FUNCTION APPROXIMATION

This article presents a survey of techniques used for function approximation and free form surface reconstruction. A comparative study is performed between classical interpolation methods and two methods based on neural networks. We show that the neural networks approach provides good approximation and provides better results than classical techniques when used for reconstructing smoothly varying surfaces.

The interpolation problem, in its strict sense, may be stated as follows (1): Given a set of N different points: $\{x_i \in \Re^m | i = 1, 2, \ldots, N\}$ and a corresponding set of N real numbers: $\{d_i \in \Re^n | i = 1, 2, \ldots, N\}$ find a function F:

$$F : \Re^m \rightarrow \Re^n | F(x_i) = d_i, i = 1, 2, \ldots, N \tag{1}$$

where m and n are integers. The interpolation surface is constrained to pass through all the data points. The interpolation function can take the form:

$$F(x) = \sum_{i=1}^{N} w_i \phi(x, x_i) \tag{2}$$

where $\{\phi(x, x_i) | i = 1, 2, \ldots, N\}$ is a set of N arbitrary functions known as the radial basis functions. Inserting the interpolation conditions, we obtain the following set of simultaneous linear equations for the unknown coefficients (weights) of the expansion w_i:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdot & \cdot & \phi_{1N} \\ \phi_{21} & \phi_{22} & \cdot & \cdot & \phi_{2N} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \phi_{N1} & \phi_{N2} & \cdot & \cdot & \phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \cdot \\ \cdot \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \cdot \\ \cdot \\ d_N \end{bmatrix} \tag{3}$$

where

$$\phi_{ji} = \phi(x, x_i), j, i = 1, 2, ..N \tag{4}$$

Let the $N \times 1$ vectors $\boldsymbol{d}$ and $\boldsymbol{W}$ defined as

$$\boldsymbol{d} = [d_1, d_2, \ldots, d_N]^t \tag{5}$$

$$\boldsymbol{W} = [w_1, w_2, \ldots, w_N]^t \tag{6}$$

represent the desired response vector and the linear weight vector, respectively. Let

$$\Phi = \{\phi_{i,j}, i, j \in [1, N]\} \tag{7}$$

denote the $N \times N$ interpolation matrix. Hence, Eq. (3) can be rewritten as

$$\Phi \boldsymbol{W} = \boldsymbol{d} \tag{8}$$

Provided that the data points are all distinct, the interpolation matrix Φ is positive definite (1). Therefore, the weight vector $\boldsymbol{W}$ can be obtained by

$$\boldsymbol{W} = \Phi^{-1}\boldsymbol{d} \tag{9}$$

where Φ^{-1} is the inverse of the interpolation matrix Φ. Theoretically speaking, a solution to the system in Eq. (9) always exits. Practically speaking, however, the matrix Φ can be singular. In such cases, regularization theory can be used; where the matrix Φ is perturbed to $\Phi + \lambda\mathbf{I}$ to assure positive definiteness (1).

Based on the interpolation matrix Φ, different interpolation techniques are available (2–5). Some of these techniques will be reviewed in next section.

CLASSICAL INTERPOLATION TECHNIQUES

Shepard's Interpolation

Shepard formulated an explicit function for interpolating scattered data (16). The value of the modeling function is calculated as a weighted average of data values according to their distance from the point at which the function is to be evaluated. Shepard's expression for globally modeling a surface is

$$S(x,y) = \begin{cases} \dfrac{\sum_{i=1}^{n} Z_i/r_i^2}{\sum_{i=1}^{n} 1/r_i^2} & r_i \neq 0 \\ z_i & r_i = 0 \end{cases} \tag{10}$$

where r_i is the standard distance metric:

$$r_i = [(x-x_i)^2 + (y-y_i)^2]^{1/2} \tag{11}$$

The above algorithm is global; local Shepard methods can be formed by evaluating S(x, y) from the weighted data values within a disk of radius R from (x, y). A function $\Psi(r)$ is defined that ensures the local behavior of the interpolating method by calculating a surface model for any $r <= R$, and which also weights the points at $r <= R/3$ more heavily, as follows:

$$\Psi(r) = \begin{cases} 1/r & 0 < r < R/3 \\ 27(r/R - 1)^2/4R & R/3 < r \le R \\ 0 & R < r \end{cases} \tag{12}$$

Therefore, for points that are within R distance of (x, y), the surface is given by

$$S(x,y) = \begin{cases} \dfrac{\sum_{i=1}^{n} z_i \psi(r_i)^2}{\sum_{i=1}^{n} \psi(r_i)^2} & r_i \neq 0 \\ z_i & r_i = 0 \end{cases} \tag{13}$$

A usable number of points must fall within the local region of radius R for this method to be applicable. Shepard's method is simple to implement and can be localized, which is advantageous for the large cloud data sets (3).

Thin Plate Splines

The method of thin plate splines proposed by Shepard (2) and the multiquadric method proposed by Hardy (5) can both be classified as interpolating functions composed of a sum of radial basis functions. The basis functions are radially symmetric about the points at which the interpolating function is evaluated. Conceptually, the method is simple to understand in terms of a thin, deformable plate passing through the data points collected off the surface of the object. The thin plate spline radial basis functions are obtained from the solution of minimizing the energy of the thin plate constrained to pass through loads positioned at the cloud data set. The modeling surface is constructed from the radial basis functions $\beta_i(x, y)$ by expanding them in a series of $(n + 3)$ terms with c_i coefficients:

$$S(x,y) = \sum_{i=1}^{n} c_i \beta_i(x,y) \tag{14}$$

where the basis functions are given by

$$\beta_i(x,y) = r_i^2 ln(r_i) \tag{15}$$

The modeling surface function $S(x, y)$ has the form derived in Harder and Desmairis (11)

$$S(x,y) = a_0 + a_1 x + a_2 y + \sum_{i=1}^{n} c_i r_i^2 ln(r_i) \tag{16}$$

The coefficients are determined by substituting the discrete data set into and solving the resulting set of linear equations:

$$\sum_{i=1}^{n} c_i = 0 \tag{17}$$

$$\sum_{i=1}^{n} x_i c_i = 0 \tag{18}$$

$$\sum_{i=1}^{n} y_i c_i = 0 \tag{19}$$

$$f(x_i, y_i) = a_0 + a_1 x + a_2 y + \sum_{i=1}^{n} c_i r_i^2 ln(r_i) \tag{20}$$

