and isotropic Laplacian operator ∇^2 in equations 4 and 5 with the anisotropic and inhomogeneous operator $\boldsymbol{\nabla} \cdot \mathbf{D}(\mathbf{x}) \boldsymbol{\nabla}$. For example, the biharmonic interpolant becomes

$$\begin{aligned} (\boldsymbol{\nabla} \cdot \mathbf{D}(\mathbf{x}) \boldsymbol{\nabla})^2 q(\mathbf{x}) &= 0, \quad \mathbf{x} \notin \mathcal{X}; \\ q(\mathbf{x}_k) &= f_k, \quad \mathbf{x}_k \in \mathcal{X}. \end{aligned} \tag{6}$$

Unfortunately, most of the efficiency of multigrid methods may be lost for equation 6 with inhomogeneous and anisotropic coefficients. While significantly more difficult to implement, a multigrid method may for this case be no more efficient than the simplest conjugate-gradient method.

A second potentially undesirable feature of biharmonic interpolation is the overshoot of sample values f_k illustrated in Figure 2c. Unlike the harmonic interpolant, the biharmonic interpolant is unbounded by minimum and maximum sample values in the set $\mathcal{F}$, and it is generally impossible to predict the minimum and maximum interpolated values of $q(\mathbf{x})$ before solving

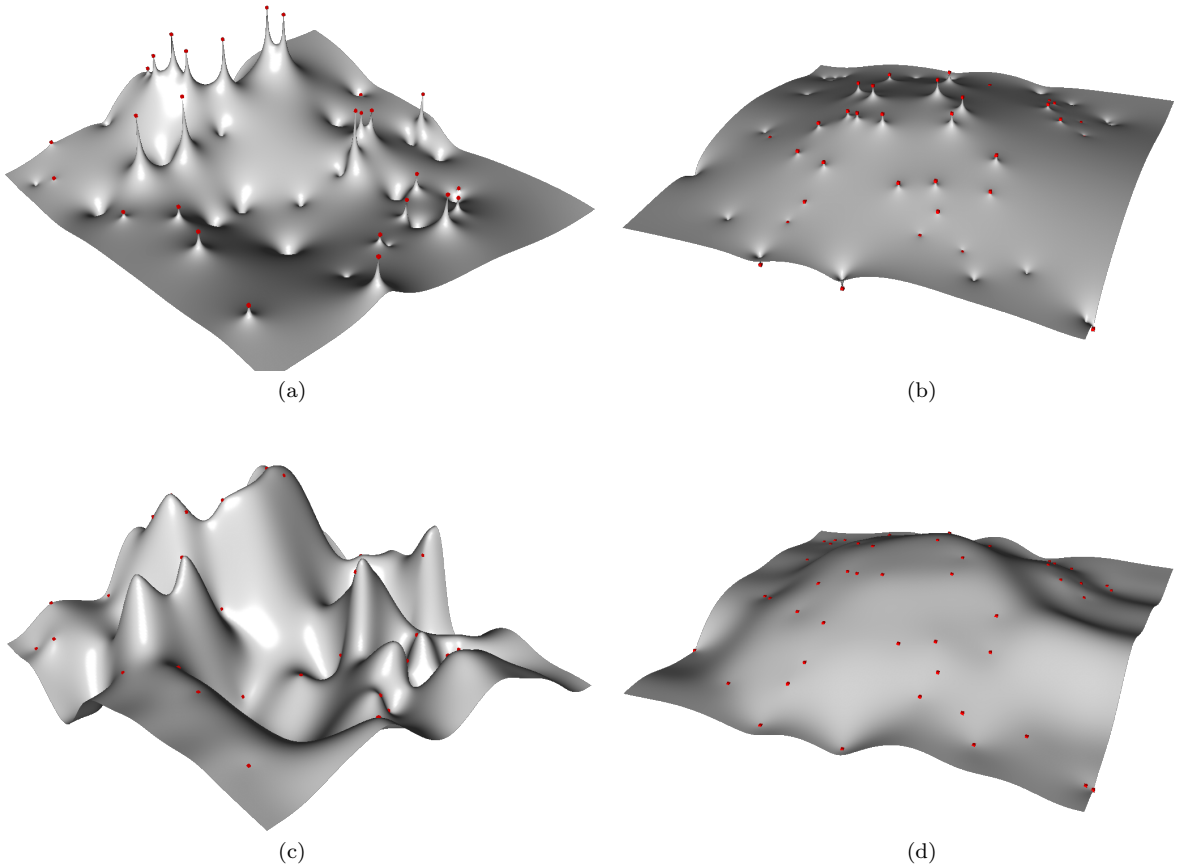


Figure 2. Harmonic (a and b) and biharmonic (c and d) interpolation of scattered samples of rough and smooth functions.

the biharmonic equations 5. In this respect, biharmonic interpolation is less robust than harmonic interpolation.

To reduce the amount of overshoot, we may construct a weighted combination of the harmonic equations 4 and the biharmonic equations 5. The weight in this combination is sometimes called *tension* (Mitášová and Luboš, 1993), and for any given set $\mathcal{K}$ of scattered data it is difficult to know what tension is appropriate before solving the equations.

1.2 RBF methods for interpolation

The harmonic and biharmonic interpolants are closely related to another family of interpolants of the form

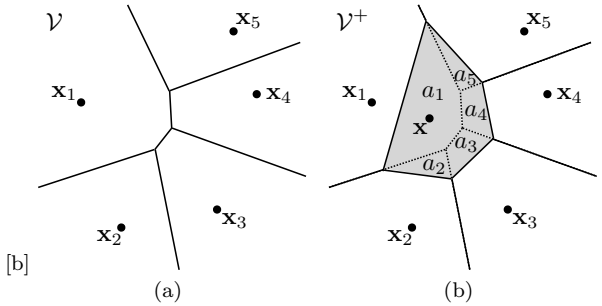
$$q(\mathbf{x}) = \sum_{k=1}^{K} w_k \phi(\|\mathbf{x} - \mathbf{x}_k\|), \tag{7}$$

where $\phi(r) : \mathbb{R} \to \mathbb{R}$ is a *radial basis function* (RBF) of distance r (Dyn, 1987; Carr et al., 1997). For example, in two dimensions, the thin-plate spline (Duchon, 1977) corresponds to the RBF $\Phi(\mathbf{x}) \equiv \phi(\|\mathbf{x}\|) = \|\mathbf{x}\|^2 \log\|\mathbf{x}\|$, which is a fundamental solution of the biharmonic equation $(\nabla^2)^2\Phi(\mathbf{x}) = 0$.

To determine the weights w_k in equation 7, we use the interpolation conditions $q(\mathbf{x}_k) = f_k$ to obtain a system of K linear equations for the K unknown weights. For an RBF without compact support this system of equations is dense, not sparse, and the computational cost of solving them is $O(K^3)$.

In other words, the cost of RBF interpolation grows quickly with the number K of known samples, in part because each interpolated sample depends through equation 7 on every known sample. Like the PDE methods described above, RBF methods for interpolation are *global*. If a known sample $(f_k, \mathbf{x}_k)$ in the set $\mathcal{K}$ is added, removed, or modified, all interpolated values $q(\mathbf{x})$ must be recomputed, no matter how great the distance $\|\mathbf{x} - \mathbf{x}_k\|$.

To reduce this cost we might use an RBF with compact support (Franke, 1982; Schaback, 2005) to obtain a sparse system of equations. But the resulting interpolation methods raise a new question. For what distance $\|\mathbf{x} - \mathbf{x}_k\|$ should the RBF $\Phi(\mathbf{x} - \mathbf{x}_k)$ go to zero? That



[b]

Figure 3. In natural neighbor interpolation at point $\mathbf{x}$, the weight for sample value f_k is proportional to the area $a_k(\mathbf{x})$ of intersection of (a) the Voronoi polygon for $\mathbf{x}_k$ in $\mathcal{V}$ and (b) the Voronoi polygon for $\mathbf{x}$ in $\mathcal{V}^+$.

is, for a given point $\mathbf{x}$, which nearby samples should be used in equation 7 to construct the interpolant $q(\mathbf{x})$? The answer to this question may depend in a complicated way on the distribution of the sample points $\mathcal{X}$.

1.3 Natural neighbor interpolation

An alternative *local* method that specifically answers the question of which nearby samples are relevant is Sibson's (1981) *natural neighbor* interpolation. Sibson defined his interpolant $q(\mathbf{x})$ in terms of Voronoi diagrams, as illustrated for $\mathbb{R}^2$ in Figure 3. In n dimensions, a Voronoi diagram decomposes $\mathbb{R}^n$ into convex neighborhoods of points $\mathbf{x}$ that are nearer to one known sample point $\mathbf{x}_k$ than to any other.

