Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 237984, 9 pages
http://dx.doi.org/10.1155/2013/237984

Research Article

Firefly Algorithm for Polynomial Bézier Surface Parameterization

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

[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 21 June 2013; Accepted 7 August 2013

Academic Editor: Xin-She Yang

A classical issue in many applied fields is to obtain an approximating surface to a given set of data points. This problem arises in Computer-Aided Design and Manufacturing (CAD/CAM), virtual reality, medical imaging, computer graphics, computer animation, and many others. Very often, the preferred approximating surface is polynomial, usually described in parametric form. This leads to the problem of determining suitable parametric values for the data points, the so-called surface parameterization. In real-world settings, data points are generally irregularly sampled and subjected to measurement noise, leading to a very difficult nonlinear continuous optimization problem, unsolvable with standard optimization techniques. This paper solves the parameterization problem for polynomial Bézier surfaces by applying the firefly algorithm, a powerful nature-inspired metaheuristic algorithm introduced recently to address difficult optimization problems. The method has been successfully applied to some illustrative examples of open and closed surfaces, including shapes with singularities. Our results show that the method performs very well, being able to yield the best approximating surface with a high degree of accuracy.

1. Introduction

Obtaining a curve or surface that approximates a given cloud of data points is a classical problem in several scientific and technological domains such as computer-aided design and manufacturing (CAD/CAM), virtual reality, medical imaging, computer graphics, computer animation, and many others. In real-world settings, data points come from real measurements of an existing geometric entity, as it typically happens in the construction of car bodies, ship hulls, airplane fuselage, and other free-form objects [1–8]. This process is also applied in the shoes industry, in archeology (reconstruction of archeological assets), in medicine (computed tomography), and in many other fields. The primary goal is to convert the real data from a physical object into a fully usable digital model, a process called reverse engineering. Such digital models are usually easier and cheaper to modify than their real counterparts, leading to a significant reduction of the costs associated with the processing and manufacturing time of the real goods they represent. Furthermore, due to their inherent digital nature, they become available anytime and anywhere, a very valuable feature in our current *digital-world* era.

Data points in reverse engineering are usually acquired through laser scanning and other digitizing methods (light digitizers, coordinate measuring machines, CT scanners, and tactile scanners) and are, therefore, subjected to measurement noise, irregular sampling, and other artifacts [7, 9]. Consequently, a good fitting of data is generally based on approximation schemes (where the curve/surface is expected to pass *near* the data points) rather than on interpolation (where the curve/surface is constrained to pass *through* all input data points). 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 irregularly sampled noisy data points.

There are two key components for a good approximation of data points: a proper choice of the approximating function

and a suitable parameter tuning. The usual models for data fitting in CAD/CAM and other industrial fields are free-form parametric entities, such as Bézier, B-spline, and NURBS, as they have a great flexibility and can represent well any smooth shape with only a few parameters, thus leading to substantial savings in terms of computer memory and storage capacity [10–17].

In this paper we focus particularly on the case of polynomial Bézier surfaces, a kind of free-form splines very popular in fields such as CAD/CAM and computer graphics. Bézier splines were developed independently in the early 60s by Paul de Casteljau and Pierre Bézier for the CAD systems of the French automotive companies Citröen and Renault, respectively. Mathematically, they are based on the Bernstein polynomials (see Section 4 for details), developed as early as 1912 but whose applicability to engineering design was unknown until the 60s. A Bézier curve is a linear combination of the Bernstein polynomials and vector coefficients called control points. The curve follows approximately the shape of its control polygon (the collection of segments joining the control points), and hence, it reacts to the movement of its control points by following a push-pull effect. This powerful feature was fundamental for the popularization of free-form curves and surfaces for interactive design. The generalization of this idea to surfaces leads to the Bézier surfaces, which are linear combinations of the control points (now arranged in a three-dimensional net) and the so-called tensor-product basis functions (given by the products of all possible combinations of univariate Bernstein polynomials in surface parameters u and v, resp.).

Although nowadays Bézier splines have been overtaken by the B-splines (developed during the 70s and of which the Bézier splines are a particular case), they played a key role in the current development of computer design. In addition to their historical value, they are still widely used today for different purposes, such as computer fonts (e.g., TrueType fonts, PostScript), computer animation (for simple movements of objects in programs such as Adobe Flash), and computer design (Adobe Photoshop, Corel Draw, Adobe illustrator). The reader is referred to [18–20] for further details about the subject. See also [21] for a nice historical approach written by some of the most prominent figures in the field.