Hardy's Multiquadric Interpolation. Hardy (5) proposed a method for interpolating scattered data that employs the summation of equations of quadratic surfaces that have unknown coefficients. The multiquadric basis functions are the hyperbolic quadrics

$$\phi_i = (r_i^2 + b^2)^{1/2} \tag{21}$$

where b is a constant and r_i is the standard Euclidean distance metric. The summation of a series of hyperbolic quadrics have been found to perform best compared with the other members of the multiquadrics family. The modeling surface is given by

$$S(x,y) = \sum_{i=1}^{n} c_i \phi_i = \sum_{i=1}^{n} c_i [(x-x_i)^2 + (y-y_i)^2 + b^2]^{1/2} \tag{22}$$

The cloud data set is substituted into Eq. (22), giving set of linear equations

$$z_i = \sum_{i=1}^{n} c_i [(x_j - x_i)^2 + (y_j - y_i)^2 + b^2]^{1/2} \quad j = 1, \ldots, n \tag{23}$$

NEURAL NETWORK AS UNIVERSAL APPROXIMATOR

Although classification is a very important form of neural computation, neural networks could be used to find an approximation of a multivariable function $F(x)$ (6). This could be approached through a supervised training of an input-output mapping from a data set. The learning proceeds as a sequence of iterative weight adjustments until a weight vector is found that satisfies certain criterion.

In a more formal approach, multilayer networks can be used to map $\Re^n$ into $\Re$ by using P examples of the function $F(x)$ to be approximated by performing nonlinear mapping with continuous neurons in the first layer, and then computing the linear combination by the single node of the output layer as follows:

$$y = \Gamma[VX] \tag{24}$$

$$O = W^t y \tag{25}$$

where V and W are the weight matrices for hidden and output layer respectively, and $\Gamma[\cdot]$ is a diagonal operator matrix consisting of nonlinear squashing functions $\phi(\cdot)$

$$\Gamma = \begin{bmatrix} \phi(\cdot) & 0 & 0 & \cdot & 0 \\ 0 & \phi(\cdot) & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \phi(\cdot) \end{bmatrix} \tag{26}$$

A function $\phi(\cdot): \Re \rightarrow [0, 1]$ is a squashing function if (1) it is nondecreasing, (2) $lim_{\lambda\rightarrow\infty}\ \phi(\lambda) = 1$, (3) $lim_{\lambda\rightarrow-\infty}\ \phi(\lambda) = 0$. Here we have used a bipolar squashing function of the form

$$\phi(x) = \frac{2}{1+e^{-\lambda x}} - 1 \tag{27}$$

The studies of Funanashi (8), Hornik, Stinchcombe and White (7) prove that multilayer feedforward networks perform as a class of universal approximators. Although the concept of nonlinear mapping, followed by linear mapping, pervasively demonstrates the approximating potential of neural networks, the majority of the reported studies have dealt with second layer also providing the nonlinear mapping (6,7). The general network architecture performing the nested nonlinear scheme consists of a single hidden layer and a single output O such that

$$O = \Gamma(W\Gamma[VX]) \tag{28}$$

This standard class of neural networks architecture can approximate virtually any multivariable function of interest provided that sufficiently many hidden neurons are available.

Approximation Using Multilayer Networks

A 2-layer network was used for surface approximation. The x and y coordinates of the data points were the input to the network, while the function value $F(x, y)$ was the desired response d.

The learning algorithm applied was the error back-propagation learning technique. This technique calculates an error signal at the output layer and uses the signal to adjust network weights in the direction of the negative gradient descent of the network error E so that, for a network with I neurons in the input layer, J neurons in the hidden layer, and K neurons the output layer, the weight adjustment is as follows:

$$\Delta w_{kj} = -\eta\frac{\partial E}{\partial w_{kj}}, k = 1, 2, \ldots, K \quad j = 1, 2, \ldots J \tag{29}$$

$$\Delta v_{ji} = -\eta\frac{\partial E}{\partial v_{ji}}, j = 1, 2, \ldots, J \quad i = 1, 2, \ldots I \tag{30}$$

where

$$E = \frac{1}{2}\sum_{k=1}^{K}(d_k - O_k)^2 \tag{31}$$

The size J of the hidden layer is one of the most important considerations when solving actual problems using multilayer feedforward networks. The problem of the size choice is under intensive study, with no conclusive answers available thus far for many tasks. The exact analysis of the issue is rather difficult because of the complexity of the network mapping and the nondeterministic nature of many successfully completed training procedures (6). Here, we tested the network using different number of hidden neurons. The degree of accuracy reflected by mean square error was chosen to be 0.05. Results are provided later in this paper.

Approximation Using Functional Link Networks

Instead of carrying out a multistage transformation, as in multilayer networks, input/output mapping can also be achieved through an artificially augmented single-layer network. The concept of training an augmented and expanded network leads to the so-called functional link network as introduced by Pao (1989) (10). Functional link networks are single-layer neural networks that can handle linearly nonseparable tasks by using appropriately enhanced representation. This enhanced representation is obtained by augmenting the input by higher order terms that are generally nonlinear functions of the input.

The functional link network was used to approximate the surfaces by enhancing the 2-component input pattern (x, y) by 26 orthogonal components such as xy, $\sin(n\pi x)$, $\cos(n\pi y)$, *etc.* for $n = 1, 2, \ldots, m$. The output of the network can be expressed as follows:

$$\begin{aligned} F(x,y) = & xw_x + yw_y + xyw_{xy} \\ & + \sum_{i=1}^{m}\sin(i\pi x)w_{xi} + \cos(i\pi x)w_{xi} \\ & + \sum_{i=1}^{m}\sin(i\pi y)w_{yi} + \cos(i\pi y)w_{yi} \end{aligned} \tag{32}$$

The basic mathematical theory indicates that the functional expansion model should converge to a flat-net solution if a large enough number of additional independent terms are used.