The Voronoi diagram is therefore closely related to *nearest neighbor* interpolation. For any $\mathbf{x}$, the nearest neighbor interpolant $p(\mathbf{x})$ is simply the value f_k corresponding to the nearest sample point $\mathbf{x}_k$. That is,

$$p(\mathbf{x}) = f_k \,|\, k = \stackrel{\arg\min}{j} \|\mathbf{x} - \mathbf{x}_j\|. \tag{8}$$

Note that nearest neighbor interpolation is well defined for only those points $\mathbf{x}$ with a single nearest sample point $\mathbf{x}_k$. At points $\mathbf{x}$ that are equidistant from two or more sample points, the nearest neighbor interpolant $p(\mathbf{x})$ is discontinuous and undefined.

Natural neighbor interpolation is more complex. Let $\mathcal{V}$ denote the Voronoi diagram of the set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$ of sample points, and let $\mathcal{V}^+$ denote the Voronoi diagram of the augmented set $\mathcal{X}^+ = \{\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}$. As illustrated in Figure 3 for $\mathbb{R}^2$, each point in the Voronoi diagrams $\mathcal{V}$ and $\mathcal{V}^+$ has a corresponding Voronoi polygon.

Now let $a_k(\mathbf{x})$ denote the area of intersection of the polygon for $\mathbf{x}$ in $\mathcal{V}^+$ with the polygon in $\mathcal{V}$ for a sample point $\mathbf{x}_k$. The natural neighbor interpolant is then

$$q(\mathbf{x}) = \frac{\sum_{k=1}^{K} a_k(\mathbf{x}) f_k}{\sum_{k=1}^{K} a_k(\mathbf{x})}. \tag{9}$$

When the number K of known samples is large, most of the areas $a_k(\mathbf{x})$ in equation 9 are typically zero, and the sum can be limited to only a subset of neighbor samples. These natural neighbors can be found efficiently using a Delaunay triangulation, which is the geometric dual of the Voronoi diagram of the sample points.

For points $\mathbf{x}$ that lie outside the convex hull of the set $\mathcal{X}$ of sample points, areas $a_k(\mathbf{x})$ in equation 9 become infinite, which makes computation of the natural neighbor interpolant difficult. In practice, this difficulty is sometimes overcome by adding additional ghost points (Bobach et al., 2008) to create a convex hull outside the region of interest.

Figure 4 illustrates nearest and natural neighbor interpolation for sparsely sampled rough and smooth functions. In natural neighbor interpolation, four ghost points have been added at the corners of a square outside the region displayed in Figures 4c and 4d.

Whereas discontinuities in the piecewise-constant nearest neighbor interpolants are clearly apparent, the natural neighbor interpolants appear to be continuous. Indeed, as shown by Sibson (1981) and others (e.g., Farin, 1990), natural neighbor interpolants are continuous everywhere and have continuous gradients everywhere except at the sample points $\mathbf{x} = \mathbf{x}_k$. Moreover, like harmonic and biharmonic interpolation, natural neighbor interpolation has linear precision.

Also apparent in Figure 4 is that the minimum and maximum values of the interpolants $q(\mathbf{x})$ are bounded by minimum and maximum values f_k of the known samples. In this sense natural neighbor interpolants are robust, with no unpredictable oscillation or overshoot.

When implemented with appropriate data structures, including the Delaunay triangulation mentioned above, natural neighbor interpolation is efficient. Much of this efficiency is due to its local property. As noted above, most of the areas $a_k(\mathbf{x})$ in equation 9 are typically zero and need not be computed. (A pathological exception would be the case where all sample points $\mathbf{x}_k$ lie on the convex hull of the set $\mathcal{X}$.) In contrast to PDE and RBF interpolants, the natural neighbor interpolant at some point $\mathbf{x}$ typically does not depend on every known sample in $\mathcal{K}$.

In development of a method for image-guided interpolation, I seek to retain the desirable properties — linear precision, smoothness between sample points, robustness, and locality — of natural neighbor interpolation cited above. This development leads to a new method for *blended neighbor interpolation* that combines elements of both natural neighbor and PDE methods.

2 BLENDING NEAREST NEIGHBORS

Suppose that, for a set $\mathcal{K}$ of known samples, we have constructed two functions defined as follows:

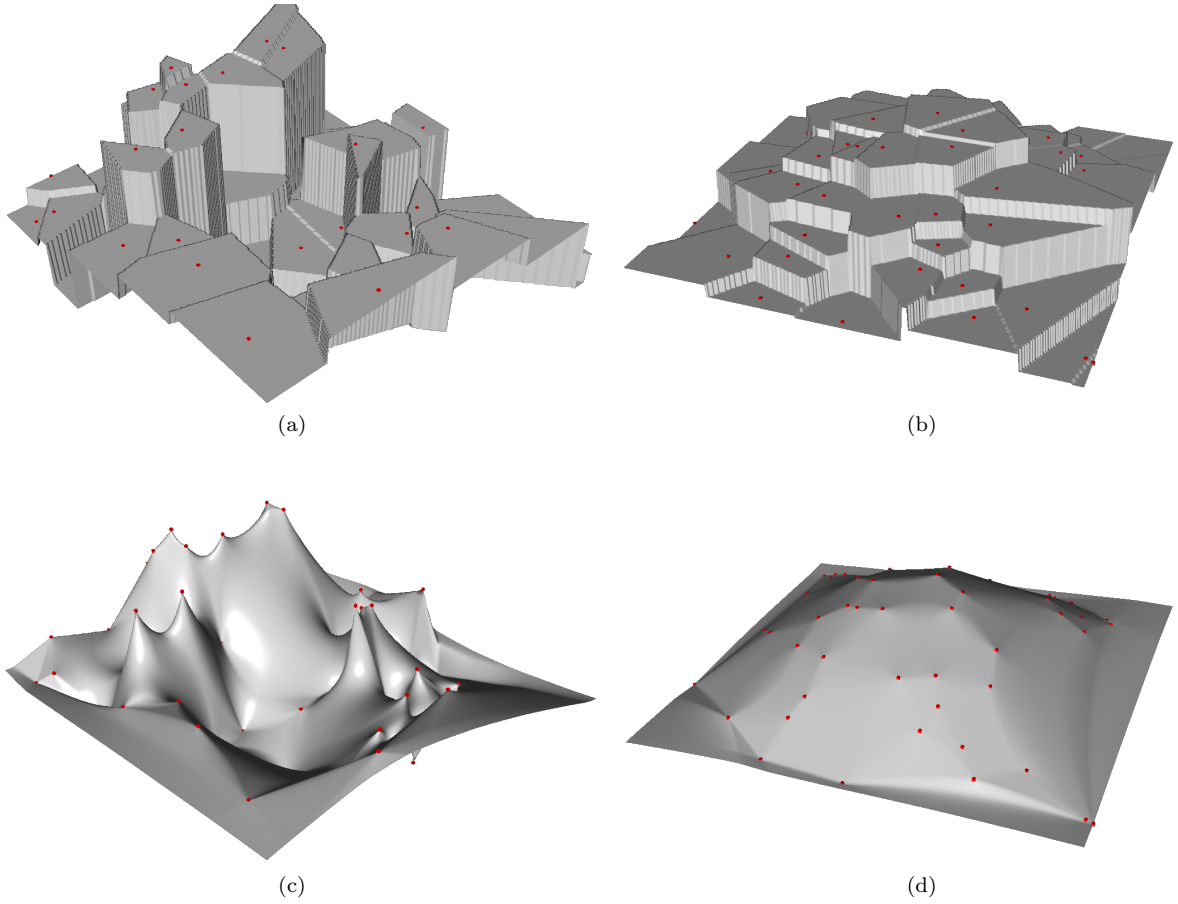


Figure 4. Nearest neighbor (a and b) and natural neighbor (c and d) interpolation of scattered samples.

$d(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is the distance $\|\mathbf{x} - \mathbf{x}_k\|$ from $\mathbf{x}$ to the nearest known sample point $\mathbf{x}_k$, and

$p(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is the value f_k corresponding to the sample point $\mathbf{x}_k$ nearest to the point $\mathbf{x}$.

In other words, $d(\mathbf{x})$ is the *distance map* and $p(\mathbf{x})$ is the *nearest neighbor interpolant* for the scattered data $\mathcal{K}$.