Best approximation methods make commonly use of least-squares techniques [1, 8, 10, 13, 14, 22–28], where the goal is to obtain the relevant parameters of the polynomial approximating surface that fits the data points better in the least-squares sense. This problem is far from being trivial: because the surface is parametric, we are confronted with the problem of obtaining a suitable parameterization of the data points [18, 20]. As remarked in [29], the selection of an appropriate parameterization is essential for a good fitting. Unfortunately, it also becomes a very hard problem, specially for the cases of irregularly sampled noisy data points. In fact, it is well known that it leads to a very difficult overdetermined continuous nonlinear optimization problem. 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. Finally, it is usually a multimodal problem as well, because of the potential existence of several (global or local) optima of the objective function.

In this context, the present paper describes a new method to solve this challenging parameterization problem for free-form polynomial Bézier surfaces. Our method applies a powerful nature-inspired metaheuristic algorithm, called firefly algorithm, introduced recently by Professor Yang (Cambridge University) to solve difficult optimization problems. The trademark of the firefly algorithm is its search mechanism, inspired by the social behavior of the swarms of fireflies and the phenomenon of bioluminescent communication. The paper shows that this approach can be effectively applied to obtain an optimal approximating Bézier surface to a given set of noisy data points, provided that an adequate representation of the problem and a proper selection of the parameters are carried out. To check the performance of our approach, it has been applied to some illustrative examples of open and closed surfaces, including shapes with singularities. Our results show that the method performs very well, being able to yield the best approximating surface with a high degree of accuracy.

The structure of this paper is as follows: in Section 2 the previous work in the field is briefly reported. Then, the fundamentals and main ideas of the firefly algorithm, the method used in this paper, are briefly explained in Section 3. Our proposed firefly-based method for data fitting with Bézier surfaces is described in Section 4. The section begins with the description of the problem to be solved. Then, the application of the firefly algorithm to solve it is explained in detail. Some illustrative examples of its application to open and closed surfaces, including shapes with singularities, along with some implementation details are reported in Section 5. The paper closes with the main conclusions of this contribution and our plans for future work in the field.

2. Previous Work

The problem of data fitting through free-form parametric surfaces has been the subject of research for many years [1, 20, 30–35]. One of the most important problems regarding this issue is the surface parameterization, that is, the computation of suitable parametric values for the fitting surface to data points. In many practical situations, it is advisable to obtain a parameterization as similar as possible to the arc-length parameterization. The ultimate reason for this is that a constant step on the parametric domain automatically translates into a constant distance along an arc-length parameterized curve on the surface. In other words, for constant parameter intervals, the curve on the surface exhibits a point spacing that is as uniform as possible. Therefore, this parameterization is very convenient for surface interrogation issues, such as surface intersections or measuring distances on a surface [36, 37]. For instance, it has been traditionally applied in metrology for design and manufacturing, to collect measurement data from industrial parts of the designed and manufactured products. Many other industrial operations also require a uniform parameterization. For example, in computer controlled milling operations, the curve path followed by the milling machine must be parameterized such

that the cutter neither speeds up nor slows down along the path [9]. This property is only guaranteed when the curve path is parameterized with the arc-length parameterization. Consequently, this has been the preferred and most classical choice for surface parameterization.

Some recent papers have shown that the application of Artificial Intelligence techniques can achieve remarkable results regarding this parameterization problem [2, 5, 6, 38–40]. Most of these methods rely on some kind of neural networks, either standard neural networks [38], Kohonen's SOM (Self-Organizing Maps) nets [29, 39], or the Bernstein Basis Function (BBF) network [40]. In the case of surfaces, the network is used exclusively to order the data and create a grid of control vertices with quadrilateral topology [39]. After this preprocessing step, any standard surface reconstruction method (such as those referenced in the bibliography) has to be applied. In some other cases, the neural network approach is combined with partial differential equations [29] or other approaches. The generalization to functional networks (an extension of neural networks where the weights are replaced by functions) is also analyzed in [2, 5, 6, 41].

Due to their good behavior for complex optimization problems involving ambiguous and noisy data, there has recently been an increasing interest in applying nature-inspired optimization techniques (such as metaheuristics and evolutionary methods) to this problem. However, there are still few works reported in the literature. A previous paper in [42] describes the application of genetic algorithms and functional networks yielding pretty good results for both curves and surfaces. 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 are described in [4, 10] for particle swarm optimization (PSO), [3, 27, 28, 43] for genetic algorithms (GA), [44, 45] for artificial immune systems, [46] for estimation of distribution algorithms, and [11] for hybrid GA-PSO techniques. The method used in this paper also belongs to this category, as described in next section.