HYPERSURFACE RECONSTRUCTION AS AN ILL-POSED PROBLEM

For the following reason, the strict interpolation procedure described here may be a poor strategy for training of function approximators (for certain classes) because of poor new data generalization: When the number of data points in the training set is much larger than the number of degrees of freedom of the underlying physical process, and we are constrained to have as many basis functions as data points, the problem is overdetermined. Consequently, the algorithm may end up fitting misleading variations because of noise in the input data, and thereby result in a degraded generalization performance (1).

The approximation problem belongs to a generic class of problems called *inverse problems.* An inverse problem may be *well posed* or *ill posed.* The term *well posed* has been used in applied mathematics since the time of Hadamard in the early 1900s. To explain what we mean by this term, assume that we have a domain X and a range Y taken to be metric spaces, and that they are related by a fixed but unknown mapping F. The problem of reconstructing the mapping F is said to be well posed if three conditions are satisfied (1):

1. *Existence.* For every input vector $\boldsymbol{x} \in X$, there does exist an output $y = F(\mathbf{x})$, where $y \in Y$.
2. *Uniqueness.* For any pair of input vectors $\boldsymbol{x}, \boldsymbol{t} \in X$, we have $F(\boldsymbol{x}) = F(\boldsymbol{t})$ if, and only if, $\boldsymbol{x} = \boldsymbol{t}$.
3. *Continuity.* The mapping is continuous, that is for any $\epsilon > 0$ there exists $\delta = \delta(\epsilon)$ such that the condition $d_X(\boldsymbol{x}, \boldsymbol{t}) < \delta$ implies that $d_Y(F(\boldsymbol{x}), F(\boldsymbol{t})) < \epsilon$, where $d(\cdot\,,\ \cdot)$ is the distance between two arguments in their respective spaces.

If these conditions are not satisfied, the inverse problem is said to be *ill posed.* Function approximation is an ill posed inverse problem for the following reasons. First there is not as much information in the training data as we really need to reconstruct the input-output mapping uniquely, hence the uniqueness criterion is violated. Second, the presence of noise or imprecision in the input data adds uncertainty to the reconstructed input-output mapping in such a way that an output may be produced outside the range for a giving input inside the domain, in other words, when the continuity condition is violated.

Regularization Theory

Tikhonov (12) proposed a method called *regularization* for solving ill-posed problems. In the context of approximation problems, the basic idea of regularization is to stabilize the solution by means of some auxiliary nonnegative functional that embeds prior information, for example, smoothness constraints on the input-output mapping (that is, a solution to the approximation problem), and thereby make an ill-posed problem into a well-posed one (1,11).

According to Tikhonov's regularization theory, the function F is determined by minimizing a cost functional $\mathscr{E}(F)$, so called because it maps functions (in some suitable function space) to the real line. It involves two terms:

1. *Standard Error Term.* This first term, denoted by $\mathscr{E}_s(F)$, measures the standard error between the desired response d_i and the actual response y_i for training example $i = 1, 2, \ldots, N$. Specifically,

$$\begin{aligned}\mathscr{E}_s(F) &= \sum_{i=1} N(d_i - y_i)^2 \\ &= \sum_{i=1} N[d_i - F(\mathbf{x}_i)]^2\end{aligned} \tag{33}$$

2. *Regularization Term.* This second term, denoted by $\mathscr{E}_c(F)$, depends on the geometric properties of the approximating function $F(\mathbf{x})$. Specifically, we write

$$\mathscr{E}_c(F) = \|\mathbf{P}F\|^2 \tag{34}$$

where $\mathbf{P}$ is a linear differential operator. Prior information about the form of the solution is embedded in the operator $\mathbf{P}$, which naturally makes the selection of $\mathbf{P}$ problem dependent. $\mathbf{P}$ is referred to as a stabilizer in the sense that it stabilizes the solution F, making it smooth and therefore continuous.

The analytical approach used for the situation described here draws a strong analogy between linear differential operators and matrices, thereby placing both types of models in the same conceptual framework. Thus the symbol $\|.\|$ denotes a norm imposed on the function space to which $\mathbf{P}F$ belongs. By a function space we mean a normed vector space of functions. Ordinarily, the function space used here is the L^2 space that consists of all real-valued functions $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathscr{R}^p$, for which $|f(\boldsymbol{x})|^2$ is Lebesgue integrable. The function $f(\boldsymbol{x})$ denotes the actual function that defines the underlying physical process responsible for the generation of the input-output pair. Strictly speaking, we require the function $f(\boldsymbol{x})$ to be a member of a reproducing kernel Hilbert space (RKHS) with a reproducing kernel in the form of the Dirac delta distribution (14). The simplest RKHS that satisfies the previously mentioned conditions is the space of rapidly decreasing, infinitely continuous differentiable functions, that is, the classical space S of rapidly decreasing test functions for the Shawrz theory of distributions, with finite $\mathbf{P}$-induced norm

$$H_p = \{f \in S : \|\mathbf{P}f\| < \infty\} \tag{35}$$

where the norm of $\mathbf{P}f$ is taken with respect to the range of $\mathbf{P}$, assumed to be another Hilbert space. The principal of regularization may now be stated as follows: Find the function $F(\mathrm{x})$ that minimizes the cost functional $\mathscr{E}(F)$ defined by

$$\begin{aligned}\mathscr{E}(F) &= \mathscr{E}_s(F) + \lambda\mathscr{E}_c(F) \\ &= \sum_{i=1} N[d_i - F(\mathbf{x}_i)]^2 + \lambda\|\mathbf{P}F\|^2\end{aligned} \tag{36}$$

where λ is a positive real number called regularization parameter.

Regularization Networks

Poggio et al., (13) suggested some form of prior information about the input-output mapping that would make the learning problem well posed so that the generalization to new data is feasible. They also suggested a network structure that they called *regularization network.* It consists of three layers. The first layer is composed of input (source) nodes whose number is equal to the dimension p of the input vector $\boldsymbol{x}$. The second layer is a hidden layer, composed of nonlinear units that are connected directly to all the nodes in the input layer. There is one hidden unit for each data point x_i, $i = 1, 2, \ldots, N$, where N is the number of training examples. The activation of the individual hidden units are defined by Green's function $G(x, x_i)$ given by (20)

$$G(x, x_i) = \exp\left[-\frac{1}{2\sigma_i^2}\sum_{k=1}^{p}(x_k - x_{i,k})^2\right] \tag{37}$$

This Green's function is recognized to be a multivariate Gaussian function. Correspondingly, the regularized solution takes on the following special form:

$$F(x) = \sum_{i=1}^{N} w_i \exp\left[-\frac{1}{2\sigma_i^2}\sum_{k=1}^{p}(x_k - x_{i,k})^2\right] \tag{38}$$

which consists of a linear superposition of multivariate Gaussian basis functions with center x_i (located at the data points) and widths σ_i (1).