With these two functions, the natural neighbor interpolant is defined by

$$q(\mathbf{x}) = \frac{\int h(\mathbf{x},\mathbf{y})\, p(\mathbf{y})\, \mathrm{d}^n\mathbf{y}}{\int h(\mathbf{x},\mathbf{y})\, \mathrm{d}^n\mathbf{y}}, \tag{10}$$

where

$$h(\mathbf{x},\mathbf{y}) \equiv \begin{cases} 1, & \text{if } \|\mathbf{x}-\mathbf{y}\| \le d(\mathbf{y}); \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

This definition of natural neighbor interpolation is equivalent to Sibson's (1981) definition in equation 9. For any interpolated point $\mathbf{x}$, the function $h(\mathbf{x},\mathbf{y})$ defined in equation 11 restricts the integrals in equation 10 to precisely the shaded polygon illustrated in Figure 3b.

We may interpret equation 10 in two different ways. The first and perhaps most obvious interpretation is that numerous $p(\mathbf{y})$-weighted infinitesimal elements $\mathrm{d}^n\mathbf{y}$ are gathered into a single interpolated value $q(\mathbf{x})$. The second interpretation is that a single $p(\mathbf{y})$-weighted infinitesimal element $\mathrm{d}^n\mathbf{y}$ is scattered into numerous interpolated values $q(\mathbf{x})$. These two interprations are illustrated in Figure 5.

Both the gather and scatter interpretations of equation 10 imply that the natural neighbor interpolant $q(\mathbf{x})$ is obtained by smoothing the nearest neighbor interpolant $p(\mathbf{x})$. And in both interpretations the smoothing filter $h(\mathbf{x},\mathbf{y})$ varies spatially according to equation 11. However, the circular region of support for $h(\mathbf{x},\mathbf{y})$ used in scattering is simpler than the polygonal region used in gathering.

In a discrete approximation of natural neighbor interpolation (Park et al., 2006), scattering may be more efficient than gathering. In this approximation, the functions in equations 10 and 11 are sampled uniformly and the integrals become sums. Gathering then implies an outer loop over sampled $\mathbf{x}$ and an inner loop over sam-

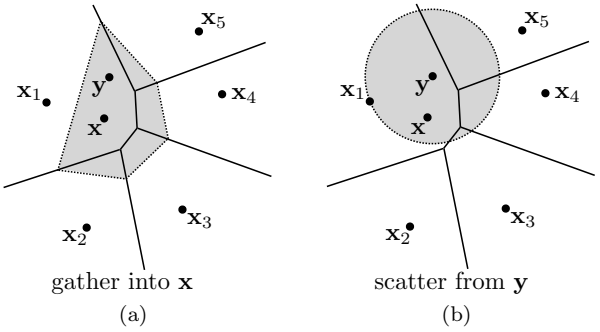


Figure 5. Natural neighbor interpolation as defined by equations 10 and 11 is either (a) for each $q(\mathbf{x})$ a gathering of many $p(\mathbf{y})$, or equivalently (b) a scattering of each $p(\mathbf{y})$ into many $q(\mathbf{x})$ within a circle of radius $d(\mathbf{y})$.

pled $\mathbf{y}$. Because $d(\mathbf{y})$ in equation 11 varies inside the inner gathering loop over $\mathbf{y}$, it may be costly to determine the range of sampled $\mathbf{y}$ for which the smoothing filter $h(\mathbf{x}, \mathbf{y})$ is non-zero. Scattering instead of gathering interchanges the loops, so that the range of sampled $\mathbf{x}$ for which $h(\mathbf{x}, \mathbf{y})$ is non-zero is circular and can therefore be computed more efficiently.

The computational complexity of the digital smoothing filter is $O(LM)$, where L is the average number of non-zero samples in $h(\mathbf{x}, \mathbf{y})$ and M is the number of sampled values in $p(\mathbf{x})$ and $q(\mathbf{x})$. Equation 11 implies that the factor L grows with distances in the map $d(\mathbf{x})$. In two dimensions, L grows with distance squared, proportional to the area of circles like that shown in Figure 5b. In three dimensions, L grows with distance cubed, proportional to the volume of spheres. The total $O(LM)$ cost of discrete natural neighbor interpolation therefore grows quickly with distances in $d(\mathbf{x})$.

2.1 Blending with PDEs

To reduce this potentially high cost, I propose an alternative smoothing filter, one with a factor L that grows only linearly with distance $d(\mathbf{x})$. To apply this filter, we solve the partial differential equation

$$q(\mathbf{x}) - \frac{1}{2} \boldsymbol{\nabla} \cdot d^2(\mathbf{x}) \boldsymbol{\nabla} q(\mathbf{x}) = p(\mathbf{x}), \tag{12}$$

with some suitable (e.g., zero-slope) boundary conditions for the domain of interest.

At known sample points $\mathbf{x}_k$ for which $d(\mathbf{x}_k) = 0$, the solution to equation 12 is clearly $q(\mathbf{x}_k) = p(\mathbf{x}_k) = f_k$. At these known sample points the nearest neighbor and natural neighbor interpolants are equal to the known sample values; the interpolation conditions are satisfied.

To show that elsewhere $q(\mathbf{x})$ is a continuous smoothed version of $p(\mathbf{x})$, let us assume momentarily that the distance d is constant and then consider the Fourier transform of equation 12:

$$Q(\mathbf{k}) = \frac{P(\mathbf{k})}{1 + \frac{1}{2} d^2 k^2}. \tag{13}$$

The factor $k^2 = \mathbf{k} \cdot \mathbf{k}$ in the denominator implies attenuation of high frequencies. This attenuation increases with distance d so that (now ignoring our assumption of constant d) smoothing is most significant at points $\mathbf{x}$ for which distances $d(\mathbf{x})$ to the nearest known sample points are largest.

I use the name *blended neighbor interpolation* to describe the solution of equation 12, because it blends nearest neighbors much like equation 10 does in natural neighbor interpolation. The difference lies in the smoothing filter $h(\mathbf{x}, \mathbf{y})$, defined explicitly by equation 11, but implicitly by the partial differential equation 12. Both filters produce a continuous interpolant $q(\mathbf{x})$ by smoothing the discontinous nearest neighbor interpolant $p(\mathbf{x})$, and the amount of smoothing increases with distances $d(\mathbf{x})$ to known sample points.

Figure 6 illustrates discrete natural neighbor and blended neighbor interpolants. As expected, these interpolants are similar but not identical. Small ridges, grooves and bumps in the discrete natural neighbor interpolants are caused by discretization on a rectangular grid of circles like that in Figure 5b. These artifacts and the circular creases in Figures 6a and 6b are not apparent in the discrete blended neighbor interpolants in Figures 6c and 6d.

Discrete approximation of the blending equation 12 yields a large sparse system of equations that are best solved by an iterative method. The number of simple conjugate-gradient iterations required to converge to a solution is roughly proportional to the maximum distance $d(\mathbf{x})$. For any dimension n of points $\mathbf{x} \in \mathbb{R}^n$, the computational complexity of discrete blended neighbor interpolation grows only linearly with distance. For sample points scattered at large distances, this linear dependence contrasts favorably with the quadratic dependence in 2D (or cubic dependence in 3D) for discrete natural neighbor interpolation via equations 10 and 11.

The factor $\frac{1}{2}$ and exponent 2 in $d^2(\mathbf{x})$ of equation 12 were chosen carefully to ensure that the interpolant $q(\mathbf{x})$ has linear precision. Because the nearest neighbor interpolant $p(\mathbf{x})$ is piecewise constant, it is easy to verify that inside the Voronoi neighborhood of any sample point $\mathbf{x}_k$, where $d(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_k\|$ and $p(\mathbf{x}) = f_k$, the linear function

$$q_k(\mathbf{x}) = f_k + \mathbf{g}_k^T (\mathbf{x} - \mathbf{x}_k) \tag{14}$$

is a solution to equation 12. Therefore, if scattered data are obtained by sampling a linear function $f(\mathbf{x}) = f_0 + \mathbf{g}_0^T (\mathbf{x} - \mathbf{x}_0)$, then the continous blended neighbor interpolant $q(\mathbf{x})$ [the union of all piecewise linear $q_k(\mathbf{x})$], must equal that linear function.

