Journal of Electronic Imaging 11(2), 157–176 (April 2002).

Scattered data interpolation methods for electronic imaging systems: a survey

Isaac Amidror
Laboratoire de Systèmes Périphériques
Ecole Polytechnique Fédérale de Lausanne
Lausanne, Switzerland

Abstract. *Numerous problems in electronic imaging systems involve the need to interpolate from irregularly spaced data. One example is the calibration of color input/output devices with respect to a common intermediate objective color space, such as XYZ or $L^*a^*b^*$. In the present report we survey some of the most important methods of scattered data interpolation in two-dimensional and in three-dimensional spaces. We review both single-valued cases, where the underlying function has the form $f:\mathbb{R}^2\rightarrow\mathbb{R}$ or $f:\mathbb{R}^3\rightarrow\mathbb{R}$, and multivalued cases, where the underlying function is $\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$ or $\mathbf{f}:\mathbb{R}^3\rightarrow\mathbb{R}^3$. The main methods we review include linear triangular (or tetrahedral) interpolation, cubic triangular (Clough–Tocher) interpolation, triangle based blending interpolation, inverse distance weighted methods, radial basis function methods, and natural neighbor interpolation methods. We also review one method of scattered data fitting, as an illustration to the basic differences between scattered data interpolation and scattered data fitting.* © 2002 SPIE and IS&T. [DOI: 10.1117/1.1455013]

1 Introduction

The problem of scattered data interpolation consists of constructing a continuous function of two, three, or more independent variables that interpolates data values which are only known at some scattered points in the two-dimensional (2D) plane [or, respectively, in the three-dimensional (3D) or *N*D space]. The need for interpolation from irregularly spaced data occurs in many different fields, such as medical imaging, meteorological or geological modeling, cartography, and computer aided geometric design. For example, in meteorology weather measurements are available from irregularly located observation stations; and in geostatistics and geology aquifer properties and layer structures are studied from data that is only available at a few given locations. The data at these few locations must be interpolated to the nodes of an underlying grid in order to allow the use of 2D or 3D scientific visualization tools to illustrate the variation in the data. Interpolation is also required for the construction of isolines or contour maps based on the available data.

These examples concern the interpolation of single-valued data, where the underlying function has the form $f:\mathbb{R}^2\rightarrow\mathbb{R}$ or $f:\mathbb{R}^3\rightarrow\mathbb{R}$. But other cases may require the interpolation of multivalued data, where the underlying function is of the form $\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$ or $\mathbf{f}:\mathbb{R}^3\rightarrow\mathbb{R}^3$. As an illustration to the case of $\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$, consider the geometric correction of aerial or satellite views by "landmark based morphing."[1] Airborne and satellite images are often distorted because of the earth curvature and the oblique viewing angles. In this case we are given an original, distorted image that includes a few identifiable ground control points (pillars, road junctions, etc.). For each of these scattered control points we know both the distorted (x,y) coordinates in the image, and the real (u,v) coordinates on a geographic map. Our task is, therefore, to interpolate between these few known points in order to obtain the underlying geometric correction transformation $\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$, that will allow us later to obtain the rectified image by resampling. Similar cases occur also in medical imaging, where it is required to match images obtained from a patient at different times or with different imaging techniques, or to compare them with a standard anatomical data set. In all of these bivariate two-valued cases ($\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$) we actually need to solve two instances of the bivariate single-valued interpolation problem ($f:\mathbb{R}^2\rightarrow\mathbb{R}$), one for each of the two destination coordinates.

Another example, in the field of electronic color reproduction systems, illustrates the case of $\mathbf{f}:\mathbb{R}^3\rightarrow\mathbb{R}^3$. This example concerns the calibration of color input/output devices (such as a scanner and a printer) with respect to a common intermediate objective color space (such as *XYZ* or $L^*a^*b^*$). In such cases the input device calibration is often based on establishing a mapping between the 3D device-dependent *RGB* color space of the input device and the chosen 3D objective color space (say, the *XYZ* space).[2] This is done by using color patches from a standard color catalog, such as the Pantone catalog:[3] each color patch is fed to the input device to obtain its device *RGB* values, and then measured by a spectrophotometer to obtain its *XYZ* values. This gives us a set of several hundred points that are scattered within the 3D input *RGB* space, to each of which there are associated three scalar values—the coordinates of the same patch in the *XYZ* color space. In this case the value at each point of the scattered point set is itself a 3D

Paper 20029 received Apr. 5, 2000; revised manuscript received Oct. 10, 2000; accepted for publication Oct. 2, 2001.
1017-9909/2002/$15.00 © 2002 SPIE and IS&T.

quantity, so that we actually need to solve three instances of the single-valued problem—one for each of the *XYZ* components.

In the present survey we will mainly concentrate on the basic problem of scattered data interpolation for single-valued, scalar quantities [such as the geographic elevation over irregularly spaced points (x_i, y_i) in the plane, or the temperature at scattered points (x_i, y_i, z_i) in space]. However, if an interesting vector or matrix expression is available for the multivalued case, we will also mention it.

The problem of scattered data interpolation in two or more independent variables has been addressed in many papers and book chapters that are dispersed in various scientific disciplines. And indeed, numerous methods with many variants have been devised to solve this problem.[4–9] Our aim in the present paper is to briefly survey those methods which are most relevant for the needs of electronic imaging systems, and to provide useful references with more detailed accounts on these methods, for the benefit of the interested readers. It should be noted that the concepts involved in scattered data interpolation are largely inspired from the fundamental concepts in the interpolation of regularly spaced data. Readers who desire a short introduction on classical interpolation methods in this elementary case may consult, for example, Chap. 11 in Ref. 10 or Refs. 11–13.

The different approaches to the interpolation of scattered data can be classified into global methods, in which each interpolated value is influenced by all of the data, and local methods, in which the interpolated value is only influenced by the values at ''nearby'' points from the scattered point set. Global methods are practically limited to small data sets due to the computational efforts they require; moreover, an addition or deletion of a data point, or a correction in any of the coordinates of a data point, will modify the interpolated values throughout the entire domain of definition. Local methods, on the other hand, are capable of treating much larger data sets, and they are less sensitive to data modifications, but they may become quite complex, too, if a smooth result is required.

The methods surveyed in the present paper will be classified into the following categories:

1. triangulation (or tetrahedrization) based methods (Sec. 2);
2. inverse distance weighted methods (Sec. 3);
3. radial basis function methods (Sec. 4); and
4. natural neighbor methods (Sec. 5).

Finally, in Sec. 6, we will review one method of scattered data fitting, the regression method, as an illustration to the basic difference between scattered data interpolation methods and scattered data fitting methods: While interpolation schemes construct functions that pass through the given data points, data fitting schemes produce functions that maintain the nature and the trends of the input data, but only pass near the data points. The main criteria used for comparing the different methods include smoothness (derivative continuity: C^0, C^1), possibility of extrapolation, local versus global technique, accuracy and fitting ability, performance (efficiency, speed), suitability for large point sets, extension to three or more dimensions, the need for a preprocessing step (triangulation, Voronoi tesselation, etc.), simplicity of use, particular artifacts known, and various method-family specific criteria.

Notations: Except where indicated otherwise, the following notations will be adopted throughout this review: We assume that we are given a set of n distinct points $P_1, \ldots, P_n$ that are scattered in an N-dimensional Euclidean space (usually $N=2$ or 3). Each point P_i is located at the vector position $\mathbf{x}_i$, and has a numerical value z_i. Our task (in the single-valued case) is to find good interpolation functions $\hat{f}(\mathbf{x})$ such that $z_i = \hat{f}(\mathbf{x}_i)$ for all $i=1,...,n$.

2 Triangulation (or Tetrahedrization) Based Methods

The interpolation methods belonging to this category operate in two steps: First, the scattered point set is triangulated (in the 2D case) or tetrahedrized (in the 3D case); and then, an interpolation scheme is used within each triangle (or tetrahedron). These methods are therefore always local. A good source of information on triangulation, tetrahedrization and their derived scattered data interpolation methods is Chap. 20 in Ref. 10.

2.1 *The Triangulation (or Tetrahedrization) of a Scattered Point Set*

Let us start first with the 2D case. Clearly, a given point set in the plane has many different triangulations. It may be therefore desirable to find among these triangulations an optimal one which avoids as much as possible poorly shaped triangles (such as thin and elongated triangles; see Ref. 10, p. 433 and Fig. 20.19 there). [We say ''as much as possible'' because this is not always possible, especially in the vicinity of the external boundaries of the given point set (Ref. 8, p. 135).] One of the ''nicer'' candidates is the Delaunay triangulation, i.e., the triangulation obtained by connecting all the neighboring points in the Voronoi diagram of the given point set [Ref. 14, pp. 186–187; Ref. 15, p. 28]. [Note that the Delaunay triangulation is not necessarily unique. For example, in the case of four points forming a square either diagonal produces a valid Delaunay triangulation. Such points are always cocircular (Ref. 15, p. 28).] Intuitively, we may say that a triangulation is ''nice'' if it consists of triangles that are close to being equilateral. Therefore, we may compare different triangulations of the given point set by finding the minimal angle of each triangle: the triangulation that has the largest minimal angle would be considered the better one (this is called the ''maxmin'' angle criterion). And it turns out that of all possible triangulations, the Delaunay triangulation is the one that is guaranteed to produce the largest minimal angle (Ref. 14, pp. 188–189; Ref. 15, p. 29). [Note, however, that it is not guaranteed that a Delaunay triangulation contains no small angles. A simple example of a Delaunay triangulation that necessarily contains very small angles is shown in (Ref. 14, p. 291).] Furthermore, the Delaunay triangulation is the only one having the property that the circle circumscribing any of its triangles contains no other point of the scattered data set, and hence, no other triangle (Ref. 14, p. 188; Ref. 10, pp. 444–446). There exist several different techniques for constructing a Delaunay triangulation of a given scattered point set in the plane. An overview on these algorithms is given in Ref. 10, pp. 448–453.

A similar situation exists also in the 3D case: Since there may be many different tetrahedrizations for any scattered point set in space, it may be desirable to find among them an optimal tetrahedrization that avoids as much as possible poorly shaped tetrahedra (see Ref. 10, p. 489 and Fig. 20.62 there). Here, too, the Delaunay tetrahedrization has some particularly nice properties. For example, only a Delaunay tetrahedrization ensures that the enclosing sphere of any of its tetrahedra contains no other point of the scattered data set (and hence, no other tetrahedra).[16] In practice this implies that most tetrahedra are neither ‘‘too flat’’ nor ‘‘too long,’’ so that the volume on which the tetrahedral interpolation will be performed has a reasonable shape. In the 3D case, too, there exist several techniques for constructing the Delaunay tetrahedrization of a given scattered point set; an overview on these algorithms is given in Ref. 10, pp. 490–491. Optimal tetrahedrizations that are based on other optimization criteria than the Delaunay method are briefly reviewed in Ref. 17, pp. 12–13.

Once the given scattered point set has been triangulated (or tetrahedrized), we still need a procedure that finds in which triangle (or tetrahedron) an arbitrary new point P is located, in order to be able to find its value by interpolation within that triangle (or tetrahedron). This can be done by the following search procedure (described here for the 3D case):

- Let G_i be a point belonging to a known tetrahedron T.
- P belongs to the same tetrahedron if there is no intersection between segment G_iP and any surface of T. In this case we have found the tetrahedron to which P belongs.
- If an intersection with one of the tetrahedron's surfaces exists, a new point G_{i+1} is chosen in the tetrahedron adjacent to T, which shares with T its intersected surface. The procedure is repeated until the correct tetrahedron is found.

Another possible procedure is mentioned in Ref. 18, p. 56.

Having described how to find the triangle (or tetrahedron) in which is located an arbitrary point P, we can proceed now to the different interpolation methods inside this triangle (tetrahedron).

2.2 Linear Triangular (or Tetrahedral) Interpolation

Let us start with the 2D case, where the scattered points (x_i, y_i) are located in the x,y plane; their values can be visualized as altitudes z_i over this plane. Therefore, the triangulation of the scattered points in the x,y plane induces a piecewise triangular surface over the plane, whose nodes are the points (x_i, y_i, z_i). This is a continuous surface made up of flat triangular pieces that are joined along edges (which is an obvious generalization of broken line functions in one dimension). Such a surface is often called a triangulated irregular network. This surface represents, therefore, a piecewise interpolation scheme in which a bivariate linear interpolation* is applied within each triangle. Let points P_1, P_2, and P_3, located at $\mathbf{x}_1=(x_1,y_1)$, $\mathbf{x}_2=(x_2,y_2)$ and $\mathbf{x}_3=(x_3,y_3)$, be the three vertices of any of the underlying triangles in the x,y plane, and let their values be z_1, z_2, and z_3. Then, the value z at any arbitrary point P located at $\mathbf{x}=(x,y)$ within this triangle can be found as follows.

Suppose that the plane which is defined by the three points (x_1,y_1,z_1), (x_2,y_2,z_2), and (x_3,y_3,z_3) is given by

$$z=ax+by+c. \tag{1}$$

By inserting the x,y,z values of each of the three known points into this equation we obtain a linear system of equations

$$z_1=ax_1+by_1+c,$$

$$z_2=ax_2+by_2+c,$$

$$z_3=ax_3+by_3+c.$$

The coefficients a,b,c of the plane of Eq. (1) can be found, therefore, by solving this system of equations. Once these coefficients are known, Eq. (1) gives us the interpolated value z for any point P located at $\mathbf{x}=(x,y)$ within this triangle.

Alternatively, the value z at any arbitrary point P within the triangle can be also found by *barycentric interpolation*: Let P_1, P_2, and P_3 be the three vertices of the triangle, located at $\mathbf{x}_1=(x_1,y_1)$, $\mathbf{x}_2=(x_2,y_2)$ and $\mathbf{x}_3=(x_3,y_3)$. Then, the location $\mathbf{x}=(x,y)$ of any point P in the x,y plane can be uniquely expressed as a weighted average of the locations of these three vertices

$$\mathbf{x}=a_1\mathbf{x}_1+a_2\mathbf{x}_2+a_3\mathbf{x}_3, \tag{2}$$

where $a_1+a_2+a_3=1$. (Note that if the point P is situated inside the triangle, then we also have $a_i>0$, $i=1,2,3$.) The weights a_1, a_2, a_3 are called the *barycentric coordinates* of point P (Ref. 15, pp. 23–26). [This term is due to the physical interpretation of the point P as the center of gravity of the triangle $P_1P_2P_3$ when weights a_i are attached to its vertices P_i (barys means in Greek ‘‘heavy,’’ hence, the term barycenter: center of gravity).]