3. The Firefly Algorithm

The firefly algorithm is a nature-inspired metaheuristic algorithm introduced in 2008 by Yang to solve optimization problems [47, 48] (see also [49] for a recent modified version of this algorithm). The algorithm is based on the social flashing behavior of fireflies in nature. The key ingredients of the method are the variation of light intensity and formulation of attractiveness. In general, the attractiveness of an individual is assumed to be proportional to their brightness, which in turn is associated with the encoded objective function. The reader is kindly referred to [50] for a comprehensive review of the firefly algorithm and other nature-inspired metaheuristic approaches. See also [51] for a gentle introduction to metaheuristic applications in engineering optimization.

In the firefly algorithm, there are three particular idealized rules, which are based on some of the major flashing characteristics of real fireflies [47]. They are

(1) all fireflies are unisex, so that one firefly will be attracted to other fireflies regardless of their sex;

(2) the degree of attractiveness of a firefly is proportional to its brightness, which decreases as the distance from the other firefly increases due to the fact that the air absorbs light. For any two flashing fireflies, the less brighter one will move towards the brighter one. If there is not a brighter or more attractive firefly than a particular one, it will then move randomly;

(3) the brightness or light intensity of a firefly is determined by the value of the objective function of a given problem. For instance, for maximization problems, the light intensity can simply be proportional to the value of the objective function.

The distance between any two fireflies i and j, at positions $\mathbf{X}_i$ and $\mathbf{X}_j$, respectively, can be defined as a Cartesian or Euclidean distance as follows:

$$r_{ij} = \left\| \mathbf{X}_i - \mathbf{X}_j \right\| = \sqrt{\sum_{k=1}^{D} \left(x_{i,k} - x_{j,k} \right)^2}, \tag{1}$$

where $x_{i,k}$ is the k-th component of the spatial coordinate $\mathbf{X}_i$ of the i-th firefly and D is the number of dimensions.

In the firefly algorithm, as attractiveness function of a firefly j one should select any monotonically decreasing function of the distance to the chosen firefly, for example, the exponential function:

$$\beta = \beta_0 e^{-\gamma r_{ij}^{\mu}} \quad (\mu \geq 1), \tag{2}$$

where r_{ij} is the distance defined as in (1), β_0 is the initial attractiveness at $r = 0$, and γ is an absorption coefficient at the source which controls the decrease of the light intensity.

The movement of a firefly i which is attracted by a more attractive (i.e., brighter) firefly j is governed by the following evolution equation:

$$\mathbf{X}_i = \mathbf{X}_i + \beta_0 e^{-\gamma r_{ij}^{\mu}} \left(\mathbf{X}_j - \mathbf{X}_i \right) + \alpha \left(\sigma - \frac{1}{2} \right), \tag{3}$$

where the first term on the right-hand side is the current position of the firefly, the second term is used for considering the attractiveness of the firefly to light intensity seen by adjacent fireflies, and the third term is used for the random movement of a firefly in case there are not any brighter ones. The coefficient α is a randomization parameter determined by the problem of interest, while σ is a random number generator uniformly distributed in the space $[0, 1]$.

The method described in previous paragraphs corresponds to the original version of the firefly algorithm (FFA), as originally developed by its inventor. Since then, many different modifications and improvements on the original version have been developed, including the discrete FFA, multiobjective FFA, chaotic FFA, parallel FFA, elitist FFA, Lagrangian FFA, and many others, including its hybridization with other techniques. The interested reader is referred to the nice paper in [52] for a comprehensive, updated review and taxonomic classification of the firefly algorithms and all its variants and applications.

4. The Proposed Method

A free-form polynomial parametric surface is defined as [18, 19]:

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{ij}\phi_i(u)\,\varphi_j(v), \tag{4}$$

where $\{\mathbf{P}_{ij}\}_{i,j}$ are vector coefficients in $\mathbb{R}^3$ (usually referred to as the control points as they roughly control the shape of the surface), $\{\phi_i(u)\varphi_j(v)\}_{i,j}$ are the tensor-product functions obtained from two sets of basis functions (or *blending functions*) $\{\phi_i(u)\}_i$, and $\{\varphi_j(v)\}_j$, and (u, v) are the surface parameters, usually defined on a bounded rectangular domain $[\alpha_u, \beta_u] \times [\alpha_v, \beta_v] \subset \mathbb{R}^2$. Note that in this paper vectors are denoted in bold.