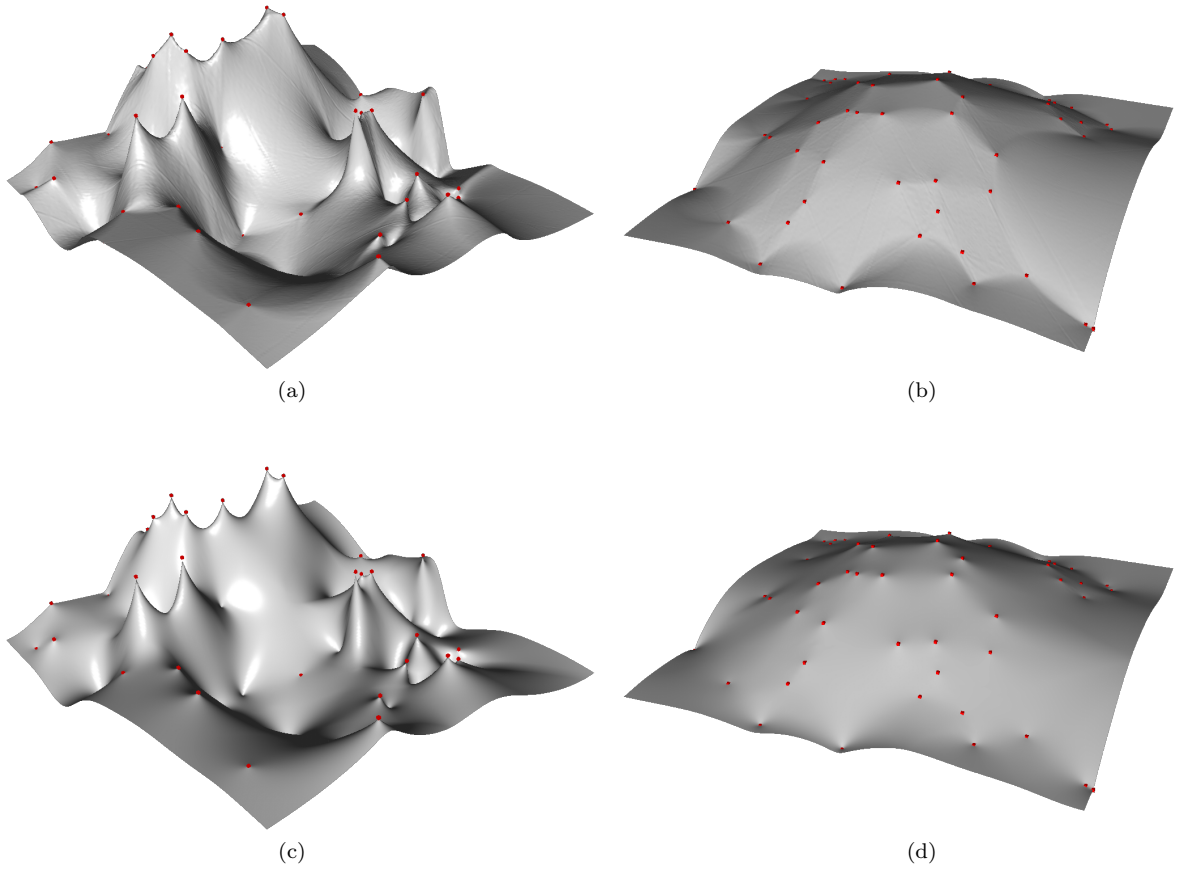


Figure 6. Discrete natural neighbor (a and b) and blended neighbor (c and d) interpolation of scattered samples.

2.2 Alternative blending PDEs

We may obtain alternatives to the blending equation 12 by simply modifying its coefficients. For example, we might restrict the variable coefficient $d(\mathbf{x})$ in equation 12 to not exceed some specified maximum distance. This maximum would limit the extent of smoothing and thereby also limit the number of iterations required to converge to a solution $q(\mathbf{x})$. Linear precision of the blended neighbor interpolant would be lost for large distances but maintained where perhaps most desirable, at locations $\mathbf{x}$ where the density of known samples is high and distances $d(\mathbf{x})$ are relatively small.

A related alternative is to replace the constant coefficient $\frac{1}{2}$ in equation 12 with a smaller value. In doing so, we lose linear precision for all $\mathbf{x}$, but gain smoothness at the known sample points $\mathbf{x}_k$, where $\boldsymbol{\nabla} q(\mathbf{x}_k) = 0$. In effect, decreasing this coefficient reduces the amount of smoothing performed by the blending equation 12. Indeed, in the limit as this coefficient goes to zero, no smoothing is performed and the blended neighbor interpolant $q(\mathbf{x})$ equals the nearest neighbor interpolant $p(\mathbf{x})$.

More generally, we may solve

$$q(\mathbf{x}) - \frac{1}{e}\,\boldsymbol{\nabla} \cdot d^2(\mathbf{x})\,\boldsymbol{\nabla}\, q(\mathbf{x}) = p(\mathbf{x}), \qquad (15)$$

for some constant $e \geq 2$. Figures 6c and 6d correspond to the choice $e = 2$, for interpolants with linear precision.

Figure 7 displays blended neighbor interpolants for $e = 4$ and $e = 8$. While smoother at the known sample points $\mathbf{x}_k$, these interpolants have flat spots, small plateaus at those points, and cannot precisely interpolate data sampled from linear functions.

The choice $e = 3$ is interesting because, inside the Voronoi neighborhood of any known sample point $\mathbf{x}_k$, equation 15 then has the solution

$$q_k(\mathbf{x}) = f_k + h_k \|\mathbf{x} - \mathbf{x}_k\|, \qquad (16)$$

for some constant h_k. Although the solution $q_k(\mathbf{x})$ lacks linear precision, its dependence on the radial distance $\|\mathbf{x} - \mathbf{x}_k\|$ reminds us of RBF methods for interpolation. In particular, the RBF $\phi(\|\mathbf{x} - \mathbf{x}_k\|) = \|\mathbf{x} - \mathbf{x}_k\|$ corresponds to the 3D biharmonic interpolant (Sandwell, 1987).

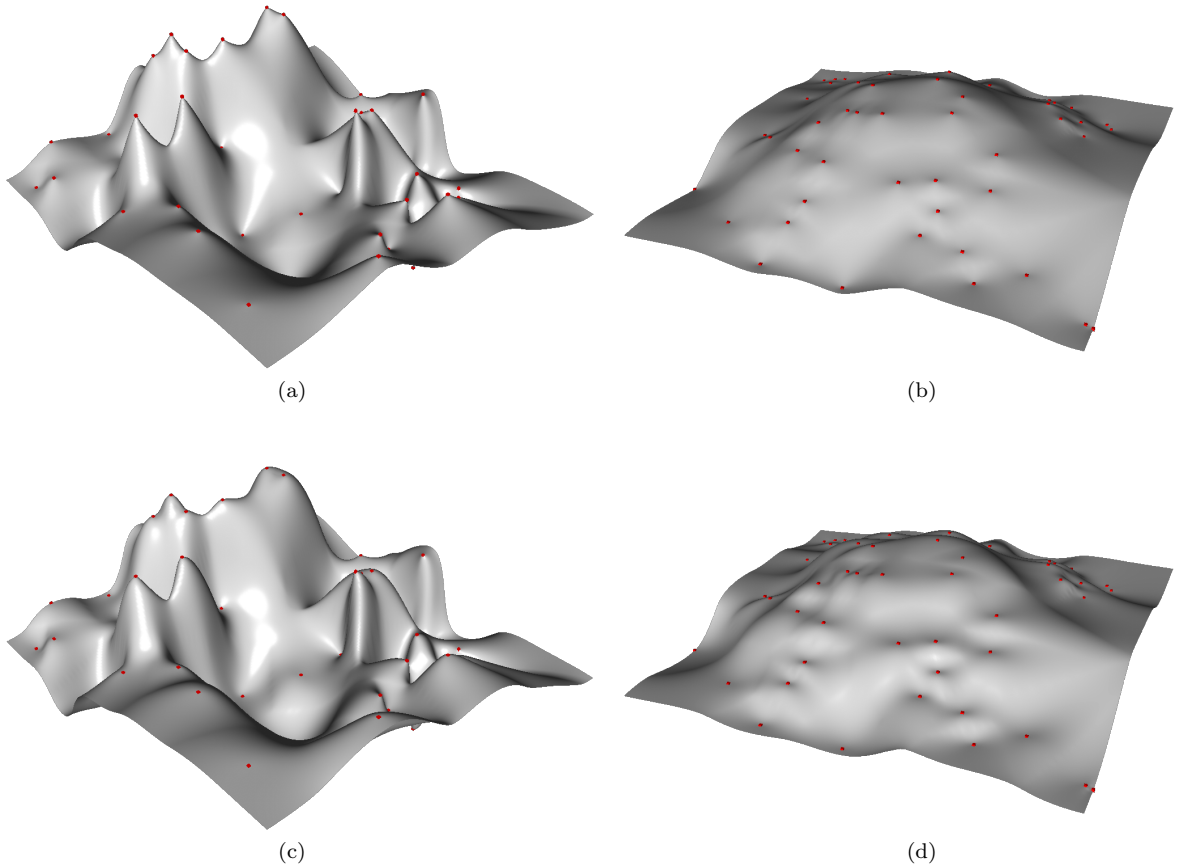


Figure 7. Blended neighbor of interpolation of scattered samples with equation 15 for $e = 4$ (a and b) and $e = 8$ (c and d).