Now, for any linear (or affine) function g of $\mathbf{x}=(x,y)$ we clearly have from Eq. (2):

$$g(\mathbf{x})=a_1g(\mathbf{x}_1)+a_2g(\mathbf{x}_2)+a_3g(\mathbf{x}_3)$$

*We prefer to avoid the popular terms ‘‘bilinear interpolation’’ and ‘‘trilinear interpolation’’ because of their ambiguity. In algebra, a bilinear function is a function that is linear in each of its two variables separately, such as the function $f(x,y)=xy$. This function is linear in x for any fixed y_0, and linear in y for any fixed x_0; but obviously, the surface defined by this function is *not* a plane, but a hyperbolic paraboloid (see, for example, the section on bilinear interpolation in Ref. 15, pp. 231–233). Therefore, we call the 2D linear function that represents a plane, $f(x,y)=ax+by$, a ‘‘bivariate linear function.’’

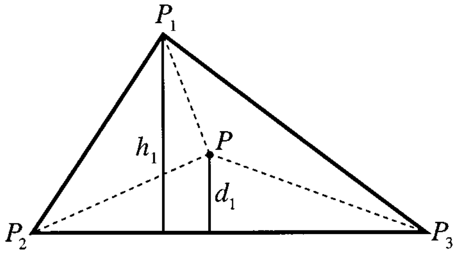


Fig. 1 Triangle $P_1P_2P_3$ and the interpolation point P inside it. By connecting point P to each of the vertices P_1, P_2, and P_3 we obtain a subdivision of the triangle into three subtriangles, PP_2P_3, P_1PP_3 and P_1P_2P. The distances h_i and d_i are shown for $i=1$.

and in particular, since our interpolating function $z=\hat{f}(x,y)$ is indeed linear, we obtain that the value z at the point P, i.e., at the location $\mathbf{x}=(x,y)$, is simply the weighted average of the values z_i at the locations of the vertices P_i, $\mathbf{x}_i=(x_i,y_i)$:

$$z=a_1z_1+a_2z_2+a_3z_3 \tag{3}$$

with the *same* weights a_1,a_2,a_3 as in Eq. (2), and $a_1+a_2+a_3=1$. Since the locations of the points P, P_1, P_2, P_3 are known the weights a_1,a_2,a_3 can be found by solving the linear system of equations defined by Eq. (2) for a_1, a_2, and a_3:

$$a_1x_1+a_2x_2+a_3x_3=x,$$

$$a_1y_1+a_2y_2+a_3y_3=y,$$

$$a_1+a_2+a_3=1.$$

The solution of these equations is

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4}$$

or, more explicitly, using the Cramer rule

$$a_1=D_1/D, \quad a_2=D_2/D, \quad a_3=D_3/D, \tag{5}$$

where

$$D=\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}, \quad D_1=\begin{vmatrix} x & x_2 & x_3 \\ y & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix},$$

$$D_2=\begin{vmatrix} x_1 & x & x_3 \\ y_1 & y & y_3 \\ 1 & 1 & 1 \end{vmatrix}, \quad D_3=\begin{vmatrix} x_1 & x_2 & x \\ y_1 & y_2 & y \\ 1 & 1 & 1 \end{vmatrix}.$$

Note that the weights a_i can be interpreted geometrically as area ratios $a_i=A_i/A$, where A is the area of the triangle $P_1P_2P_3$ and A_i is the area of the subtriangle obtained when the interpolation point P is substituted for the vertex P_i (see Fig. 1): $A=\text{area}(P_1P_2P_3)$, $A_1=\text{area}(PP_2P_3)$, $A_2=\text{area}(P_1PP_3)$, $A_3=\text{area}(P_1P_2P)$ (Ref. 15, p. 24). Moreover, since both of the triangles in each ratio a_i have the same basis, a_i equals also the ratio between the heights of these triangles. In other words, if h_i is the distance of vertex P_i from the opposite triangle edge and d_i is the distance of the interpolation point P from that edge, then $a_i=d_i/h_i$.

It is important to note that in spite of the form difference between the results obtained by Eq. (1) and those obtained by barycentric coordinates [Eqs. (3) and (5)], in fact, the linear interpolation values obtained by both methods are identical: since the bivariate linear polynomial passing through three distinct given points in space is *unique*.

Let us proceed now to the 3D case. Here, the scattered points (x_i,y_i,z_i) are located in the 3D space, and their values can no longer be visualized along an additional dimension. A useful intuition for illustrating this case could be the temperature t_i measured at each of the scattered points (x_i,y_i,z_i) in the 3D space. In this case the scattered points (x_i,y_i,z_i) are first tetrahedrized, and then a linear interpolation is applied within each tetrahedron.

Given the four vertices P_1,P_2,P_3,P_4 of any such tetrahedron, namely: the locations (x_i,y_i,z_i) of its four vertices and their values t_i, the value t at any arbitrary point P located at $\mathbf{x}=(x,y,z)$ within this tetrahedron can be found by writing the trivariate linear interpolation polynomial

$$t=ax+by+cz+d \tag{6}$$

and finding its coefficients a,b,c,d by solving the linear system of equations:

$$t_1=ax_1+by_1+cz_1+d,$$

$$t_2=ax_2+by_2+cz_2+d,$$

$$t_3=ax_3+by_3+cz_3+d,$$

$$t_4=ax_4+by_4+cz_4+d.$$

Alternatively, the value t at any point P located at $\mathbf{x}=(x,y,z)$ within the tetrahedron can be found by *barycentric interpolation* as a weighted average of the values t_i:

$$t=a_1t_1+a_2t_2+a_3t_3+a_4t_4, \tag{7}$$

where the weights a_1, a_2, a_3, a_4 are the *barycentric coordinates* of the point P in the tetrahedron $P_1P_2P_3P_4$ whose vertice locations are $\mathbf{x}_1,\ldots,\mathbf{x}_4$, namely

$$\mathbf{x}=a_1\mathbf{x}_1+a_2\mathbf{x}_2+a_3\mathbf{x}_3+a_4\mathbf{x}_4 \tag{8}$$

with $a_1+a_2+a_3+a_4=1$. Like in the 2D case, it can be shown that a_1, a_2, a_3, a_4 are given by

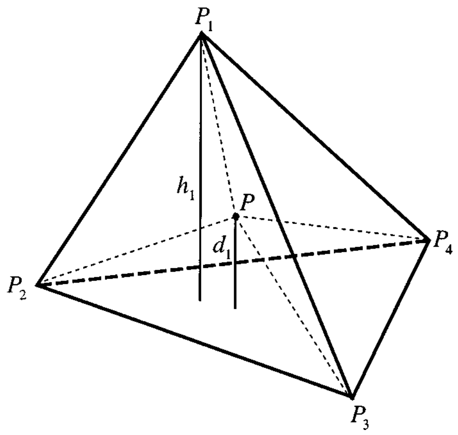


Fig. 2 Tetrahedron $P_1P_2P_3P_4$ and the interpolation point P inside it. By connecting point P to each of the vertices P_1, P_2, P_3, and P_4 we obtain a subdivision of the tetrahedron into four subtetrahedra, $PP_2P_3P_4$, $P_1PP_3P_4$, $P_1P_2PP_4$, and $P_1P_2P_3P$. The distances h_i and d_i are shown for $i=1$.

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{9}$$

or, more explicitly, by

$$a_1=D_1/D, \quad a_2=D_2/D, \quad a_3=D_3/D, \quad a_4=D_4/D, \tag{10}$$

where D, D_1, D_2, D_3, and D_4 are straightforward generalizations of their 2D counterparts given in Eq. (5).

Note that the weights a_i can be interpreted geometrically as volume ratios $a_i=V_i/V$, where V is the volume of the tetrahedron $P_1P_2P_3P_4$ and V_i is the volume of the subtetrahedron obtained when the interpolation point P is substituted for the vertex P_i (see Fig. 2): $V=\text{volume}(P_1P_2P_3P_4)$, $V_1=\text{volume}(PP_2P_3P_4)$, $V_2=\text{volume}(P_1PP_3P_4)$, $V_3=\text{volume}(P_1P_2PP_4)$, and $V_4=\text{volume}(P_1P_2P_3P)$. Moreover, since both of the tetrahedra in each ratio a_i have the same basis, a_i equals also the ratio between the heights of these tetrahedra. In other words, if h_i is the distance of vertex P_i from the opposite tetrahedron face and d_i is the distance of the interpolation point P from that face, then $a_i=d_i/h_i$.

Once again, in spite of the form difference between the results obtained by Eq. (6) and those obtained by barycentric coordinates [Eqs. (7) and (10)], the linear interpolation is unique, and the interpolated values obtained by the different formulas are identical.

2.3 *Linear Multivalued Triangular (or Tetrahedral) Interpolation*

Equations (6) or (7) represent a *single-valued* tetrahedral interpolation. For a three-valued case such as the *RGB* to *XYZ* conversion, where to each point in the scattered point set within the *RGB* space there correspond three values in the *XYZ* space, one may use Eq. (7) three times—once for each of the *XYZ* coordinates (using each time the same weights a_i).

However, the linear three-valued tetrahedral interpolation can be performed also in a different way, which is not based on barycentric coordinates, and which can be expressed in an elegant and compact matrix form. Let us first present this method in the 2D case, i.e., for the interpolation of a 2D mapping $\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$ from the x,y plane into the destination u, v plane.

Suppose that the point P for which we want to estimate the value of the mapping is situated at $\mathbf{x}=(x,y)$ inside the triangle formed by the points P_1, P_2, and P_3 of our known data set, that are located, respectively, at $\mathbf{x}_1=(x_1,y_1)$, $\mathbf{x}_2=(x_2,y_2)$, and $\mathbf{x}_3=(x_3,y_3)$ (see Fig. 3; note that throughout this discussion we do not distinguish between points P_i and the corresponding vectors $\mathbf{x}_i$ that emanate from the origin and point to P_i). Let us express point $\mathbf{x}$ in terms of the affine coordinate system that is generated by the vectors $\mathbf{x}_2-\mathbf{x}_1$ and $\mathbf{x}_3-\mathbf{x}_1$, the vectors which define the two triangle edges emanating from point $\mathbf{x}_1$. We have, therefore (see Fig. 3):

$$\mathbf{x}=\mathbf{x}_1+(\mathbf{x}_2-\mathbf{x}_1)a_2+(\mathbf{x}_3-\mathbf{x}_1)a_3, \tag{11}$$

where a_2,a_3 are the coordinates of point $\mathbf{x}$ in this new

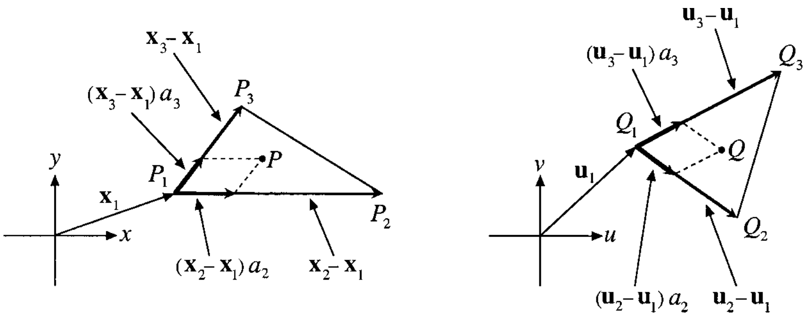

$\mathbf{x}_3-\mathbf{x}_1$
$(\mathbf{x}_3-\mathbf{x}_1)a_3$
P_3
y
P_1
P
$\mathbf{x}_1$
x
P_2
$(\mathbf{x}_2-\mathbf{x}_1)a_2$
$\mathbf{x}_2-\mathbf{x}_1$
$\mathbf{u}_3-\mathbf{u}_1$
$(\mathbf{u}_3-\mathbf{u}_1)a_3$
Q_3
Q_1
v
$\mathbf{u}_1$
Q
u
Q_2
$(\mathbf{u}_2-\mathbf{u}_1)a_2$
$\mathbf{u}_2-\mathbf{u}_1$

Fig. 3 Triangle $P_1P_2P_3$ in the x,y plane is mapped by the 2D transformation $\mathbf{f}:\mathbb{R}^2\rightarrow\mathbb{R}^2$ into triangle $Q_1Q_2Q_3$ in the u,v plane. The point P is mapped by linear triangular interpolation into the destination point Q. The coordinates of points P_1, P_2, P_3 and P are $\mathbf{x}_1=(x_1,y_1)$, $\mathbf{x}_2=(x_2,y_2)$, $\mathbf{x}_3=(x_3,y_3)$, and $\mathbf{x}=(x,y)$, and the coordinates of points Q_1, Q_2, Q_3 and Q are $\mathbf{u}_1=(u_1,v_1)$, $\mathbf{u}_2=(u_2,v_2)$, $\mathbf{u}_3=(u_3,v_3)$, and $\mathbf{u}=(u,v)$.