The output layer consists of a single linear unit, and is fully connected to the hidden layer. The weights of the output layer are the unknown coefficients of the expansion, defined in terms of the Green's functions $G(x; x_i)$ and the regularization parameter λ by

$$w = (G + \lambda I)^{-1} d \tag{39}$$

This regularization network assumes that the Green's function $G(x; x_i)$ is positive definite for all i. Provided that this condition is satisfied, which it is in the case of the $G(x; x_i)$ having the Gaussian form, for example, then the solution produced by this network will be an optimal interpolant in the sense that it minimizes the functional $\mathscr{E}(F)$. Moreover, from the viewpoint of approximation theory, the regularization network has three desirable properties (17):

1. The regularization network is a universal approximator in that it can approximate arbitrarily well any multivariate continuous function on a compact subset $\mathscr{R}^p$, given a sufficiently large number of hidden units.
2. Since the approximation scheme derived from regularization theory is linear in the unknown coefficients, it follows that the regularization network has the best approximation property. This means that given an unknown nonlinear function f, there always exists a choice of coefficients that approximates f better than all other possible choices.
3. The solution computed by the regularization network is optimal. Optimality here means that the regularization minimizes a functional that measures how much the solution deviates from its true value as represented by the training data.

RESULTS

We now quantitatively compare the performance of the classic techniques shown in Section 1 with the neural network approaches using synthetic range data for four typical free-form surface patches suggested by Bradley and Vickers (3), and two surfaces suggested in this paper. The six test surfaces are:

1. Surface 1:

$$z = \sin(0.5x) + \cos(0.5y) \tag{40}$$

2. Surface 2:

$$z = \sin(x) + \cos(y) \tag{41}$$

3. Surface 3:

$$z = e^{-(x-5)^2 + (y-5)^2/4} \tag{42}$$

4. Surface 4:

$$z = \tanh(x + y - 11) \tag{43}$$

5. Surface 5:

$$z = \text{Rectangular Box} \tag{44}$$

6. Surface 6:

$$z = \text{Pyramid} \tag{45}$$

The first two surfaces are bivariate sinusoidal functions, with Surface 2 having twice the spatial frequency of Surface 1. Many consumer items are composed of smoothly varying free-form patches similar to Surface 1, while Surface 2 is less common. Surface 3 has a peaked form, and Surface 4 is smooth with a sharp ridge diagonally across it. Surfaces 5 and 6 have

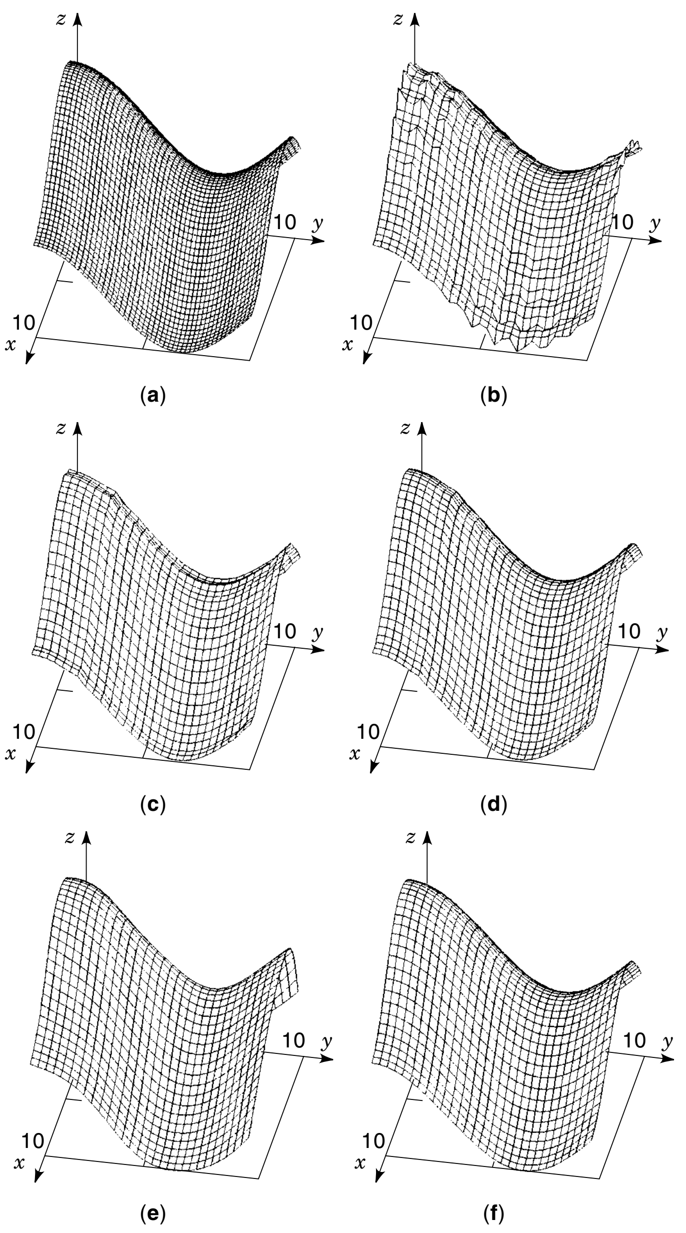

Figure 1. Comparison of reconstruction of Surface 1 using all methods: (a) original surface; (b) Shepard Interpolation; (c) Thin Plate B-spline; (d) Hardy's Multiquadric Interpolation; (e) MultiLayer Neural Network; and (f) Functional link Neural Network.