An alternative blending equation that maintains linear precision while admitting RBF-like solutions is

$$q(\mathbf{x}) - \frac{1}{e} d^{2-e}(\mathbf{x}) \, \boldsymbol{\nabla} \cdot d^{e}(\mathbf{x}) \, \boldsymbol{\nabla} \, q(\mathbf{x}) = p(\mathbf{x}), \qquad (17)$$

for any constant $e \geq 2$. The special case $e = 2$ again corresponds to equation 12 above.

The blending equation 17 may be useful because, inside the Voronoi neighborhood of any known sample point $\mathbf{x}_k$, it has a solution of the form

$$q_k(\mathbf{x}) = f_k + \mathbf{g}_k^T(\mathbf{x} - \mathbf{x}_k) + h_k \|\mathbf{x} - \mathbf{x}_k\|^r, \qquad (18)$$

for some constants $\mathbf{g}_k$, h_k and r. Though not obvious, this solution may be easily verified by substitution into equation 17.

The constant exponent r in the radial term of equation 18 depends on the constant e in equation 17 and on the dimension n of points $\mathbf{x} \in \mathbb{R}^n$. For example,

$$\begin{aligned} &\text{1D:} \quad r = 1 \\ &\text{2D:} \quad r = \frac{1}{2}\left(\sqrt{e^2 + 4e} - e\right) \\ &\text{3D:} \quad r = \frac{1}{2}\left(\sqrt{e^2 + 6e + 1} - e - 1\right). \end{aligned} \qquad (19)$$

These exponents r are not the same as those typically used in RBF methods, and we should not expect blended neighbor interpolants to be as smooth as conventional RBF interpolants.

For all equations 19, $\lim_{e \to \infty} r = 1$, so that the radial term in equation 18 approaches the 3D RBF $\phi(\|\mathbf{x} - \mathbf{x}_k\|) = \|\mathbf{x} - \mathbf{x}_k\|$. This limit suggests that we might want to increase the constant e in equation 17 from $e = 2$ to $e = 4$ or $e = 8$. However, very large constants e reduce the accuracy of finite-difference approximations to equation 17. In the examples shown in this paper, I use only the simpler equation 15.

Recall that my goal in blended neighbor interpolation with equations 12, 15 or 17 is not to approximate RBF or biharmonic methods. Rather, I blend nearest neighbors using these partial differential equations because they (1) yield an interpolant much like the natural neighbor interpolant, (2) can be solved efficiently using a simple conjugate-gradient method, and (3) can be readily extended to interpolation in tensor fields derived from images.

2.3 Summary

Blended neighbor interpolation is a two-step process.

Blended neighbor interpolation

Step 1: solve

$$\begin{aligned} \boldsymbol{\nabla}\, d(\mathbf{x}) \cdot \boldsymbol{\nabla}\, d(\mathbf{x}) = 1, \quad & \mathbf{x} \notin \mathcal{X}; \\ d(\mathbf{x}) = 0, \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{20}$$

for

$d(\mathbf{x})$: the distance $\|\mathbf{x} - \mathbf{x}_k\|$ from $\mathbf{x}$ to the nearest known sample point $\mathbf{x}_k$, and
$p(\mathbf{x})$: the known value f_k corresponding to the sample point $\mathbf{x}_k$ nearest to the point $\mathbf{x}$.

Step 2: for a specified constant $e \geq 2$, solve

$$q(\mathbf{x}) - \frac{1}{e}\, \boldsymbol{\nabla} \cdot d^2(\mathbf{x})\, \boldsymbol{\nabla}\, q(\mathbf{x}) = p(\mathbf{x}), \tag{21}$$

for the blended neighbor interpolant $q(\mathbf{x})$.

When computing $d(\mathbf{x})$ in step 1, it is straightforward to simultaneously compute the nearest neighbor interpolant $p(\mathbf{x})$. Finite-difference approximation of the eikonal equation 20 is unnecessary, because efficient exact solutions are possible. Park et al. (2006), suggest using a kD tree for this purpose.

After computing $d(\mathbf{x})$ and $p(\mathbf{x})$ on a uniformly sampled grid, I use an iterative conjugate-gradient method to solve a finite-difference approximation of equation 21 for the blended neighbor interpolant $q(\mathbf{x})$.

3 BLENDING IN TENSOR FIELDS

In image-guided interpolation I replace Euclidean distance with time, so that "nearest" corresponds to paths of minimum time between points $\mathbf{x}$ and $\mathbf{x}_k$, not minimum distance $\|\mathbf{x} - \mathbf{x}_k\|$. The result is again a two-step process.

Image-guided blended neighbor interpolation

Step 1: solve

$$\begin{aligned} \boldsymbol{\nabla}\, t(\mathbf{x}) \cdot \mathbf{D}(\mathbf{x})\, \boldsymbol{\nabla}\, t(\mathbf{x}) = 1, \quad & \mathbf{x} \notin \mathcal{X}; \\ t(\mathbf{x}) = 0, \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{22}$$

for

$t(\mathbf{x})$: the minimum traveltime from $\mathbf{x}$ to the nearest known sample point $\mathbf{x}_k$, and
$p(\mathbf{x})$: the value f_k corresponding to the sample point $\mathbf{x}_k$ nearest to the point $\mathbf{x}$.

Step 2: for a specified constant $e \geq 2$, solve

$$q(\mathbf{x}) - \frac{1}{e}\, \boldsymbol{\nabla} \cdot t^2(\mathbf{x}) \mathbf{D}(\mathbf{x})\, \boldsymbol{\nabla}\, q(\mathbf{x}) = p(\mathbf{x}), \tag{23}$$

for the blended neighbor interpolant $q(\mathbf{x})$.

3.1 Computing the tensor field

The metric tensor field $\mathbf{D}(\mathbf{x})$ in equations 22 and 23 is the link between distance and time. It represents the coherence, orientation, and dimensionality of features in the image that will guide interpolation. Intuitively, this tensor field alters interpolation so that known sample values within spatially coherent image features are given more weight than values on opposite sides of such features or where the image is less coherent.

In some applications, a suitable $\mathbf{D}(\mathbf{x})$ is readily available. For example, to track white matter in diffusion-tensor magnetic-resonance images (DT-MRI), Jbabdi et al. (2008) choose $\mathbf{D}(\mathbf{x})$ to be simply the inverse of acquired tensor-valued images. For scalar-valued images, including most seismic images, $\mathbf{D}(\mathbf{x})$ may be computed from structure tensors $\mathbf{S}(\mathbf{x})$, which are smoothed outer products of gradient vectors (van Vliet and Verbeek, 1995).

In two dimensions, each tensor in the field $\mathbf{D}(\mathbf{x})$ is a 2×2 symmetric positive-definite matrix

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix}. \tag{24}$$

Equations 22 and 23 imply that the tensor elements d_{11}, d_{12}, and d_{22} have units of velocity squared.

Because the units of time t are arbitrary in equations 22 and 23, I scale the tensor field $\mathbf{D}(\mathbf{x})$ so that the maximum eigenvalue (maximum velocity squared) in any of these matrices is one. Eigenvalues less than one therefore imply slower velocities in directions of the corresponding eigenvectors.

In directions in which velocities are slow, two points that are nearby in the Euclidean distance map $d(\mathbf{x})$ may be far apart in the time map $t(\mathbf{x})$, the solution to equation 22 computed in step 1. Time, not distance, now determines which neighboring known sample points $\mathbf{x}_k$ are nearest in step 1, and the amount of blending of nearest neighbors performed in step 2.

Figure 8 shows examples of tensor fields $\mathbf{D}(\mathbf{x})$ computed from two different 2D scalar-valued seismic images to guide interpolation of scattered data. Each ellipse represents one of the symmetric positive-definite 2×2 tensors that I computed for every sample in these images.

For both images, I computed the displayed tensor fields $\mathbf{D}(\mathbf{x})$ from a structure tensor field $\mathbf{S}(\mathbf{x})$ by

$$\mathbf{D}(\mathbf{x}) = s \frac{\mathbf{S}^{-1}(\mathbf{x})}{1 - c(\mathbf{x})}, \tag{25}$$

where the constant scale factor s ensures that the maximum eigenvalue in $\mathbf{D}(\mathbf{x})$ is one. The function $c(\mathbf{x})$ is a measure of coherence computed from structure tensors $\mathbf{S}(\mathbf{x})$ using the method suggested by Fehmers and Höcker(2003).