In this work we will focus on the particular case of free-form polynomial Bézier surfaces. In this case, (4) becomes

$$\mathbf{S}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u)\,\Psi_j^n(v), \tag{5}$$

where the blending functions $\Psi_k^d(\omega)$ are the Bernstein polynomials of index k and degree d, given by

$$\Psi_k^d(\omega) = \binom{d}{k}\omega^k(1-\omega)^{d-k}, \tag{6}$$

where

$$\binom{d}{k} = \frac{d!}{k!\,(d-k)!}, \tag{7}$$

and the surface parameters u, v are defined on the unit square $[0, 1] \times [0, 1]$. Note that, by convention, $0! = 1$.

Let us suppose now that we are given a set of data points $\{\mathbf{Q}_{k,l}\}_{k=1,\ldots,p;l=1,\ldots,q}$ in an ξ-dimensional space (usually $\xi = 2$ or $\xi = 3$). Our goal is to obtain the free-form polynomial Bézier surface $\mathbf{S}(u, v)$ that fits the data points better in the discrete least-squares sense. To do so, we have to compute the control points $\{\mathbf{P}_{i,j}\}_{i=0,\ldots,m;j=0,\ldots,n}$ of the approximating surface by minimizing the least-squares error, E, defined as the sum of squares of the residuals:

$$E = \sum_{k=1}^{p}\sum_{l=1}^{q}\left(\mathbf{Q}_{k,l} - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_k)\,\Psi_j^n(v_l)\right)^2. \tag{8}$$

In the case of irregularly sampled data points $\{\mathbf{Q}_r\}_{r=1,\ldots,R}$, our method will work in a similar way by simply replacing the previous expression (8) by

$$E = \sum_{r=1}^{R}\left(\mathbf{Q}_r - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_r)\,\Psi_j^n(v_r)\right)^2. \tag{9}$$

The least-squares minimization of either (8) or (9) leads to the system of equations:

$$\langle\mathbf{Q}\rangle = \langle\mathbf{P}\rangle \cdot \Xi, \tag{10}$$

where $\langle\mathbf{Q}\rangle$ corresponds to the vectorization of the set of data points $\{\mathbf{Q}_{k,l}\}_{k=1,\ldots,p;l=1,\ldots,q}$ (alternatively, $\{\mathbf{Q}_r\}_{r=1,\ldots,R}$), $\langle\mathbf{P}\rangle$ corresponds to the vectorization of the set of control points $\{\mathbf{P}_{i,j}\}_{i=0,\ldots,m;j=0,\ldots,n}$, and Ξ is a matrix given by $\Xi_{i,j} = \Psi^n(v_j) \odot \Psi_0^m(\mathbf{u})$, with $\Psi^d(\omega_k) = (\Psi_0^d(\omega_k), \ldots, \Psi_D^d(\omega_k))$, $\Psi_k^d(\Theta) = (\Psi_k^d(\theta_1), \ldots, \Psi_k^d(\theta_K))$, for any $\Theta = (\theta_1, \ldots, \theta_K)$, and $\odot$ represents the tensor product of vectors. The indices in (10) vary in the ranges of values indicated throughout the section.

The algebraic solution of (10) is given by, $\mathbf{P} = \Xi^+ \cdot \mathbf{Q}$, where Ξ^+ denotes the Moore-Penrose pseudoinverse of Ξ. Due to the fact that the blending functions are nonlinear in u and v, the least-squares minimization of the errors is a strongly nonlinear problem, with a large number of unknowns for large sets of data points. Our strategy for solving the problem consists of applying the firefly algorithm to determine suitable parameter values for the least-squares minimization of functional E according to either (8) or (9). However, in order to do it, some previous steps must be carefully carried out.

(1) First of all, we need an adequate representation of the unknowns of the problem. Because of the tensor-product structure of the free-form Bézier surfaces, the fireflies in our method can be encoded as either strings of two sorted real-coded vectors on the interval $[0, 1]$ of length p and q, respectively, for organized data points, or as sorted real-coded vectors of length R for the case of irregularly sampled data points. All fireflies are initialized with sorted uniformly distributed random numbers on the coordinate parametric domain.

(2) The objective function corresponds to the evaluation of the least-squares function given by either (8) or (9). Since this error function does not consider the number of data points, we also compute the RMSE (root-mean squared error), given by

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{p}\sum_{l=1}^{q}\left(\mathbf{Q}_{k,l} - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_k)\,\Psi_j^n(v_l)\right)^2}{p\cdot q}}, \tag{11}$$

for (8) or, alternatively by:

$$\text{RMSE} = \sqrt{\frac{\sum_{r=1}^{R}\left(\mathbf{Q}_r - \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{i,j}\Psi_i^m(u_r)\,\Psi_j^n(v_r)\right)^2}{R}}, \tag{12}$$

for (9) and report our results by using these error criteria.