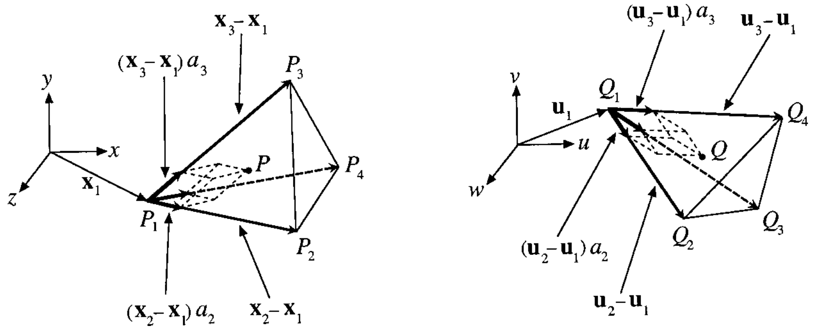

$\mathbf{x}_3-\mathbf{x}_1$
$(\mathbf{x}_3-\mathbf{x}_1)a_3$
P_3
y
x
z
$\mathbf{x}_1$
P
P_4
P_1
P_2
$(\mathbf{x}_2-\mathbf{x}_1)a_2$
$\mathbf{x}_2-\mathbf{x}_1$
$(\mathbf{u}_3-\mathbf{u}_1)a_3$
$\mathbf{u}_3-\mathbf{u}_1$
v
$\mathbf{u}_1$
Q_1
Q_4
u
Q
w
Q_2
Q_3
$(\mathbf{u}_2-\mathbf{u}_1)a_2$
$\mathbf{u}_2-\mathbf{u}_1$

Fig. 4 Tetrahedron $P_1P_2P_3P_4$ in the *x,y,z* space is mapped by the 3D transformation $\mathbf{f}:\mathbb{R}^3\rightarrow\mathbb{R}^3$ into tetrahedron $Q_1Q_2Q_3Q_4$ in the u,v,w space. The point P is mapped by linear tetrahedral interpolation into the destination point Q The coordinates of points P_1, P_2, P_3, P_4, and P are $\mathbf{x}_1=(x_1,y_1,z_1)$, $\mathbf{x}_2=(x_2,y_2,z_2)$, $\mathbf{x}_3=(x_3,y_3,z_3)$, $\mathbf{x}_4=(x_4,y_4,z_4)$, and $\mathbf{x}=(x,y,z)$, and the coordinates of points Q_1, Q_2, Q_3, Q_4, and Q are $\mathbf{u}_1=(u_1,v_1,w_1)$, $\mathbf{u}_2=(u_2,v_2,w_2)$, $\mathbf{u}_3=(u_3,v_3,w_3)$, $\mathbf{u}_4=(u_4,v_4,w_4)$, and $\mathbf{u}=(u,v,w)$.

coordinate system (note that $0\leqslant a_2,a_3\leqslant 1$ and $a_2+a_3\leqslant 1$, since point $\mathbf{x}$ is situated inside the triangle). In other words:

$$\begin{pmatrix}x\\ y\end{pmatrix}=\begin{pmatrix}x_1\\ y_1\end{pmatrix}+\begin{pmatrix}x_2-x_1\\ y_2-y_1\end{pmatrix}a_2+\begin{pmatrix}x_3-x_1\\ y_3-y_1\end{pmatrix}a_3$$

or

$$x=x_1+(x_2-x_1)a_2+(x_3-x_1)a_3,$$

$$y=y_1+(y_2-y_1)a_2+(y_3-y_1)a_3.$$

This can be reformulated as

$$\begin{pmatrix}x\\ y\end{pmatrix}=\begin{pmatrix}x_1\\ y_1\end{pmatrix}+\begin{pmatrix}x_2-x_1 & x_3-x_1\\ y_2-y_1 & y_3-y_1\end{pmatrix}\begin{pmatrix}a_2\\ a_3\end{pmatrix},$$

and hence, we obtain

$$\begin{pmatrix}a_2\\ a_3\end{pmatrix}=\begin{pmatrix}x_2-x_1 & x_3-x_1\\ y_2-y_1 & y_3-y_1\end{pmatrix}^{-1}\begin{pmatrix}x-x_1\\ y-y_1\end{pmatrix}. \tag{12}$$

Let now $\mathbf{u}_1=(u_1,v_1)$, $\mathbf{u}_2=(u_2,v_2)$, and $\mathbf{u}_3=(u_3,v_3)$ be the known images under the mapping $\mathbf{f}$ of points $\mathbf{x}_1=(x_1,y_1)$, $\mathbf{x}_2=(x_2,y_2)$, and $\mathbf{x}_3=(x_3,y_3)$ in the destination u, v plane. These points define the image of our triangle in the destination plane. Using linear triangulation interpolation, our interpolation point $\mathbf{x}=(x,y)$ is mapped to the point $\mathbf{u}=(u,v)$, inside the destination triangle, that has the same relative coordinates a_2,a_3 with respect to the new coordinate system defined by the destination triangle (see Fig. 3). We have, therefore, in analogy with Eq. (11):

$$\mathbf{u}=\mathbf{u}_1+(\mathbf{u}_2-\mathbf{u}_1)a_2+(\mathbf{u}_3-\mathbf{u}_1)a_3 \tag{13}$$

or

$$u=u_1+(u_2-u_1)a_2+(u_3-u_1)a_3,$$

$$v=v_1+(v_2-v_1)a_2+(v_3-v_1)a_3.$$

This can be reformulated as

$$\begin{pmatrix}u\\ v\end{pmatrix}=\begin{pmatrix}u_1\\ v_1\end{pmatrix}+\begin{pmatrix}u_2-u_1 & u_3-u_1\\ v_2-v_1 & v_3-v_1\end{pmatrix}\begin{pmatrix}a_2\\ a_3\end{pmatrix}.$$

Inserting here the values of a_2 and a_3 from Eq. (12) we obtain, therefore

$$\begin{pmatrix}u\\ v\end{pmatrix}=\begin{pmatrix}u_1\\ v_1\end{pmatrix}+\begin{pmatrix}u_2-u_1 & u_3-u_1\\ v_2-v_1 & v_3-v_1\end{pmatrix}\times\begin{pmatrix}x_2-x_1 & x_3-x_1\\ y_2-y_1 & y_3-y_1\end{pmatrix}^{-1}\begin{pmatrix}x-x_1\\ y-y_1\end{pmatrix}. \tag{14}$$

This is the interpolated image $\mathbf{u}$ of our point $\mathbf{x}$ in the destination u, v plane.

The 3D case, i.e., the interpolation of a 3D mapping $\mathbf{f}:\mathbb{R}^3\rightarrow\mathbb{R}^3$ from the *x,y,z* space into the destination u, v, w space, is a straightforward generalization of the 2D case, where the source and destination triangles are replaced by a source and a destination tetrahedra (see Fig. 4). The 3D tetrahedral linear interpolation is defined, therefore, by the following pair of equations, which are the 3D analogs of Eqs. (11) and (13):

$$\mathbf{x}=\mathbf{x}_1+(\mathbf{x}_2-\mathbf{x}_1)a_2+(\mathbf{x}_3-\mathbf{x}_1)a_3+(\mathbf{x}_4-\mathbf{x}_1)a_4, \tag{15}$$

$$\mathbf{u}=\mathbf{u}_1+(\mathbf{u}_2-\mathbf{u}_1)a_2+(\mathbf{u}_3-\mathbf{u}_1)a_3+(\mathbf{u}_4-\mathbf{u}_1)a_4, \tag{16}$$

where $\mathbf{x}_i$ are four points of the given data set that define the tetrahedron in the source *x,y,z* space in which the interpolation point $\mathbf{x}$ is located, and $\mathbf{u}_i$ are their known images under the mapping $\mathbf{f}$ in the destination u, v, w space (see Fig. 4). The interpolated image $\mathbf{u}=(u,v,w)$ of the point $\mathbf{x}=(x,y,z)$ is given, therefore, by the 3D analog of Eq. (14):

$$\begin{pmatrix} u\\ v\\ w\end{pmatrix} = \begin{pmatrix}u_1\\ v_1\\ w_1\end{pmatrix} + \begin{pmatrix} u_2-u_1 & u_3-u_1 & u_4-u_1\\ v_2-v_1 & v_3-v_1 & v_4-v_1\\ w_2-w_1 & w_3-w_1 & w_4-w_1\end{pmatrix} \times \begin{pmatrix} x_2-x_1 & x_3-x_1 & x_4-x_1\\ y_2-y_1 & y_3-y_1 & y_4-y_1\\ z_2-z_1 & z_3-z_1 & z_4-z_2\end{pmatrix}^{-1} \begin{pmatrix}x-x_1\\ y-y_1\\ z-z_1\end{pmatrix}. \quad (17)$$

This is the requested interpolated value **u** in the destination space.

This three-valued variant of tetrahedral linear interpolation method can be found in literature in Refs. 18 and 19.

Evaluation: The linear interpolation methods we have seen earlier are relatively simple to implement (provided that good triangulation or tetrahedrization routines are available), and they run quite quickly. However, they suffer from a few considerable drawbacks.

(1) Although piecewise linear interpolation guarantees a continuous result, it introduces derivative discontinuities on the boundaries between adjacent triangles (or tetrahedra). In other words, the resulting piecewise interpolating function is C^0 continuous but not C^1 continuous (smooth). This may lead to some objectionable, clearly visible artifacts such as a faceted appearance (Ref. 20, p. 735).

(2) Since a triangulation (or tetrahedrization) only covers the convex hull of the scattered point set, extrapolation beyond the convex hull—whenever required—is not possible with the linear interpolation scheme [see an illustrated example in (Ref. 19, pp. 319–320)].

(3) Since triangulation (or tetrahedrization) of a scattered point set is not unique, a different triangulation will result in different interpolated values at each intermediate point.

(4) The use of the Delaunay method guarantees an optimal triangulation (or tetrahedrization) in the sense that we have seen earlier. However, this triangulation (tetrahedrization) is not necessarily best suited for all applications. In some cases it is desired that the triangulation uses a predefined set of edges; in such particular cases a *constrained triangulation* may be desirable (Ref. 21, pp. 230–234). In other cases, in order to best suit some known characteristics or requirements of the underlying phenomenon, the best suited triangulation (or tetrahedrization) may depend on the actual values of the data set points, and not only on their locations. Overviews on *data-dependent triangulations* or *data-dependent tetrahedrizations* can be found, respectively, in Ref. 10, pp. 453–467; Ref. 14, p. 205; or in Ref. 10, pp. 494–502.

(5) Finally, as already mentioned, the triangulation (or tetrahedrization) of a scattered point set gives the *convex hull* of the given data set. In cases where the domain is not convex (for example, the color gamut of certain devices) the triangulation may yield, therefore, triangles (or tetrahedra) that are completely or partially outside the domain, where interpolations may give meaningless results. In such cases it may be desired to only triangulate the interior of the data set boundary, and not its full convex hull. Recipes for doing so are given in Ref. 22, p. 168; Ref. 17, p. 8. Note, however, that the important relationship between the ‘‘maxmin’’ angle criterion and the Delaunay triangulation (see Sec. 2.1) is only guaranteed for the convex case (Ref. 10, p. 444); also, the search algorithm presented at the end of Sec. 2.1 may not work in nonconvex cases.

2.4 *The Clough–Tocher Method: A Cubic Triangular Interpolation*

As we have seen earlier, one of the main drawbacks of piecewise linear interpolation schemes is that they are only C^0 continuous but not C^1 continuous (i.e., they are not smooth across the boundaries between triangles or tetrahedra). C^1 continuity obviously requires piecewise interpolation schemes based on polynomials of higher order than 1. For example, in the 2D case the planar triangular surfaces of the linear interpolation should be replaced by curved triangular surfaces, whose slopes to both sides of each boundary can be adjusted and made equal.

In order to guarantee C^1 continuity across boundaries, we need to have more information at each of the triangle vertices or edges. Typically, we need to know at each point, in addition to its location and its value, its gradient (i.e., in the case of a 2D scattered point set, the x- and y-partial derivatives, or the tangent plane, or equivalently, its normal vector). However, since the assumption of given gradients is not always realistic, in most cases they will have to be estimated from the given point values (Ref. 15, p. 301). This can be done by considering the known values not only in the vertices of the triangle in question, but also in its neighbors; a few possible methods for estimating the gradients, both local and global methods, are discussed and compared in Ref. 23; Ref. 22, pp. 173–176; and Ref. 17, pp. 8–9. Some of these gradient estimation methods can be considered as an additional preprocessing step, to be performed immediately after the triangulation step.

Several different interpolation methods have been devised along these lines (see, for example, the minimum norm network method in Ref. 10, p. 475 and Ref. 22, p. 170; the Powell–Sabin method, Ref. 15, p. 305; and a survey of several methods in Ref. 24, pp. 50–52). We will describe here one of the most popular and widely used methods, the Clough–Tocher method. This method was originally designed for the 2D case, as a tool for the finite element method (Ref. 25, p. 84; Ref. 26, p. 19), but it was later extended also to the general ND case.[27] We will describe it here for the basic 2D case.

Like piecewise linear interpolation, the Clough–Tocher interpolation method, too, is based on a triangulation of the given scattered point set. However, it significantly improves on the linear triangular interpolation methods in that it uses a *cubic* interpolation scheme within each triangle using bivariate cubic polynomials.

In order to create a piecewise C^1-continuous interpolation surface, the Clough–Tocher method requires for each triangle the value and the gradient (i.e., two partial derivatives) at each of the three vertices, as well as the normal derivative at the midpoint of each of the three edges in order to ensure that the normal slope matches across triangle boundaries (Ref. 25, p. 84). The aim is to impose these 12 given or estimated constraints (three per vertex + one per edge) on the cubic polynomial defined on our triangle. However, this is not possible since a bivariate cubic polynomial is determined by only ten coefficients

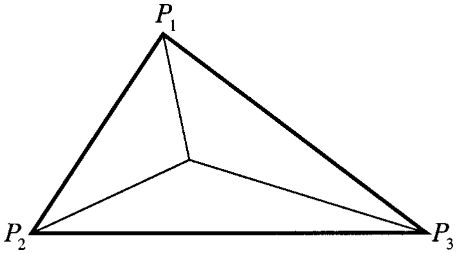


Fig. 5 In the Clough–Tocher method each triangle $P_1P_2P_3$ from the original triangulation is split into three minitriangles.

$$p(x,y)=a_1x^3+a_2x^2y+a_3xy^2+a_4y^3+a_5x^2 +a_6xy+a_7y^2+a_8x+a_9y+a_{10}$$

and can therefore only satisfy ten constraints.

Nevertheless, it turns out that we can still achieve our goal by splitting each triangle of the original triangulation into three *minitriangles* by joining each of the three vertices to the centroid (see Fig. 5),[†] and defining a bivariate cubic polynomial on each of the three minitriangles. This construction gives us more degrees of freedom for satisfying our constraints; we will see later how our constraints can now be imposed, and what is their geometric significance.

The key idea behind the Clough–Tocher method is, therefore, to split each cubic polynomial patch into three cubic polynomial subpatches in order to satisfy the C^1-continuity constraints with neighbors. Specifically, a *cubic Bézier patch* is defined over each minitriangle.