Alternative measures of coherence (e.g., Bahorich and Farmer, 1995) in the range $0 \leq c(\mathbf{x}) < 1$ could be used instead. The significance of the divisor $1 - c(\mathbf{x})$ is

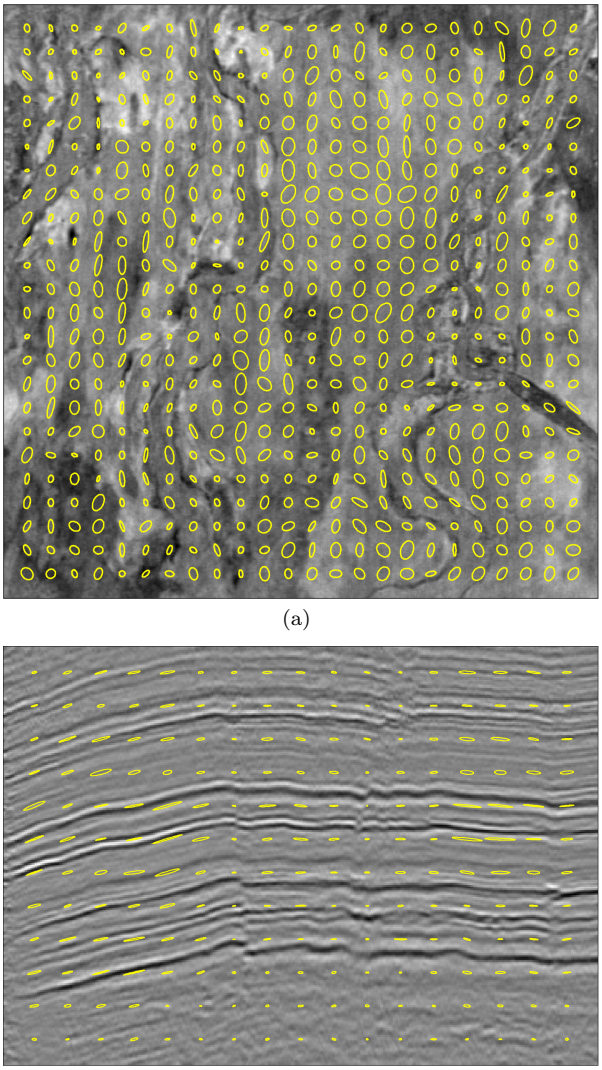


Figure 8. Tensor fields $\mathbf{D}(\mathbf{x})$ computed from 2D seismic images. The upper image of subsurface channels (a) was used to guide the interpolation in Figure 1. Geologic layers apparent in the lower image (b) guide the interpolation of scattered data in examples below.

that it increases eigenvalues of $\mathbf{D}(\mathbf{x})$, thereby decreasing times $t(\mathbf{x})$, at locations where features in images are most coherent. Locations and directions of high coherence correspond to large ellipses in Figure 8.

Eigenvalues and corresponding ellipses are smaller where features are less coherent; e.g., near discontinuities in laterally coherent features and in the noisy lower part of the image in Figure 8b.

3.2 Tensor-guided interpolation

Given a set $\mathcal{K}$ of scattered data, tensor fields $\mathbf{D}(\mathbf{x})$ like those displayed in Figure 8 guide blended neighbor interpolation with the two-step process of equations 22 and 23. Figure 9 illustrates this process with an example of interpolation guided by a seismic image.

In this example, 21 scattered samples in Figure 9a have colors corresponding to values that alternate vertically while decreasing from left to right. I painted these samples interactively using a digital 3×3-pixel paintbrush to make the sample points more clearly visible than if only one pixel per sample were painted. Such interactive painting can be useful in geologic interpretation of seismic images, either with or without additional scattered data obtained from borehole measurements.

I computed the time map $t(\mathbf{x})$ in Figure 9b by solving a finite-difference approximation to the eikonal equation 22 using an iterative algorithm similar to that proposed by Jeong et al. (2007). Other suitable algorithms include variants of fast-marching methods (Sethian, 1999; Sethian and Vladimirsky, 2005; Konukoglu et al., 2007) and sweeping methods (Tsai et al., 2003; Qian et al., 2007). Ridges in the time map $t(\mathbf{x})$ are aligned with the near-vertical faults, because times increase rapidly where the eigenvalues of the tensor field $\mathbf{D}(\mathbf{x})$ (the ellipses displayed in Figure 8b) are small.

While computing the time map $t(\mathbf{x})$, I also computed the nearest neighbor interpolant $p(\mathbf{x})$ displayed in Figure 9c. It may be possible to compute $p(\mathbf{x})$ directly from $t(\mathbf{x})$, but I have not found an efficient and stable way to do so. Instead, I compute $p(\mathbf{x})$ as I solve for times $t(\mathbf{x})$.

Specifically, I begin with a time map $t(\mathbf{x}) = \infty$. Then, for each known sample point $\mathbf{x}_k$ and value f_k, I use the eikonal equation 22 to compute times $t_k(\mathbf{x})$, while ignoring any other known sample points. Where $t_k(\mathbf{x}) < t(\mathbf{x})$, I update the time $t(\mathbf{x}) = t_k(\mathbf{x})$ and the value $p(\mathbf{x}) = f_k$. After all known samples have been processed, $t(\mathbf{x})$ is the time map and $p(\mathbf{x})$ is the nearest neighbor interpolant that we seek in step 1.

For efficiency in step 1, I processed the known samples in a random order and computed times $t_k(\mathbf{x})$ only where those times could possibly be lower than the minimum time maintained in the time map $t(\mathbf{x})$. For the first known sample, all times in $t(\mathbf{x})$ must be updated. But as more known samples are processed, the size of the region in which $t_k(\mathbf{x}) < t(\mathbf{x})$ decreases. Randomizing the order improves the rate of decrease in cases where the scattered data may be ordered in some way, as for data collected in boreholes.

The same randomizing technique is used in the classic quicksort algorithm for sorting a sequence of values (Cormen et al., 2001). For N values, the randomized quicksort algorithm has an $O(N^2)$ worst-case computational complexity, but for large N this worst case is highly unlikely. The average-case complexity for a randomized quicksort is $O(N \log N)$. Likewise, for K known samples and N image samples, the worst-case complex-