(3) We also need to choose the degree of the approximating surface, which in turn depends on the number of control points. This value is chosen according to the complexity of the shape of the underlying function of data. In general, a small amount of control points is needed for simple, smooth shapes, while

a large number of control points must be selected for complicated, twisted, or irregular shapes. Since this number is unknown a priori, it is advisable to start with a low number of control points for each parametric coordinate and increase it until the error reaches values below a prescribed threshold, which generally depends on both the underlying surface and the application domain.

(4) Regarding the firefly algorithm, some control parameters should be set up. As usual when working with metaheuristic techniques, the choice of suitable control parameters is very important as it determines the performance of the method at large extent. It is also challenging, because it is strongly problem dependent. In this paper, our choice is based on a large collection of empirical results. These control parameters are

(a) the number of fireflies, n_f: this value is set up to $n_f = 100$ fireflies in all examples of this paper. We also tried larger populations of fireflies (up to 1000 individuals) but found that our results do not change significantly. Since larger populations mean larger computation times with no remarkable improvement at all, we found this value to be appropriate in our simulations;

(b) the number of iterations, n_{iter}: this number is another parameter of the method that has to be determined in order to run the algorithm until the convergence of the minimization of the error is achieved. In general, the firefly algorithm does not need a large number of iterations to reach the global optima. This also happens in this case. In all our simulations, we found that $n_{\text{iter}} = 10$ is a suitable value, as larger values for this parameter does not improve our results;

(c) the initial attractiveness, β_0: some theoretical results suggest that $\beta_0 = 1$ is a good choice for many optimization problems. We also take this value in this paper, with very good results, as it will be discussed in next section;

(d) the absorption coefficient, γ: it is set up to $\gamma = 0.5$ in this paper, as this value provides a quick convergence of the algorithm to the optimal solution;

(e) the potential coefficient, μ: although any positive value can be used for this parameter, the light intensity varies according to the inverse square law. Therefore, we choose $\mu = 2$ accordingly;

(f) the randomization parameter, α. This parameter varies on the interval $[0, 1]$ and allows us to determine the degree of randomization introduced in the algorithm. This stochastic component is necessary in order to allow new solutions appear and avoid getting stuck in a local minimum. However, larger values introduce large perturbations on the evolution of the firefly and, therefore, delay convergence to the global optima. Consequently, it is advisable to select values in between. In this work, we take $\alpha = 0.5$.

After the selection of those parameters, the firefly algorithm is performed iteratively for the given number of iterations. To remove the stochastic effects and avoid premature convergence, 20 independent executions have been carried out for each choice of the surface degree. Then, the firefly with the best (i.e., minimum) fitness value is selected as the best solution to the problem.

5. Experimental Results

To check the performance of our method described previously, it has been tested with a large collection of examples with excellent results in all cases. To keep the paper at manageable size, in this section we consider only three of them. They have been primarily chosen to reflect the diversity of situations to which the method can be applied. The examples correspond to both open and closed surfaces, including shapes with singularities. As the reader will see, they clearly show the good performance of our approach.

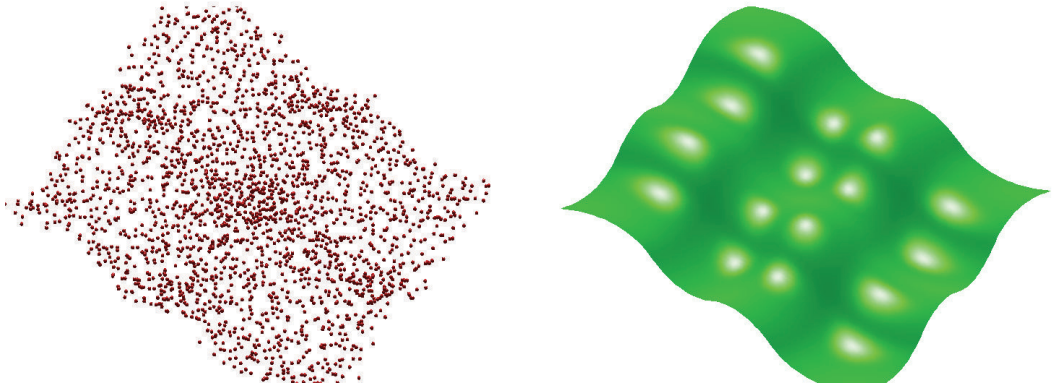
Examples in this paper are shown in Figures 1, 2, and 3. For each example, two different pictures are displayed: on the left, we show the original cloud of input data points, represented as small red points; on the right, the best approximating Bézier surface, as obtained with our firefly-based method, is displayed. Our input consists of sets of irregularly sampled data points (this fact can readily be seen from simple visual inspection of the point clouds on the left), which are also affected by measurement noise of low to medium intensity (signal-to-noise ratio of 15 : 1, 25 : 1, and 10 : 1, resp.). In all examples, no information about the data points parameterization is available at all. In fact, no information about the structure and properties of the underlying surface of data is either assumed or known beyond the data points.