The triangular Bézier patch is a 2D generalization of the Bézier curve. While the most straightforward 2D generalization of the Bézier curve consists of rectangular Bézier patches (see Chap. 15 in Ref. 15), a more useful form for our needs is the generalization of the Bézier curve into triangular Bézier patches (Ref. 28; Ref. 15, Chap. 17). These patches are defined over a triangle with known vertices using barycentric coordinates; the generalization to Bézier tetrahedra in the 3D case is straightforward. The triangular Bézier patch is defined in terms of Bernstein polynomials (Ref. 15, pp. 44, 283), hence, the name Bernstein–Bézier polynomial. An mth degree Bernstein–Bézier polynomial defined over a triangle is of the form (Ref. 26, p. 19)[††]

$$p(u,v,w)=\sum_{\substack{0\leq i,j,k\leq m\\ i+j+k=m}} \frac{m!}{i!j!k!}b_{i,j,k}u^iv^jw^k,$$

where u, v, w are the barycentric coordinates within the triangle $P_1P_2P_3$; the coefficients $b_{i,j,k}$ are called the *Bézier ordinates* of p. We will be interested here only in *cubic* Bernstein–Bézier polynomials (i.e., with $m=3$), whose form is therefore

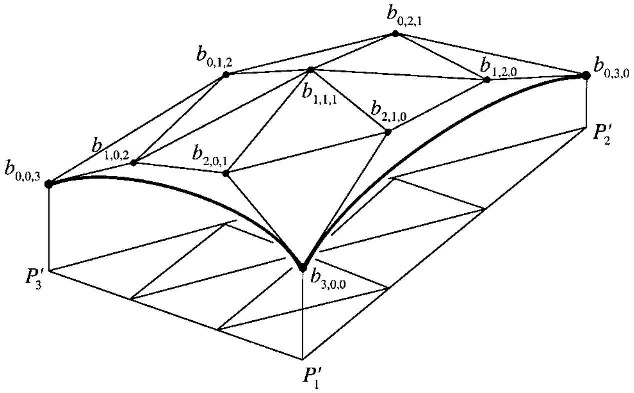


Fig. 6 A cubic Bézier patch over minitriangle $P_1'P_2'P_3'$. The curved surface of the patch lies beneath the triangulated surface defined by the control points $b_{i,j,k}$. The partition of the floor triangle $P_1'P_2'P_3'$ is the orthogonal projection of the triangulated surface onto the *x,y* plane. (Adapted from Ref. 26, p. 20, with permission from Elsevier Science.)

$$p(u,v,w)=b_{3,0,0}u^3+3b_{2,1,0}u^2v+3b_{1,2,0}uv^2+b_{0,3,0}v^3 +3b_{0,2,1}v^2w+3b_{0,1,2}vw^2+b_{0,0,3}w^3 +3b_{1,0,2}uw^2+3b_{2,0,1}u^2w+6b_{1,1,1}uvw \quad (18)$$

(see Fig. 6). Note that these cubic polynomials have exactly ten coefficients $b_{i,j,k}$: $b_{3,0,0}$, $b_{0,3,0}$, and $b_{0,0,3}$ are the known values at the triangle vertices, and the seven other Bézier ordinates are the 2D analogs of the control points of a Bézier curve [see the one-dimensional (1D) Bézier curve over each of the three triangle boundaries in the figure!]. Although the control points are not included in our given scattered point set, and their values $b_{i,j,k}$ are not known *a priori*, we will give them values by imposing constraints on the cross-boundary derivatives—just as we do in piecewise Bézier curves in order to impose a smooth behavior across their segment boundaries. As we have seen above, it turns out that the ten coefficients $b_{i,j,k}$ are not sufficient in order to ensure C^1 continuity across the three boundaries of the triangle $P_1P_2P_3$ since they do not give us enough degrees of freedom as we would wish. Hence the idea of splitting each of the original triangles of our triangulation into three minitriangles, and defining a cubic Bézier patch over each of these minitriangles. In order to avoid confusion between triangles and minitriangles we will use for minitriangles a notation with primes, e.g., $P_1'P_2'P_3'$.

Let us see now the geometric interpretation of the constraints that we need to impose in order to ensure C^1 continuity everywhere on the piecewise interpolating surface. As we can see in Figs. 6–8, the Bézier ordinates form within each minitriangle a triangular net, that divides each minitriangle edge into three equal segments; this triangular net is known as the *control net*. We will call the nine small triangles that are defined by the control net within each minitriangle *microtriangles*. Note that the microtriangles are not part of the Bézier patch itself, and they only belong to its control net. The curved Bézier patch itself is located beneath the triangulated surface defined by the control net (see Fig. 6), just as a 1D Bézier curve lies below its control polygon. Now, the constraints we impose in order to ensure

[†] Note that any other interior point other than the centroid would also do; the centroid is only chosen for symmetry reasons (Ref. 28, p. 108).

[††] Note that although the polynomial $p(u,v,w)$ looks trivariate, it is in fact only bivariate, since barycentric coordinates impose that $u+v+w=1$.

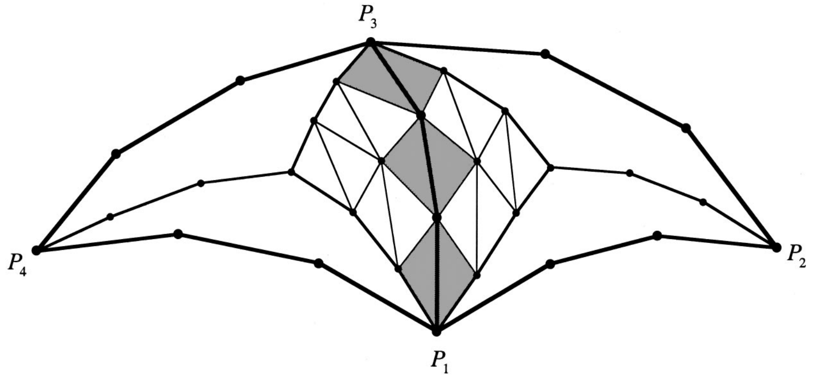


Fig. 7 The constraint imposed by the Clough–Tocher method in order to ensure C^1 continuity: every two neighboring microtriangles (shown shaded) that lie to both sides of a minitriangle edge must be coplanar. (Adapted from Ref. 26, p. 20, with permission from Elsevier Science.)

C^1 continuity can be geometrically interpreted as a 2D generalization of the constraints used in the 1D case to impose a smooth behavior across segment boundaries in a piecewise Bézier curve: Just as in the 1D case the two control line segments to both sides of each boundary must be collinear, so in our 2D case every two neighboring microtriangles that lie to both sides of a minitriangle edge must be coplanar (note that there are three pairs of such coplanar microtriangles along each minitriangle edge). This imposes also that all neighboring microtriangles that surround a vertex of a minitriangle are coplanar. This is illustrated in Figs. 7–9, where each shaded pair of microtriangles is coplanar.

The planes on which lie the microtriangles are determined by the values of the Bézier ordinates in the three minitriangles that make up the triangle $P_1P_2P_3$ (see Fig. 9). The *locations* of the ten Bézier ordinates within each minitriangle are fully determined by the control net: they are located at the minitriangle vertices, at the 1/3 and 2/3 points of each edge, and at the center of the minitriangle. The *values* of these Bézier ordinates are determined from our given (or estimated) constraints—the value and gradient at each vertex P_i and the normal slope at the edge midpoints, as follows (Ref. 28, p. 108): The values of the Bézier ordinates denoted in Fig. 8 by "●" are determined from the data (value and gradient) at the triangle vertices P_1, P_2, and P_3; for example, the values of the Bézier ordinates located at the $\frac{1}{3}$ and $\frac{2}{3}$ points of the edge P_1P_2 are determined by the tangent planes at P_1 and P_2, respectively. The three ordinates denoted by "▲" can be then determined from the estimated crossboundary derivative (the normal derivative at the midpoint of each of the three edges of the triangle $P_1P_2P_3$): each of them is located on the plane defined by a microtriangle "●,●,▲" that is determined by two already known ordinates ● and by the cross-boundary derivative. Then, the ordinates denoted by

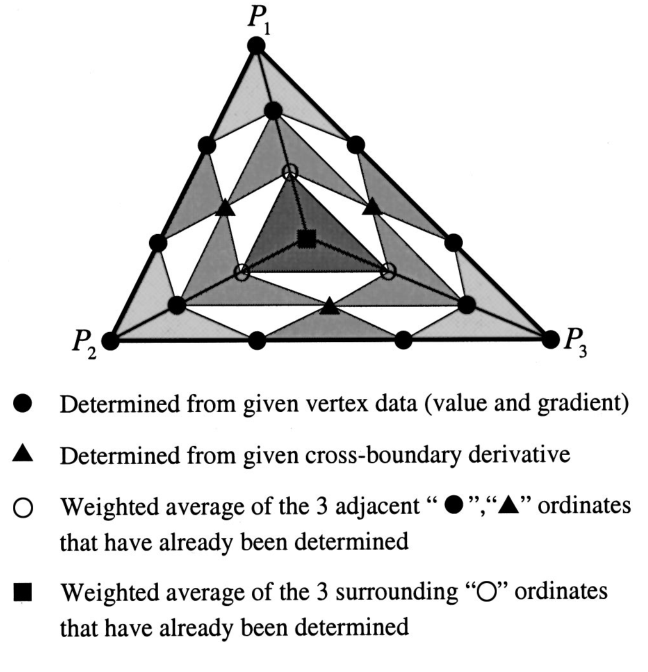

P_1
P_2
P_3
● Determined from given vertex data (value and gradient)
▲ Determined from given cross-boundary derivative
○ Weighted average of the 3 adjacent "●","▲" ordinates that have already been determined
■ Weighted average of the 3 surrounding "○" ordinates that have already been determined

Fig. 8 Top view of one triangle of the original triangulation, showing its three minitriangles and its 19 Bézier ordinates (control points). Each of the three minitriangles consists of nine microtriangles; crossboundary coplanar microtriangles are shown shaded. Note that all neighboring microtriangles that surround a common vertex of a minitriangle are coplanar: If the vertex in question is not a point of the triangulation (as in the center of this figure) the number of microtriangles surrounding it is 3, but if the vertex is a point of the triangulation, the number of surrounding triangles may be larger. (Adapted from Ref. 26, p. 21, with permission from Elsevier Science.)

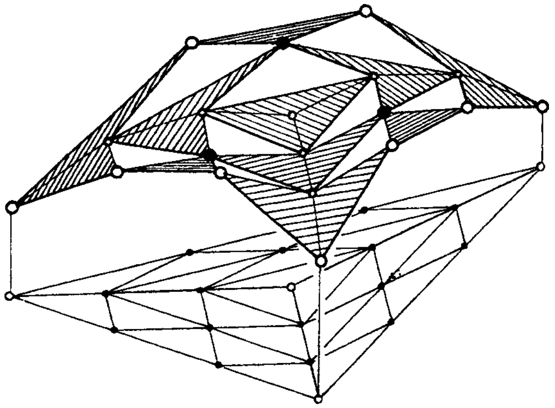


Fig. 9 A 3D view of Fig. 8. (Adapted from Ref. 24, p. 51, with permission from Elsevier Science.)

"○" can be determined by the three adjacent ordinates ● and ▲ that have already been calculated (since the two adjacent "▲, ●, ○" microtriangles must be coplanar). And finally, the Bézier ordinate at the central point "■" is determined by its three surrounding ○ points so that the three central microtriangles be coplanar. Note that the fact that makes this construction possible is that we do not need to impose restrictions at the third vertex of each minitriangle (the center point): as we have just seen, the value and the slope there are automatically derived from the vertex and edge-midpoint data of the triangle $P_1P_2P_3$. Hence, the subdivided surface now has enough degrees of freedom to allow C^1 continuity of the overall interpolant (Ref. 29, p. 53; Ref. 26, p. 22).

Once we have understood the logic behind the Clough–Tocher interpolation method, we can proceed to practical computation issues. The precise formulas for calculating the Bézier ordinates $b_{i,j,k}$ that satisfy our constraints are given in Ref. 15, pp. 305–306; Ref. 26, pp. 22–23; Ref. 28, pp. 108–109; and Ref. 29, pp. 52–54. Once the values $b_{i,j,k}$ are known, we can insert them back into Eq. (18) in order to obtain the specific triangular Bézier subpatch over the minitriangle in question; three such subpatches are needed to cover the triangle $P_1P_2P_3$ of our original triangulation. By proceeding in a similar way for all the other triangles, we obtain a C^1-continuous piecewise interpolating surface that is defined over the triangulation of our given scattered point set.

There exist several variants and generalizations of the Clough–Tocher method. A 3D (or rather *N*D) version of the method is described in Ref. 27; Ref. 28, pp. 116–117. A variant of the Clough–Tocher method which is closer to C^2 continuity has been proposed in Ref. 26. Another variant of the Clough–Tocher method has been proposed for cases in which it is known that the underlying function has discontinuities (or discontinuous derivatives) along certain boundaries.[30]

Evaluation: Since the Clough–Tocher interpolation method is local, it has the advantage of speed: even large scatter point sets can be interpolated quite rapidly. Also, like in the linear triangulation-based interpolation methods, a correction in any of the given data points only influences the interpolated values within the triangles (or tetrahedra) that have this data point as a vertex [see Fig. 5(a) in Ref. 26]. The Clough–Tocher method gives a smooth (C^1 continuous) interpolation surface (or volume), which brings out local trends in the data set quite accurately. However, it still depends on the quality of the underlying triangulation (or tetrahedrization): see points (4)–(5) at the end of Sec. 2.3.

2.5 *Triangle Based Blending*

The last triangulation-based method for scattered data interpolation that we describe here is the triangle based blending technique.[31] Once again, we need first to obtain a triangulation of our given scattered point set. Then, for each point (x_i, y_i) of the scattered point set we calculate a corresponding quadratic polynomial interpolant

$$z=\hat{f}(x,y)=a_1x^2+a_2xy+a_3y^2+a_4x+a_5y+a_6$$

that passes through the point (x_i, y_i) itself and its five nearest neighbors within the scattered point set. [As we have seen following Eq. (1), this can be simply done by inserting the *x,y,z* values of each of the six points into the polynomial equation, in order to obtain a linear system of six equations for the six coefficients $a_1,\ldots,a_6$.]

Now, in order to find the interpolated value *z* at an arbitrary point (*x,y*) we have to find first in which triangle of the triangulation this point is located. Then, the value *z* is determined as the weighted average of the values at the point (*x,y*) of the three interpolating polynomials that correspond to the three triangle vertices (Ref. 31, p. 180):

$$z=w_1\hat{f}_1(x,y)+w_2\hat{f}_2(x,y)+w_3\hat{f}_3(x,y).$$

To ensure a continuous transition from one triangle to the next we need only ensure that each weight w_i is identically zero along the edge of the triangle opposite to the *i*th vertex. This can be achieved by making w_i proportional to the *k*th power of the distance d_i of the point (*x,y*) to the edge, for some *k*, typically 3. The weights w_i are then scaled to make $d_i=1$ at the vertex *i*. We have, therefore

$$w_i=d_i{}^k/(d_1{}^k+d_2{}^k+d_3{}^k).$$

Other variants of this method can also guarantee C^1 continuity across triangle boundaries (Ref. 31, p. 384; Ref. 6, pp. 187–188). The extension of such methods to 3D is obtained by considering tetrahedra instead of triangles.

Evaluation: Triangle based blending methods are very fast, since they are local. Another advantage of these methods is that they allow also extrapolation beyond the convex hull of the given data points, i.e., outside the region covered by the triangles. To do this one can partition the exterior of this region by extending lines outwards from the data points on the boundary, for example by bisecting the exterior angles at the corners. The extrapolated values in each such semi-infinite quadrilateral region may be then found as a weighted average of the functions $\hat{f}_i(x,y)$ of the two boundary points (see Fig. 4 in Ref. 31, p. 179).