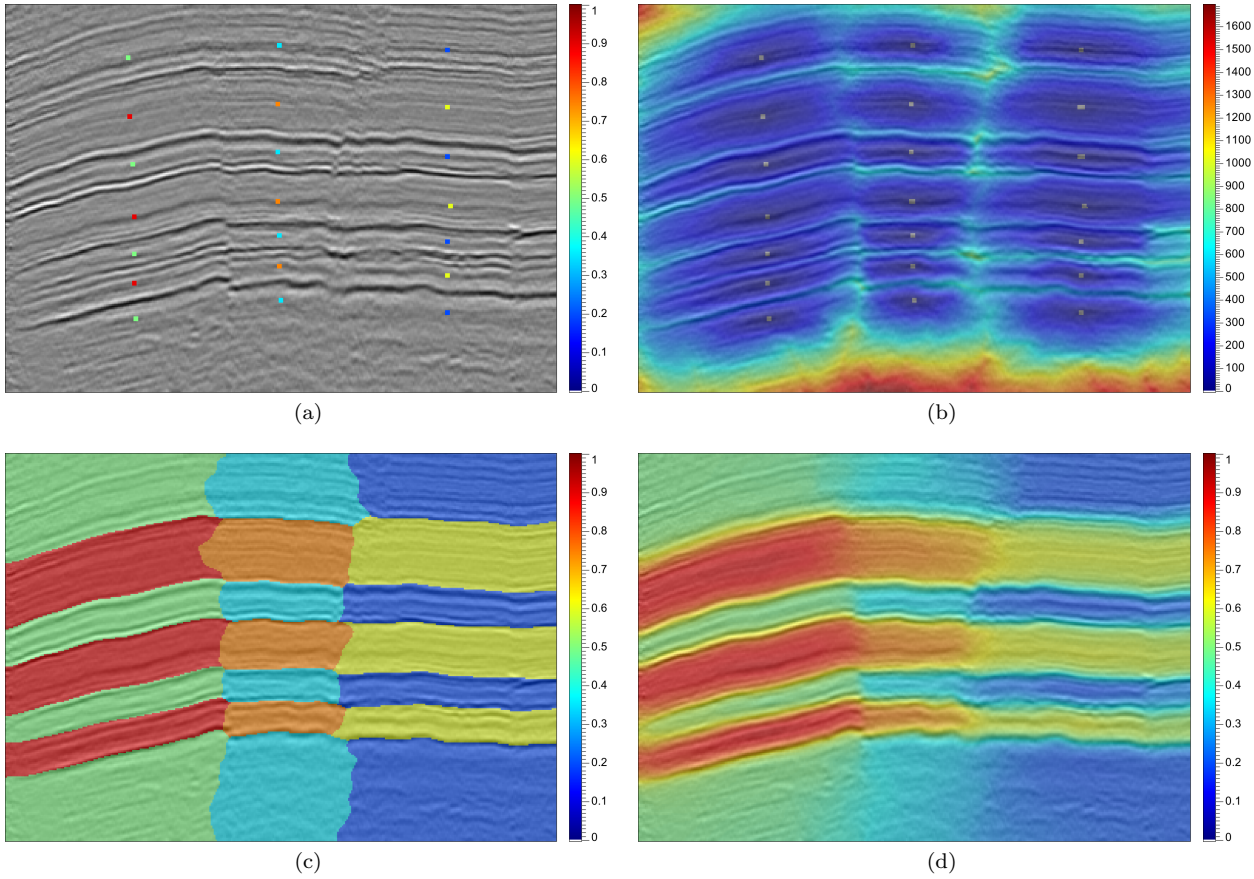


Figure 9. Image-guided blended neighbor interpolation. For a set $\mathcal{K}$ of known samples (a), we first use equation 22 to compute the time map (b) and nearest neighbor interpolant (c), and then solve equation 23 for the blended neighbor interpolant (d).

ity for my implementation of step 1 is $O(NK)$, but the average-case complexity is only $O(N \log K)$.

In step 2 I solved a finite-difference approximation to the blending equation 23 with $e = 4$ using an iterative conjugate-gradient solver. With $K = 21$ scattered samples and $N = 251 \times 357$ image samples, I performed 709 iterations to converge to the blended neighbor interpolant $q(\mathbf{x})$ displayed in Figure 9d. As expected, this blended neighbor interpolant conforms to features in the seismic image. Contours of constant color are aligned with both near-horizontal features (geologic layers) and near-vertical discontinuities (geologic faults).

To compare the image-guided process to one similar to a discrete natural neighbor interpolant (Park et al., 2006), I solved equations 20 and 21 with $e = 4$ for the same set of scattered data. In this isotropic and homogeneous case, 243 conjugate-gradient iterations were required to converge to the blended neighbor interpolant displayed in Figure 10.

As expected, without the tensor field $\mathbf{D}(\mathbf{x})$ to guide the interpolation, the blended neighbor interpolant is smooth between sample points but does not conform to image features. Both image-guided and image-ignorant interpolants honor the scattered data values, but the image-guided interpolant is more accurate where the values to be interpolated are correlated with image features.

4 DISCUSSION

Assuming that an image is available, the accuracy of image-guided interpolation depends on the extent to which the property being interpolated is correlated with image features. If no such correlation exists, then a simpler and faster image-ignorant interpolation may be more accurate than image-guided interpolation with an irrelevant image. In this case, blended neighbor interpolation is an efficient alternative to discrete natural neighbor interpolation.

In many contexts, however, an isotropic and constant tensor field is inappropriate. Even when a useful image is unavailable, it may still be possible to construct an anisotropic and inhomogeneous tensor field $\mathbf{D}(\mathbf{x})$ to guide interpolation. The proposed two-step process

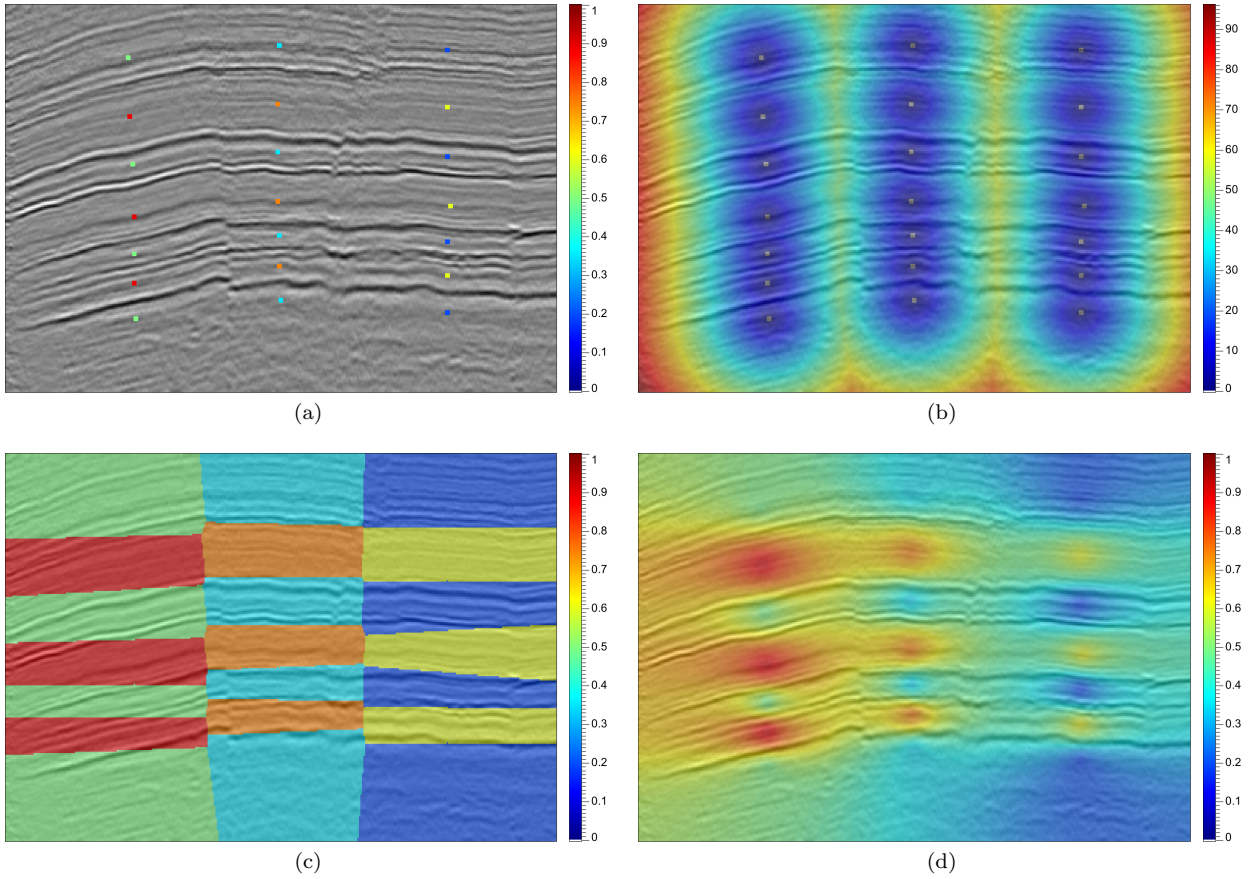


Figure 10. Image-ignorant blended neighbor interpolation. For a set $\mathcal{K}$ of known samples (a), we first use equation 20 to compute the distance map (b) and nearest neighbor interpolant (c), and then solve equation 21 for the blended neighbor interpolant (d).

might more accurately be called tensor-guided blended neighbor interpolation, because it requires only the tensor field $\mathbf{D}(\mathbf{x})$, not the image.

However, in the examples shown in this paper, I derived tensor fields from uniformly-sampled seismic images, and then interpolated scattered data on the same uniform sampling grid. In some applications, it may be desirable to interpolate scattered data with higher resolution, and nothing in the method prevents this. Image-guided blended neighbor interpolation requires only that we provide a tensor $\mathbf{D}$ for all uniformly sampled locations $\mathbf{x}$ where we interpolate.

The tensor fields used in image-guided blended neighbor interpolation are analogous to spatial correlation functions (variograms) used in kriging, a geostatistical interpolation method in which subsurface properties are modelled as random variables (e.g., Goovaerts, 1997). But the interpolation methods are otherwise rather different. Where images have less resolution than desired for geostatistical modeling, blended neighbor interpolation might be used to provide image-guided trends for kriging and other geostatistical methods.