Table 1 summarizes the main results of our computer simulations. The different examples are arranged in rows. For each example, the following data are arranged in columns: number of data points, E error value (according to (8) and (9)), the maximum of the E error (denoted by Max E and that provides a useful upper bound for that error), and RMSE error value (according to (11) and (12)). The error values are reported for each coordinate in all cases.

First observation is that, although our data points are irregularly sampled and affected by noise, the method yields very good fitting results in all cases. The RMSE is of order 10^{-3} in all cases, while the order of the least-squares E error is within the range 10^{-3}–10^{-2} and so is its maximum. Furthermore, these very small fitting errors are obtained for surfaces that are more complicated than it may seem at first sight. For instance, the surfaces of the first and third examples are apparently simple, flat, and height-map surfaces. However, a careful observation reveals that they oscillate several times, and hence, they exhibit a rich variety of hills and valleys, which have been highlighted by using an illumination model for the sake of clarity. On the other hand,

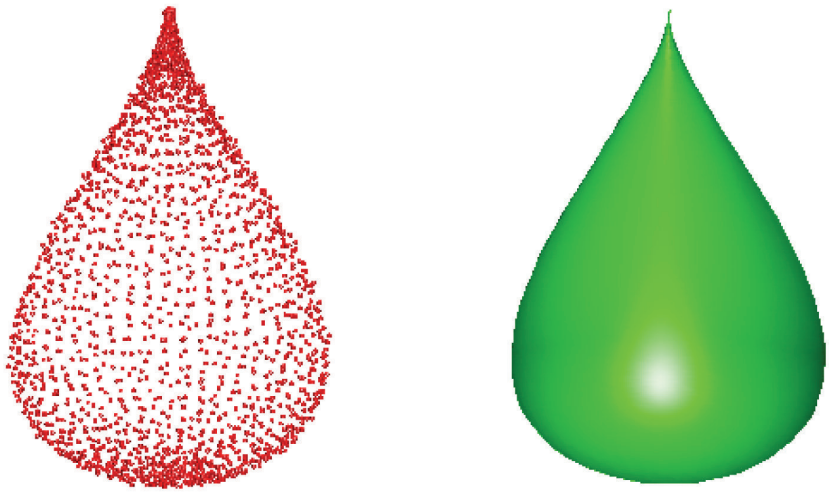
Table 1: Number of data points and error values (for each coordinate) of the three examples discussed in this paper.

Example	Number of data points	Error (E)	Error (Max E)	Error (RMSE)
Example 1	6572	x: 7.3652×10^{-2}	x: 9.5144×10^{-2}	x: 3.3476×10^{-3}
		y: 7.4303×10^{-2}	y: 9.8452×10^{-2}	y: 3.3624×10^{-3}
		z: 7.5126×10^{-2}	z: 9.9673×10^{-2}	z: 3.3811×10^{-3}
Example 2	3378	x: 5.2958×10^{-3}	x: 7.2446×10^{-3}	x: 1.2521×10^{-3}
		y: 5.1216×10^{-3}	y: 7.0237×10^{-3}	y: 1.2313×10^{-3}
		z: 5.2909×10^{-3}	z: 7.4532×10^{-3}	z: 1.2515×10^{-3}
Example 3	7312	x: 6.4191×10^{-2}	x: 8.4377×10^{-2}	x: 2.9629×10^{-3}
		y: 6.3774×10^{-2}	y: 8.3875×10^{-2}	y: 2.9532×10^{-3}
		z: 6.4746×10^{-2}	z: 9.3271×10^{-2}	z: 2.9756×10^{-3}



