Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 351648, 13 pages
http://dx.doi.org/10.1155/2014/351648

Research Article

Hybrid Functional-Neural Approach for Surface Reconstruction

Andrés Iglesias[1,2] and Akemi Gálvez[1]

[1] *Department of Applied Mathematics and Computational Sciences, E.T.S.I. Caminos, Canales y Puertos, University of Cantabria, Avenida de los Castros, s/n, 39005 Santander, Spain*
[2] *Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan*

Correspondence should be addressed to Andrés Iglesias; iglesias@unican.es

Received 28 July 2013; Accepted 8 December 2013; Published 16 January 2014

Academic Editor: Yudong Zhang

This paper introduces a new hybrid functional-neural approach for surface reconstruction. Our approach is based on the combination of two powerful artificial intelligence paradigms: on one hand, we apply the popular Kohonen neural network to address the data parameterization problem. On the other hand, we introduce a new functional network, called NURBS functional network, whose topology is aimed at reproducing faithfully the functional structure of the NURBS surfaces. These neural and functional networks are applied in an iterative fashion for further surface refinement. The hybridization of these two networks provides us with a powerful computational approach to obtain a NURBS fitting surface to a set of irregularly sampled noisy data points within a prescribed error threshold. The method has been applied to two illustrative examples. The experimental results confirm the good performance of our approach.

1. Introduction

Manufacturing industries are constantly evolving in response to the new challenges of the globalization and the growing competition in this global market. Product design is playing a central role in this process, as current customers are increasingly demanding a mass customization of the products. As a result, the geometric and aesthetic properties of the manufactured goods (shape, color, and dimensions) have to be modified frequently in order to meet the new market demands.

A major step in this process is the generation of real prototypes with different materials to explore and analyze their geometric properties and the feedback of potential customers when exposed to different variations of the final product. Prototype generation and customization can be dramatically improved by using digital technologies, in which the physical model is digitized, stored, and manipulated by computer, a process called reverse engineering [1, 2]. Typically, this process begins with data sampling by using 3D laser scanning and other digitizing devices. This technology is intensively used for the construction of car bodies, ship hulls, airplane fuselage, and other free-form objects [2–7]. The resulting data points are then fitted to mathematical entities such as curves and surfaces, usually in parametric form. The output is a very accurate digital version of the real product, which is also simpler and easier to store, analyze, and manipulate. It also simplifies the transfer and communication processes among designers, manufacturers, and providers, making the model available in just a few seconds all over the world, a key aspect in our ubiquitously connected information society era.

In this paper, we are interested in one of the most critical steps of this process, namely, the construction of surfaces of the real objects from sets of digitized data points, a field usually called *surface reconstruction*. The preferred mathematical models for design and manufacturing are the free-form parametric surfaces [3, 8–16], because they are very flexible and can be readily modified by changing a small set of parameters. In this paper, we use NURBS surfaces, the most powerful (and most difficult to deal with) free-form parametric surfaces, which have become the standard for CAD/CAM data representation in industrial settings and in many other fields, from digital effects for movies in computer animation and advertisements to the design of characters in computer graphics and video games. In reverse engineering applications, data points are usually acquired through laser

scanning and other digitizing devices and are, therefore, subjected to measurement noise, irregular sampling, and other artifacts [6, 7]. Consequently, a good fitting of data should be generally based on approximation schemes rather than interpolation [1, 13, 14, 17–19]. Because this is the typical case in many real-world industrial problems, in this paper we focus on the approximation scheme to a given set of noisy, irregularly sampled data points.

Obtaining the best approximating surface in such cases is much more troublesome than it may seem at first sight. The main reasons are as follows.

(i) The NURBS surfaces depend on many different parameters (data parameters, knots, control points, and weights) that are strongly interconnected with each other, leading to a strongly nonlinear continuous optimization problem.

(ii) It is also multivariate, as it typically involves a large number of unknown variables for a large number of data points, a case that happens very often in real-world examples.

(iii) In addition, it is also overdetermined, because we expect to obtain the approximating surface with many fewer parameters than the number of data points.

(iv) Finally, the problem is known to be multimodal; that is, the least-squares objective function can exhibit many local optima [20, 21], meaning that the problem might have several (global and/or local) good solutions.

In conclusion, we have to solve a very difficult multimodal, multivariate, high-dimensional continuous nonlinear optimization problem. A number of methods have been proposed to solve this problem (see Section 2 for details). Among them, those based on artificial intelligence techniques have received increasing attention during the last few years. Most of such methods rely on the *artificial neural networks* (ANN) formalism [22]. Since ANN methodology is actually inspired by the behavior of the human brain, it is able to reproduce some of its most typical features, such as the ability to learn from data. This explains why they have been so widely applied to data fitting problems.

Although they are very popular, the ANN are however limited in many aspects. A major drawback is their inability to reproduce mathematically the functional structure of a given problem. This limitation can be overcome with the use of a new paradigm in artificial intelligence, the so-called *functional networks* (FN) (see Section 4 for details). In short, functional networks are a generalization of the standard neural networks in which the scalar weights are replaced by neural functions. These neural functions can exhibit, in general, a multivariate character. Furthermore, different neurons can be associated with neural functions from different families of functions. These FN features allow us to reproduce exactly the functional structure of the problem by a careful choice of the functions involved, which can hereby be associated with one or several neurons of the network. This procedure yields a functional structure that is typically a replica of the underlying structure of the given problem.

1.1. Aims and Structure of the Paper. In this paper, we propose a hybrid artificial intelligence approach to solve the surface reconstruction problem. Our approach is based on the combination of two powerful artificial intelligence paradigms: on one hand, we apply the popular Kohonen neural network to address the data parameterization problem. On the other hand, we take advantage of the remarkable properties mentioned in previous paragraph by introducing a new functional network, called NURBS functional network, whose topology is specially targeted to reproduce the functional structure of the NURBS surfaces. These neural and functional networks are then applied iteratively for further surface refinement. As it will be shown later on, the hybridization of these two networks provides us with a powerful computational approach to solve the surface reconstruction problem. To check the performance of our approach, it has been applied to two illustrative examples. Our experimental results show that the method performs very well, being able to reconstruct the approximating surface of the given set of data points with a high degree of accuracy.

The structure of this paper is as follows: in Section 2, previous work regarding the surface reconstruction problem is reported. Then, some basic concepts about NURBS surfaces and the optimization problem to be solved are given in Section 3. Section 4 describes the fundamentals of the functional networks along with their main components and the differences between neural and functional networks. The proposed hybrid functional-neural method for surface reconstruction with NURBS surfaces is described in detail in Section 5. Then, some illustrative examples of its application are reported in Section 6. A comparison of our approach with other ANN alternative methods is analyzed in detail in Section 7. The paper closes with the main conclusions of this contribution and our plans for future work in the field.

2. Previous Work

Surface reconstruction has been a topic of increasing attention from the scientific community during the last 20 years, with outstanding applications in both theoretical and applied domains. Regarding the theoretical side, it is a remarkable subject in approximation theory [23, 24], statistics [25], numerical analysis [26, 27], geometric modeling [2, 8, 28], and computer-aided geometric design (CAGD) [5, 29]. In addition, there is a bulk of applications in several fields, such as computer-aided manufacturing (CAM) [6, 7], data visualization [30], cultural heritage preservation [31], virtual reality [32], medical imaging [33], and computer animation [34], to mention just a few.

In general, surface reconstruction methods are classified in terms of the available input (2D slices, isoparametric curves, clouds of points, mixed information, etc.). For instance, authors in [35–39] address the problem of obtaining a surface model from a set of given cross-sections, a classical problem in medical science, biomedical engineering, and

CAD/CAM. Other classical input data include isoparametric curves on the surface [40] and even mixed information, such as scattered points and contours [41–43] or isoparametric curves and data points [4, 5, 44].