The main disadvantage of triangle based blending methods, like in all triangulation based methods, is the need to triangulate (or to tetrahedrize) the given scattered point set as a preprocessing step before interpolation can be started.

3 Inverse Distance Weighted Methods

One of the most commonly used techniques for interpolation of scattered points is inverse distance weighted interpolation. Inverse distance weighted methods are also known as "Shepard methods" after the name of the first contributor in this field.[32] These methods are basically global, i.e., they use all of the data points to calculate each interpolated value. Their basic assumption is that the interpolated values should be influenced more by nearby points and less by the more distant points. The interpolation value at each new point *P* is a weighted average of the values of the scattered points, and the weight assigned to each scatter point diminishes as the distance from the interpolation point to the scatter point increases. As we will see, many different ways exist for choosing these weights.

Let us start, as usual, with the 2D case. Suppose we are given irregularly distributed points P_i in the plane, whose locations and values are, respectively, (x_i,y_i) and z_i. We wish to construct an interpolating function $\hat{f}(x,y)$ with $\hat{f}(x_i,y_i)=z_i$ for all i, where the influence of point P_i decreases with increasing distance between (x,y) and (x_i,y_i). Shepard's original proposal can be formulated as a weighted average of the values z_i:

$$\hat{f}(x,y)=\sum_{i=1}^{n} w_i(x,y)z_i=\sum_{i=1}^{n}\frac{h_i(x,y)}{\sum_{i=1}^{n}h_i(x,y)}z_i \tag{19}$$

with weights:

$$w_i(x,y)=\frac{h_i(x,y)}{\sum_{i=1}^{n}h_i(x,y)}0\leqslant w_i(x,y)\leqslant 1,\ \sum_{i=1}^{n} w_i(x,y)=1$$

$$\text{where,}\quad h_i(x,y)=\frac{1}{[d_i(x,y)]^k} \tag{20}$$

and where $d_i(x,y)$ is the Euclidean distance between the points (x,y) and (x_i,y_i): $d_i(x,y)=\sqrt{(x-x_i)^2+(y-y_i)^2}$, and $k\geqslant 1$ is a chosen exponent. Note that $h_i(x,y)$ is infinitely large at the data point (x_i,y_i) itself. However, it can be shown that when (x,y) approaches a data point (x_i,y_i), $\hat{f}(x,y)$ tends to the value z_i; therefore, by choosing $\hat{f}(x_i,y_i)=z_i$ for all the given data points it is guaranteed that the interpolating function $\hat{f}(x,y)$ is continuous (Ref. 32, p. 518). Another favorable property of $\hat{f}(x,y)$ is that it is bounded above by max z_i and below by min z_i (Ref. 33, pp. 255, 257).

It is interesting to note that for any fixed point P located at (x,y), the denominator in Eq. (19) can be considered as a normalization constant so that the magnitude of $\hat{f}(x,y)$ is directly proportional to the value z_i and inversely proportional to the kth power of the distance from P to P_i. In this sense, the formula is analogous to a "$1/r^k$ gravitation law"(Ref. 33, p. 256).

The simplest interpolation function $\hat{f}(x,y)$ is obtained by choosing in Eq. (20) $k=1$, i.e.,

$$h_i(x,y)=\frac{1}{\sqrt{(x-x_i)^2+(y-y_i)^2}}.$$

However, a disadvantage of this function is that although it is continuous, it has slope discontinuities in the form of a cone at each of the data points (x_i,y_i) [see Fig. 10(a) for the equivalent 1D case]. This can be cured by taking $k=2$, i.e., the square of the distances

$$h_i(x,y)=\frac{1}{(x-x_i)^2+(y-y_i)^2}.$$

However, this allows too much influence by far away points (Ref. 6, p. 186). Furthermore, it turns out that such a surface becomes flat near each of the data points (x_i,y_i), since its partial derivatives there vanish (Ref. 33, pp. 258–260). In fact, as shown in Figs. 10–11, the higher the value of the exponent k, the more pronounced this flatness becomes, yielding an effect of "terracing;" and as $k\rightarrow\infty$, the interpolating function $\hat{f}(x,y)$ approaches a step function (similar to a 2D histogram) with the values $f(x_i,y_i)=z_i$. This means that for large values of k the practical influence of the ith point is essentially limited to a certain "region of influence" around it. Moreover, as clearly shown by Fig. 11(c), when $k\rightarrow\infty$ the regions of influence of the points P_i are clearly bounded by straight line segments. Note that an individual point P_i may be given a larger region of influence than the others by simply choosing a higher value of k for its $h_i(x,y)$ in Eq. (20). This is graphically illustrated for the 1D case in Fig. 10(d), and for the 2D case in Fig. 11(b).

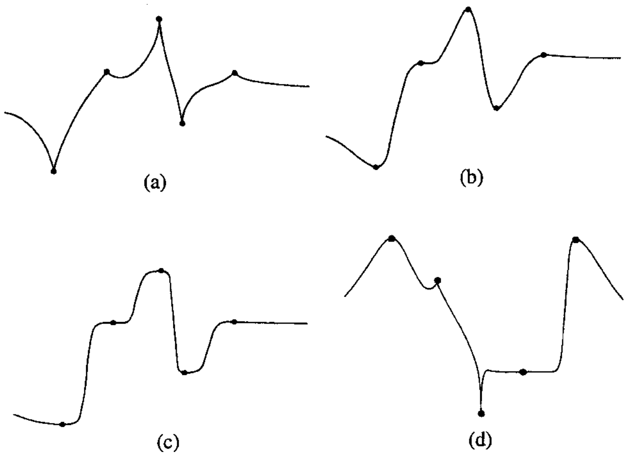


Fig. 10 A simple illustration of the behavior of a univariate Shepard interpolation function based on the 1D counterparts of Eqs. (19) and (20), for various values of the exponent *k*. (Adapted from Ref. 33, p. 259, with permission from the American Mathematical Society.) The values of *k* used in figures (a), (b), (c), and (d) are as follows (from left to right in each figure):

Figure	k_1	k_2	k_3	k_4	k_5
(a)	1	1	1	1	1
(b)	2	2	2	2	2
(c)	10	10	10	10	10
(d)	2	1	0.5	15	2

The objectionable artifact due to the flatness of $f(x,y)$ around the data points (x_i,y_i) can be cured by imposing at each data point its real partial derivatives instead of the zero partial derivatives inherent to the original method. This can be done by replacing the values z_i in Eq. (19) by tangent planes

$$g(x,y)=z_i+a_i(x-x_i)+b_i(y-y_i), \tag{21}$$

where a_i and b_i are the x-and y-partial derivatives of the underlying function $z(x,y)$ at the point (x_i,y_i). The influence of this correction on the resulting interpolation function $\hat{f}(x,y)$ is graphically illustrated in Fig. 12. Note, however, that in most cases the partial derivatives at the data points are not given, and they have to be estimated. Ways for estimating them are given in Ref. 33, pp. 263–264; Ref.

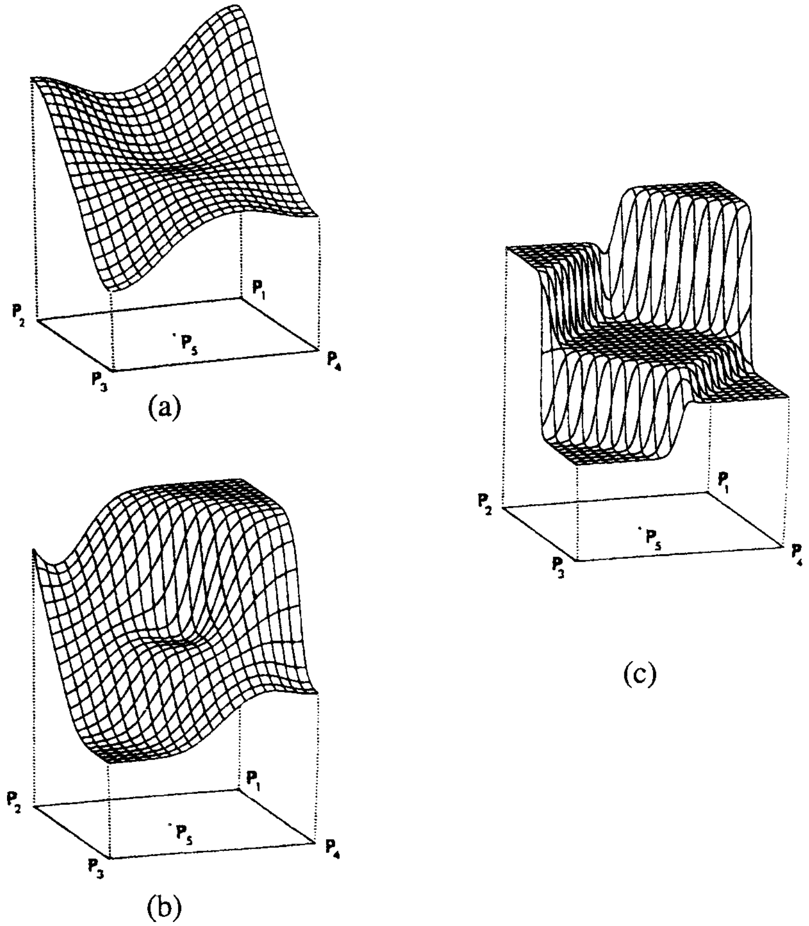


Fig. 11 A simple illustration of the behavior of a bivariate Shepard interpolation function based on Eqs. (19) and (20) for various values of the exponent *k*. (Adapted from Ref. 33, p. 260, with permission from the American Mathematical Society.) The values of *k* used in figures (a), (b), and (c) are as follows:

Figure	k_1	k_2	k_3	k_4	k_5
(a)	2	2	2	2	2
(b)	10	1	5	2	3
(c)	20	20	20	20	20

32, pp. 520–521; Ref. 7, Sec. 11; and Ref. 34, Sec. 2.4; see also the derivative estimation problem in the Clough–Tocher method (Sec. 2.4 earlier).

More generally, in order to increase the continuity of the resulting function one may replace the values z_i in Eq. (19) by $L_i z(x,y)$, where $L_i z(x,y)$ is a local approximation to the underlying function $z(x,y)$ at the "node" (x_i, y_i), such that $L_i z(x_i, y_i) = z_i$ (Ref. 35, pp. 140–141; Ref. 6, pp. 185–186). Such $L_i z(x,y)$ are called nodal functions. Other generalizations of the Shepard methods can be found in Ref. 6, p. 186.

Finally, because the inverse distance weighted methods are global, and hence, their computation is too slow for large point sets, several ways have been suggested to "localize" them. For example, it has been suggested (Ref. 6, p. 185) to use

$$h_i(x,y) = \left\{ \frac{[R - d_i(x,y)]_+}{R d_i(x,y)} \right\}^2, \qquad (22)$$

where

$$[R - d_i(x,y)]_+ = \begin{cases} R - d_i(x,y) & \text{if } d_i(x,y) < R \\ 0 & \text{if } d_i(x,y) \geqslant R \end{cases}$$

and where $d_i(x,y)$ denotes the Euclidean distance between (x,y) and (x_i, y_i), and R is a radius of influence about the node (x_i, y_i). This means that data at (x_i, y_i) only influences interpolated values at points within this radius. It can be shown that the resulting interpolant $\hat{f}(x,y)$ has continuous first partial derivatives (Ref. 35, pp. 140–141), so that it is C^1 continuous.

Evaluation: Shepard methods suffer from some problems inherent to distance-based schemes. In particular, since they are only sensitive to the distance they tend to overweight data clusters. Moreover, because they are distance-based schemes each data point has a radially symmetric influence and, hence, data features such as planes,

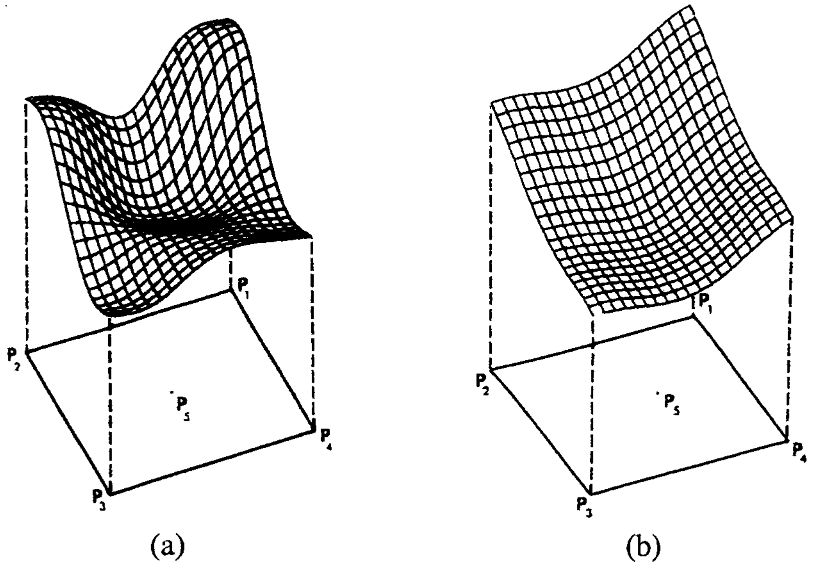


Fig. 12 (a) The Shepard interpolation function $\hat{f}(x,y)$ of Eqs. (19) and (20) with $k=2$ for all five points. The "flat spots" artifact caused by vanishing first partial derivatives is quite evident. (b) The result of imposing the estimated partial derivatives at each of the five given points. The improvement obtained by this method with respect to (a) is clearly visible. (Adapted from Ref. 33, p. 263, with permission from the American Mathematical Society.)

ridges, and valleys are obscured by this enforced isotropy around sample points (Ref. 34, p. 116). But although Shepard methods are less efficient and less accurate than some triangulation-based methods, they are among most viable candidates for extension to three or more independent variables, because the triangulation-based methods have greater complexity and storage requirements associated with their extension (Ref. 35, p. 140). Also, Shepard methods have the ability to extrapolate naturally outside the convex hull of the given data points (Ref. 32, p. 521). In fact, it has been shown that for any choice of equal exponents $n \geqslant 1$ for the $h_i(x,y)$ in Eq. (19), as the interpolation point P recedes infinitely from the cluster of the given points P_i, $\hat{f}(x,y)$ approaches the average of the values z_i (Ref. 33, pp. 257; 261–262). This is clearly illustrated in Fig. 13.