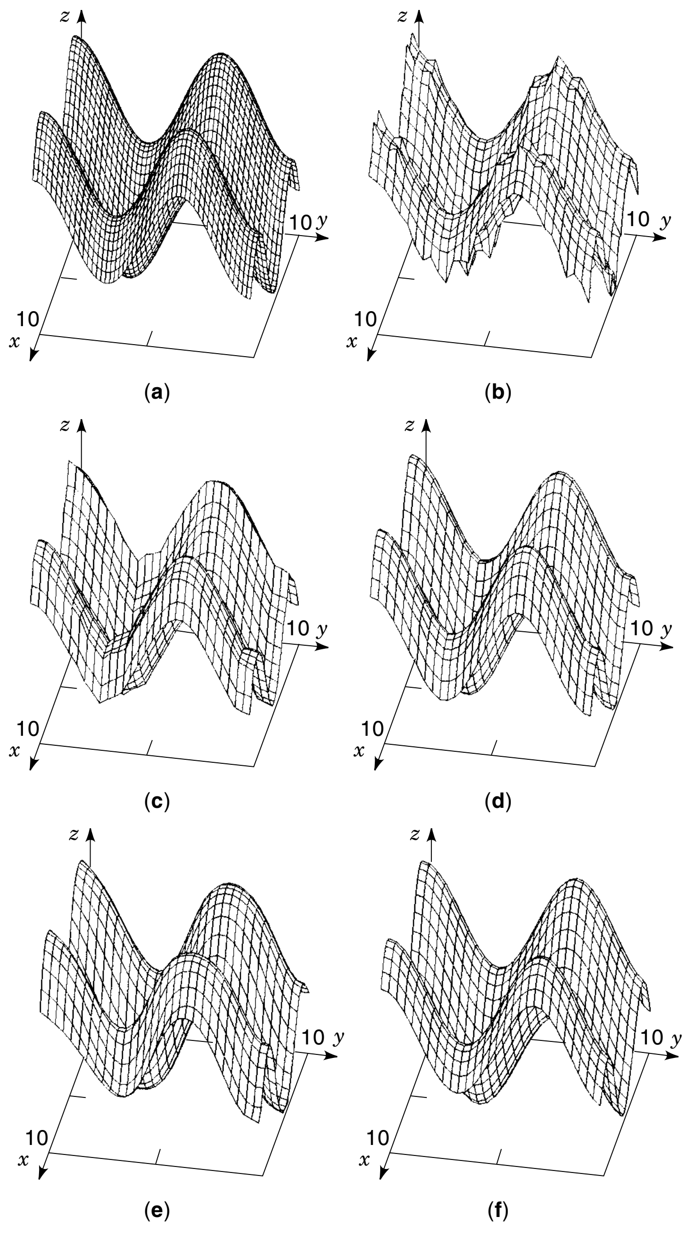

Figure 2. Comparison of reconstruction of Surface 2 using all methods: (a) original surface; (b) Shepard Interpolation; (c) Thin Plate B-spline; (d) Hardy's Multiquadric Interpolation; (e) MultiLayer Neural Network; and (f) Functional link Neural Network.

sharp edges and were included to test the modeling techniques on discontinuous surfaces. Data sets were generated, using the six surfaces, with each set consisting of 2500 points contained in a rectangular patch and with x and y varying from 0.0 to 10.0. Testing of these surface fitting methods has been done by local interpolation over 8 $\times$ 8 overlapping square regions. All methods were applied to a data set mesh of 900 points contained in the same rectangular patch as the original data set (9).

From the results, we can deduce that the Hardy's Multiquadratic Interpolation provides a large improvement over the Shepard's Interpolation and the Thin Plate Splines method for the first four surfaces, while performing the worst of the three methods for the sharp edge surfaces. The reconstructed surfaces, using all interpolation techniques, are shown in Figs. 1–6.

The interpolations using the neural networks perform exceptionally well on the first four surfaces, but because of the sharp edges in Surfaces 5 and 6, their performance was not as good. The networks seem to have difficulty with sharp transitions and discontinuities. A method for dealing with this difficulty could be to use a block training technique in which the neural network learns the surfaces in smaller patches. This should decrease learning time and make the training cycles less complicated, although it creates the need for local regions. During training, the weights are updated for each training point; the error for that training point and corresponding weight set is then calculated. When using a large number of training pairs, the weights are significantly changed from the beginning to the end of the training cycle. Therefore, the calculated error is not equivalent to the true error, which is based on this final weight set. This is because

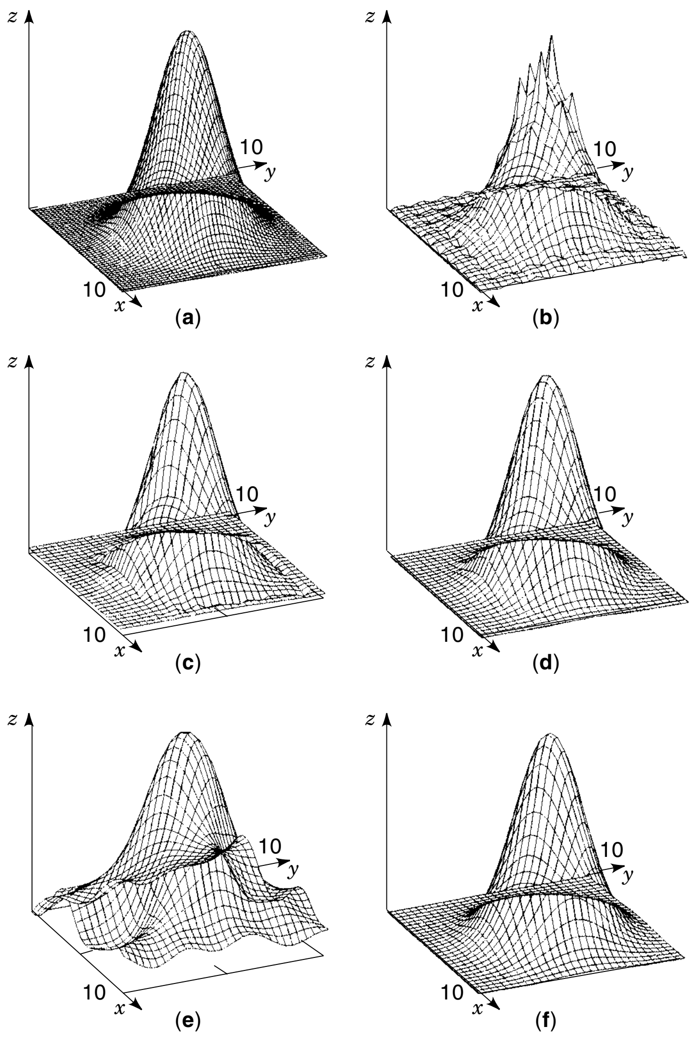

Figure 3. Comparison of reconstruction of Surface 3 using all methods: (a) original surface; (b) Shepard Interpolation; (c) Thin Plate B-spline; (d) Hardy's Multiquadric Interpolation; (e) MultiLayer Neural Network; and (f) Functional link Neural Network.