In most cases, however, the available information about the surface is typically a dense set of (usually unorganized) 3D data points obtained by using some sort of digitizing devices (see, e.g., [14, 45–48]). In that case, the reconstructed surface can be described using three different representations providing different levels of accuracy. The simplest one is given by the polygonal meshes, where the data points are used as vertices connected by lines (edges) that work together to create a 3D model, comprised of vertices, edges, and faces. Although it is the coarsest representation, it is also the most popular because of its simplicity, flexibility, and excellent performance with current graphical cards. Surface reconstruction methods with polygonal meshes can be found, for instance, in [30, 31, 47, 49–51] and references therein. The next level is given by the constructive solid geometry (CSG) models, where elementary geometries (such as spheres, boxes, cylinders, or cones) are combined in order to produce more elaborated shapes by applying some simple (Boolean) operators: union, intersection, and difference. This methodology works well but presents a low level of flexibility, being severely limited to very simple shapes. The most sophisticated and most accurate level consists of obtaining the real mathematical surface fitting the data points. This issue has been analyzed from several points of view, such as parametric methods [52], subdivision surfaces [53], function reconstruction [54, 55], implicit surfaces [48], and algebraic surfaces [56]. Other approaches are based on the application of metaheuristic techniques, which have been intensively applied to solve difficult optimization problems that cannot be tackled through traditional optimization algorithms. Recent schemes in this area involve particle swarm optimization [10, 57, 58], genetic algorithms [59–62], artificial immune systems [63, 64], estimation of distribution algorithms [65], firefly algorithm [66, 67], and hybrid techniques [68, 69].

Artificial neural networks have also been applied to this problem [45, 70], mostly for arranging the input data in case of unorganized points. After this preprocessing step, any other classical surface reconstruction method operating on organized points is subsequently applied. A work using a combination of neural networks and partial differential equation (PDE) techniques for the parameterization and reconstruction of surfaces from 3D scattered points can be found in [71]. Two previous papers by the authors have also addressed this problem by using functional networks [4, 8], a powerful generalization of neural networks based on functional equations [72, 73]. Both works show, however, that the single application of functional networks is still unable to solve the general case. The work in [4] addresses the particular case of B-spline surface reconstruction when some additional information (isoparametric curves) is available in addition to the data points. The fitting surfaces are a generalization of the Gordon surfaces [40]. The work in [8] combines functional networks with genetic algorithms in order to solve the (much simpler) polynomial Bézier case. The approach presented here solves the general NURBS surface reconstruction problem based exclusively on neural and functional networks. To the best of our knowledge, no previous method is reported in the literature providing all these features.

3. Basic Concepts and Definitions

3.1. NURBS Surfaces. Let $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_{r-1}, s_r\} \subset [a, b]$ be a nondecreasing sequence of real numbers called *knots*. $\mathcal{S}$ is called the *knot vector*. Without loss of generality, we can assume that $[a, b] = [0, 1]$. The ith *B-spline basis function* $N_{i,k}(u)$ *of order* k (or equivalently, degree $k-1$) is defined by the recurrence relations:

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } s_i \le u < s_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

with $i = 0, \ldots, r-1$ and

$$N_{i,k}(u) = \frac{u - s_i}{s_{i+k-1} - s_i} N_{i,k-1}(u) + \frac{s_{i+k} - u}{s_{i+k} - s_{i+1}} N_{i+1,k-1}(u) \tag{2}$$

for $k > 1$. Note that ith B-spline basis function of order 1, $N_{i,1}(u)$, is a piecewise constant function with value 1 on the interval $[s_i, s_{i+1})$, called the *support* of $N_{i,1}(u)$, and zero elsewhere. This support either can be an interval or reduce to a point, as knots s_i and s_{i+1} must not necessarily be different. If necessary, the convention $0/0 = 0$ in (2) is applied. Any basis function of order $k > 1$, $N_{i,k}(u)$, is a linear combination of two consecutive functions of order $k-1$, where the coefficients are linear polynomials in u, such that its order (and hence its degree) increases by 1. Simultaneously, its support is the union of the (partially overlapping) supports of the former basis functions of order $k-1$ and, consequently, it usually enlarges.

With the same notation, given a set of three-dimensional points (called *control points* as they roughly determine the shape of the curve) $\{\mathbf{P}_{ij}\}_{i=0,\ldots,m;j=0,\ldots,n}$ in a bidirectional net and two knot vectors $\mathcal{S} = \{s_0, s_1, s_2, \ldots, s_{r-1}, s_r\}$ and $\mathcal{T} = \{t_0, t_1, \ldots, t_{h-1}, t_h\}$, a *NURBS surface* $\mathbf{S}(u, v)$ *of order* (k, l) (where *NURBS* stands for Non-Uniform Rational B-Spline) is a rational B-spline parametric surface given by

$$\mathbf{S}(u, v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} \mathbf{P}_{i,j} N_{i,k}(u) N_{j,l}(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} N_{i,k}(u) N_{j,l}(v)}, \tag{3}$$

where the $\{N_{i,k}(u)\}_i$ and $\{N_{j,l}(v)\}_j$ are the B-spline basis functions of orders k and l, respectively, defined following (1) and (2), and $\{w_{i,j}\}_{i,j}$ are nonnegative scalar values called weights associated with the control points $\{\mathbf{P}_{i,j}\}_{i,j}$. Without loss of generality, parameters u, v can be assumed to take values on the interval $[0, 1]$. For a proper definition of a NURBS surface in (3), the following relationships must hold (see [29]): $r = m + k$, $h = n + l$.

In general, a NURBS surface does not interpolate any of its control points; the interpolation only occurs for nonperiodic knot vectors (in that case, the NURBS surface does interpolate the corner control points) [11, 29]. Since they are the

most common in computer graphics and industrial domains, in this work we will consider the case of nonperiodic knot vectors. Note, however, that our method does not depend on the kind of knot vectors used for the approximating surfaces.

3.2. Surface Reconstruction Problem. In clear contrast with many previous methods, in this paper we focus on the *general* surface reconstruction problem, which assumes that no other information about the problem is available beyond the data points. In particular, our problem can be stated as follows. Given a set of (usually irregularly sampled) noisy data points $\mathbf{Q}$ assumed to lie on an unknown surface $\mathbf{U}$, construct, to the extent possible, a full mathematical representation of a surface model $\mathbf{S}$ that approximates $\mathbf{U}$. Because of its remarkable applications in real-world engineering problems, we also demand such a mathematical representation to be a NURBS surface.

Mathematically speaking, we assume that we are provided with a set of data points $\{\mathbf{Q}_{\alpha,\beta}\}_{\alpha=1,\ldots,M;\beta=1,\ldots,N}$ in $\mathbb{R}^3$, with $M, N \ggg r, h$. Our goal is to obtain the NURBS surface $\mathbf{S}(u, v)$ that fits the data points better in the discrete least-squares sense. To do so, we have to compute all the parameters of the approximating surface by minimizing the least-squares error, E, defined as the sum of squares of the residuals:

$$E = \sum_{\alpha=1}^{M}\sum_{\beta=1}^{N}\left(\mathbf{Q}_{\alpha,\beta} - \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}\mathbf{P}_{i,j}N_{i,k}(u_\alpha)N_{j,l}(v_\beta)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}N_{i,k}(u_\alpha)N_{j,l}(v_\beta)}\right)^2. \tag{4}$$

In the case of scattered data points $\{\mathbf{Q}_\mu\}_{\mu=1,\ldots,R}$, our method will work in a similar way by simply replacing the previous expression (4) by

$$E = \sum_{\mu=1}^{R}\left(\mathbf{Q}_\mu - \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}\mathbf{P}_{i,j}N_{i,k}(u_\mu)N_{j,l}(v_\mu)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}N_{i,k}(u_\mu)N_{j,l}(v_\mu)}\right)^2. \tag{5}$$

The minimization of either (4) or (5) leads to the system of equations:

$$\mathbf{Q}^{\mathbf{v}} = \mathbf{M}\cdot\mathbf{P}^{\mathbf{v}}, \tag{6}$$

where the symbol $(\cdot)^{\mathbf{v}}$ denotes the vectorization operator of the given matrix and $\mathbf{M}$ in (6) is a matrix given by $\mathbf{M}_{i,j}(u_\alpha, v_\beta) = [((\mathbf{R}^{\mathbf{v}}_{j,l}(u_\alpha, v_\beta))^{\mathbf{T}} \otimes R_{i,k}(\mathbf{u}^{\mathbf{T}}, v_\beta))^{\mathbf{v}}]^{\mathbf{T}}$, with $\mathbf{u} = (u_1, \ldots, u_M)$. $R_{i,j}(u, v)$ is given as

$$R_{i,j}(u, v) = \frac{w_{i,j}N_{i,k}(u)\,N_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}N_{i,k}(u)\,N_{j,l}(v)} \qquad i = 0, \ldots, m; \quad j = 0, \ldots, n \tag{7}$$

while $\otimes$ and $(\cdot)^{\mathbf{T}}$ represent the outer product operator and the transpose of a vector or matrix, respectively. The indices in (4)–(7) vary in the ranges of values indicated throughout the section.