The performance of Shepard methods is very dependent on an appropriate weight function $w_i(x,y)$ (Ref. 6, p. 186). It is instructive to see the list of shortcomings of pure inverse distance weighting as mentioned in Shepard's original paper (Ref. 32, pp. 518–519), and the remedies he suggests for each of them in his weight functions (Ref. 32, pp. 519–521). The weight function of Eq. (22) is reported to have an accuracy comparable to other local methods (Ref. 35, p. 139) and is therefore a good starting point, provided that the parameter R can be well chosen for the given scattered point set. See Ref. 8, Sec. 4.1 for an available implementation of a modified Shepard method, including both 2D and 3D versions.

4 Radial Basis Function Methods

Radial basis function methods are another example of global interpolation methods for scattered points. The initial ideas, dating from the late 1960's, are due to Hardy.[36] Basically, these methods can be characterized as follows: For each point (x_i, y_i) of the data set simply choose some func-

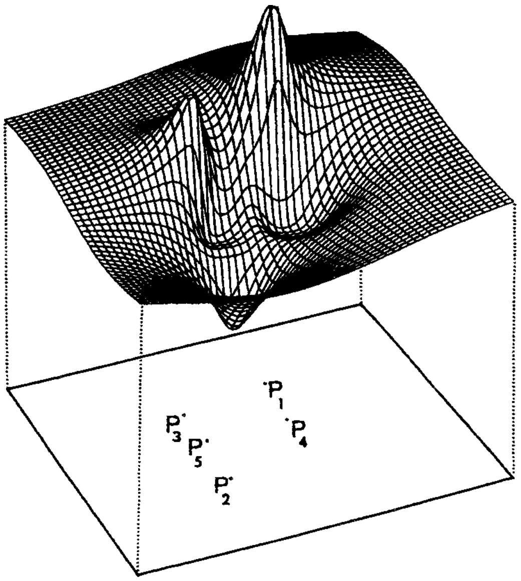


Fig. 13 An illustration of the behavior of a bivariate Shepard interpolation function based on Eqs. (19) and (20) for various values of the exponent k. (Adapted from Ref. 33, p. 262, with permission from the American Mathematical Society.) This figure clearly illustrates that the partial derivatives at each point P_i are zero. Since the figure is plotted with large margins around the given points, it also shows the asymptotic behavior of the interpolating function $\hat{f}(x,y)$ away from the given points: clearly, $\hat{f}(x,y)$ tends in all directions to the average of the values z_i. The point locations (x_i, y_i), their corresponding values z_i and the values of k used for each of them are as follows:

i	1	2	3	4	5
(x_i, y_i)	(0,0)	(1,1)	(1.2,0.2)	(0,0.5)	(1,0.5)
z_i	4	0	3	1	1
k_i	2.2	2.5	3	4	4

tion $h_i(x,y)$, and then determine coefficients a_i so that $\hat{f}(x,y)=\Sigma_i a_i h_i(x,y)$ interpolates the data. However, as we will see later, appropriate choices of functions $h_i(x,y)$ are not easy to make. For example, polynomial functions give a very bad interpolating surface with many overshoots and ripples between the given data points; and even those functions that are known to work well are not always easy to justify mathematically. The most useful functions $h_i(x,y)$ are radial functions of one variable, $h(r)$, so that the basis function associated with each data point (x_i,y_i) is of the form $h_i(x,y)=h(d_i)$, where d_i is the distance between (x,y) to (x_i,y_i). This explains the name of this method family.

In Hardy's original method, each data point (x_i,y_i) is associated with a quadratic function $h_i(x,y)=[(x-x_i)^2+(y-y_i)^2+c^2]^{1/2}$, which represents a circular two-sheet hyperboloid (or, when $c=0$, a conic surface) centered at (x_i,y_i). Alternatively, one may associate with each data point (x_i,y_i) a quadratic function $h_i(x,y)=[(x-x_i)^2+(y-y_i)^2+c^2]$, which represents a circular paraboloid surface centered at (x_i,y_i). The multiquadric surface[§] defined by the sum

$$z=\sum_{i=1}^{n} a_i[(x-x_i)^2+(y-y_i)^2+c^2]^{1/2} \tag{23}$$

is, therefore, a shifted sum of circular two-sheet hyperboloids, while the multiquadric surface

$$z=\sum_{i=1}^{n} a_i[(x-x_i)^2+(y-y_i)^2+c^2] \tag{24}$$

is a shifted sum of circular paraboloids. The coefficients a_i determine the sign and the flatness of each quadratic term in the sum, and c is a small constant value to be specified by the user. Other multiquadric surfaces consisting of hyperbolic paraboloids, one-sheet hyperboloids, etc., can be also constructed in a similar way.

Now, since we know the location (x_i,y_i) and the value z_i for each of the n given points, we can insert them into Eq. (23) [or Eq. (24)], obtaining a set of n linear equations for the n unknown coefficients a_i:

$$\begin{aligned} z_1&=a_1[(x_1-x_1)^2+(y_1-y_1)^2+c^2]^{1/2}+\ldots+a_n[(x_1-x_n)^2\\ &\quad+(y_1-y_n)^2+c^2]^{1/2}\\ &\vdots\\ z_n&=a_1[(x_n-x_1)^2+(y_n-y_1)^2+c^2]^{1/2}+\ldots+a_n[(x_n-x_n)^2\\ &\quad+(y_n-y_n)^2+c^2]^{1/2}\end{aligned}$$

or in matrix notation

$$\mathbf{z}=M\mathbf{a}. \tag{25}$$

The solution of these equations is given by

$$\mathbf{a}=M^{-1}\mathbf{z}.$$

This calculation may be considered as a preinterpolation stage (much like the triangulation step in triangulation-based methods). Once the coefficients $a_1,\ldots,a_n$ are known, Eq. (23) [or, respectively, Eq. (24)] becomes a smooth interpolating surface, that gives us the interpolated value z for any point (x,y).

This multiquadric surface can be understood geometrically as a weighted sum of ''hills'' or ''hollows'' that are centered at the point locations (x_i,y_i), and whose steepness is determined by the weights a_i (see, for example, Fig. 1 in Ref. 37, p. 165). Clearly, a weighted sum of such quadratic terms (two-sheet hyperboloids, paraboloids, etc.) is C^∞ continuous (Ref. 6, p. 192); a sum of cones, however, is only C^0 continuous.

Hardy does not conclude which class of quadratic functions $h_i(x,y)$ gives the best results, and he suggests that the choice will depend on the nature of the underlying topography (its smoothness, angularity, etc.) (Ref. 36, pp. 1907–1908). He notes, however, that quadratics other than the cone tend to displace the maxima and minima of the underlying topography (Ref. 36, p. 1914). This can be cured by considering also the partial derivatives (surface tangents) if we know a certain number m of points (data points or not) where the slope of the underlying surface is zero (e.g., points that are hilltops or the bottom of a hollow) (Ref. 36, p. 1910). Let us illustrate this, for the sake of simplicity, in the 1D case. In this case Eq. (23) is reduced to

$$z=\sum_{i=1}^{n} a_i[(x-x_i)^2+c^2]^{1/2}.$$

Adding an expression for a polynomial series we obtain

$$z=\sum_{i=1}^{n} a_i[(x-x_i)^2+c^2]^{1/2}+\sum_{i=1}^{m} b_i x^i. \tag{26}$$

By differentiating Eq. (26) and equating the derivative to zero we have

$$\sum_{i=1}^{n} a_i[(x-x_i)^2+c^2]^{-1/2}(x-x_i)+\sum_{i=1}^{m} i b_i x^{i-1}=0. \tag{27}$$

Now, since we know the locations x_i and the values z_i for each of the n given points, we can insert them into Eq. (26), obtaining a set of n linear equations with $n+m$ unknowns, $a_1,\ldots a_n,b_1,\ldots b_m$. Similarly, by inserting the locations x_i of the m points at which the slope is zero into Eq. (27), we obtain a system of m equations with the same $n+m$ unknowns. We have, therefore, a system of $n+m$ equations (n coordinate equations and m slope equations) with $n+m$ unknowns. After the equations are solved for the unique values of $a_1,\ldots a_n$, $b_1,\ldots b_m$, these coeffi-

[§] The term multiquadric surface means a sum of quadric surfaces. A quadric (surface) is the 2D generalization of a conic (curve): just as a conic has the implicit equation $q(x,y)=0$, where q is a quadratic polynomial in x and y, so a quadric has the implicit equation $q(x,y,z)=0$, where q is quadratic in x,y, and z (Ref. 15, p. 298).

cients are inserted into Eq. (26) to form the required interpolation function. The resulting function is a multiquadric curve (or surface, in the 2D case) on which an additional polynomial, normally of a low degree, is added to impose zero slopes at the desired maxima and minima.

In a further generalization of these methods, the derivative conditions can be replaced by other constraints such as polynomial precision of order m (i.e., the ability to reproduce exactly any polynomial up to order m; this may be important in some fields such as computer aided geometric design or approximation theory): If we denote by $h_i(x,y) = h(d_i)$ the basis function associated with each data point (x_i, y_i), and by $\{q_i(x,y)\}$ a basis for the bivariate polynomials of degree$<m$, then the form of the interpolation function is (Ref. 8, p. 135):

$$z=\sum_{i=1}^{n} a_i h_i(x,y)+\sum_{i=1}^{m} b_i q_i(x,y)$$

and the $n+m$ coefficients a_i and b_i are found by solving the following n equations:

$$z_1=\sum_{i=1}^{n} a_i h_i(x_1,y_1)+\sum_{i=1}^{m} b_i q_i(x_1,y_1)$$

$$\vdots$$

$$z_n=\sum_{i=1}^{n} a_i h_i(x_n,y_n)+\sum_{i=1}^{m} b_i q_i(x_n,y_n)$$

along with the following m constraints that guarantee polynomial precision:

$$0=\sum_{i=1}^{n} a_i q_1(x_i,y_i)$$

$$\vdots$$

$$0=\sum_{i=1}^{n} a_i q_m(x_i,y_i).$$

It is interesting to note that although radial basis function methods were known to work well since the late 1960's, until the 1980's no mathematical theory existed for them, not even concerning the existence of a solution to the equations in all possible point configurations (Ref. 8, p. 147). Only in 1986 it has been shown that the system of equations is indeed nonsingular.

Note that in principle this method could be used with any family of functions $h_i(x,y)$, such as polynomials or trigonometric series of a certain order, whose coefficients a_i would be determined by a linear system of equations like Eq. (25) (see, for example, Ref. 4, pp. 208–209). The particularity of the multiquadric functions is that they give a system of equations that is rather well conditioned, and which is guaranteed to be nonsingular and hence to have a solution. For other function families, where the existence of a good solution for the coefficients a_i is not guaranteed, a best fit approach can still be used to find the "best" coefficients a_i, as explained in Sec. 6.

Finally, the extension of radial basis function methods to 3D or higher dimensions is rather straightforward.

Evaluation: Radial basis function methods are some of the most elegant schemes from a mathematical point of view, and they often work very well (Ref. 8, p. 135). In terms of fitting ability and visual smoothness, the most impressive method in this family is the original multiquadric method due to Hardy. Hardy's method was verified as being capable of fitting the underlying surface very accurately, and furthermore, it is quite stable with respect to the choice of the parameter c (Ref. 6, p. 191). However, since these methods require a preprocessing stage of solving a system of at least n linear equations with as many unknowns, with a full matrix, in order to obtain the coefficients of the interpolating function, these methods are not suitable for large data sets with more than a few hundred points (Ref. 8, p. 135). Various ways have been proposed to "localize" these methods in order to allow the treatment of larger data sets (Ref. 8, p. 136). For example, one may decompose the domain of the scattered point set in order to solve many small systems of linear equations rather than one global system, and subsequently blend the resulting local interpolants. A historical review of the multiquadric method with more than 100 references is given in Ref. 37.

5 Natural Neighbor Interpolation Methods

The last family of scattered data interpolation methods we describe here, known as natural neighbor interpolation, was first introduced in Ref. 38. This interpolation approach is local, and is based on the Voronoi tesselation of the given scattered point set—which is the geometric dual structure of the Delaunay triangulation (or tetrahedrization) (Ref. 21, p. 203). The Voronoi tesselation partitions the plane (or space) into an exhaustive and disjoint set of tiles (polygons or, respectively, polytopes), each tile T_i enclosing one point $\mathbf{x}_i$ of the given point set. The tile T_i is defined as the area (or volume) that is closer to the scatter point $\mathbf{x}_i$ than to any other scatter point

$$T_i=\{\mathbf{x}\in\mathbb{R}^2 | d(\mathbf{x},\mathbf{x}_i)\leqslant d(\mathbf{x},\mathbf{x}_j)\forall j=1,\ldots n\},$$

where $d(\mathbf{a,b})$ denotes the Euclidean distance between the points $\mathbf{a}$ and $\mathbf{b}$. Point $\mathbf{x}_i$ of our point set is said to be a natural neighbor of point $\mathbf{x}_j$ of the set if their respective tiles T_i and T_j have a common edge or point of contact. The number of natural neighbors of point $\mathbf{x}_i$ (that is not on the external boundary) is at least $N+1$, where N is the space dimension, and at most $n-1$, where n is the number of points in the set, and it varies from point to point. In the 2D case the average number of natural neighbors (the number of edges in a Voronoi polygon) does not exceed 6 (Ref. 21, p. 205).

Before interpolation can be started, a preprocessing step is required for constructing the Voronoi tesselation of the given scattered point set (see Ref. 13, pp. 149–160; Ref. 21 pp. 205–214; and Ref. 38, p. 31).