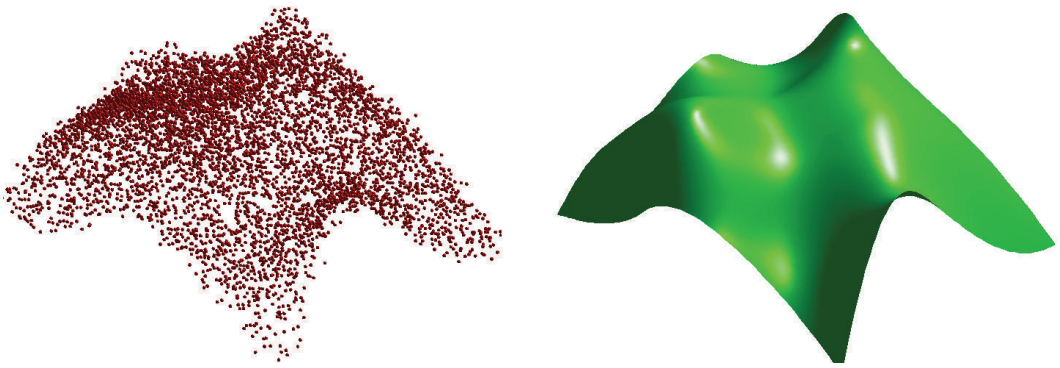
(a) (b)

Figure 1: Applying the firefly algorithm to Bézier surface approximation of data points: (a) original data points; (b) best approximating Bézier surface.



(a) (b)

Figure 2: Applying the firefly algorithm to Bézier surface approximation of data points: (a) original data points; (b) best approximating Bézier surface.



(a) (b)

Figure 3: Applying the firefly algorithm to Bézier surface approximation of data points: (a) original data points; (b) best approximating Bézier surface.

the second surface is a closed surface with a strong singularity at its uppermost part, where many data points concentrate in a very small volume. This is usually a very challenging feature for free-form parametric surfaces, which typically tend to distribute the control points by following a rectangular topology. Clearly, such a distribution is not adequate for this surface. To our delight, the proposed method identifies this situation automatically and rearranges the control points by itself to adapt to the underlying structure of data points. In opinion of the authors, this is a striking and very remarkable feature of this method and shows its ability to capture the real behavior of data points even under unfavorable conditions.

To summarize, a visual inspection of the three figures clearly shows that our method yields a very good approximating surface to data points in all cases. This fact is validated by the numerical results reported in Table 1, which confirm the good behavior of the method. From these examples and many other not reported here for the sake of brevity, we conclude that the presented method performs very well, with remarkable capability to provide a satisfactory, accurate solution to our parameterization problem with polynomial Bézier 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 for Windows 8 operating system.

6. Conclusions and Future Work

This paper introduces a new method to address the surface parameterization problem, that is, to compute a suitable parameterization of a set of data points in order to construct the free-form parametric surface approximating such data points better in the least-squares sense. This is a challenging problem that appears recurrently in reverse engineering for computer design and manufacturing and in many other industrial fields. Very often, data points in real-world settings are irregularly sampled and subjected to measurement noise, leading to a very difficult nonlinear continuous optimization problem, which cannot be solved by using standard optimization techniques. To overcome this limitation, this paper proposes a new method based on a powerful nature-inspired metaheuristic algorithm called firefly algorithm, introduced recently to solve difficult optimization problems. The method has been successfully applied to solve the parameterization problem for polynomial Bézier surfaces. The paper discusses the main issues in this problem, such as the solution representation and the selection of suitable control parameters. To check the performance of our approach, it has been applied to some illustrative examples of open and closed surfaces, including shapes with singularities. Our results show that the method performs very well, being able to yield the best approximating surface with a high degree of accuracy.

As mentioned in Section 3, the original firefly algorithm has been improved and modified in many different ways. Some of its variants have shown to be more efficient than the original version, meaning that the presented approach can arguably be improved with new, optimized features for better performance. An illustrative example is given by a very recent version called memetic self-adaptive firefly algorithm [53], whose new capabilities (the use of self-adaptation strategies on the control parameters, a new population model based on elitism, and the hybridization with a local search heuristics) improve the original firefly algorithm significantly. The application of many of these variants to our parameterization problem along with a comparative analysis of their performance is part of our future work. We are also interested to extend this method to other families of surfaces, such as the B-splines and NURBS, where the existence of additional parameters (such as knots and weights) can modify our procedure significantly. The application of this method to some interesting real-world problems in industrial settings is also part of our plans for future work.

Conflict of Interests

The authors of this paper have no current or past, direct or indirect financial relationship with any commercial identity

mentioned in this paper that might lead to any conflict of interests. They are solely mentioned here 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 Reference 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 research work. Special thanks are due to the Editor and the anonymous reviewers for their useful comments and suggestions that allowed us to improve the final version of this paper.

References

[1] R. E. Barnhill, *Geometry Processing for Design and Manufacturing*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 1992.