It is worthwhile to mention that since the lengths of $\mathbf{Q}^{\mathbf{v}}$ and $\mathbf{P}^{\mathbf{v}}$ are, respectively, $M \times N$ and $(m + 1) \times (n + 1)$, the system (6) is overdetermined. Premultiplication of both sides by $\mathbf{M}^{\mathbf{T}}$ yields

$$\mathbf{M}^{\mathbf{T}}\cdot\mathbf{Q}^{\mathbf{v}} = \mathbf{M}^{\mathbf{T}}\cdot\mathbf{M}\cdot\mathbf{P}^{\mathbf{v}} = \boldsymbol{\Delta}\cdot\mathbf{P}^{\mathbf{v}}, \tag{8}$$

where $\boldsymbol{\Delta} = \mathbf{M}^{\mathbf{T}}\cdot\mathbf{M}$. Note also that the tensor-product basis functions $R_{i,j}(u, v)$ are generally continuous and nonlinear, so the minimization of (8) leads to the continuous nonlinear optimization problem:

$$\min_{\{\mathbf{P}_{i,j}\}\subset\mathbb{R}^3;\{w_{i,j}\}\geq 0;\{(u_\alpha,v_\beta)\}\subset\mathrm{Dom}(\mathbf{S});\{s_v\},\{t_\gamma\}\subset[0,1]} \left\|\mathbf{M}^{\mathbf{T}}\cdot\mathbf{Q}^{\mathbf{v}} - \boldsymbol{\Delta}\cdot\mathbf{P}^{\mathbf{v}}\right\|^2, \tag{9}$$

where $\|\cdot\|$ represents the Euclidean norm.

4. Functional Networks

Roughly speaking, a *functional network* is a generalization of the standard neural network in which the scalar weights are replaced by neural functions. Functional networks were firstly introduced in 1998 by Castillo in [72] as a way to enhance the neural networks with new capabilities. Since then, they have been successfully applied to several problems in science and engineering. The interested reader is referred to [73, Chapter 9] (Chapter 9) for in-depth explanation about functional networks along with several illustrative examples and applications. In this section we describe the main components of a functional network. Differences between neural and functional networks are also discussed in this section.

4.1. Components of a Functional Network. As an explanatory example, Figure 1(a) shows the functional network of the associative operation F between two real numbers; that is, function F satisfies

$$F(F(x, y), z) = F(x, F(y, z)). \tag{10}$$

It can be proved that the general solution of this equation is given by [73]

$$F(x, y) = f^{-1}[f(x) + f(y)], \tag{11}$$

where $f(x)$ is an arbitrary continuous and strictly monotonic function, which can be replaced only by $c\ f(x)$, where c is an arbitrary constant. Such a solution can be represented by the functional network in Figure 1(b). Note that, because of the uniqueness (except arbitrary constants) of the solution, both networks do actually represent the same problem. In other words, the functional network in Figure 1(b), is equivalent to (but arguably simpler than) that in Figure 1(a). From Figure 1(b) the main components of a functional network become clear.

(i) Several Layers of Storing Units

(a) A Layer of Input Units. This first layer contains the input information. In this figure, this input layer consists of the units x and y.

(b) A Set of Intermediate Layers of Storing Units. They are not neurons but units storing intermediate information. This

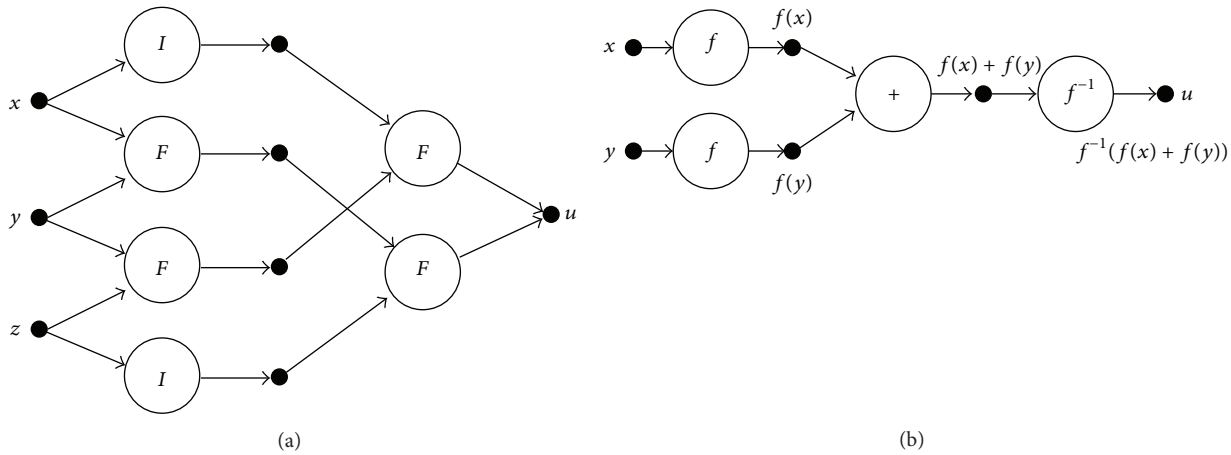


Figure 1: The associativity functional network: (a) original network; (b) simplified network.

set is optional and allows more than one neuron output to be connected to the same unit. In Figure 1(b), there are two intermediate layers of storing units, which are represented by small circles in black.

(c) A Layer of Output Units. This last layer contains the output information. In Figure 1(b), this output layer is reduced to the unit $u = f^{-1}(f(x) + f(y))$.

(ii) One or More Layers of Neurons or Computing Units. A neuron is a computing unit which evaluates a set of input values, coming from the previous layer, of input or intermediate units, and gives a set of output values to the next layer, of intermediate or output units. Neurons are represented by circles with the name of the corresponding neural function inside. For example, in Figure 1(b), we have three layers of neurons. The first one gives outputs of functions with one variable. The second layer exhibits the sum operator of its two inputs. The last layer computes the inverse of the first layer function applied to output of the previous layer.

(iii) A Set of Directed Links. They connect the input or intermediate layers to its adjacent layer of neurons and neurons of one layer to its adjacent intermediate layers or to the output layer. Connections are represented by arrows, indicating the information flow direction. We remark here that information flows in only one direction, from the input layer to the output layer.

All these elements together form the *network architecture* or *topology* of the functional network, which defines the functional capabilities of the network.

4.2. Differences between Functional and Neural Networks. In next paragraphs, we discuss the differences between functional and neural networks and the advantages of using functional networks instead of standard neural networks.

(1) In neural networks, each neuron returns an output $y = f(\sum w_{ik}x_k)$ that depends only on the value $\sum w_{ik}x_k$, where $x_1, x_2, \ldots, x_n$ are the received inputs (see Figure 2(a)). Therefore, their neural functions have only one argument. In contrast, neural functions in functional networks can have several arguments, as shown in Figure 2(b).

(2) In neural networks, the neural functions are *univariate*: neurons can show different outputs but all of them represent the same values. In functional networks, the neural functions can be *multivariate*.

(3) In a given functional network, the neural functions can be *different* (such as functions f_1, f_2, and f_3 in Figure 2(b)), while in neural networks they *are identical*.

(4) In neural networks, there are weights, which must be learned. These weights do not appear in functional networks, where neural functions are learned instead.

(5) In neural networks *the neuron outputs are different*, while in functional networks *neuron outputs can be coincident*. This fact leads to a set of functional equations, which have to be solved [73, 74]. These functional equations impose strong constraints leading to a considerable reduction in the degrees of freedom of the neural functions. In most cases, this implies that neural functions can be reduced in dimension or expressed as functions of smaller dimensions.

All these features show that the functional networks exhibit more interesting possibilities than the neural networks. This implies that some problems can be solved more efficiently by using functional networks instead of neural networks.

5. Our Method

In this section, we describe the proposed method for solving the surface reconstruction problem indicated in Section 3.2. Firstly, a general overview of the method is presented. Then, each step of the method is discussed in detail.