Once the Voronoi tesselation of our point set has been constructed, all that we need in order to evaluate the inter-

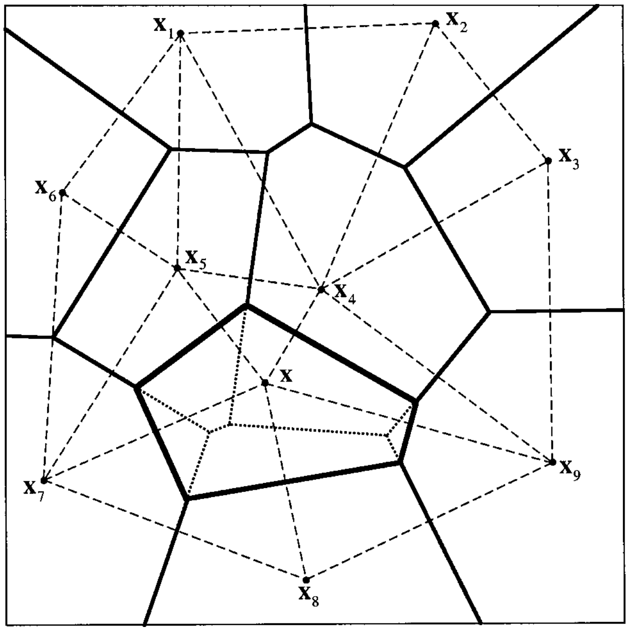


Fig. 14 The Voronoi tiles of the scattered point set $\{\mathbf{x}_1,\ldots,\mathbf{x}_9\}$. When a new point $\mathbf{x}$ is added to the tesselation, it "craves" its own tile by "stealing" territory from the surrounding data points. Bold lines indicate the new tile of point $\mathbf{x}$; the dotted lines inside it show the old tile boundaries before the addition of point $\mathbf{x}$. For the sake of completeness, the Delaunay triangles associated with the Voronoi tesselation are also shown, indicated by thin dashed lines; the triangle edges are perpendicular bisectors of the tile edges.

polated value at a new point $\mathbf{x}$ is simply to add $\mathbf{x}$ to the set of scatter points, letting it "crave" its own tile from the tiles of the surrounding data points (see Fig. 14). Thus, the new tile of point $\mathbf{x}$ is

$$T(\mathbf{x})=\{\mathbf{z}\in\mathbb{R}^2|d(\mathbf{z},\mathbf{x})\leqslant d(\mathbf{z},\mathbf{x}_j)\forall j=1,\ldots,n\}$$

and its intersections with the old tiles are:

$$T_i(\mathbf{x})=T(\mathbf{x})\cap T_i\,.$$

Note that the intersections $T_i(\mathbf{x})$ are nonempty only for neighboring tiles T_i from which the new tile $T(\mathbf{x})$ has "stolen" territory. The actual calculation of the areas (or volumes) of the subtiles $T_i(\mathbf{x})$ for each neighboring point $\mathbf{x}_i$ is a complicated but reasonably efficient geometrical computation (Ref. 38, p. 31). The detailed procedure for the 2D case can be found in Ref. 34, p. 82; for higher dimensions see Ref. 39, p. 658.

Now, denoting the area (or the volume) of a tile T by area(T), the natural neighbor interpolant at the point $\mathbf{x}$ could be defined as

$$\hat{f}(\mathbf{x})=\sum_i h_i(\mathbf{x})z_i\,, \tag{28}$$

where

$$h_i(\mathbf{x})=\frac{\text{area}[T_i(\mathbf{x})]}{\text{area}[T(\mathbf{x})]}\quad 0\leqslant h_i(\mathbf{x})\leqslant 1,\ \sum_i h_i(\mathbf{x})=1.$$

[Note that $h_i(\mathbf{x})$ varies smoothly between 1 at the point $\mathbf{x}_i$ itself and 0 where the interpolation point $\mathbf{x}$ ceases to be a natural neighbor of $\mathbf{x}_i$; $h_i(\mathbf{x})$ is a continuous function of $\mathbf{x}$, and furthermore, $\Sigma_i h_i(\mathbf{x})\mathbf{x}_i=\mathbf{x}$ (Ref. 38, p. 28).] This interpolation method is, therefore, a weighted average of the subset of data points $\mathbf{x}_i$ which are natural neighbors of the interpolated location $\mathbf{x}$.

This method results in a surface which consists of cone-like peaks or pits at the data points $\mathbf{x}_i$ (Ref. 34, p. 155). This means that the interpolant Eq. (28) is only C^0 continuous, since its derivatives are discontinuous at the points $\mathbf{x}_i$. In order to obtain a C^1-continuous interpolant we must take into account also the gradient at each of the points; as we have already seen in the previous sections there exist several methods for estimating the gradients at the given data points. Once we have estimated the gradient $\nabla z(\mathbf{x})$ of the underlying function $z(\mathbf{x})$ at the point $\mathbf{x}_i$, we replace the values z_i in Eq. (28) by the first degree polynomial $g_i(\mathbf{x})$ that passes through z_i with the calculated slope

$$g_i(\mathbf{x})=z_i+\nabla z(\mathbf{x}_i)^T(\mathbf{x}-\mathbf{x}_i), \tag{29}$$

where T denotes the vector transpose. This will blend, using the natural neighbor weights, not only the values z_i but also their corresponding gradients. In order to guarantee the correct slopes at the points $\mathbf{x}_i$ we also replace the weights $h_i(\mathbf{x})$ in Eq. (28) by the weights

$$w_i(\mathbf{x})=\sum_i \frac{h_i(\mathbf{x})d(\mathbf{x},\mathbf{x}_i)^{-1}}{\Sigma_i h_i(\mathbf{x})d(\mathbf{x},\mathbf{x}_i)^{-1}}$$

so that we finally obtain (Ref. 7, p. 13; Ref. 38, p. 30):

$$\hat{f}(\mathbf{x})=\sum_i w_i(\mathbf{x})g_i(\mathbf{x})=\sum_i \frac{h_i(\mathbf{x})d(\mathbf{x},\mathbf{x}_i)^{-1}}{\Sigma_i h_i(\mathbf{x})d(\mathbf{x},\mathbf{x}_i)^{-1}}g_i(\mathbf{x}). \tag{30}$$

The fact that $h_i(\mathbf{x})$ is nonzero only for the neighboring points of $\mathbf{x}_i$ causes this method to be local. As we can see, the weights used in natural neighbor interpolation define the amount of influence any neighboring scatter point will have on the computed value at the interpolation point. The weight depends on the area of influence (i.e., the area of the Voronoi polygons) of the surrounding scatter points: a larger area results in a larger weight or influence of the corresponding scatter point on the interpolated value.

This natural neighbor interpolant has some remarkable properties (Ref. 7, p. 13); among others, it can be shown (although the proof is rather involved) that it is indeed C^1 continuous, and that its values depend continuously on the data points $\mathbf{x}_i$.

Evaluation: Natural neighbor interpolation methods share with triangulation-based methods the unhappy burden of requiring a preprocessing stage that partitions the given scattered point set into a network of exhaustive and disjoint cells, but also the happy consequence of being *local*. However, all their other properties are sufficiently different to justify their classification in separate method families.

Natural neighbor interpolation is a weighted average method where the weights are area-based, as opposed to other methods (such as inverse distance methods) where

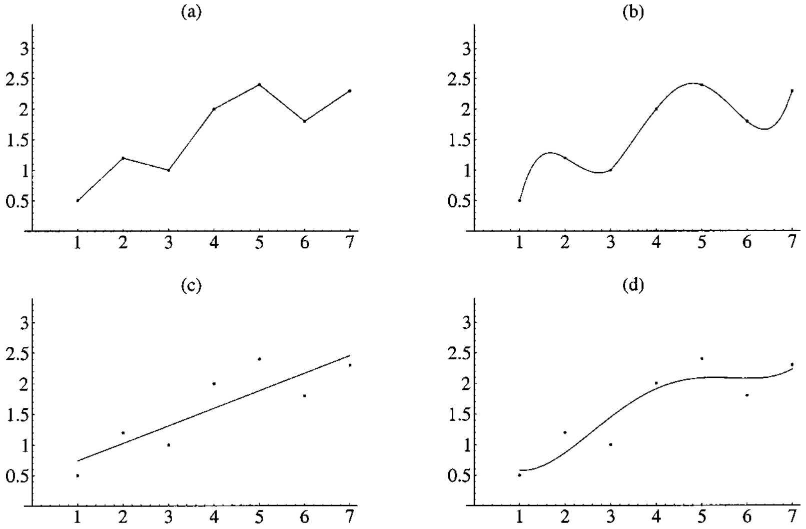


Fig. 15 The difference between interpolation (top figures) and best fit (bottom figures). In all figures the same seven data points are used. (a) Linear interpolation. (b) Fourth-order polynomial interpolation. (c) Linear best fit. (d) Best fit using polynomials up to order 4.

distance-based weights are used. Area-based weights are superior to distance-based weights because they compensate for data density variation—whereas distance-based interpolation is only sensitive to the distance and hence may overweight data clusters (Ref. 34, p. 82). Also, as compared to triangular-based interpolation methods, each point $\mathbf{x}_i$ is situated *within* its region of influence (Voronoi tile), and not on a vertex between regions (triangles). The fact that the tiles correspond to an area of influence about the given points is intuitively more appealing. All this makes the natural neighbor approach a robust method of scattered point interpolation, which performs equally well in clustered and in sparse areas of the given point set (Ref. 34, pp. 155–161).

This approach also avoids another pitfall of other methods, namely: the arbitrary definition of the interpolation subset [such as the radius R in Eq. (22) and in similar localization schemes]. It may be interesting to note that even the simpler C^0-continuous variant of this approach [Eq. (28)] gives better results than its C^0-continuous triangulation-based counterpart (Sec. 2.2)—since the natural neighbor method has derivative discontinuities only at the points $\mathbf{x}_i$, whereas the linear triangulation-based interpolation method has derivative discontinuities along all the triangle borders, and not only at the points $\mathbf{x}_i$.

However, in spite of its advantages, this approach is more computationally intensive and, hence, slower than other approaches, and all the more so in higher dimensions.

6 Scattered Data Interpolation Versus Scattered Data Fitting

Data fitting is an alternative approach to interpolation in finding intermediate values of an underlying function whose values are only known at a limited number of points. The approach of data fitting significantly differs from that of interpolation: While interpolation methods approximate the underlying function by finding a curve (or a surface, etc.) that passes *through* the known data points, data fitting methods find an approximating curve (or surface, etc.) that best fits the known data points according to some specified criteria. This means that the approximating function will pass *close* to the known values, but not necessarily exactly through them. This does not mean, however, that the quality of approximations calculated between the known points by data fitting is worse than in the case of interpolation: best fit methods have the advantage of being able to specify the range of the overall error, and their main concern is to minimize it. Furthermore, although data fitting misses the given points, it usually gives a smoother function than interpolation (see Figs. 15 and 16). The choice between interpolation and best fit approaches depends, therefore, on the nature and the specific needs of each particular application.

In general, interpolation through the data points is desirable if the values of the data points are highly accurate and reliable (e.g., if they are based on precise measurements), and if their number and density are not too high (see Fig. 15). However, when we are given a dense "cloud" of data points with relatively high noise (such as statistical data), the natural approach would be to fit the data by a smooth approximating curve (or surface, etc.) that passes in the center of the "data cloud" like a skeleton, and shows the average tendencies of the data. In such cases interpolation through all data points is not desirable, since it would result in high oscillations and poor approximation between the points, as shown in Fig. 16. Best fit approximation also appears in a natural way in overdetermined situations, i.e., when we have more data points than free parameters in the curve equation, so that we obtain a linear system with more equations than variables. This situation will be illustrated later.

Data fitting methods have been largely discussed in literature; see, for example, Ref. 4 where both global and local methods are reviewed. As an example of scattered data fitting we will describe here the regression method. We will explain this method by showing how it is used to establish a transformation that converts *RGB* input values of a

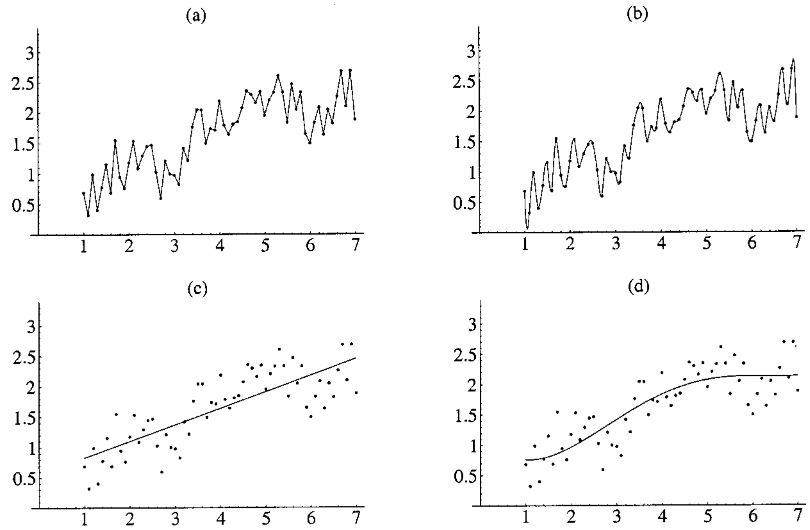


Fig. 16 Same as Fig. 14, when we are given a dense cloud of data points with relatively high noise (such as statistical data). In such cases interpolation through all points gives high oscillations (top), and the natural approach would be to fit the data by an approximating curve (bottom).

scanner into the corresponding *XYZ* values, based on a set of irregularly spaced points (e.g., Pantone color patches) for which the input values *R*,*G*,*B* as well as the target values *X*,*Y*,*Z* are known by measurement (Ref. 40, Chap. 3).

Assume that our scattered data set consists of eight sample points, whose coordinates in the input *RGB* space and their corresponding *XYZ* values are known to be

$$\begin{array}{lll} \text{Point } 1: & R_1,G_1,B_1 & X_1,Y_1,Z_1 \\ \vdots & \vdots & \vdots \\ \text{Point } 8: & R_8G_8,B_8 & X_8,Y_8,Z_8. \end{array}$$

The basic assumption of the regression method is that the conversion between the two color spaces (*RGB* and *XYZ*) can be approximated by three equations of the form

$$t=f_{a_1,\ldots,a_k}(x,y,z),$$

where the independent variables *x*,*y*,*z* stand for our input *R*,*G*,*B* values and the dependent variable *t* represents the resulting value of *X* (or *Y* or *Z*) in the destination color space.

Note that $f_{a_1,\ldots,a_k}(x,y,z)$ represents here an *equation family* with several free parameters $a_1,\ldots,a_k$. We are free to select the equation family according to our best judgment; it will be the task of the regression to give us the parameters of the *one* equation from this family that gives the best least square fit for the known sample points. For example, in cases where the underlying relationship seems to be exponential (or logarithmic), it would be reasonable to select an exponential (or, respectively, a logarithmic) equation family. In our present example we assume that the equation family consists of polynomials, since the solution for this case is simpler.