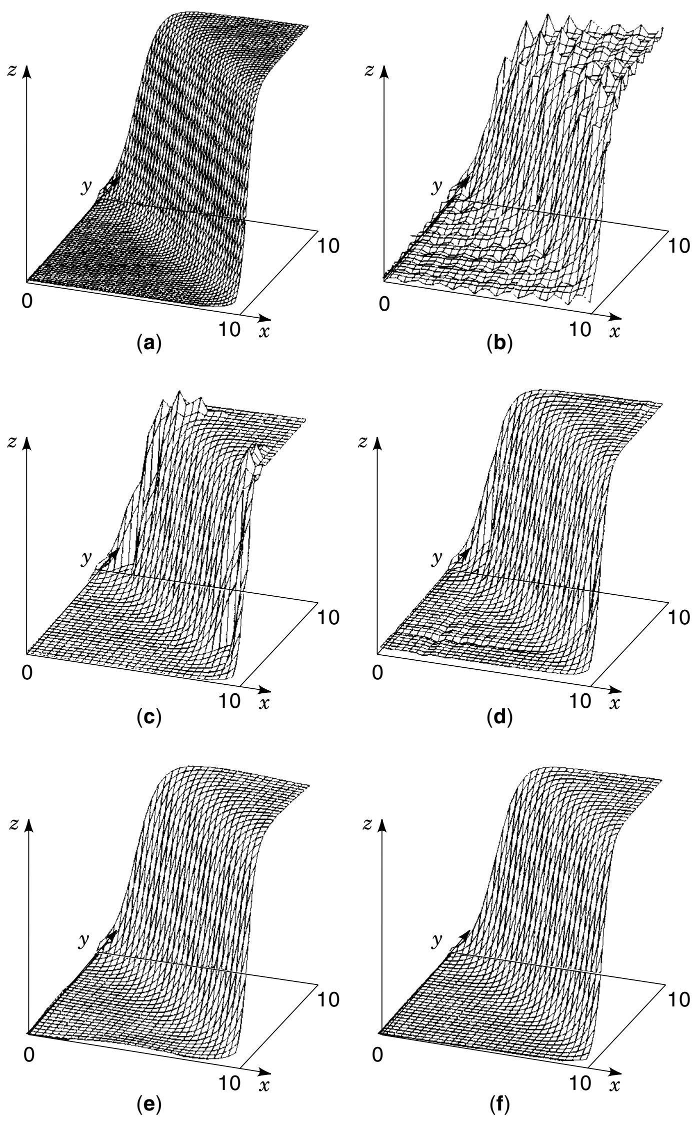

Figure 4. Comparison of reconstruction of Surface 4 using all methods: (a) original surface; (b) Shepard Interpolation; (c) Thin Plate B-spline; (d) Hardy's Multiquadric Interpolation; (e) MultiLayer Neural Network; and (f) Functional link Neural Network.

each error calculation has been made with a different weight set. A correction for this can be done by simply recalculating the true error using the final weight set at the end of each training cycle.

It is clear that the neural networks method produces smooth surfaces as compared with those produced by Hardy's Multiquadratic Interpolation without the need of constructing local regions. In general, the neural networks approach is far superior in terms of the ease of implementation. The results also show the potential of Functional Link Neural Network as an approximator; it is easier to implement than the MultiLayer Neural Network and faster to converge (9). However, neither of the two networks performed well on Surface 6, possibly because of the presence of two discontinuities.

A new approach for free-form surface modeling is to combine neural networks with classical techniques to create a new hybrid interpolation method. The surface can be divided into smoothly varying portions and areas of sharp transitions. The neural network could then be used to approximate these smoothly varying areas, while a classical technique like Hardy's Multiquadric Interpolation could be used on the areas of sharp transition.

APPLICATIONS

Recently, laser range finders (also known as 3-D laser scanners) have been employed to scan multiple views of an object. The scanner output is usually an unformatted file of large size (known as the "cloud of data") (3,31). In order to use the cloud of data for surface reconstruction, a registration technique is implemented that makes correspondence with the actual surface. Laser scanners are convenient in applications where the object is irregular but in general smooth. The corre-