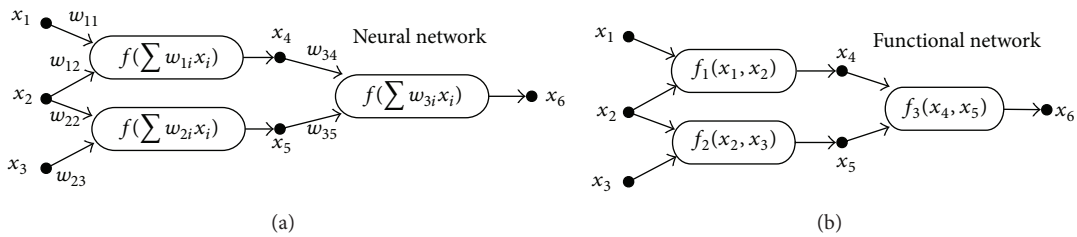


Figure 2: Differences between (a) neural networks and (b) functional networks.

5.1. Overview of the Method. The graphical workflow in Figure 3 summarizes the main steps of our method. The initial input consists of a set of irregularly sampled noisy 3D data points assumed to lie on an unknown surface. The goal is to obtain the NURBS surface that approximates these data points optimally. To this purpose, we need to solve two important subproblems: data parameterization and surface approximation. To address data parameterization, we firstly perform a principal component analysis (PCA) to obtain a parametric plane that accounts for the variability of data. Then, the data points are projected onto that parametric plane. A 2D surface parameterization is subsequently obtained by applying a Kohonen neural network to the set of projected 2D data points. Then, we apply our NURBS functional network to compute all other parameters of the NURBS surface approximating the data points for this initial parameterization. We call that surface a base surface. The next step consists of computing the fitting error for the approximating surface. Finally, the data points are projected onto the new base surface in order to yield a new (more accurate) parameterization. All previous steps are repeated iteratively until a stopping criterion is reached. Usual stopping criteria are that the fitting error becomes smaller than a given threshold value or that successive iterations of this reconstruction pipeline no longer improve current solutions.

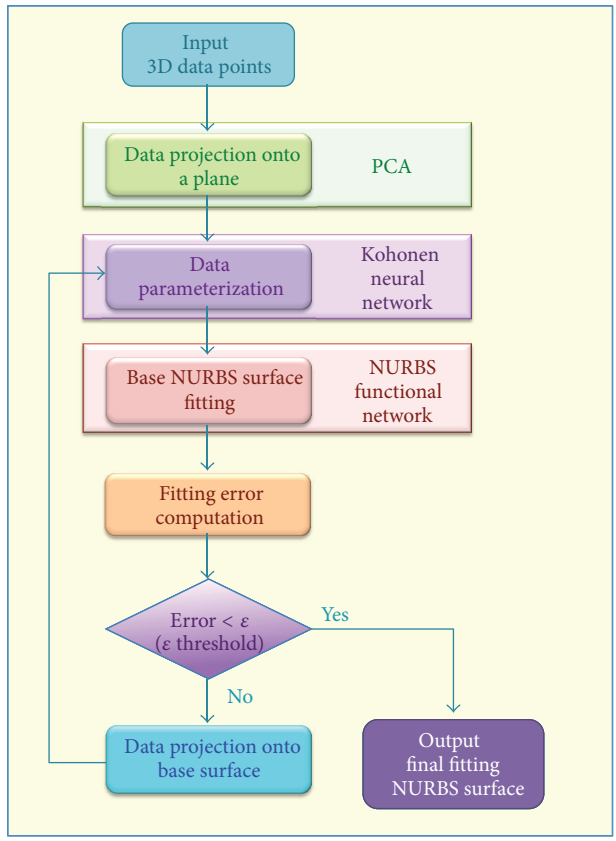


Figure 3: Graphical workflow of the proposed method.

5.2. Data Parameterization. The parameterization step consists of establishing the relationships among the data points in the surface parametric domain. This process is essential for a good fitting of data points. Some standard procedures are given by the uniform, chord length and centripetal parameterizations. However, these methods are only suitable for data points distributed in a uniform grid and tend to fail for unorganized, irregularly sampled data. An alternative procedure is based on the idea of projecting the data points onto an additional surface, usually called base surface, reflecting the distribution of data points and then computing a parameterization by using the projected 2D points. The simplest case of this approach consists of using a parametric plane [11], usually orthogonal to the main viewing direction of the digitizing device. A better alternative is to use a suitable 3D surface for data projection [14], usually a coarse approximation of final fitting surface, which is expected to be modified by successive improvements of this initial surface.

In our method, we combine these ideas to develop a refinement process in iterative fashion. At the initial stage, we project the data points onto a parametric plane reflecting the variability of data. This parametric plane is computed by using the principal component analysis (PCA), a very popular descriptive technique in data analysis and many other fields. PCA is a powerful statistical method aimed at performing dimensionality reduction by using the analysis of the correlation of data. A great advantage of this method is its nonparametric nature, meaning that it does not require any parameter tuning in order to get an output. Furthermore, the answer is unique and available regardless of the way the data has been recorded or obtained. Other main reasons for our choice are that PCA is very efficient at preserving distances between the points and each principal component has the highest variance possible under the constraint that it is uncorrelated with preceding components. In fact, the first principal component corresponds to a line passing through the multidimensional mean. It also minimizes the sum of

squares of the distances to the points from that line. Because of these properties, the parametric plane obtained by PCA is very adequate for the initial 2D projection.

Projected 2D points are then used for an initial surface parameterization by applying a Kohonen neural network (also called self-organizing map (SOM) neural network). This is a very popular artificial neural network for unsupervised learning. The interested reader is referred to the nice books in [75, 76] for a comprehensive overview about the Kohonen neural network, its fundamentals and mathematical basis, along with the most important variants and modifications and several interesting applications.

Algorithm 1. Kohonen neural network for 2D data parameterization.

(1) Initialize position Ψ of all neurons Φ with random values within the surface boundaries.

(2) Initialize the weights $\sigma_{i,j}$ randomly around the centroid of the input space.

(3) Pick a new input sample Q_δ.

(4) Compute

$$d_j = \sqrt{\sum_{i=1}^{R} \left(\Psi_i(\delta) - \sigma_{i,j}(\delta)\right)^2}. \tag{12}$$

(5) Select the active neuron γ for which $d_\gamma = \min_{i=1,\ldots,R}(d_i)$. The winner is the closest active neuron to the sample data Q_δ, with parametric coordinates $(\tilde{i}, \tilde{j})$.

(6) Compute the neighborhood of neuron γ as follows. The radius ρ is given by

$$\rho(\gamma) = \frac{R}{2e^{[\sum_{i=1}^{\delta}(1/10+i/10^4)]}}. \tag{13}$$

The bubble neighborhood of the winner neuron is defined as all neurons at positions (i, j) such that

$$\left|i - \tilde{i}\right| + \left|j - \tilde{j}\right| < \rho\left(1 - \frac{\delta}{\theta}\right), \tag{14}$$

where θ is the run length.

(7) Update the weight of neuron γ and all neurons in its neighborhood as

$$\sigma_{i,j}(\delta + 1) = \sigma_{i,j}(\delta) + \eta(\delta)\left(\Psi_i(\delta) - \sigma_{i,j}(\delta)\right) \tag{15}$$

with

$$\eta(\delta) = \zeta_\delta \frac{1}{\sqrt{2\pi}e^{-\delta^2/2}}, \tag{16}$$

where the learning rate ζ_δ is given by

$$\zeta_\delta = 1 - \left(1 - \zeta_0\left(1 - \frac{\delta}{\theta}\right)\right)^{\sigma_\delta}. \tag{17}$$

(8) Increment δ.

(9) Repeat the steps (3)–(8) until δ reaches the limit value.