In developing blended neighbor interpolation I retained most of the desirable features of natural neighbor interpolation, but I gave up locality. That is, the blended neighbor interpolant depends on all scattered known samples, even those that are very far way, as finite-difference approximations to the blending equations 21 or 23 yield a sparse system of linear equations that we must solve simultaneously. In practice, the neighborhood in blended neighbor interpolation is quite limited and errors in assuming a local region of influence for each known sample may be less than those due to the use of an iterative solver for the system of blending equations.

But instead of solving the blending equation 23, why not extend the strictly local scattering method of discrete natural neighbor interpolation to handle tensor fields? The computational cost of such an extension would be high. In tensor fields the simple scattering disc of Figure 5b becomes an irregularly shaped region that

must be computed numerically by solving an eikonal equation 22 for every image sample. And even supposing that this region could be computed quickly for every sample, the computational cost of scattering remains proportional to the areas (or volumes) of such regions in 2D (or 3D).

In contrast, the cost of solving the blending equations with the simplest conjugate-gradient method grows only linearly with distances or times to nearest known samples. The number of iterations required for such an iterative solver to converge depends in part on the accuracy required in the blended neighbor interpolant. The number of iterations might be reduced by the use of preconditioners, including multigrid methods, but in my experience these techniques have yielded only moderate improvements in efficiency when tensor fields are as inhomogeneous and anisotropic as those displayed in Figure 8.

Closely related to the problem of image-guided interpolation are the problems of computing geodesic distances and interpolation on surfaces embedded in a 3D space (Kimmel and Sethian, 1998; Boissonnat and Flötotto, 2004; Surazhasky et al., 2005) or manifolds in higher dimensions (Bronstein et al., 2007). For example, when interpolating scattered geophysical data acquired on a global scale, one might use non-Euclidean distances measured on the earth's surface. Surfaces on which geodesic distances are computed correspond to metric tensor fields, but tensor fields computed as in equation 25 need not correspond to any surface. Nevertheless, improved algorithms for computing geodesic distances may lead to better algorithms for image-guided interpolation.

5 CONCLUSION

Blended neighbor interpolation of scattered data is similar to the classic method of natural neighbor interpolation, in that both methods smooth a nearest neighbor interpolant, and the extent of smoothing grows with distance to the nearest known sample point.

The interpolants are similar but not identical, and the difference between the two methods lies in their smoothing filters. In blended neighbor interpolation a smoothing filter is implied by the solution of a partial differential equation. In natural neighbor interpolation the smoothing filter explicitly computes weighted sums of nearest neighbor sample values.

When Euclidean distances are used, the weights in natural neighbor interpolation are simply the areas of polygons, and can be computed efficiently with suitable data structures. However, in non-Euclidean metric tensor fields, these areas must be computed numerically, and for this case blended neighbor interpolation is an efficient alternative to natural neighbor interpolation.

In image-guided interpolation we derive metric tensor fields from images, so that the blended neighbor interpolant conforms to image features, while retaining many of the attractive features of the natural neighbor interpolant.

ACKNOWLEDGMENTS

Thanks to Luming Liang for many thoughtful discussions of the ideas presented in this paper, and to Myoung Jae Kwon, Ken Larner, and Derek Parks for helpful reviews. WesternGeco provided the seismic data displayed in Figures 1 and 8a. The U.S. Department of Energy provided the seismic data displayed in Figures 8b, 9 and 10.

REFERENCES

Bahorich, M.S., and S.L. Farmer, 1995, 3-D seismic coherency for faults and stratigraphic features: The coherence cube: The Leading Edge, **14** 1053–1058.

Bobach, T., G. Farin, D. Hansford and G. Umlauf, 2008, Natural neighbor extrapolation using ghost points: Computer-Aided Design, in press.

Boissonnat, J.-D., and J. Flötotto, 2004, A coordinate system associated with points scattered on a surface: Computer-Aided Design, **36**, 161–174.

Bronstein, A.M., M.M. Bronstein and R. Kimmel, 2007, Weighted distance maps computation on parametric three-dimensional manifolds: Journal of Computational Physics, **225**, 771–784.

Carr, J.C., W.R. Fright and R.K. Beatson, 1997, Surface interpolation with radial basis functions for medical imaging: IEEE Transactions on Medical Imaging, **16**, 96–107.

Cormen, T.H., C.E. Leiserson and R.L. Rivest, 2001, Introduction to algorithms, second edition: MIT Press.

Duchon, J., 1977, Splines minimizing rotation-invariant seminorms in Sobolev spaces, *in* W. Schempp and K. Zeller, eds., Constructive Theory of Functions of Several Variables, Lecture Notes in Mathematics 571, Springer-Verlag, 85–100.

Dyn, N., 1987, Interpolation of scattered data by radial functions, *in* C.K. Chui and L.L. Schumaker and F. Utreras, eds., Topics in Multivariate Approximation: Academic Press, 47–61.

Farin, G., 1990, Surfaces over Direchlet tesselations: Computer Aided Geometric Design, **7**, 281–292.

Fehmers, G.C., C.F.W. Höcker, 2003, Fast structural interpretations with structure-oriented filtering: Geophysics, **68**, 1286–1293.

Foster, M.P., and A.N. Evans, 2008, An evaluation of interpolation techniques for reconstructing ionospheric TEC maps: IEEE Transactions on Geoscience and Remote Sensing, **46**, 2153–2164.

Franke, R., 1982, Smooth interpolation of scattered data by local thin plate splines: Computers & Mathematics with Applications, **8**, 237-281.

Gáspár, C., 1999, Multigrid technique for biharmonic interpolation with application to dual and multiple reciprocity method: Numerical Algorithms, **21**, 165–183.

Goovaerts, P., 1997, Geostatistics for natural resources evaluation: Oxford University Press.

Jbabdi, S., P. Bellec, R. Toro, J. Daunizeau, M. Pélégrini-Issac and H. Benali, 2008, Accurate anisotropic fast marching for diffusion-based geodesic tractography: International Journal of Biomedical Imaging, **2008**, 1–12.

Jeong, W.-K., P.T. Fletcher, R. Tao and R.T. Whitaker, 2007, Interactive visualization of volumetric white matter connectivity in DT-MRI using a parallel-hardware Hamilton-Jacobi solver: IEEE Transactions on Visualization and Computer Graphics, **13**, 1480–1487.

Kimmel, R., and J.A. Sethian, 1998, Computing geodesic paths on manifolds: Proceedings of the National Academy of Sciences, **95**, 8431–8435.

Konukoglu, E., M. Sermesant, O. Clatz, J.-M. Perat, H. Delingette and N. Ayache, 2007, A recursive anisotropic fast marching approach to reaction diffusion equation: application to tumor growth modeling: Information processing in medical imaging, **20**, 687–699.

Mitášová, H. and M. Luboš, 1993, Interpolation by regularized spline with tension: I. theory and implementation: Mathematical Geology, **25**, 641–655.

Park, S.W., L. Linsen, O. Kreylos, J.D. Owens, B. Hamann, 2006, Discrete Sibson interpolation: IEEE Transactions on Visualization and Computer Graphics, **12**, 243–253.

Qian, J., Y.-T.. Zhang, and H.-K. Zhao, 2007, A fast sweeping method for static convex Hamilton-Jacobi equations: Journal of Scientific Computing, **31**, 237–271.

Sandwell, D.T., 1987, Biharmonic spline interpolation of GEOS-3 and SEASAT altimeter data: Geophysical Research Letters, **14**, 139–142.

Schaback, R., 2005, Multivariate interpolation by polynomials and radial basis functions: Constructive Approximation, **21**, 293–317.

Sethian, J.A, 1999, Fast marching methods: SIAM Review, **41**, 199–235.

Sethian, J.A, and A. Vladimirsky, 2005, Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms: SIAM Journal of Numerical Analysis, **41**, 325–363.

Sibson, R., 1981, A brief description of natural neighbor interpolation, *in* V. Barnett, ed., Interpreting Multivariate Data: John Wiley & Sons, 21–36.

Surazhsky, V., T. Surazhsky, D. Kirsanov, S.J. Gortler, and H. Hoppe, 2005, Fast exact and approximate geodesics on meshes, *in* J. Marks, ed., ACM SIGGRAPH 2005 Papers: ACM, 553–560.

Tsai, Y.-H.R., L.-T. Cheng, S. Osher and H.-K. Zhao, 2003, Fast sweeping algorithms for a class of Hamilton-Jacobi equations: SIAM Journal of Numerical Analysis, **41**, 673–694.

van Vliet, L.J., and P.W. Verbeek, 1995, Estimators for orientation and anisotropy in digitized images: Proceedings of the first annual conference of the Advanced School for Computing and Imaging ASCI'95, Heijen (The Netherlands), 442–450.