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

[3] 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, pp. 56–76, 2012.

[4] 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.

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

[6] 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.

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

[8] T. Varady and R. Martin, "Reverse engineering," in *Handbook of Computer Aided Geometric Design*, pp. 651–681, North-Holland, Amsterdam, The Netherlands, 2002.

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

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

[11] A. Galvez 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.

[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 and Brain Proceedings (ICNNB '05)*, pp. 905–909, Beijing, China, October 2005.

[13] 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.

[14] 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.

[15] W. Li, S. Xu, G. Zhao, and L. P. Goh, "Adaptive knot placement in *B*-spline curve approximation," *Computer Aided Design*, vol. 37, no. 8, pp. 791–797, 2005.

[16] H. Park, "An error-bounded approximate method for representing planar curves in *B*-splines," *Computer Aided Geometric Design*, vol. 21, no. 5, pp. 479–497, 2004.

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

[18] G. Farin, *Curves and Surfaces for CAGD*, Morgan Kaufmann, San Francisco, Calif, USA, 5th edition, 2002.

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

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

[21] D. F. Rogers, *An Introduction to NURBS: With His Historical Perspective*, Morgan Kaufmann, 2000.

[22] G. E. Hölzle, "Knot placement for piecewise polynomial approximation of curves," *Computer-Aided Design*, vol. 15, no. 5, pp. 295–296, 1983.

[23] W. Ma and J. 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.

[24] 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.

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

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

[27] 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 International and Applications*, pp. 162–169, IEEE Computer Society Press, 1999.

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

[29] 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.

[30] M. Alhanaty and M. Bercovier, "Curve and surface fitting and design by optimal control methods," *Computer Aided Design*, vol. 33, no. 2, pp. 167–182, 2001.

[31] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, Oxford, Miss, USA, 1993.

[32] T. Lyche and K. Mørken, "Knot removal for parametric *B*-spline curves and surfaces," *Computer Aided Geometric Design*, vol. 4, no. 3, pp. 217–230, 1987.

[33] M. J. D. Powell, "Curve fitting by splines in one variable," in *Numerical Approximation to Functions and Data*, J. G. Hayes, Ed., Athlone Press, London, UK, 1970.

[34] J. R. Rice, *Numerical Methods, Software and Analysis*, Academic Press, New York, NY, USA, 2nd edition, 1993.

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

[36] 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.

[37] A. Gálvez, J. Puig-Pey, and A. Iglesias, "A differential method for parametric surface intersection," in *Computational science and its applications—ICCSA 2004*, vol. 3044 of *Lecture Notes in Computer Science*, pp. 651–660, Springer, Berlin, Germany, 2004.

[38] 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.

[39] M. Hoffmann, "Numerical control of Kohonen neural network for scattered data approximation," *Numerical Algorithms*, vol. 39, no. 1–3, pp. 175–186, 2005.

[40] G. K. Knopf and J. Kofman, "Free-form surface reconstruction using Bernstein basis function networks," in *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE '99)*, vol. 9, pp. 797–802, ASME Press, November 1999.

[41] A. Iglesias and A. Galvez, "Applying functional networks to fit data points from *B*-spline surfaces," in *Proceedings of the Computer Graphics International (CGI '01)*, pp. 329–332, IEEE Computer Society Press, Hong Kong, China, 2001.

[42] A. Galvez, 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 *Lecture Notes in Computer Science*, pp. 680–693, 2007.

[43] 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.

[44] A. Galvez, A. Iglesias, and A. Avila, "Immunological-based approach for accurate fitting of 3D noisy data points with Bezier surfaces," in *Proceedings of the International Conference on Computational Science (ICCS '13)*, vol. 18, pp. 50–59, Procedia Computer Science, 2013.

[45] 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.

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

[47] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications*, vol. 5792 of *Lectures Notes in Computer Science*, pp. 169–178, Springer, Berlin, Germany, 2009.

[48] X. S. Yang, "Firey algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[49] S. L. Tilahun and H. C. Ong, "Modified firefly algorithm," *Journal of Applied Mathematics*, vol. 2012, Article ID 467631, 12 pages, 2012.

[50] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Frome, UK, 2nd edition, 2010.

[51] X.-S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, Wiley & Sons, New Jersey, NJ, USA, 2010.

[52] I. Fister, I. Fister Jr., X. S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*. In press.

[53] I. Fister, X. S. Yang, J. Brest, and I. Fister Jr., "Memetic self-adaptive firefly algorithm," in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, X. S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, Eds., pp. 73–102, Elsevier, 2013.