A very remarkable feature of the Kohonen neural network is that it incorporates a neighborhood function to preserve the topological properties of the input space [75]. This property is very useful to generate a 2D grid for parameterization, where the neurons represent the grid nodes. The neural network is trained by using the projected 2D data so that its topology eventually reflects their shape and neighborhood relations. To this purpose, each neuron contains geometrical information about the coordinates of the associated node and the topological relations with its neighbors. It is important to remark that the connections among the neurons do not change during the training stage; instead, the changes occur on the geometrical values stored in the neurons. In other words, the network performs a topological ordering of the competitive neurons such that the neighboring neurons represent clusters in the two-dimensional space. Eventually, the weights will specify cluster centers whose distribution approximates the distribution of data points and hence a suitable 2D parameterization. Because of these good properties, the Kohonen neural network has already been used in several ways for data parameterization in some previous works [49, 70, 71, 77]. The algorithm used in this paper combines some of the best features of previous methods. It is briefly summarized in Algorithm 1.

5.3. Surface Fitting. A major property of functional networks is their ability to reproduce the functional structure of the underlying mathematical function of the data points. In this paper, we introduce a new functional network (depicted in Figure 4) especially designed to reproduce the functional structure of NURBS surfaces: the NURBS functional network. Its workflow can be traced graphically by proceeding upwardly in Figure 4: given the surface parameter values u and v and two orders k and l, what this functional network essentially does is to compute the values of the basis functions N_{ik} at u and N_{jl} at v and then the bivariate tensor-product basis functions $N_{ik}(u)N_{jl}(v)$, $(i = 0, \ldots, m;\ j = 0, \ldots, n)$. Each bivariate basis function is then multiplied twice, firstly by the scalar weight $w_{i,j}$ and then by such a weight and its associated vector weight $\mathbf{P}_{i,j}$. Summation on indices i and j is applied to both expressions to obtain the numerator and denominator of (3), to which the quotient operator is subsequently applied. Note that two different (scalar and vector) time operators are considered, to account for the multiplication by $w_{i,j}$ and $\mathbf{P}_{i,j}$, respectively. Note also that those vectors $\mathbf{P}_{i,j}$ play the role of weights of the neural functions as well, with the meaning that a functional network with d-dimensional vectors as weights can be understood as d parallel functional networks with scalar weights. Note finally that the first three layers of this functional network apply functions to either the same or independent arguments of the previous layer, so NURBS functional networks are well suited for partial bottom-up parallelization.

Now, we use the surface parameterization obtained in previous steps to compute an approximating surface to

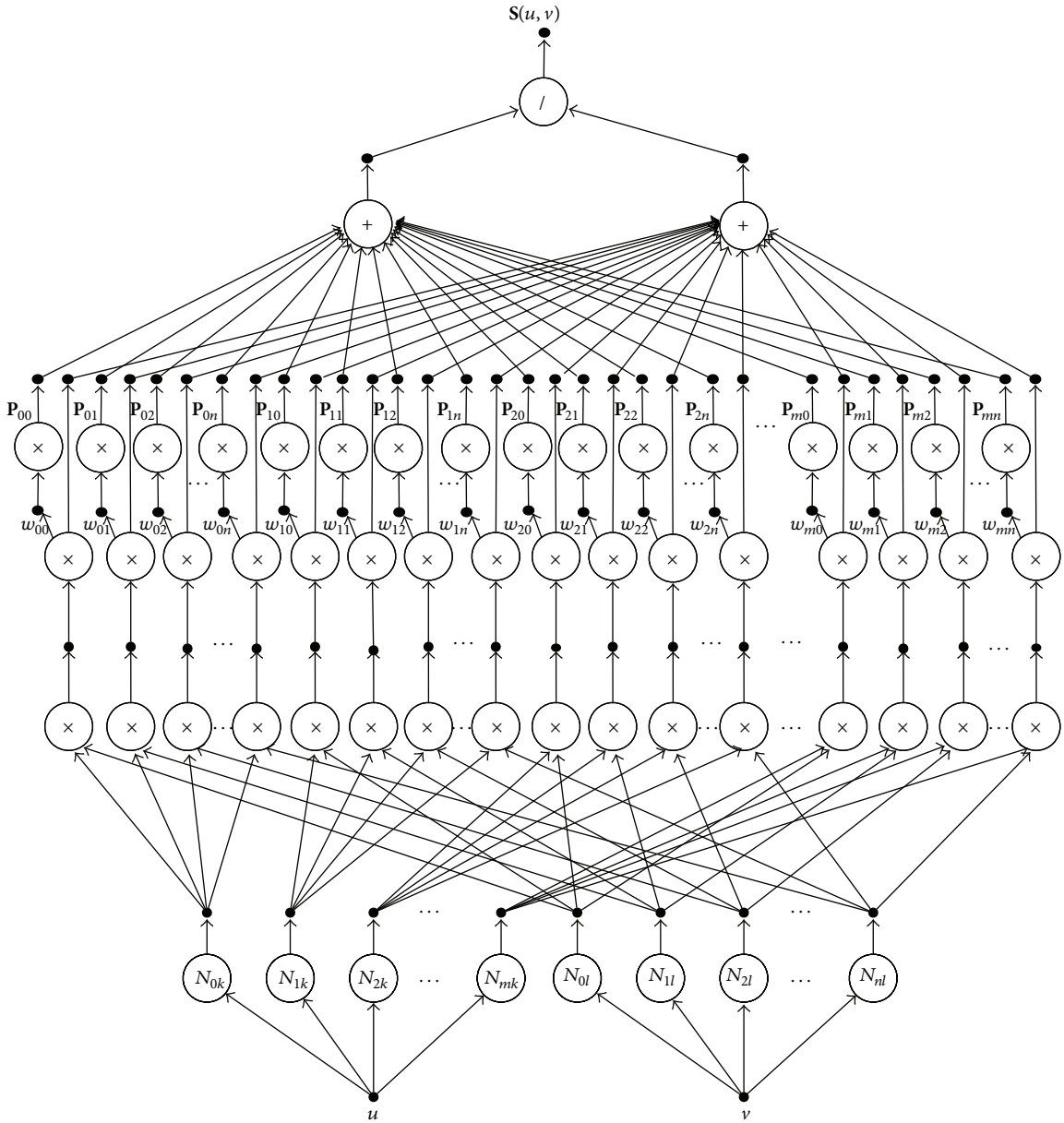


Figure 4: Graphical representation of the NURBS functional network.

the data points. This functional network is trained by supervised learning in which couples of input-output values $\mathbf{D} = \{(I_i, O_i) \mid i = 1, \ldots, R\}$, corresponding to the surface parameters $\{(u_\mu, v_\mu)\}_\mu$ and their associated data points $\mathbf{Q}_\mu$, are presented to the network. Unlike ANN, where the neuron functions are assumed to be fixed and known and only the weights are learned, in functional networks the functions are learned during the structural learning (which obtains the simplified network and functional structures) and estimated during the parametric learning (which consists of obtaining the optimal neuron function from a given family). In our case, the structural learning corresponds to the topology of the NURBS functional network, where the input data are the order (k, l) and the length (r, h) of the knot vectors of the approximating surface. These values determine the number of neurons in each layer of the functional network.

The parametric learning concerns the estimation of the neuron functions. It is usually accomplished by considering linear combinations of given functional families and estimating the associated parameters from the available data. For our choice of B-spline basis functions during the structural learning, the parametric learning is required to determine the scalar and vector weights of our NURBS functional network. Vector weights are learned by using expression (8) where $\mathbf{\Delta}$ is a symmetric square matrix. This system is solved by using the singular value decomposition (SVD), which provides the best numerical answer for the minimization problem in (9) in those cases in which the exact solution is not possible

Table 1: Number of data points and input parameters of the approximating NURBS surfaces for the two examples discussed in this paper.

Example	Number of data points	Order	(r, h)	Iterations
Mask	11830	(4, 4)	(17, 14)	2
Umbrella	8926	(3, 3)	(6, 27)	4

(see [78] for details). Finally, the scalar weights are obtained by least-squares minimization of expression (9), where all other relevant parameters of this minimization problem have already been obtained as described in previous steps.

5.4. Surface Refinement. Once the approximating surface is obtained, the initial parametric plane used for data parameterization is replaced by this approximating surface, which becomes the new base surface. Data points are then projected onto this base surface by computing the nearest point on the surface to each 3D data point by following the procedure indicated in [79]. A new data parameterization and surface fitting steps are computed according to Sections 5.2 and 5.3, respectively. The resulting fitting surface becomes the new base surface and so on. In general, this procedure yields a more refined (i.e., more accurate) fitting surface to data points.