Suppose that we have chosen the following equation family:

$$\begin{aligned} t&=f_{a_1,\ldots,a_6}(x,y,z)\\ &=a_1x+a_2y+a_3z+a_4xy+a_5yz+a_6zx, \end{aligned} \tag{31}$$

i.e., a family of second-order polynomials in *x*, *y*, and *z*. This family has six free coefficients, $a_1,\ldots,a_6$; the aim of the method is, therefore, to find the specific values of these parameters for which the best fit occurs. It is important to note that the number of sample points must be higher than the number of parameters; in our case we have, indeed, 8 >6. Inserting the known values of our eight sample points into Eq. (31) we obtain, therefore, an overdetermined set of eight equations for the six unknowns $a_1,\ldots,a_6$**

$$t_1=a_1x_1+a_2y_1+a_3z_1+a_4x_1y_1+a_5y_1z_1+a_6z_1x_1$$

$$\vdots$$

$$t_8=a_1x_8+a_2y_8+a_3z_8+a_4x_1y_8+a_5y_8z_8+a_6z_8x_8,$$

where x_i, y_i, z_i stand for R_i, G_i, B_i, and t_i stands, for example, for X_i. This can be expressed in the matrix form as follows:

**Note that if the number of sample points were equal to the number of parameters we would obtain here a linear system of *n* equations with *n* unknowns; this system could be directly solved for the coefficients a_i for which Eq. (31) becomes an interpolating function that passes through all the given points. However, as we have already seen in the previous section, it is not generally guaranteed that this system of equations is nonsingular and has a solution, and even if it is nonsingular, it may be very ill-conditioned with an unstable solution. As we have seen, this method does work for some particular function families, like in the case of Hardy's multiquadric functions, but in the general case where a good solution is not guaranteed, it is best to have an overdetermined set of equations and to find the best coefficients a_i using the best fit approach described here.

$$\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & z_1 & x_1y_1 & y_1z_1 & z_1x_1 \\ x_2 & y_2 & z_2 & x_2y_2 & y_2z_2 & z_2x_2 \\ x_3 & y_3 & z_3 & x_3y_3 & y_3z_3 & z_3x_3 \\ x_4 & y_4 & z_4 & x_4y_4 & y_4z_4 & z_4x_4 \\ x_5 & y_5 & z_5 & x_5y_5 & y_5z_5 & z_5x_5 \\ x_6 & y_6 & z_6 & x_6y_6 & y_6z_6 & z_6x_6 \\ x_7 & y_7 & z_7 & x_7y_7 & y_7z_7 & z_7x_7 \\ x_8 & y_8 & z_8 & x_8y_8 & y_8z_8 & z_8x_8 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix}$$

or in short

$$\mathbf{t} = \mathbf{Ma}.$$

Now, it has been shown in literature (see, for example, Ref. 41, p. 1049) that the coefficients **a** that yield the best least square fit (i.e., that minimize the global square error on the sample points) are given by

$$\mathbf{a} = (\mathbf{M}^T\mathbf{M})^{-1}(\mathbf{M}^T\mathbf{t}),$$

where $\mathbf{M}^T$ is the transpose of matrix **M**. By inserting these values of $a_1, \ldots, a_6$ into equation family (31) above we obtain, therefore, the specific equation from this equation family that gives the best least-square approximation for the component *X* of the target color space. This polynomial regression method should be repeated two more times, for the *Y* and for the *Z* components.

Evaluation: Clearly, since the coefficients a_i are obtained by a global least square error minimization, the polynomial obtained may not map the sample *RGB* points to their original *XYZ* values. And yet, the data fit obtained by this method is normally good. The quality of this method depends on the relationship between the source and target spaces (i.e., the complexity of the underlying, unknown function), the number and location of the known sample points, the number of terms in the polynomial and its order, and the measurement errors. This method is ideal for transformations with linear relationship. For nonlinear color space conversion, this method does not guarantee uniform accuracy across the entire space, and some regions (such as dark colors) may have larger errors than other areas. In general, the accuracy improves as the number of terms in the equation increases, the trade-offs being the higher computation cost and lower processing speed (Ref. 40, p. 62).

Finally, this method is *global*, and hence, it suffers from the drawbacks mentioned in Sec. 1.

Note that instead of repeating the regression method three times, for the *X*, *Y*, and *Z* components, it is also possible to apply the method directly to the full 3D mapping $RGB \rightarrow XYZ$. For example, we may choose the equation family as follows:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \\ k_3 \end{pmatrix}$$

with the 12 free coefficients $a_{i,j}$ and k_i $(1 \leqslant i,j \leqslant 3)$. Note that in this example the regression will give as an approximation to the underlying 3D mapping a trivariate linear (or rather *affine*) mapping.

As an alternative to the approach of regression for scattered data fitting, one may formulate the problem in terms of the optimization approach, where the cost function is the total error (in terms of Euclidian distance) over all the samples, and the optimal coefficients a_i are those that minimize this cost function.

7 Conclusion

In this paper we survey some of the most important methods of scattered data interpolation in 2D and 3D that can be of interest in electronic imaging systems. The methods reviewed include triangulation (or tetrahedrization) based methods, inverse distance weighed methods, radial basis function methods, natural neighbor methods, as well as one data fitting method for the sake of comparison.

Triangulation based methods are local and, hence, capable of treating efficiently large data sets. They are computationally simple (once the prerequisite task of triangulation or tetrahedrization has been accomplished) and relatively accurate. Their serious drawback is the necessity of triangulating (or tetrahedrizing) the given data set prior to the actual interpolation step. We have presented several linear variants and one cubic variant of this family of methods; the latter has the advantage of being C^1 continuous and not only C^0 continuous as the simpler linear variants.

Inverse distance weighting methods are usually less efficient and less accurate than good triangulation-based methods, and they suffer from some well-known artifacts. However, due to their simplicity they are among the most commonly used techniques, in 2D as well as in higher dimensions.

Radial basis functions, and in particular Hardy's original multiquadric interpolation method, are among the more promissing methods in terms of fitting ability and visual smoothness. However, they require the solution of a linear equation system with at least as many equations and unknowns as the number of data points.

Natural neighbor interpolation methods are robust and perform equally well in clustered and in sparse areas of the given point set. However, in spite of their advantages, they are more computationally intensive and, hence, slower than other methods. They also require a preprocessing step that constructs the Voronoi tesselation of the given point set.

Finally, although not really passing through the data points, data fitting methods usually give smoother functions than interpolation, and they minimize the overall error. They should typically be used when we are given a "cloud" of data points with relatively high noise, or in overdetermined situations.

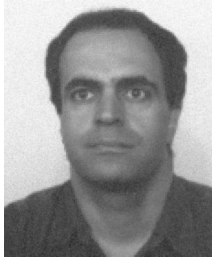
This review is certainly not complete, and it does not include all the existing methods. Just as one example, we did not mention here the family of stochastic process methods (such as "kriging") which is often used in the mining and geology community (Ref. 8, pp. 148–149). Such methods are less attractive for applications in electronic imaging systems since they necessitate the stationarity of input data, and their overall performance has been shown to be inferior to other methods such as Hardy's multiquadric method

(Ref. 37, pp. 198–199, 205). Readers who are interested in a wider spectrum of methods for scattered data interpolation may consult references such as Ref. 34, Sec. 2.5, or Refs. 4–8.

Clearly, none of the existing methods is universally satisfactory, and understanding their limitations is an important key in applying them successfully. It remains, therefore, the task of the designer or the engineer to select among all the different available methods the one that best suits his specific application. Several general guidelines can be found, for example, in Ref. 12, Chap. 6. Typical criteria for such a choice may include the type, the density, the amount and the particular properties of the data (see Sec. 3 in Ref. 8); the availability of working algorithms, software packages and computer programs (see Sec. 4 in Ref. 8); the computing environment available (including parameters such as speed, memory, etc.); the smoothness (C^0 or C^1 continuity) and the precision desired; the computational cost; and the programing efforts required. We hope the present review will be of help in the evaluation of the different possibilities and in taking a more founded decision.

References

1. D. N. Fogel, ''Image rectification with radial basis functions: applications to RS/GIS data integration,'' *Proceedings of the Third International Conference on Integrating GIS and Environmental Modeling*, Santa Fe (1996) http://www.ncgia.ucsb.edu/~fogel/mqregdoc.html
2. V. Ostromoukhov, R. D. Hersch, C. Péraire, P. Emmel, and I. Amidror, ''Two approaches in scanner-printer calibration: colorimetric space based vs. 'closed loop,' '' *Proc. SPIE* **2170**, 133–142 (1994).
3. *Pantone Color Formula Guide 1000*, Pantone Inc., New Jersey (1991–1992).
4. L. L. Schumaker, ''Fitting surfaces to scattered data,'' in *Approximation Theory II*, G. G. Lorentz, C. K. Chui, and L. L. Schumaker, Eds., pp. 203–268, Academic, New York, (1976).
5. R. E. Barnhill, ''Representation and approximation of surfaces,'' in *Mathematical Software III*, J. R. Rice, Ed., pp. 69–120, Academic, New York (1977).
6. R. Franke, ''Scattered data interpolation: tests of some methods,'' *Math. Comput.* **38**, 181–200 (1982).
7. P. Alfeld, ''Scattered data interpolation in three or more variables,'' in *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker, Eds., pp. 1–33, Academic, New York (1989).
8. R. Franke and G. M. Nielson, ''Scattered data interpolation and applications: a tutorial and survey,'' in *Geometric Modelling Methods and Applications*, H. Hagen and D. Roller, Eds., pp. 131–160, Springer, Berlin (1991).
9. T. A. Foley and H. Hagen, ''Advances in scattered data interpolation,'' *Surv. Math. Ind.* **4**(2), 71–84 (1994).
10. G. M. Nielson, H. Hagen, and H. Müller, *Scientific Visualisation*, IEEE, New York (1997).
11. R. N. Bracewell, *Two-Dimensional Imaging*, pp. 247–257, Prentice-Hall, NJ, Englewood Cliffs (1995).
12. P. Lancaster and K. Salkauskas, *Curve and Surface Fitting*, Academic, London (1986).
13. T. M. Lehmann, C. Gönner, and K. Spitzer, ''Survey: interpolation methods in medical image processing,'' *IEEE Trans. Med. Imaging* **18**(11), 1049–1075 (1999).
14. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry—Algorithms and Applications*, Springer, Berlin (1997).
15. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, 4th ed., Academic, San Diego (1997).
16. J. D. Boissonnat, ''An hierarchical representation of objects: the Delaunay tree,'' *Proceedings of the 2nd ACM Symposium on Computational Geometry*, pp. 260–268, Yorktown Heights (June 1986).
17. R. E. Barnhill, ''Surfaces in computer aided geometric design: a survey with new results,'' *Comput. Aided Des.* **2**, 1–17 (1985).
18. P. C. Hung, ''Colorimetric calibration in electronic imaging devices using a look-up table model and interpretations,'' *J. Electron. Imaging* **2**(1), 53–61 (1993).
19. S. I. Nin, J. M. Kasson, and W. Plouffe, ''Printing CIELAB images on a CMYK printer using tri-linear interpolation,'' *Proc. SPIE* **1670**, 316–324 (1992).
20. J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed., Wesely, Reading, MA (1990).
21. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer, New York (1985).
22. G. M. Nielson and R. Franke, ''Surface construction based upon triangulations,'' in *Surfaces in CAGD*, R. N. Barnhill and W. Boehm, Eds., pp. 163–177, North Holland, Amsterdam (1983).
23. S. A. Stead, ''Estimation of gradients from scattered data,'' *Rocky Mt. J. Math.* **14**(1), 265–279 (1984).
24. W. Böhm, G. Farin, and J. Kahmann, ''A survey of curve and surface methods in CAGD,'' *Comput. Aided Des.* **1**, 1–60 (1984).
25. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ (1973).
26. G. Farin, ''A modified Clough–Tocher interpolant,'' *Comput. Aided Des.* **2**, 19–27 (1985).
27. A. J. Worsey and G. Farin, ''An *n*-dimensional Clough–Tocher interpolant,'' *Constructive Approximation* **3**, 99–110 (1987).
28. G. Farin, ''Triangular Bernstein–Bézier patches,'' *Comput. Aided Des.* **3**, 83–127 (1986).
29. G. Farin, ''Smooth interpolation to scattered 3D data,'' *Surfaces in CAGD*, R. N. Barnhill and W. Boehm, Eds., pp. 43–63, North Holland, Amsterdam (1983).
30. D. Salesin, D. Lischinski, and T. DeRose, ''Reconstructing illumination functions with selected discontinuities,'' *Proc. of the Third Eurographics Workshop on Rendering*, pp. 99–112 (1992).
31. D. H. McLain, ''Two dimensional interpolation from random data,'' *Comput. J. (UK)* **19**(2), 178–181 (1976); Errata, p. 384.
32. D. Shepard, ''A two-dimensional interpolation function for irregularly spaced data,'' *Proceedings of the 23rd ACM National Conference*, pp. 517–524, ACM, NY (1968).
33. W. J. Gordon and J. A. Wixom, ''Shepard's method of 'metric interpolation' to bivariate and multivariate interpolation,'' *Math. Comput.* **32**(141), 253–264 (1978).
34. D. F. Watson, *Contouring: A Guide to the Analysis and Display of Spatial Data*, Pergamon, Oxford (1992).
35. R. J. Renka, ''Multivariate interpolation of large sets of scattered data,'' *ACM Trans. Math. Softw.* **14**(2), 139–148 (1988).
36. R. L. Hardy, ''Multiquadric equations of topography and other irregular surfaces,'' *J. Geophys. Res.* **76**(8), 1905–1915 (1971).
37. R. L. Hardy, ''Theory and applications of the multiquadric-biharmonic method, 20 years of discovery 1968–1988,'' *Comput. Math. Appl.* **19**(8/9), 163–208 (1990).
38. R. Sibson, ''A brief description of natural neighbour interpolation,'' in *Interpreting Multivariate Data*, V. Barnett Ed., pp. 21–36, Wily, Chichester (1981).
39. J. Braun and M. Sambridge, ''A numerical method for solving partial differential equations on highly irregular evolving grids,'' *Nature (London)* **376**, 655–660 (1995).
40. H. R. Kang, *Color Technology for Electronic Imaging Devices*, SPIE, Bellingham, WA, 1997.
41. E. W. Weisstein, *CRC Concise Encyclopedia of Mathematics*, CRC Press, Boca Raton, FL (1999).



Isaac Amidror received his BSc degree in mathematics from the Hebrew University of Jerusalem, Israel, and his MSc degree in computer science from the Weizmann Institute of Science in Rehovot, Israel. He received a Japanese government scholarship for a two-year research period in the Computer Science Department of the Toyohashi University of Technology in Japan. After having worked a few years in industry (notably in the fields of laser printing and digital typography), he received his PhD degree in the Swiss Federal Institute of Technology, in Lausanne, Switzerland. He has published numerous scientific papers and is the author of a recent book on the theory of the moiré phenomenon (http://lspwww.epfl.ch/books/moire/). His research interests include the mathematical foundations of moiré phenomena, document security, colour image reproduction, image processing, and digital typography.