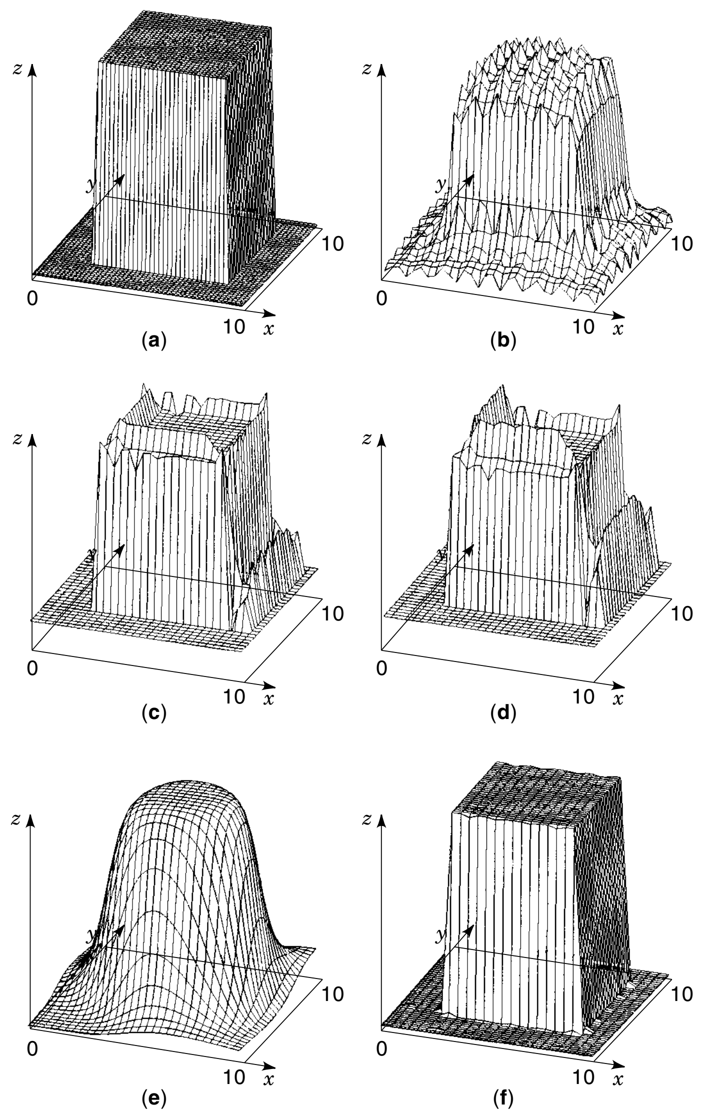

Figure 5. Comparison of reconstruction of Surface 5 using all methods: (a) original surface; (b) Shepard Interpolation; (c) Thin Plate B-spline; (d) Hardy's Multiquadric Interpolation; (e) MultiLayer Neural Network; and (f) Functional link Neural Network.

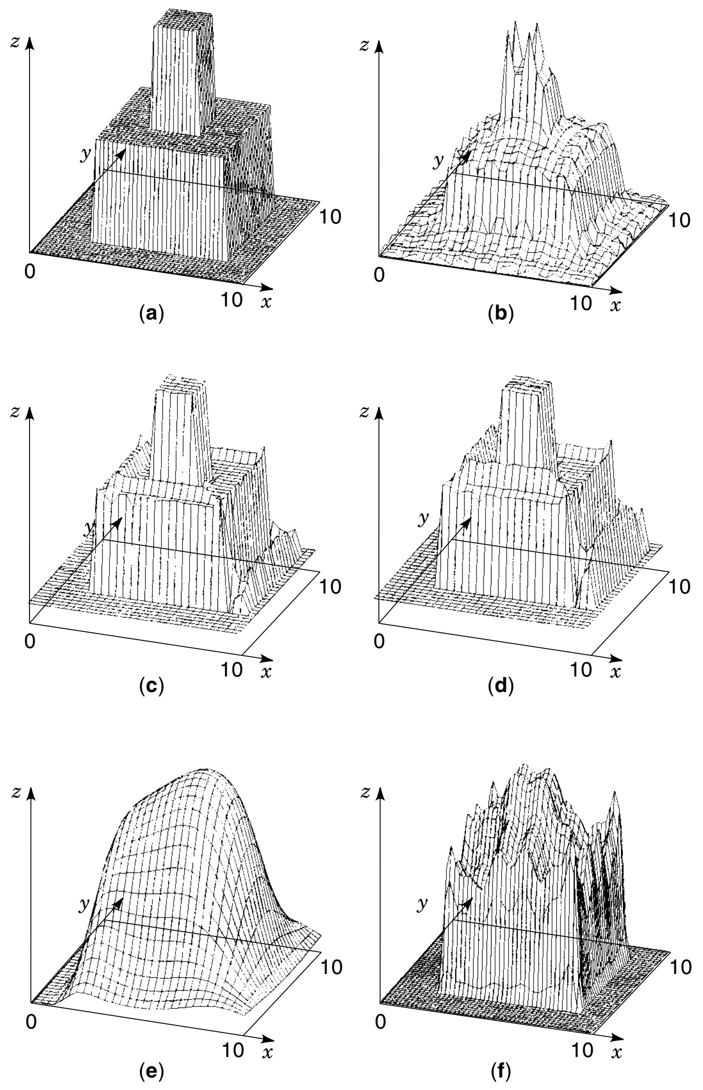

Figure 6. Comparison of reconstruction of Surface 6 using all methods: (a) original surface; (b) Shepard Interpolation; (c) Thin Plate B-spline; (d) Hardy's Multiquadric Interpolation; (e) MultiLayer Neural Network; and (f) Functional link Neural Network.

sponding surfaces of these objects are denoted by "free-form surfaces." As Besl (31) stated, a free-form surface is not constrained to be piecewise planar, piecewise quadratic, piecewise superquadratic, or cylindrical. However, the free-form surface is smooth in the sense that the surface normal is well defined and continuous everywhere (except at isolated cusps, vertices, and edges). Common examples of free-form surface shapes include human faces, cars, airplanes, boats, clay models, sculptures, and terrain.

Historically, free-form shape matching using 3-D data was done earliest by Faugeras and his group at INRIA (33), where they demonstrated effective matching with a Renault auto part (steering knuckle) in the early 1980s. This work popularized the use of quaternions for least squares registration of corresponding 3-D point sets in the computer vision community. The alternative use of the singular value decomposition (SVD) algorithm was not as widely known at that time. The primary limitation of this work was that it relied on the probable existence of reasonably large planar regions within a free-form shape.

There are a number of studies dealing with global shape matching or registration of free-form surfaces on limited classes of shapes. For example, there have been a number of studies on point sets with known correspondence (31–35). Bajcsy (34) is an example of studies on polyhedral models and piecewise models.

As we indicated previously, the output of the laser scanner (the cloud of data) is in the form of a sparse, unformatted file. The goal is to build a model of the physical surface using this data. Two issues need to be examined: (1) how to fit a surface using the data from a single view; and (2) how to merge the data from multiple views to build an overall 3-D model for the object. Bradley and Vickers (3), surveyed a number of algorithms for surface reconstruction using the cloud of data of one viewpoint. Results were shown on basic surfaces like sinusoids and exponentials. They also suggested an algorithm for surface modeling based on the following steps: (1) divide the cloud of data into meshes of smaller sizes; (2) fit a surface using a subset of data points on each mesh; and (3) merge the surfaces of the meshes together. This approach has been shown to be convenient for simple surfaces with little or no occlusion.