This process is repeated iteratively until a stopping criterion is reached. Usual stopping criteria are that the fitting error becomes smaller than a given threshold value or that successive iterations of this reconstruction pipeline no longer improve current solutions. In the former case, we assume that an error threshold value is provided as an input of the problem (as it usually happens in many industrial problems) and we compute the fitting error according to either (4) or (5). In the latter case, we compare the fitting error between successive iterations, and the process is stopped when no further improvement is achieved.

6. Illustrative Examples

Our method has been tested with several examples of different clouds of data points. To keep the paper at manageable size, we discuss here two of them. Examples in this paper are shown in Figures 5-6. For each example, two different pictures are displayed: at (a), we show the original cloud of input data points, represented as small red points; at (b), the best approximating NURBS surface, is shown as obtained with our functional-neural approach. Our input consists of sets of irregularly sampled data points (this fact can readily be seen from simple visual inspection of the point clouds at (a)), which are also affected by measurement noise of a signal-to-noise ratio of 15 : 1 in all examples. In this paper, a fitting error threshold value $\epsilon = 10^{-3}$ is considered. The other relevant parameters of the approximating NURBS surfaces are reported in Table 1, where the examples are arranged in rows. For each example, the following data are arranged in columns: number of data points, order of the approximating NURBS surface, length of knot vectors, and number of iterations required to obtain the fitting surface with a fitting error below the given threshold.

A simple visual inspection of the figures clearly shows that our method yields a very good approximating surface to data points in all cases. The low number of iterations required to obtain the fitting surface for the given threshold also confirms the good behavior of the method. From these examples and many others not reported here for the sake of brevity, we conclude that the presented method performs very well, with remarkable capability to provide a satisfactory solution to the general reconstruction problem with NURBS surfaces.

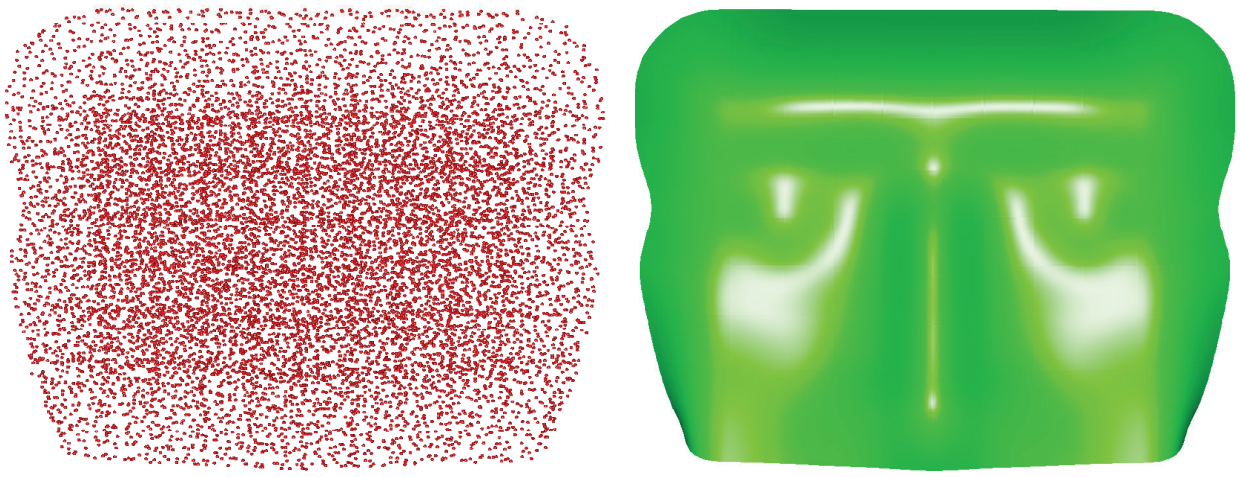
Regarding the implementation issues, all computations in this paper have been performed on a 2.9 GHz. Intel Core i7 processor with 8 GB. of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab*, version 2010b.

7. Comparison with Other Approaches

In this section, we compare the presented method with other alternative approaches for surface reconstruction based on neural networks. A careful revision of the literature in the field gives six previous contributions in the field, the works in [4, 8, 49, 70, 71, 77]. This small number is a clear indication of the difficulty and originality of the present work.

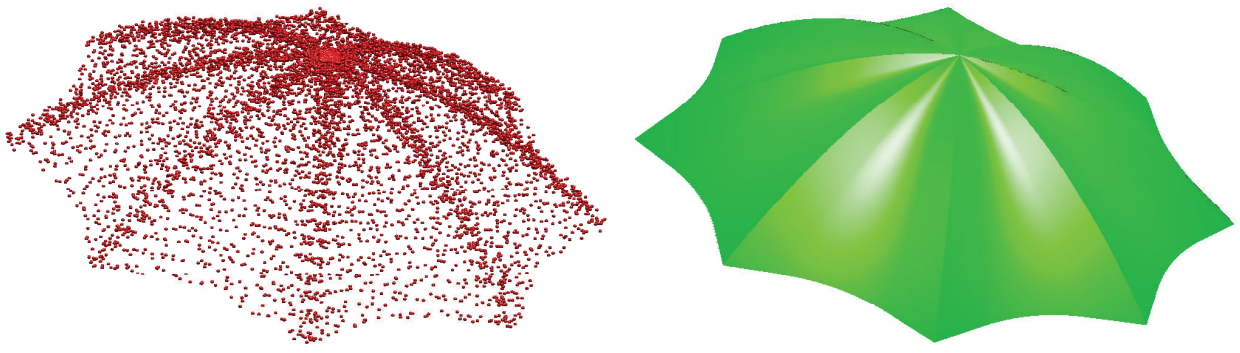
Comparative results of these methods are summarized in Table 2. The different methods are arranged in rows and sorted by year of publication. For each reported method, the columns give a brief description about its main features. Columns 2, 4, and 6 give a binary answer to three different questions: whether or not the indicated method provides algorithms for the subproblems of data parameterization, surface fitting, and support for NURBS surfaces, respectively. Answer *true* is marked with a check (✓), otherwise with symbol (×). Wherever a positive answer is found, a short description about the specific techniques incorporated in that method for the reported subproblem is given in columns 3 and 5, respectively. Note that all methods except [4, 8] address the data parameterization subproblem with a Kohonen neural network, and all methods except [70, 77] provide a procedure to compute the approximating surface, either as a Bézier surface [8], a B-spline surface [4, 71], or a polygonal mesh [49]. The works in [70, 77] assume that any traditional surface fitting technique will be used for this particular subproblem.

The most important difference of this method with respect to alternative approaches is that previous methods do not provide support for NURBS surfaces (see column 6 in Table 2). On the contrary, NURBS surfaces are fully supported in our method. In fact, the examples shown in the paper are more difficult to reconstruct through simple polynomial surfaces and require a larger number of parameters. In clear contrast, our method requires only a small number of parameters. Furthermore, the solution is reached with a small number of iterations. To the best of our knowledge, this is the first neural-based approach providing full support for all



(a) (b)

Figure 5: Mask example: (a) original data points; (b) fitting surface.



(a) (b)

Figure 6: Umbrella example: (a) original data points; (b) fitting surface.

steps of the general surface reconstruction problem by using NURBS surfaces.

8. Conclusions and Future Work

This paper proposes a new hybrid functional-neural approach to solve the surface reconstruction problem. In our approach, we combine two powerful artificial intelligence paradigms: the popular Kohonen neural network to address the data parameterization problem and a new functional network, called NURBS functional network, to reproduce the functional structure of the NURBS surfaces. These neural and functional networks are then applied iteratively for further surface refinement. The hybridization of these two networks provides us with a powerful computational approach to solve the surface reconstruction problem. The performance of our approach has been tested by its application to two illustrative examples. Our results show that the method performs very well, being able to reconstruct the approximating surface of the given set of data points with a high degree of accuracy and a low number of iterations.

The main limitation of this approach is that it requires some initial input such as the order of the approximating NURBS surface and the length of knot vectors, which are strongly dependent on the particular set of data points. Consequently, their optimal values might be difficult to choose for end users, thus preventing the method for automatic, human-independent reconstruction. This limitation opens the door for future research in the area in order to develop efficient algorithms for automatic determination of those optimal

Table 2: Comparison of the proposed method with other alternative methods based on neural networks for the surface reconstruction problem.

Author, year, and reference	Data parameterization	Method used	Surface fitting	Method used	NURBS supported?
Hoffmann and Várady (1998) [70]	√	Kohonen neural network	×		×
Yu (1999) [49]	√	Kohonen neural network	√	Polygonal mesh (edge swap)	×
Hoffmann (1999) [77]	√	Modified Kohonen neural network	×		×
Barhak and Fischer (2001) [71]	√	(i) Kohonen neural network (ii) Partial differential equations (PDE)	√	(i) Gradient descent algorithm (GDA) (ii) Random surface error correction (RSEC)	×
Iglesias et al. (2004) [4]	×		√	Tensor-product functional network	×
Gálvez et al. (2007) [8]	√	Genetic algorithms	√	Functional network	×
This method (2013)	√	Kohonen neural network	√	NURBS functional network	√

values. Another important limitation of the method occurs when the data points cannot be unambiguously projected onto the base surface. This problem can arise with closed surfaces often represented in implicit form. In those cases, the PCA method might fail to extract the real tendency of data. We are currently working on new strategies to overcome this problem. Future work also includes the extension of this approach to other kinds of approximating surfaces as well as the possible application of this methodology to some interesting industrial problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper. Any commercial identity mentioned in this paper is cited solely for scientific purposes.

Acknowledgments

This research has been kindly supported by the Computer Science National Program of the Spanish Ministry of Economy and Competitiveness, Project ref. no. TIN2012-30768, Toho University (Funabashi, Japan), and the University of Cantabria (Santander, Spain). The authors are particularly grateful to the Department of Information Science of Toho University for all the facilities given to carry out this work. Special thanks are owed to the editor and the four anonymous reviewers for their useful comments and suggestions that allowed us to improve the final version of this paper.

References

[1] T. Várady, R. R. Martin, and J. Cox, "Reverse engineering of geometric models—an introduction," *CAD Computer Aided Design*, vol. 29, no. 4, pp. 255–268, 1997.

[2] T. Varady and R. Martin, "Reverse engineering," in *Handbook of Computer Aided Geometric Design*, G. Farin, J. Hoschek, and M. Kim, Eds., pp. 651–681, Elsevier Science, 2002.

[3] R. E. Barnhill, *Geometric Processing for Design and Manufacturing*, SIAM, Philadelphia, Pa, USA, 1992.

[4] A. Iglesias, G. Echevarría, and A. Gálvez, "Functional networks for B-spline surface reconstruction," *Future Generation Computer Systems*, vol. 20, no. 8, pp. 1337–1353, 2004.

[5] A. Iglesias and A. Gálvez, "A new artificial intelligence paradigm for computer aided geometric design," in *Artificial Intelligence and Symbolic Computation*, vol. 1930 of *Lectures Notes in Artificial Intelligence*, pp. 200–213, 2001.

[6] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*, Springer, Heidelberg, Germany, 2002.

[7] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, and W. Wang, "Industrial geometry: recent advances and applications in CAD," *CAD Computer Aided Design*, vol. 37, no. 7, pp. 751–766, 2005.

[8] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, and J. Espinola, "Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation," in *Computational Science and Its Applications—ICCSA 2007*, vol. 4706 of *Lectures Notes in Computer Science*, pp. 680–693, 2007.

[9] G. E. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*, Morgan Kaufmann, San Francisco, Calif, USA, 5th edition, 2001.

[10] A. Gálvez and A. Iglesias, "Efficient particle swarm optimization approach for data fitting with free knot B-splines," *CAD Computer Aided Design*, vol. 43, no. 12, pp. 1683–1692, 2011.

[11] J. Hoschek and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, 1993.

[12] J. Ling and S. Li, "Fitting B-spline curves by least squares support vector machines," in *Proceedings of the International Conference on Neural Networks & Brain Proceedings (ICNNB '05)*, pp. 905–909, IEEE Press, Beijing, China, October 2005.

[13] T. C. M. Lee, "On algorithms for ordinary least squares regression spline fitting: a comparative study," *Journal of Statistical Computation and Simulation*, vol. 72, no. 8, pp. 647–663, 2002.

[14] W. Y. Ma and J. P. Kruth, "Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces," *Computer-Aided Design*, vol. 27, no. 9, pp. 663–675, 1995.

[15] W. P. Wang, H. Pottmann, and Y. Liu, "Fitting B-spline curves to point clouds by curvature-based squared distance minimization," *ACM Transactions on Graphics*, vol. 25, no. 2, pp. 214–238, 2006.

[16] H. P. Yang, W. P. Wang, and J. G. Sun, "Control point adjustment for B-spline curve approximation," *CAD Computer Aided Design*, vol. 36, no. 7, pp. 639–652, 2004.

[17] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, Oxford, UK, 1993.

[18] H. Park and J.-H. Lee, "B-spline curve fitting based on adaptive curve refinement using dominant points," *CAD Computer Aided Design*, vol. 39, no. 6, pp. 439–451, 2007.

[19] L. A. Piegl and W. Tiller, "Least-squares B-spline curve approximation with arbitrary end derivatives," *Engineering with Computers*, vol. 16, no. 2, pp. 109–116, 2000.

[20] D. L. B. Jupp, "Approximation to data by splines with free knots," *SIAM Journal on Numerical Analysis*, vol. 15, no. 2, pp. 328–343, 1978.

[21] J. R. Rice, *The Approximation of Functions*, vol. 2, Addison-Wesley, Reading, Mass, USA, 1969.

[22] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Reading, Mass, USA, 1991.

[23] M. G. Cox, "Algorithms for spline curves and surfaces," in *Fundamental Developments of Computer-Aided Geometric Design*, L. Piegl, Ed., pp. 51–76, Academic Press, London, UK, 1993.

[24] R. H. Franke and L. L. Schumaker :, "A bibliography of multivariate approximation," in *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker, and F. I. Utreras, Eds., Academic Press, New York, NY, USA, 1986.

[25] N. R. Draper and H. Smith, *Applied Regression Analysis*, Wiley-Interscience, 3rd edition, 1998.

[26] D. R. Forsey and R. H. Bartels, "Surface fitting with hierarchical splines," *ACM Transactions on Graphics*, vol. 14, pp. 134–161, 1995.

[27] H. Pottmann, S. Leopoldseder, and M. Hofer, "Approximation with active bspline curves and surfaces," in *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, pp. 8–25, IEEE Computer Society Press, 2002.

[28] M. Eck and H. Hoppe, "Automatic reconstruction of B-spline surfaces of arbitrary topological type," in *Proceedings of the Computer Graphics Conference (SIGGRAPH '96)*, pp. 325–334, August 1996.

[29] L. Piegl and W. Tiller, *The NURBS Book*, Springer, Berlin, Germany, 1997.

[30] M. Prasad, A. Zisserman, and A. Fitzgibbon, "Single view reconstruction of curved surfaces," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06)*, vol. 2, pp. 1345–1354, IEEE Computer Society Press, Los Alamitos, Calif, USA, June 2006.

[31] M. Levoy, K. Pulli, B. Curless et al., "The digital Michelangelo project: 3D scanning of large statues," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*, pp. 131–144, New Orleans, La, USA, July 2000.

[32] M. C. Leu, X. Peng, and W. Zhang, "Surface reconstruction for interactive modeling of freeform solids by virtual sculpting," *CIRP Annals—Manufacturing Technology*, vol. 54, no. 1, pp. 131–134, 2005.

[33] C. V. Alvino and A. J. Yezzi Jr., "Tomographic reconstruction of piecewise smooth images," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '04)*, vol. 1, pp. I576–I581, Los Alamitos, Calif, USA, July 2004.

[34] J. Rossignac and B. Borrel, "Multi-resolution 3D approximations for rendering complex scenes," in *Geometric Modeling in Computer Graphics*, pp. 455–465, Springer, 1993.

[35] C. L. Bajaj, F. Bernardini, and G. Xu, "Automatic reconstruction of surfaces and scalar fields from 3D scans," in *Proceedings of the 22nd Annual ACM Conference on Computer Graphics and Interactive Techniques*, pp. 109–118, August 1995.

[36] C. L. Bajaj, E. J. Coyle, and K.-N. Lin, "Arbitrary topology shape reconstruction from planar cross sections," *Graphical Models and Image Processing*, vol. 58, no. 6, pp. 524–543, 1996.