BIBLIOGRAPHY

1. S. Haykin, *Neural Networks: A Comprehensive Foundation,* Indianapolis, IN: Macmillan College, 1994.
2. D. Shepard, A two-dimensional interpolation function for irregularly spaced data, *Proc. ACM Nat. Conf.,* 1964, pp. 517–524.
3. C. Bradley and G. Vickers, Free-form surface reconstruction for machine vision rapid prototyping, *Opt. Eng.,* **32** (9): 2191–2200, September 1993.
4. B. Bhanu and C. C. Ho, CAD-based 3-D object representation for robot vision, *IEEE Comput.,* **20** (8): 19–36, 1987.
5. R. L. Hardy, Multiquadratic equations of topography and other irregular surfaces, interpolation using surface splines. *J. Geophys.,* **76**: 1971.
6. J. Zurada, *Introduction to Artificial Neural Systems,* St. Paul, MN: West, 1992.
7. K. Hornik and M. Stinchombe, Multilayer feedforward networks as universal approximators, *Neural Networks,* 359–366, 1989.
8. K. I. Funanashi, On the approximate realization of continuous mappings by neural networks, *Neural Networks,* 183–192, 1989.
9. M. N. Ahmed and A. A. Farag, Free form surface reconstruction using neural networks, *Proc. of ANNIE'94,* **1**: 51–56, 1994.
10. P. H. Pao, *Adaptive Pattern Recognition and Neural Networks,* New York: Addison Wesley, 1989.
11. R. Courant and D. Hilbert, *Methods and Mathematical Physics,* vols. 1 and 2, New York: Wiley, 1970.
12. A. N. Tikhonov, On solving incorrectly posed problems and method of regularization, *Doklady Akademii Nauk USSR,* **151**: 501–504, 1963.
13. T. Poggio and F. Girosi, Networks for approximation and learning, *Proc. IEEE,* **78**: 1481–1497, 1990.
14. T. Poggio, A theory of how the brain might work, *Cold Spring Harbor Symp. Quantitative Biol.,* **55**: 899–910, 1990.
15. T. Poggio and S. Edelman, A network that learns to recognize three-dimensional objects, *Nature,* (London) **343**: 263–266, 1990.

16. T. Poggio and F. Girosi, Regularization algorithms for learning that are equivalent to multilayer networks, *Science,* **247**: 978–982, 1990.

17. T. Poggio, V. Torre, and C. Koch, Computational vision and regularization theory, *Nature* (London) **317**: 314–319, 1985.

18. D. S. Broomhead and D. Lowe, Multivariable functional interpolation and adaptive networks, *Complex Systems,* **2**: 321–355, 1988.

19. D. S. Broomhead, D. Lowe, and A. R. Webb, A sum rule satisfied by optimized feedforward layered networks, *RSRE Memorandum No. 4341, Royal Signals and Radar Establishment,* Malvern, UK, 1989.

20. M. J. D. Powell, Radial basis functions for multivariate interpolation: a review, *IMA Conf. Algorithms Approximation Functions Data, RMCS,* Shivenham, UK, 1985, pp. 143–167.

21. M. J. D. Powel, Radial basis function approximations to polynomials, *Numerical Analysis 1987 Proc.,* Dundee, UK, 1988, pp. 223–241.

22. D. Lowe, Adaptive radial basis function nonlinearities, and the problem of generalization, *1st IEE Int. Conf. Artificial Neural Networks,* London, UK, 1989, pp. 171–175.

23. D. Lowe, On the iterative inversion of RBF networks: a statistical interpretation, *2nd IEE Int. Conf. Artificial Neural Networks,* Bournemouth, UK, 1991, pp. 29–33.

24. D. Lowe and A. R. Webb, Adaptive networks, dynamical systems, and the predictive analysis of time series, *1st IEE Int. Conf. Artificial Neural Networks,* London, UK, 1989, pp. 95–99.

25. F. Girosi and T. Poggio, Representative properties of networks: Kolmogorov's theorem is irrelevant, *Neural Comput.,* **1**: 465–469, 1989.

26. F. Girosi and T. Poggio, Networks and best approximation property, *Biol. Cybern.,* **63**: 169–176, 1990.

27. C. N. Dorny, *A Vector Space Approach to Models and Optimization,* New York: Wiley, 1975.

28. P. Simard and Y. LeCun, Reverse TDNN: An architecture for trajectory generation, *Advances Neural Inf. Process. Syst.* **4**: 579–588, 1992.

29. A. N. Tikhonov, On regularization of ill-posed problems, *Doklady Akadermii Nauk USSR,* **153**: 49–52, 1973.

30. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed Problems,* Washington, DC: Winston, 1977.

31. P. J. Besl, Free-form Surface Matching, in H. Freeman (ed.), *Machine Vision for Three-Dimensional Scenes,* San Diego: Academic Press, 1990, pp. 25–71.

32. P. Besl and D. McKay, A method for registration of 3-D shapes, *IEEE Trans. Pattern Anal. Mach. Intel.,* **PAMI-14**: 239–256, Feb. 1992.

33. O. D. Faugeras and M. Herbert, The representation, recognition, and locating of 3-D objects, *Int. J. Robotics Res.,* **5** (3): 27–52, 1986.

34. R. Bajcsy and F. Solina, Three-dimensional object representation, *Proc. 1st. Int. Conf. Comput. Vision,* (London), June 8–11, 1989, pp. 231–240.

35. D. Terzopoulos et al., Elastically deformable models, *Comput. Graphics,* **21** (4): 205–214, 1987.

36. R. Franke, Scattered data interpolation, *Math Comput.,* **38**: 181–200, 1982.

MOHAMED N. AHMED
SAMEH M. YAMANY
ALY A. FARAG
University of Louisville

FUNCTIONS, BESSEL. See BESSEL FUNCTIONS.
FUNDING RESEARCH. See RESEARCH INITIATIVES.
FUSES, ELECTRIC. See ELECTRIC FUSES.
FUSE TRANSFORMER PROTECTION. See TRANSFORMER PROTECTION.