[37] M. Jones and M. Chen, "A new approach to the construction of surfaces from contour data," *Computer Graphics Forum*, vol. 13, no. 3, pp. 75–84, 1994.

[38] D. Meyers, S. Skinnwer, and K. Sloan, "Surfaces from contours," *ACM Transactions on Graphics*, vol. 11, no. 3, pp. 228–258, 1992.

[39] H. Park and K. Kim, "Smooth surface approximation to serial cross-sections," *CAD Computer Aided Design*, vol. 28, no. 12, pp. 995–1005, 1996.

[40] W. J. Gordon, "Spline-blended surface interpolation through curve networks," *Journal of Mathematics and Mechanics*, vol. 18, no. 10, pp. 931–952, 1969.

[41] H. Fuchs, Z. M. Kedem, and S. P. Uselton, "Optimal surface reconstruction from planar contours," *Communications of the ACM*, vol. 20, no. 10, pp. 693–702, 1977.

[42] T. Maekawa and K. H. Ko, "Surface construction by fitting unorganized curves," *Graphical Models*, vol. 64, no. 5, pp. 316–332, 2002.

[43] V. Savchenko, A. Pasko, O. Okunev, and T. Kunii, "Function representation of solids reconstructed from scattered surface points and contours," *Computer Graphics Forum*, vol. 14, no. 4, pp. 181–188, 1995.

[44] G. Echevarría, A. Iglesias, and A. Gálvez, "Extending neural networks for B-spline surface reconstruction," in *Computational Science—ICCS*, vol. 2330 of *Lectures Notes in Computer Science*, pp. 305–314, 2002.

[45] P. Gu and X. Yan, "Neural network approach to the reconstruction of freeform surfaces for reverse engineering," *Computer-Aided Design*, vol. 27, no. 1, pp. 59–64, 1995.

[46] B. Guo, "Surface reconstruction: from points to splines," *CAD Computer Aided Design*, vol. 29, no. 4, pp. 269–277, 1997.

[47] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics (ACM)*, vol. 26, no. 2, pp. 71–78, 1992.

[48] C. T. Lim, G. M. Turkiyyah, M. A. Ganter, and D. W. Storti, "Implicit reconstruction of solids from cloud point sets," in *Proceedings of the 3rd Symposium on Solid Modeling and Applications*, pp. 393–402, May 1995.

[49] Y. Yu, "Surface reconstruction from unorganized points using self-organizing neural networks," in *Proceedings of IEEE Visualization '99*, pp. 61–64, 1999.

[50] M. S. Floater, "Parametrization and smooth approximation of surface triangulations," *Computer Aided Geometric Design*, vol. 14, no. 3, pp. 231–250, 1997.

[51] Č. Oblonšek and N. Guid, "A fast surface-based procedure for object reconstruction from 3D scattered points," *Computer Vision and Image Understanding*, vol. 69, no. 2, pp. 185–195, 1998.

[52] R. M. Bolle and B. C. Vemuri, "On three-dimensional surface reconstruction methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 1, pp. 1–13, 1991.

[53] F. J. M. Schmitt, B. A. Barsky, and W.-H. Du, "An adaptive subdivision method for surface fitting from sampled data," *Computer Graphics (ACM)*, vol. 20, no. 4, pp. 179–188, 1986.

[54] T. A. Foley, "Interpolation of scattered data on a spherical domain," in *Algorithms for Approximation II*, J. C. Mason and M. G. Cox, Eds., pp. 303–310, Chapman and Hall, London, UK, 1990.

[55] S. Sclaroff and A. Pentland, "Generalized implicit functions for computer graphics," *ACM Siggraph Computer Graphics*, vol. 25, no. 4, pp. 247–250, 1991.

[56] V. Pratt, "Direct least-squares fitting of algebraic surfaces," *Computer Graphics (ACM)*, vol. 21, no. 4, pp. 145–152, 1987.

[57] A. Gálvez, A. Cobo, J. Puig-Pey, and A. Iglesias, "Particle swarm optimization for Bézier surface reconstruction," in *Computational Science—ICCS 2008*, vol. 5102 of *Lectures Notes in Computer Science*, pp. 116–125, 2008.

[58] A. Gálvez and A. Iglesias, "Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points," *Information Sciences*, vol. 192, pp. 174–192, 2012.

[59] A. Gálvez, A. Iglesias, and J. Puig-Pey, "Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction," *Information Sciences*, vol. 182, no. 1, pp. 56–76, 2012.

[60] M. Sarfraz and S. A. Raza, "Capturing outline of fonts using genetic algorithms and splines," in *Proceedings of the 5th International Conference on Information Visualization (IV' 01)*, pp. 738–743, IEEE Computer Society Press, 2001.

[61] F. Yoshimoto, M. Moriyama, and T. Harada, "Automatic knot adjustment by a genetic algorithm for data fitting with a spline," in *Proceedings of the International Conference on Shape Modeling*, pp. 162–169, IEEE Computer Society Press, 1999.

[62] F. Yoshimoto, T. Harada, and Y. Yoshimoto, "Data fitting with a spline using a real-coded genetic algorithm," *CAD Computer Aided Design*, vol. 35, no. 8, pp. 751–760, 2003.

[63] A. Gálvez, A. Iglesias, and A. Andreina, "Discrete Bézier curve fitting with artificial immune systems," in *Intelligent Computer Graphics 2012*, vol. 441 of *Studies in Computational Intelligence*, pp. 59–75, 2013.

[64] E. Ülker and A. Arslan, "Automatic knot adjustment using an artificial immune system for B-spline curve approximation," *Information Sciences*, vol. 179, no. 10, pp. 1483–1494, 2009.

[65] X. Zhao, C. Zhang, B. Yang, and P. Li, "Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation," *CAD Computer Aided Design*, vol. 43, no. 6, pp. 598–604, 2011.

[66] A. Gálvez and A. Iglesias, "Firey algorithm for polynomial Bézier surface parameterization," *Journal of Applied Mathematics*, vol. 2013, Article ID 237984, 9 pages, 2013.

[67] A. Gálvez and A. Iglesias, "Firey algorithm for explicit B-spline curve fitting to data points," *Mathematical Problems in Engineering*, vol. 2013, Article ID 528215, 12 pages, 2013.

[68] A. Gálvez and A. Iglesias, "A new iterative mutually-coupled hybrid GA-PSO approach for curve fitting in manufacturing," *Applied Soft Computing*, vol. 13, no. 3, pp. 1491–1504, 2013.

[69] A. Gálvez and A. Iglesias, "From nonlinear optimization to convex optimization through firey algorithm and indirect approach with applications to CAD/CAM," *The Scientific World Journal*, vol. 2013, Article ID 283919, 10 pages, 2013.

[70] M. Hoffmann and L. Várady, "Free-form surfaces for scattered data by neural networks," *Journal for Geometry and Graphics*, vol. 2, no. 1, pp. 1–6, 1998.

[71] J. Barhak and A. Fischer, "Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 1, pp. 1–16, 2001.

[72] E. Castillo, "Functional networks," *Neural Processing Letters*, vol. 7, no. 3, pp. 151–159, 1998.

[73] E. Castillo, A. Iglesias, and R. Ruiz-Cobo, *Functional Equations in Applied Sciences*, Elsevier Science, Amsterdam, The Netherlands, 2005.

[74] E. Castillo and A. Iglesias, "Some characterizations of families of surfaces using functional equations," *ACM Transactions on Graphics*, vol. 16, no. 3, pp. 296–318, 1997.

[75] T. Kohonen, *Self-Organization and Associative Memory*, vol. 8 of *Springer Series in Information Sciences*, Springer, New York, NY, USA, 3rd edition, 1989.

[76] T. Kohonen, *Self-Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*, Springer, Berlin, Germany, 3rd edition, 2001.

[77] M. Hoffmann, "Modified Kohonen neural network for surface reconstruction," *Publicationes Mathematicae*, vol. 54, pp. 857–864, 1999.

[78] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, Cambridge University Press, Cambridge, UK, 2nd edition, 1986.

[79] J. Puig-Pey, A. Gálvez, A. Iglesias, J. Rodríguez, P. Corcuera, and F. Gutiérrez, "Some applications of scalar and vector fields to geometric processing of surfaces," *Computers & Graphics*, vol. 29, no. 5, pp. 723–729, 2005.